Tectonic Motions and Earthquake Deformation in Greece from GPS Measurements

Peter John Clarke
Exeter College

Thesis submitted for the degree of D.Phil.
Faculty of Physical Sciences, University of Oxford

Michaelmas Term, 1996

Supervised by Dr. P.C. England & Dr. B.E. Parsons

Abstract

Tectonic Motions and Earthquake Deformation in Greece from GPS Measurements

Peter John Clarke
Exeter College, Oxford

D.Phil. Michaelmas 1996

Sites in a 66-station geodetic network in central Greece have been occupied up to six times since 1989 using GPS surveying, and accurate positions have been computed using fiducially-improved or precise orbits. Site velocities are calculated under the assumptions that they are constant with time, after correcting for co-seismic effects, and that the position of the fixed base station (and hence the entire network) may be subject to small errors. Low-order polynomial expressions do not fit the velocity field well. The pattern of observed strain closely resembles that derived from independent geodetic observations made over a hundred-year time-scale. Significant geodetic strain is observed across the Gulf of Korinthos, even after the co-seismic displacement field of the $M_s = 6.2$ 1995 Egion earthquake has been removed by forward modelling. Geodetic strain is higher in the western than eastern Gulf, in contrast to the seismic strain which is similar throughout. Seismic strain matches geodetic strain in the east, but a significant deficit of seismic moment exists in the west which may represent a high earthquake hazard in the medium term.

The $M_s = 6.6$ 1995 Grevena earthquake struck a previously seismically quiet region well covered by a recent triangulation / trilateration survey. Ninety-one points from this network were reoccupied with GPS immediately after the earthquake, and site displacements computed. To invert for the earthquake source parameters from the geodetic displacement field, a novel inversion scheme is used which combines the Monte-Carlo and simplex approaches. *A priori* parameters are not required, even though the inverse problem is strongly nonlinear. The resulting focal mechanism agrees well with the global CMT solution and locally observed aftershocks, but implies a significantly higher scalar moment than do seismological studies. A network for observing post-seismic deformation has been established, which in view of the low background seismicity seems likely to provide significant results.

Extended Abstract

Tectonic Motions and Earthquake Deformation in Greece from GPS Measurements

Peter John Clarke
Exeter College, Oxford

D.Phil. Michaelmas 1996

Tectonic motion of the continents is not confined to narrow zones at plate boundaries but is characterised by wide regions of distributed seismic deformation. This arises as a result of the differing strengths and depth-dependent behaviour of oceanic and continental lithospheric materials. In addition, the presence of a ductile zone in the continental lower crust, beneath the brittle seismogenic upper crust, may decouple surface displacements from those of the lithosphere as a whole and permit post-seismic stresses to relax, and thus influence the seismic cycle.

The Aegean region is the most seismically active part of Europe and one of the most rapidly extending basins of the world. Tectonic motion is related to the gravitational spreading of thickened Anatolian and Aegean continental crust as the interface with the subducting African oceanic crust migrates southward with respect to Eurasia. Normal fault systems in central Greece provide the opportunity to study crustal deformation at various stages of the seismic cycle. The balance between inter-seismic geodetic build-up of regional strain and co-seismic localisation of strain has important consequences in terms of seismic hazard. Also, direct comparison between the co-seismic deformation inferred from seismology and that measured by geodesy assists with the understanding of historical seismological evidence. Post-seismic transient strains are expected to be small, but may have important effects and so must be quantified and modelled.

Geodetic surveying methods have been used to detect tectonic motions for many years, but terrestrial surveying techniques are limited by poor precision over long distances and so many years must elapse before a detectable signal has built up. Space geodetic techniques have recently been developed that are precise over long distances, but the equipment for SLR and VLBI measurements is bulky and expensive. The relative GPS carrier phase surveying technique is precise, portable and relatively cheap, and so offers the possibility to detect crustal tectonic motions over the time-scale of a few years by repeated temporary

occupation of markers situated in bedrock. The precision of earlier GPS surveys is limited by poor satellite orbit information, but can be improved using the technique of fiducial GPS, in which orbit parameters are estimated along with site coordinates, while the coordinates of 'fiducial' sites, that are well constrained by other space geodetic methods, and have velocities constrained similarly or by geological information, are held fixed. Later surveys can be processed using readily-available post-processed 'precise' orbits.

Version 3.4 of the Bernese GPS Software is used in this study. The double-difference observable (station-to-station, satellite-to-satellite) is used, with integer ambiguity parameters estimated and fixed where possible, with the aid of a model of ionospheric total electron content when necessary. Zenith tropospheric signal delays are estimated using a stochastic model. Baselines are chosen to maximise the number of common observations between stations at each end of each baseline, minimise the total length of processed baselines, and to minimise receiver type mixing. This strategy promotes ease of processing, reduces the likelihood of undetected cycle slips, and minimises receiver-type – induced errors. GPS observations are combined in daily networks to yield daily coordinate sets and their covariances, which are then combined using network adjustment software to generate a campaign coordinate set.

Relative GPS surveys yield site coordinates in a global reference frame, but relative to the base station at each epoch. An error in positioning over the mark at the base station will cause a translational error in the realisation of the reference frame at that epoch, but these offsets can be estimated if there are sites occupied three or more times, and the assumption of constant site velocities is made. To justify this assumption, temporal displacement discontinuities such as co-seismic deformation must first be eliminated by modelling, leaving only inter-seismic accumulation of strain. Translational errors will affect attempts to fit the time series of site coordinates by a low-order polynomial velocity field.

Sites in a 66-station network covering central Greece have been occupied with GPS up to six times over the interval 1989 – 1996. The network bounds the two major extensional features of the region, the Gulfs of Korinthos and Evvia. Data from the first three epochs (June 1989, October 1991 and May 1993) are processed using fiducially-improved satellite orbits, whereas later epochs (June 1995, October 1995 and May 1996) are processed using CODE precise orbits. Unfortunately, coordinates from the first epoch, even from the subset of sites which was occupied with dual-frequency GPS receivers, are too unreliable to be of use in short-term deformation studies. Later surveys of the Central Greece Network utilised dual-frequency receivers throughout and yield excellent results, although the three latest surveys only cover the sites around the Gulf of Korinthos.

The $M_s = 6.2$ 15 June 1995 Egion earthquake is the only event to cause significant co-seismic displacements of any sites in the Central Greece Network, and these displacements are removed from coordinate sets obtained after this date using a forward model based on the source mechanism of Bernard *et al.* (1996) (which is based on a combination of seismological, GPS, SAR and tectonic data), leaving inter-seismic

displacements only. Inter-seismic baseline length changes are smooth with time, demonstrating that the scale of the GPS reference frame is maintained through time. Whole-network translations at each epoch are estimated using a least-squares criterion to fit smooth velocities to each site, and attributed to possible errors in base station positioning. The translated coordinate sets can be fitted by temporally uniform site velocities, within the expected error bounds of GPS-derived coordinates. Whole-network rotations do not significantly improve the fit and so have not been applied.

The site velocities are expressed in a Europe-fixed reference frame using the NUVEL–1NNR and ITRF plate and site velocity models, which include the base station of the Central Greece Network (DION). All sites in the network show significant motion with respect to 'stable' Europe, indicating that deformation must occur to the northwest of the network in northwest Greece and Albania.

Polynomial velocity fields of up to 4^{th} order are unable to fit the time series of translated site coordinates, indicating that deformation is localised at a scale much smaller than the region. Analysis of uniform strain rates within small polygonal regions reveals significant high strains within the western Gulf of Korinthos which are not matched elsewhere, even in the eastern Gulf of Korinthos or Gulf of Evvia. Overall, the pattern of inter-seismic strain resembles that observed over the interval 1892 – 1992 by Davies *et al.* (1996), indicating that secular strain may be measured equally well over short time-scales in which no earthquakes take place or longer time-scales in which the effects of several earthquakes are averaged out.

The Gulf of Korinthos is the best-constrained part of the Central Greece Network by virtue of its longer span of occupation, and also exhibits the highest strain rates. As one of the most densely-populated regions of Greece, it has a good record of historical seismicity which can be related to the geodetic extension using the method of Kostrov (1974). Uncertainty in the scalar moment of historical earthquakes arises principally from instrumental imperfection and errors in the $M_s - M_0$ relationship for older events, and in the amount of strain released in aftershocks and pre- or post-seismic creep for all events. Additional strain release may have occurred in earthquakes too small to be included in the historical record, but previous workers have shown that this is unlikely to affect the total strain by more than 50% (Ambraseys & Jackson, 1990).

Sites in the northern Peleponnessos show very little relative motion, so across-Gulf extension can be easily studied in a reference frame in which these sites are fixed. Strike-perpendicular site velocities observed with GPS over the interval 1991 – 1996 increase smoothly from east to west, and the velocities obtained from triangulation / GPS data over the interval 1892 – 1992 by Davies *et al.* (1996) are commensurate with this trend. On average, geodetic extension rates in the western Gulf (12.7 ± 1.0 mm yr^{-1}) are approximately twice those in the east (6.4 ± 1.0 mm yr^{-1}).

In contrast, the rate of seismic moment release this century is marginally higher in the eastern Gulf of Korinthos than in the west, and the 'seismic' extension rates calculated in the east on the basis of a 10 km or 15 km seismogenic layer (6.0 ± 2.4 mm yr^{-1} or 4.0 ± 1.6 mm yr^{-1}) agree with the geodetic extension

rate, with a closer match for the higher rate based on a 10 km layer. In the western Gulf, the rate of seismic moment release implies extension rates of 3.0 ± 1.2 mm yr^{-1} (15 km layer) to 4.5 ± 1.8 mm yr^{-1} (10 km layer), significantly smaller even than the lowest geodetic extension rate observed at this end of the Gulf. The frequency of large earthquakes in the western Gulf during the period 1690 – 1890 has been higher than this century, and so it seems likely that the deficit of seismic strain release in the western Gulf will be met by several such earthquakes in the medium term. For a 10 km seismogenic layer, the total moment release required to eliminate the deficit of seismic strain is 22×10^{18} N m (for the maximum possible seismic strain that has already occurred), and for a 15 km layer the required moment release may at the extremes of possibility be as high as 52×10^{18} N m, equivalent to more than six earthquakes the size of the 24 February 1981 Alkyonides mainshock.

The $M_s = 6.6$ 13 May 1995 Kozani – Grevena earthquake struck a region of low historical seismicity in northwestern Greece in which geodetic strain has not previously been quantified. The epicentre is surrounded by a recently-occupied Hellenic Army Geodetic Service triangulation / trilateration network, from which pillars selected on the basis of a forward model of the expected co-seismic deformation were occupied with GPS immediately after the event. Co-seismic horizontal displacements are obtained by differencing the pre-seismic conventionally-surveyed and post-seismic GPS coordinate sets.

Surface displacements caused by uniform slip on a fault plane can be computed using the elastic model of Okada (1985), but the inverse problem is highly non-linear. A variation on previous inversion algorithms is proposed, in which the downhill simplex method of Nelder & Mead (1965) is augmented by the performance of several hundred inversions from randomly-selected starting points. The need for good *a priori* parameters is thus removed, and a more complete understanding of confidence limits is built up. The algorithm is tested using geodetic data obtained before and after the 1981 Alkyonides earthquake sequence.

The algorithm is then used to estimate the full set of source parameters of the 1995 Kozani – Grevena earthquake from the geodetic displacements obtained as above. Because the pre-seismic and post-seismic coordinates are expressed in different reference frames, scale, rotation and translation parameters are also estimated. Extensive stability tests are conducted on the solution using the actual data, synthetic data based on the actual site distribution, and synthetic gridded data.

The geodetic earthquake source mechanism agrees well with the Harvard CMT solution and studies of local aftershocks (Hatzfeld *et al.*, 1996) in all respects save the scalar moment corresponding to the geodetic solution (16.3×10^{18} N m for 1.2 m of displacement on the fault), which is over twice that of the CMT solution (7.6×10^{18} N m). This discrepancy cannot be explained purely by the cumulative effect of aftershocks, so pre- or post-seismic creep may be significant. Such a high proportion of aseismic creep relative to the seismic moment, if repeated for all earthquakes in the western Gulf of Korinthos, would not eliminate the discrepancy between geodetic and seismic strain that occurs in this area.

The minimum and maximum depth extents of faulting in the model are 2.8 km and 13.5 km respectively. Faulting does not propagate to the surface because near-surface deformation is distributed within a layer of unconsolidated sediments of mid-Pliocene and younger ages, which cover much of the region. The maximum depth of faulting is consistent with the aftershock distribution and expected values for the seismogenic layer thickness.

The model fault scarp does not correspond to any of the clear-cut surface features in the region, but to a poorly-defined scarp within the zone of soft surficial sediments, close to the location of the most prominent co-seismic ground cracking. Without geodetic observation and modelling it is likely that such a feature would have been overlooked in the search for faults associated with possible seismic hazard.

Post-seismic transient deformation completes the seismic cycle. Surface deformation caused by relaxation within a visco-elastic layer of the stresses induced by co-seismic elastic deformation is calculated using the model of Rundle (1982) and later co-workers, for the case of co-seismic motion in an elastic layer overlying a visco-elastic half-space. Post-seismic motion after the 1981 Alkyonides earthquakes is used as an example, and motion after the 1995 Grevena and Egion earthquakes is computed to aid geodetic network design. Relaxation by aseismic creep localised on the fault plane or its down-dip extension as a shear zone may also occur, and can be distinguished from visco-elastic distributed deformation by its temporal and spatial variation. When designing networks for the study of post-seismic displacements, consideration must be given to monument stability and the independent measurement of secular strain, because the post-seismic signal is small.

This thesis demonstrates that multiple-occupation GPS studies can quantify inter-seismic crustal deformation as precisely as and in agreement with medium-term GPS / triangulation studies, but in a far shorter time-scale, providing that co-seismic displacements of sites can be accurately modelled. Post-seismic transient deformation after normal-faulting earthquakes such as those studied is smaller in magnitude and can only be detected with GPS or more accurate techniques. Future work will include measurement and further modelling of post-seismic motion after the 1995 Grevena and Egion earthquakes, and integration of GPS and SAR data concerning the two. Also, the strain field in northwestern Greece and the northern Aegean Sea will be quantified by comparison of future GPS observations with existing terrestrial surveying measurements to determine the limits of the Aegean deformational zone and to estimate seismic hazard in the region.

Declaration

The contents of this thesis are all my own work, except where otherwise stated. The views and opinions expressed herein are mine and not necessarily those of any other person or body unless so attributed.

Acknowledgements

I gratefully acknowledge funding from the Natural Environment Research Council, the European Community, the University of Oxford, Exeter College (Oxford), and the Burdett-Coutts Fund (Department of Earth Sciences, University of Oxford) for various fieldwork, conference and living expenses incurred during the course of my time at Oxford. Equipment has been loaned by the Universities of Newcastle, Nottingham, and Aston, the Institut de Physique du Globe de Paris (IPGP), the National Technical University of Athens (NTUA), Ashtech Europe Ltd., and the NERC Geophysical Equipment Pool, Edinburgh (NERC GEP).

A large number of people (too many to name individually) have helped in one or more GPS campaigns in Greece, and as well as those mentioned below who have additional cause to be thanked I would particularly like to mention Paul Kearney (NERC GEP), Alan Wright (Global Surveys Ltd.) and Ingrid Hunstadt (Instituto Geofisico Nationale, Rome) who were all of great help in the field. Many others deserve to be thanked for other help, whether practical or theoretical, but the following people (grouped by geography) must be emphasised:

The University of Newcastle: Professor Paul Cross enabled me to spend some time there to learn about surveying and GPS, and has since been a source of much advice and encouragement in these matters. Paul Denys and Paul Cruddace (who later helped with GPS fieldwork) introduced me to the Bernese GPS processing software and provided many useful hints and tips on this and other surveying matters. The network adjustment software packages 'L3D' and 'Geodetic Suite' were provided by the University of Newcastle.

The University of Nottingham: Professor Vidal Ashkenazy and Dr. Terry Moore allowed me to attend two modules from the M.Sc. course in Surveying which were an invaluable introduction to geodesy. Dr. Richard Bingley has assisted with GPS fieldwork, planning, and also advice on processing GPS data.

Rest of the UK: Dr. James Jackson (University of Cambridge) and Professor Mike Leeder (University of Leeds) provided geological insight in both fields and tavernas in the Grevena and Egion regions. Dr. Jackson and Professor Nick Ambraseys (University College, London) have also made available unpublished data concerning historical earthquakes in the Gulf of Korinthos.

The National Technical University of Athens, Greece: Professor George Veis, Professor Haris Billiris and Assistant Professor Demitris Paradissis have provided logistical support during GPS fieldwork, and much advice on the details of Greek geodetic procedures. Professor Paradissis has in addition assisted in the computation of site displacements in cases where Hellenic Army Geodetic Service data have been used, and has also provided copies of relevant diploma theses and other unpublished data. Many students and technicians of NTUA have been invaluable in the field, not least for their linguistic skills, and I would particularly like to thank Mikhalis Tsoulakis and Stelios Felekis.

Rest of the world: Dr. Pierre Briole (Institut de Physique du Globe de Paris, France) and Dr. Denis Hatzfeld (Université Joseph Fourier, Grenoble, France) have been partners for discussion concerning the Grevena and Egion earthquakes, and sources of useful preprints of papers. They and their colleagues have assisted with the Grevena and Egion GPS campaigns. Dr. John Rundle (University of Colorado, USA) provided FORTRAN-77 code to calculate post-seismic displacements. The majority of diagrams in this thesis have been prepared using the GMT software developed by Dr. Paul Wessel (University of Hawaii, USA) and Dr. Walter Smith (National Oceanic and Atmospheric Administration, USA) and its appendages.

The University of Oxford (getting closer to home again!): various people, too numerous to mention, have done their best to keep me out of the lab and sane and their contributions have been vital. For technical matters, I would like to thank Andrew Curtis (for his work on the Alkyonides earthquakes which inspired mine), Rob Davies (for 100-year strain estimates provided in advance of publication, digital elevation model data, part-processing of the June 1995 Egion data, and general insight), Stephen Bourne (for many full and frank discussions concerning GPS and geophysics in general, and for doing his fieldwork in New Zealand) and Margaret Moore (for part-processing the October 1995 Egion data, proofreading this thesis, and just being there). All these and others (especially Jeremy Hyde) have helped with GPS fieldwork in Greece. Professor John Woodhouse provided timely CMT solutions for the Grevena and Egion earthquakes. Computing support was provided by Steve Usher.

While being supervised by Dr. Barry Parsons and Dr. Philip England I have been continually grateful for their support and encouragement, and their constructive criticism and helpful suggestions done much to improve this manuscript and develop my knowledge of geophysics.

Finally I would like to thank my parents, John and Elizabeth Clarke, and my grandfather, Tom Nye (to whose memory this thesis is dedicated), for their continued love and support while I have dabbled in and out of studenthood.

Contents

List of Figures

List of Tables

Glossary of abbreviations

AS	Anti-Spoofing
C/A	Coarse / Acquisition (code)
CDP	NASA Crustal Dynamics Project
CDDIS	NASA Crustal Dynamics Data Information System
CMT	Centroid Moment Tensor
CODE	Centre for Orbit Determination in Europe
EDM	Electronic Distance Measurement
GPS	Global Positioning System
GRS	Geodetic Reference System
GGRS	Greek Geodetic Reference System
HAGS	Hellenic Army Geodetic Service
IAG	International Association of Geodesy
IERS	International Earth Rotation Service
IGS	International GPS Geodynamics Service
InSAR	Interferometric Synthetic Aperture Radar
ISC	International Seismological Centre
ITRF	IERS Terrestrial Reference Frame
JPL	Jet Propulsion Laboratory (NASA)
NASA	National Aeronautics and Space Administration (USA)
NAVSTAR	Navigation System with Time and Ranging
NGS	National Geodetic Survey (USA)
NOAA	National Oceanic and Atmospheric Administration (USA)
P	Precise (code)
PRN	Pseudo-Random Noise (code)
RINEX	Receiver Independent Exchange (format)
r.m.s.	Root Mean Square
SA	Selective Availability
SAR	Synthetic Aperture Radar
SLR	Satellite Laser Ranging
SV	Space Vehicle
SVD	Singular Value Decomposition
TEC	Total Electron Content
USCG	United States Coast Guard
USGS	United States Geological Survey
UT	Universal Time
VLBI	Very Long Baseline Interferometry
WGS	World Geodetic System

Chapter 1

Introduction

One aim of the study of geophysics is to understand the behaviour of systems within the Earth in terms of simple physical models. One such system is the continental crust under conditions of active extension, which can be observed in a variety of places, *e.g.* the Aegean basin and the Basin and Range of the United States. A complete understanding of continental crustal extension would encompass many orders of magnitude in scale from the microseconds and nanometres of dislocation processes in mineral grains to the millions of years and thousands of kilometres of extensional basin evolution. No single theory can achieve this as the length- and time-scale that is appropriate for the study of one process in the system may be wholly inappropriate for another, nor can a single measurement technique provide all the observations necessary to study all processes. However, physical considerations can dictate which theories and techniques are appropriate for the study of particular physical processes.

The purpose of this study is to apply one measurement technique, geodetic surveying, to the behaviour of the continental crust under extension over time-scales of a few years and length-scales of a few kilometres to tens of kilometres, in the region of mainland Greece. The resulting observations of deformation are then interpreted using simple physical theories of crustal mechanics, and the results of these interpretations are compared with those arising from other methods. In order to do this it is first necessary to review appropriate current theories of continental dynamics and the tectonics of the Aegean region, and discuss wider questions applying to these subjects.

1.1 Active continental crustal deformation

A former tenet of plate tectonics was that the plates behave as rigid caps or shells (the lithosphere) moving on the surface of the Earth, above a lower layer (the asthenosphere) which behaves as a fluid over long

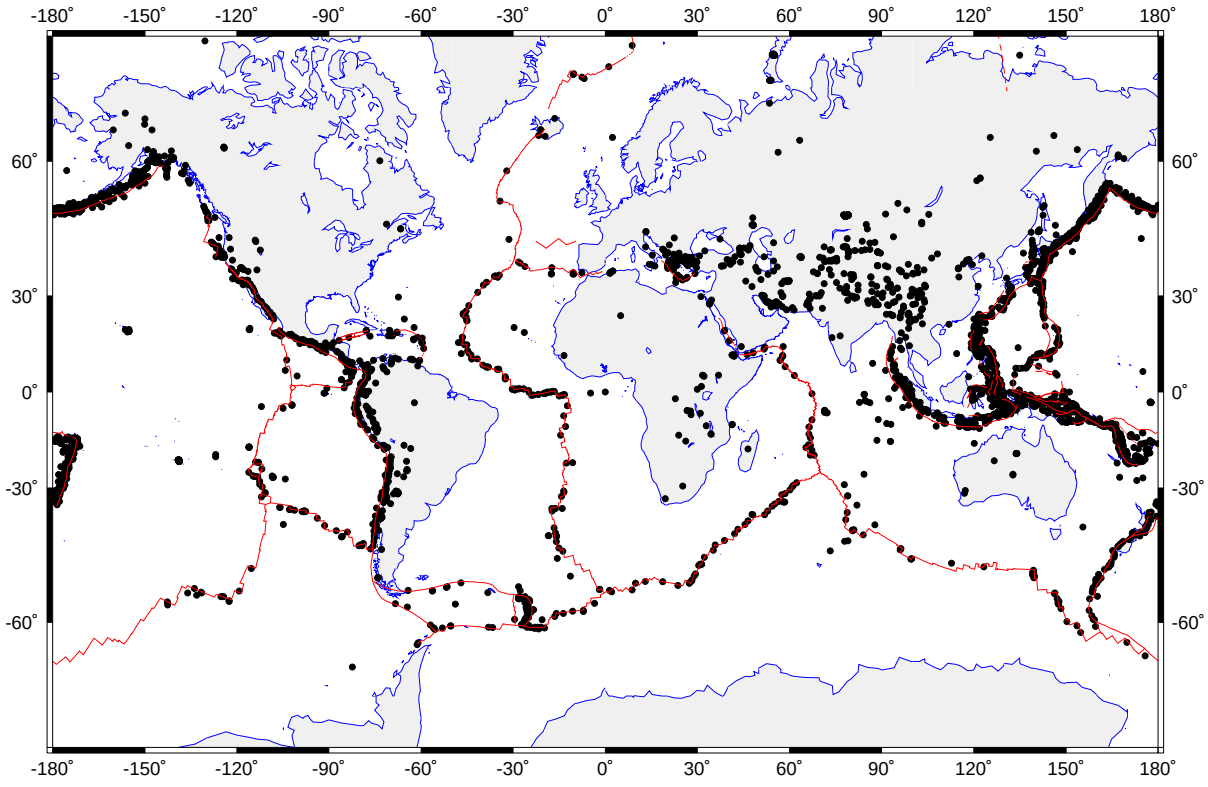

Figure 1.1: Locations of shallow earthquakes of magnitude $M_b \geq 5.5$ from the ISC catalogue 1964–1993. Approximate plate boundaries are shown in red. At oceanic–oceanic and oceanic–continental plate boundaries the zones of deformation revealed by seismicity are narrow, but for continental–continental plate boundaries such as in the Middle East and central Asia, the deformation is spread over a zone of several hundred kilometres in width.

time-scales, with the relative motion between plates taken up in very narrow zones at their boundaries. Whilst this seems to be true (to a first-order approximation) of the oceanic part of the plates (Figure 1.1), the situation in the continents is somewhat different. Here, plate boundaries are characterised by wide zones (several hundred kilometres) of distributed deformation as indicated by the seismicity, *e.g.* the boundary between the Indian and Asian plates where seismicity is distributed over a zone 2000 km wide in Tibet.

1.1.1 Vertical strength profiles and seismicity

One reason for this difference in character is that the continents and oceans are made up of different materials with different strengths and modes of failure at different depths (Chen & Molnar, 1983). To take a simplistic view, the continental crust is predominantly quartz-rich and therefore weak and able to deform

plastically by creep at relatively low temperatures (Byerlee, 1978; Scholz, 1988), whereas the oceanic crust is olivine-rich and therefore stronger and more inclined to brittle failure even at high temperatures (Goetze, 1978), so a theory can be developed based on the differences between these two materials. The lithospheric mantle beneath both continents and oceans is olivine-rich. Another reason for differing behaviour is that the geothermal gradient in the lithosphere differs between mature oceanic and continental lithosphere (as low as 10–15 K km^{-1}) and thinned continental lithosphere (as high as 25–30 K km^{-1}).

Figure 1.2 shows the strength profile of continental and oceanic lithosphere predicted by this model. It can be seen that oceanic lithosphere is several orders of magnitude stronger than continental lithosphere because of its lack of a weak lower crustal zone of plasticity, which explains the relative lack of internal deformation in the oceanic crust. More interestingly, it can be seen that in the continental lithosphere (for low to normal values of geothermal gradient), the strongest part is in the upper mantle, and there is a weak zone in the lower crust that should deform plastically (except at exceptionally high strain rates such as during an earthquake). This tallies with the observation that earthquakes, which are an expression of brittle failure, do not nucleate below depths of around 15 km in the continental crust (25 km for regions of old, thickened crust), and only in a few regions do they nucleate at greater depths in the continental lithospheric mantle (Chen & Molnar, 1983). Local variations in the maximum depth may occur as a result of lithological heterogeneity (Magistrale & Zhou, 1996).

The presence of this lower crustal plastic layer is significant because it permits stresses caused by earthquakes or lower crustal density contrasts (*e.g.* Moho topography) to relax (Kusznir & Matthews, 1988; Bird, 1991). Also, it will decouple the brittle upper crust from the lithospheric mantle to some extent (Braunmiller & Nábělek, 1996), which has important implications for basin-scale models of extension (McKenzie, 1978; Wernicke, 1981; Gans, 1987; Govers & Wortel, 1995), some of which require low-angle detachments or different amounts of stretching of the crust and mantle.

Depending on the crustal rheology, the strongest part of the continental lithosphere may behave plastically, because for certain values of thermal gradient, the brittle zone just below the base of the 'crust' in Figure 1.2 may be very thin or non-existent, yet the plastic strength of olivine at this depth is higher than the maximum brittle strength of the overlying quartzose crust. This is significant because it forms part of the justification for modelling large-scale deformation of the continental lithosphere in terms of viscous flow (England & McKenzie, 1982; England & McKenzie, 1983). If this model holds true then continental lithospheric deformation is expected to be smooth and steady over large distances, but if the upper crust is decoupled from the lithospheric mantle by a weak lower crustal zone, then small-scale motions observed on the Earth's surface may not reflect the motion at depth. An important question in crustal deformation studies is: at what length scale does this treatment become appropriate?

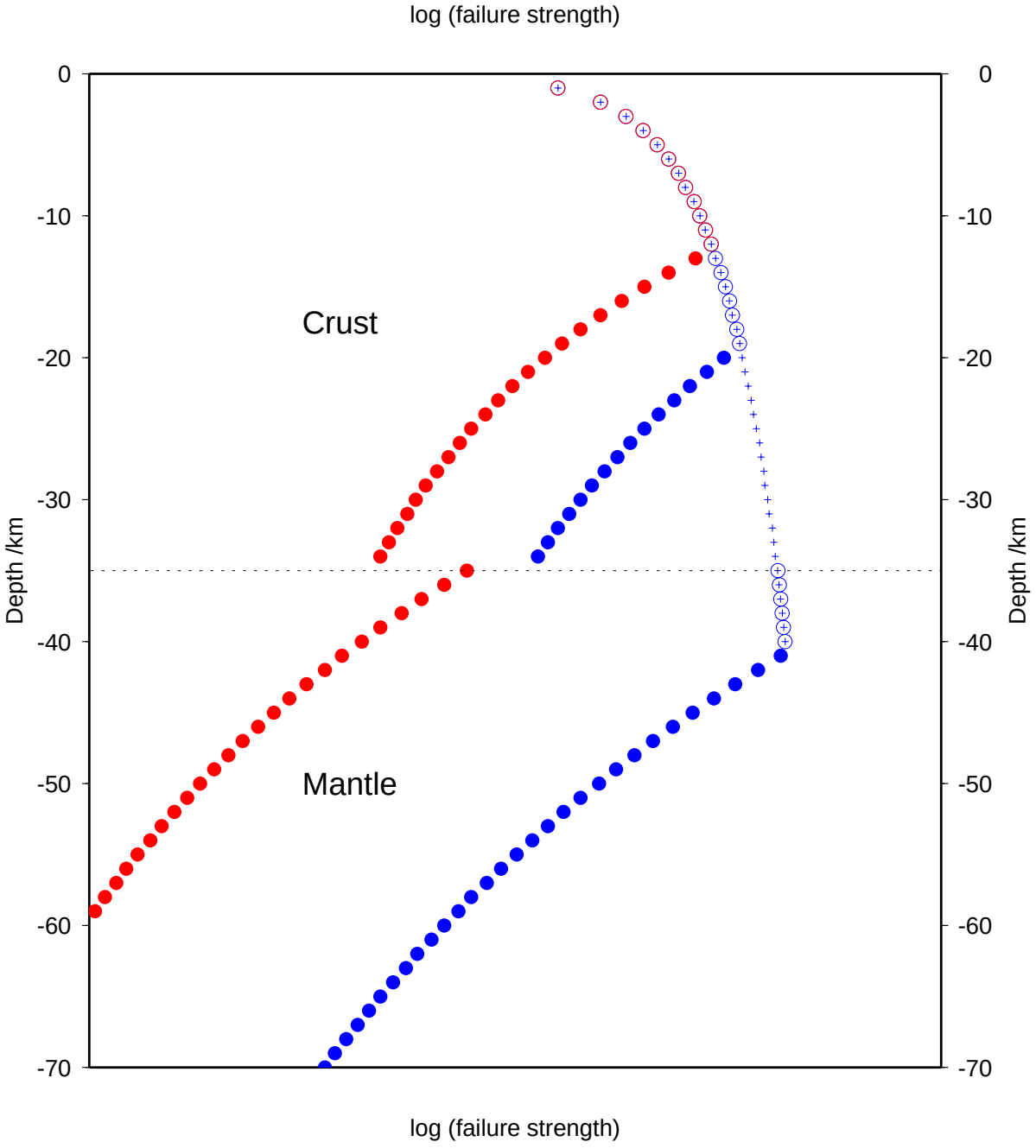

Figure 1.2: Lithospheric failure strength (arbitrary units, shown without scale) as a function of depth, for 35 km–thick quartzose crust (properties from Brace & Kohlstedt (1980)) overlying an olivine mantle (properties from Goetze (1978)) with strain rate 10^{-15} s^{-1} and thermal gradients of 15 K km^{-1} (blue) and 25 K km^{-1} (red). Open circles denote brittle failure (Byerlee's law), closed circles denote plastic failure (power law creep). Olivine-rich oceanic lithosphere (crust and mantle both composed of olivine) with thermal gradient 15 K km^{-1} (blue crosses) follows a similar pattern to continental lithosphere, but without the lower crustal zone of plasticity, so will be stronger overall (after Chen & Molnar (1983)).

1.1.2 Horizontal motions within a deforming zone

Although the crustal deformation within a continental plate boundary zone appears distributed when viewed on the scale of the zone (*e.g.* Jackson & McKenzie, 1988b; England & Jackson, 1989), it is, at the surface at least, localised onto large numbers of faults which are segmented on scales comparable with the thickness of the crust (Jackson & White, 1989). This is a consequence of the brittle nature of the upper crust, but does not imply that the strength of the brittle crust governs the overall large-scale deformation (Lamb, 1994) although it may have an influence at smaller scales (Jackson & White, 1989; Jackson, 1994). The effect of an array of spaced faults is to divide the upper crust into rigid (actually brittle-elastic) blocks that translate with respect to each other and rotate about both horizontal and vertical axes, overlying a lower layer which deforms by distributed plastic means. If this 'floating-block' model is taken too literally there is a space problem as the boundaries of the blocks collide, but in practice the blocks are not quite rigid and the difficulty can be avoided by permitting small amounts of local internal deformation near the boundaries of the blocks.

The motions of individual blocks show a somewhat chaotic pattern and do not necessarily mimic the deformation of the zone as a whole (*e.g.* Dewey & Şengör, 1979). In particular, the rotation rates of blocks will depend on their aspect ratio and their alignment to the steady motion in the lower part of the lithosphere. However, taken over a sufficiently large region (at least several blocks) and a sufficiently long time (several times the typical repeat interval for major earthquakes), the motion may reflect that of the whole zone. Using seismological methods it is not possible to measure the rotations in a global reference frame because the moment tensor only gives information about the symmetric (irrotational) part of the strain tensor (Section 3.4), but by considering a wider region it is possible to determine the relative rotations of sub-regions within the area (Holt *et al.*, 1991; Haines & Holt, 1993). With space geodetic observations, however, it is possible to determine block rotations in a global reference frame, even if the survey area is wholly within the deforming zone.

1.1.3 Vertical motions and tilting

Normal faulting will cause uplift in the foot-wall of the fault and subsidence in the hanging-wall, onto which sediment is generally deposited. Over a large number of earthquakes, finite strain will build up which will rotate both hanging-wall and foot-wall and be observable as tilting of deposited sediments. This may occur *en echelon* for a system of faults (Jackson & White, 1989). Most normal faults have dips of between 25° and 65°, and simple rock mechanics calculations (*e.g.* in Jaeger & Cook, 1976) show that this upper bound is close to the preferred angle of fracture for isotropic rock. If the extension continues for a long time, the finite rotation may cause the fault plane to rotate sufficiently far (shallower than 30°) that it is energetically more favourable to initiate a new fault than to continue slip on the old one (*e.g.* Proffett,

1977). Tilting will only occur as a result of the localisation of strain onto a fault (or array of faults) so this phenomenon will not be observed by networks which measure inter-seismic strains. In addition, the rate of tilt is smaller than the extension rate by an amount dependent on the dip of the faults, so it is not discernible using the GPS height surveying technique (which is only about half as precise as GPS horizontal surveying).

1.1.4 Kinematics *versus* dynamics

A full understanding of continental deformation would include that of the forces causing the deformation (the dynamics). For this it is necessary to relate the kinematic observations (instantaneous crustal deformation, deformation or rotation accumulated through geological time, tilting of sedimentary layers, crustal thickness, *etc.*) to dynamic physical models. As real-life geological materials are far from isotropic and homogeneous, simple models will only be an approximation to the truth, and a knowledge of the geological history of the region in question is necessary because the departures from simplicity are often the result of geological features inherited from earlier episodes of deformation. Also, the evolution of the system through geological time often provides information essential to a full understanding.

Before this dynamic modelling can be achieved, it is first necessary to characterise continental deformation by measuring its effects and by determining which parts of the continental lithosphere are subject to it. The study of individual processes such as crustal-scale faulting and the relationship between instantaneous deformation and that throughout the seismic cycle or over many earthquake repeat cycles is needed to complete this kinematic description. It is this kinematic study that forms part of the purpose of this work.

1.2 The seismic cycle and seismic hazard

An earthquake represents the near-instantaneous localisation of strain that has built up over a period of time in a volume of the Earth's crust, and the interplay between build-up, release and redistribution of strain has been termed the seismic cycle. The interval between earthquakes on an individual fault will not necessarily be constant because the local state of crustal stress will be influenced by movements on nearby faults, nor will successive earthquakes necessarily be of the same magnitude, although there are strong correlations between fault length and individual and total displacements (Walsh & Watterson, 1988). However, several authors have treated the behaviour of a fault averaged over many seismic cycles as though such characteristic earthquakes and repeat times did exist, in order to explain the growth of geological features in terms of repeated seismic activity and associated post-seismic and inter-seismic strains (King *et al.*, 1988; Armijo *et al.*, 1996).

In practice, fault and fault systems will evolve over their lifetime in response to changing tectonic conditions and rotation of the faults as finite strain builds up. The idealised model of repeating similar seismic cycles will only apply to a stable, mature, but not yet dying fault system. Such models have been applied both to lithospheric-scale faulting (Thatcher & Rundle, 1979; Thatcher & Rundle, 1984) and crustal-scale faulting (King *et al.*, 1988; Armijo *et al.*, 1996).

1.2.1 Co-seismic behaviour

During an earthquake, energy is released in the form of seismic waves and their transmission is evidence that the Earth behaves elastically over short time-scales. Crustal earthquakes normally nucleate near the base of the seismogenic layer (*i.e.* in the strongest part of the brittle crust) and the dislocation propagates primarily upwards towards the surface, but also downwards into the lower crust which may normally deform plastically but will fail in a brittle manner when subject to high strain rates. Large normal faults are generally thought to be planar to first order (*e.g.* Ward & Barrientos, 1986; Braunmiller & Nábělek, 1996) and exhibit a horizontal length scale comparable with the thickness of the crust, but some authors (*e.g.* Wernicke, 1981; Rietbrock *et al.*, 1996; Rigo *et al.*, 1996) suggest that faults may flatten out at depth as the stress is refracted by changes in rheology.

The static (permanent) deformation has been modelled using elastic dislocation theory, originally for a point dislocation source within a uniform elastic half-space (Volterra, 1907) but more recently for finite rectangular sources (*e.g.* Steketee, 1958; Okada, 1985) and for sources within horizontally layered media (Singh, 1970).

1.2.2 Post-seismic behaviour

One effect of co-seismic stress drop on the fault plane is to change the state of stress acting on the crust and upper mantle nearby, in a well-defined pattern (*e.g.* King *et al.*, 1994; Das & Scholz, 1981). Although these materials behave elastically over short time-scales, the conditions of high temperature and pressure in the lower crust are such that over longer time periods it will deform plastically (Scholz, 1988). This motion can be modelled as viscous flow in the lower crust (arising from a visco-elastic rheology such as that of a Maxwell solid) or as exponentially-decaying aseismic (but elastic) slip on a downward extension of the fault plane. These two models are hard to distinguish for strike-slip faulting, but in the case of dip-slip faulting they show different deformation characteristics (Thatcher & Rundle, 1984). The visco-elastic model shows more pronounced vertical motion in the hanging-wall at moderate distances (3 – 4 fault lengths) from the fault early on in the cycle, whereas the aseismic slip model shows smoother behaviour in this region which persists later in the cycle. While it would be possible to mimic the behaviour of the visco-elastic model by

introducing several complementing zones of slip in an aseismic slip model, this approach would introduce needless complication.

Providing the lower crust is at least as thick as the upper crust, and the faulting is on a scale comparable with the upper crustal thickness, the visco-elastic calculation is not affected by the presence or absence of a lower purely elastic layer (the upper mantle) that does not allow stresses to relax (Matsu'ura *et al.*, 1981). The effect of stress relaxation at depth is to redistribute stress within the elastic upper crust and thereby cause further surface displacements. Increased stress in the upper crust around the fault plane, whether co-seismic or post-seismic, may be sufficient to trigger significant earthquakes elsewhere within the seismogenic layer (King *et al.*, 1994; Ma & Kusznir, 1995). Stress diffusion calculations (Bott & Dean, 1973) for a purely viscous lower layer indicate that post-seismic stress pulses do not propagate far away from the source in plate tectonic terms but significantly far in terms of plate boundary zones (tens to possibly hundreds of kilometres) in the long term. Studies of post-seismic motion will allow the location and effective viscosity of any viscous or visco-elastic lower layer to be pin-pointed, which will help to answer important questions concerning the degree of coupling between crustal and lithospheric deformation.

1.2.3 Inter-seismic behaviour

Most models of the seismic cycle (*e.g.* Savage, 1983; Thatcher & Rundle, 1984) do not consider the forces driving plate motion but instead model inter-seismic deformation in purely kinematic terms, regarding the system as being driven by displacements applied at the boundaries of the model. Inter-seismic motion on the fault can be modelled as steady slip along the part of the fault plane beneath the upper crustal locked portion (Savage, 1983), with the fault cutting the entire elastic layer which overlies an inviscid fluid (corresponding to a totally relaxed or steady-state visco-elastic medium). While this approach is not satisfactory from the point of view of dynamics, it is kinematically able to describe the situation. Some models (*e.g.* King *et al.*, 1988) include the effects of differential loading caused by sediment erosion and deposition in this part of the cycle.

1.2.4 Geodetic *versus* seismic strain

Kostrov (1974) gives a method for calculating the infinitesimal strain tensor of a region by summing the moment tensors of earthquakes in that region. Using this result, it is possible to compare the strain measured by geodetic surveying over a period of time (Section 1.3) with the strain released by earthquakes in the region of the survey in the same period, for the purpose of estimating the risk of future earthquakes (Molnar & Chen, 1982). As the repeat time of earthquakes on a given fault is large (hundreds of years) and may be variable, this provides a superior estimate to one based only on the past location of earthquakes,

particularly when historical earthquakes are poorly documented. However due consideration must be given to likely errors in the moment tensors of historical earthquakes. Also, smaller earthquakes, which will not be globally recorded, may add up to 50% extra seismic moment (Ambraseys & Jackson, 1990).

It is necessary to assume a vertical dimension for the region that is affected by earthquake strain. Since most continental earthquakes are generated within the elastic upper crust, it is assumed that lower layers are able to relax stresses and so the thickness of the seismogenic layer is taken as the depth extent of the region. Co-seismic elastic deformation will in fact propagate below this level because of the high strain rates involved, so this value should be taken as a minimum bound on depth.

Even if all the possible errors in the seismology can be eliminated, the geodetic strain may not equal the seismic strain. Proximity of geodetic monuments to large earthquakes may affect their position because of local co-seismic motions which are not accounted for by the Kostrov model, which deals with the relaxed bulk strain only (*i.e.* it assumes that all stresses are relaxed) and not the immediate co-seismic strain local to the fault. Monuments further away from earthquake locations may also behave anomalously due to post-seismic displacement, although this effect is less likely to be observed if the characteristic time of post-seismic relaxation is short, because post-seismic relaxation will then only take up a small fraction of the seismic cycle. Only if these two effects can be removed is it possible to compare the inter-seismic strain, which represents the long-term build-up of stress leading to an earthquake (unless stress is relaxed anelastically by aseismic creep), with that which has been released in earthquakes already. Horizontal elements of the strain tensor only are considered, because errors in geodetic height measurements at the surface are currently too large to be of any use so the vertical components of strain cannot be estimated directly (although it would be possible to derive the non-shear component if incompressibility of the crust is assumed). If the geodetic strain so measured is greater than the seismic strain, it indicates that there is increased risk of medium-term seismic hazard in the region.

1.3 Use of geodetic surveying in geophysics

Geodetic positioning techniques have been used in geophysics both to assist in the measurement of the Earth's gravity field and to measure movements or strains of the Earth's surface *per se*. In this thesis I will only deal with the latter use. The first published application of a geodetic technique to measurements relating to crustal movements is that of Hayford & Baldwin (1907), who compared triangulation surveys made across the San Andreas Fault (California) before and after the 1906 San Francisco earthquake. Since then, geodetic methods have been applied with increasing accuracy to the problems of crustal deformation. Lambeck (1988) provides a comprehensive review of geodetic methods in geophysics.

1.3.1 Geodetic triangulation and trilateration

Classical triangulation makes use of theodolite measurements of angles between monuments (*e.g.* concrete pillars) to determine the positions of these monuments. Normally, an excess of observations is made and the resulting overdetermined system of equations is reduced to give positions using an iterative least squares method. Horizontal and vertical positioning are treated as separate issues in order to simplify the calculations, although modern computer programs are able to deal with the combined problem.

While it is in principle possible to read theodolite angles to within $0.1''$, several sources of error make this level of precision impossible in practice for inter-station observations. The most significant of these is refraction of light caused by variations in the temperature and humidity of the troposphere between theodolite and target, which can induce errors of up to $1''$ in mountainous areas where vertical thermal gradients are high. Surveyors have attempted to get around this problem by taking measurements at several times of day, or at night when temperatures are more uniform, but the best accuracies attained are still of order $0.3 - .5''$ or just under 0.5 p.p.m. for first-order networks with a station spacing of 30 – 40 km (Bomford, 1980).

For surveys larger than a few kilometres in extent, the change in direction of the vertical (the axis about which the theodolite measures angles) caused by curvature of the Earth (strictly speaking, curvature of the geoid or equipotential surface) becomes significant and the displacement computations must be carried out on a curved reference surface. The simplest global approximation to the geoid is an ellipsoid of revolution and so this is used as the basis for computation. However local deflections of the vertical caused by nearby density contrasts in the crust will cause the geoid to be irregular and thus introduce a second source of bias into the positioning which may need to be eliminated. An alternative approach for deformation studies (Frank, 1966; Bibby, 1982) that avoids this pitfall is to proceed directly from the changes in angle measurements between two epochs to estimates of shear strain of the network, but this is only applicable when triangulation is used at both epochs.

Early triangulation surveys relied on astronomical observations for absolute network position and orientation control, and hard-to-perform distance measurements using invar tape to establish scale control, the latter method being particularly inaccurate (errors of up to 10 p.p.m., and only applicable to distances of a few kilometres or less). With the advent of Electronic Distance Measurement (EDM) techniques since the 1950s, first with microwave and later with visible light radiation, it has become possible to measure distances with precisions of up to 0.1 p.p.m. if observations are made at two frequencies so that the dispersive property of atmospheric refraction at these wavelengths can be used to reduce bias, although precisions of 1 p.p.m. are achievable using only one frequency if meteorological observations are used to apply a correction factor.

1.3.2 Space geodetic techniques

The immediate advantage of all space-based methods over terrestrial surveying is that there is no longer a requirement for station intervisibility, and widely-separated points can therefore be co-located in all weather conditions. Also, much of the data-gathering process is automated which reduces the chance of blunders and enables more readings to be easily taken to improve accuracy further.

Early satellite geodetic techniques such as satellite photography using camera telescopes and TRANSIT Doppler tracking resulted in ground positions accurate to 10 – 20 m, insufficient for plate tectonic studies. The development of Satellite Laser Ranging (SLR) methods (and analogous Lunar Laser Ranging methods) in the late 1960s, based on sub-nanosecond interval timing of the two-way travel time of laser pulses transmitted from a ground station and reflected back from corner cubes on Earth-orbiting satellites, permitted an improvement in accuracy of two orders of magnitude. Because the satellites are orbiting the Earth, positions derived from SLR will be relative to the geocentre (centre of mass of the Earth). Precise absolute time control (to within microseconds) is necessary to position the satellite within its orbit, which for low-Earth-orbiting satellites is highly sensitive to the Earth's gravity field. However the accuracy of relative positioning of stations is largely independent of the baseline length between them, which makes SLR an attractive method for global and large-scale regional deformation studies despite the cost and bulk of the apparatus.

Another technique that blossomed in the late 1960s (also due to the improvements in absolute time control at this epoch) is Very Long Baseline Interferometry (VLBI). VLBI is an interferometric technique in which radio signals emitted from distant stars are recorded and time-tagged (to within 0.1 ns) separately by different radio telescopes around the globe (Figure 1.3). Because the stars are so distant (many of them are extragalactic), they do not exhibit perceptible motions with regard to each other and so may be regarded as providing an inertial frame of reference, unlike Earth-orbiting satellite methods in which the dynamics of the satellite must be considered. The delay in reception of the signal between two antennae yields estimates to centimetric precision of the baseline component along the direction of signal propagation, and as the Earth rotates, knowledge of the whole baseline vector and also of Earth orientation and rotation rate is gained. Unfortunately VLBI apparatus is extremely bulky and expensive and so is only used for global studies (plate motion measurements and definition of the global reference frame).

The most recent space geodetic positioning method that has applications in deformation studies is the Global Positioning System (GPS) of satellite navigation, first established in the 1980s but reaching full operational capability in early 1994. Details of the system are explained more fully in Chapter 2, but in essence the method used for geodetic work is an interferometric treatment of the phase observables of the L-band radio signals emitted by the satellites and observed at two or more ground stations. Because the signals are much stronger than those from radio stars, antennae can be very small, and because several

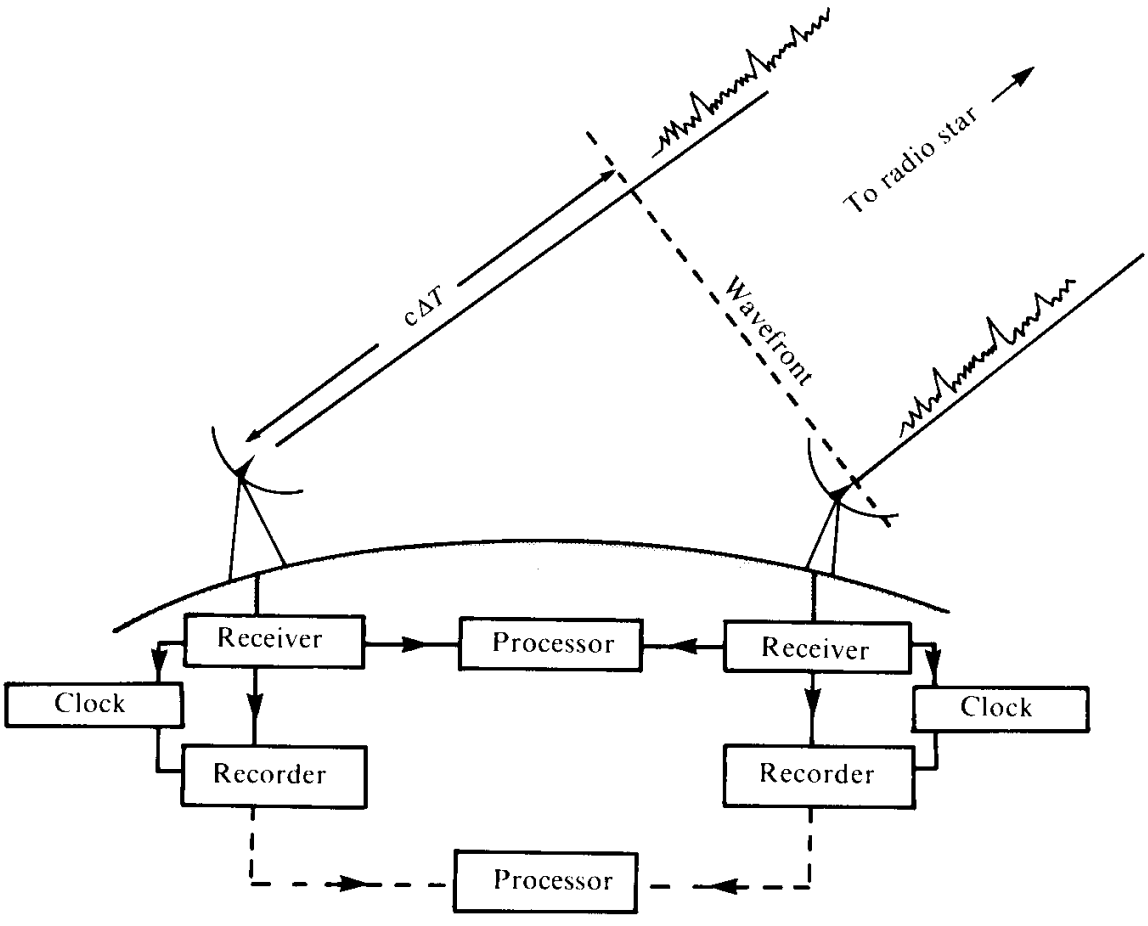

Figure 1.3: Cartoon of VLBI system configuration, showing only two stations, from Lambeck, 1988. $c\Delta T$ is the along-path component of baseline length that is measured instantaneously, and its variation with time allows the whole baseline vector to be reconstructed.

satellites are observed simultaneously, there is no requirement for absolute clock accuracy, unlike VLBI (although good short-term frequency stability is necessary), so modern systems are relatively cheap and lightweight. Observation times required for geodetic positioning are shorter than for SLR and VLBI, and so GPS is particularly suited to deployment at a large number of sites in order to carry out local or regional studies on an epoch basis. However, the satellite orbits, while more distant than those of SLR satellites and thus less sensitive to higher-order effects of the Earth's gravity field, are unable to provide as stable an inertial frame of reference as VLBI. Despite this, the greater number of global GPS sites and their probable greater stability (radio telescopes are large engineering structures; GPS antennae are small and easily anchored), GPS may provide a more stable terrestrial reference frame. A global network of permanently-observing GPS stations, many of whose positions and relative motions have also been determined by VLBI (and SLR), is used to track the satellites and this provides a link between the reference frames.

While not strictly a geodetic positioning method, the technique of differential Interferometric Synthetic Aperture Radar (InSAR or just SAR) can also be used for the study of crustal deformation. As the name suggests, SAR is also an interferometric satellite-based technique, but unlike the methods outlined above, both the transmitter and the antenna are located on the satellite and the signals reflected from the Earth's surface are recorded. The change in phase of the reflected signal between two passes of the satellite contains information about the surface deformation in the ground–satellite direction (near-vertical), and also information about the topography because the satellite positions are not identical from pass to pass so there is a stereoscopic effect, although the latter can be corrected for either by double-differencing three or more images or by using an explicit topographic elevation model. Changes in vegetation cover and groundwater affect reflection of the signal and cause spurious phase changes and loss of coherence between successive satellite images (currently 35 days apart for the ERS-1 and ERS-2 satellites), which increase with time and thus make SAR hard to use for long-term studies. Also, the size of a SAR image (approximately 100 km in width) is too small to allow SAR to be used for global or large regional studies. Offset against this disadvantage is the fact that there is no need for any ground-based monumentation, and measurements are made world-wide so deformation can be measured even where there are no existing geodetic networks.

All of these space-based methods rely on signals that must be propagated through the Earth's atmosphere and are therefore subject to refraction and delay by the ionosphere and troposphere in analogous ways to EDM and triangulation. However, the use of dual-frequency measurements (or multiple-frequency for VLBI) means that it is largely possible to remove ionospheric sources of error. To remove tropospheric error, it is much easier to model the effects of a thin vertical profile of the atmosphere, as required by space geodesy, than those of arbitrary horizontal sections, as required by classical surveying techniques. The use of differencing techniques (Subsection 2.1.5) further limits atmospheric errors.

1.3.3 Worldwide application to regional strain and plate motions

Purely terrestrial surveying techniques have been used to study long-term strain in a number of regions, *e.g.* New Zealand (Bibby, 1982; Walcott, 1984) and the western United States (Prescott *et al.*, 1979; Matsu'ura *et al.*, 1986; *etc.*). Even for such regions of high deformation rate, the inaccuracy of terrestrial techniques over large distances limits such techniques to the measurement of strain over long periods (several years to decades) and precludes the possibility of global plate motion measurements. Also, the local nature of the reference frame means that only relative rotations (about vertical axes) of crustal units can be measured, unless the survey network extends beyond the boundary of the zone of deformation.

In contrast, VLBI and SLR methods have been used effectively for global measurements of plate tectonic motion (*e.g.* Argus & Gordon, 1990; Smith *et al.*, 1990; Robbins *et al.*, 1993) which match long-term geological estimates (DeMets *et al.*, 1994) very well. Global GPS networks have recently also been used to measure plate motions (*e.g.* Blewitt *et al.*, 1992).

GPS and SLR networks have both been used for regional studies, *e.g.* California (Dong & Bock, 1989; Larsen & Reilinger, 1992 *etc.*) and the Caribbean (Dixon, 1993). The reoccupation of former terrestrial geodetic networks using GPS has also proved fruitful in a number of regions (*e.g.* Italy, Hunstad & England, manuscript in preparation; New Zealand, Bourne, 1996). With this approach, resurveys can be performed far faster than the original survey, and as the positioning errors are far smaller than for a triangulation – triangulation comparison, a shorter time is needed for a detectable tectonic signal to accumulate (although not as short as for pure GPS experiments).

1.3.4 Worldwide applications to local deformation

Since the early observations in California mentioned above, geodetic surveying has been used to measure local displacements caused by several continental earthquakes. Geodetic levelling data have been used to constrain mechanisms of normal faulting in the Basin and Range of the United States, for the 1954 M_w 7.2 Fairview Peak, Nevada (Savage & Hastie, 1969), 1959 M_w 7.3 Hebgen Lake, Montana (Savage & Hastie, 1966) and 1983 M_w 7.0 Borah Peak, Idaho (Stein & Barrientos, 1985; Ward & Barrientos, 1986) earthquakes. Although the linear distribution of levelling data meant that only a few fault parameters could be estimated, these studies have generally yielded geodetic estimates of moment that were higher than those from seismological studies.

Segall & Matthews (1988) included measurements from a trilateration network bounding the 1966 M_s 5.5 Parkfield, California earthquake in their paper on inversion mechanisms, but they do not publish results of the source parameters. Segall & Harris (1987) use a different inversion scheme for the same dataset and derive a geodetic moment which may be as much as three times the seismic moment.

The 1989 M_s 7.1 Loma Prieta, California and 1991 M_s 7.6 Valle de la Estrella, Costa Rica thrusting earthquakes (Williams *et al.*, 1993; Lundgren *et al.*, 1993) were both studied using purely GPS measurements. In the former case, all the source parameters were estimated from the geodetic displacements, although only eight stations were observed, but answers agreed reasonably with EDM, levelling and combined space and terrestrial geodetic studies (Lisowski *et al.*, 1990; Marshall *et al.*, 1991; Snay *et al.*, 1991). In the latter case, only two sites were sufficiently close to the fault plane to be of use in determining a mechanism, so only the fault plane dimensions and slip were estimated, with the slip implying a scalar moment considerably higher than the CMT and other seismological solutions.

More recently, Massonnet *et al.* (1993; 1996) have used differential Synthetic Aperture Radar (SAR) interferometry to map the co-seismic and post-seismic displacements of the 1992 Landers, California (M_w 7.3) strike-slip earthquake. While SAR only provides satellite line-of-sight displacements, and considerable technical difficulties remain, it is the only geodetic method that provides complete spatial coverage of the earthquake zone and does not require a pre-existing geodetic network. Blewitt *et al.* (1993) have also observed this earthquake using far-field continuous GPS measurements, and obtain a scalar moment consitent with geological and seismological observations.

Thatcher & Rundle (1984) used geodetic levelling data from the 1944 and 1946 Nankaido, Japan (M_s 8.1, 8.2) thrusting earthquakes in their study of post-seismic behaviour in underthrust zones, and suggested that characteristic relaxation times may be of order 40 years. Reilinger (1986) suggested a decay time of 10 years based on a multiple levelling study of the 1959 Hebgen Lake, Montana (M_s 7.5) normal earthquake. However, GPS studies of the Landers earthquake (Yu *et al.*, 1996) show that short-term relaxation over tens of days is also exhibited and may be modelled as visco-elastic behaviour. Repeated GPS measurements taken after the Loma Prieta earthquake (Savage *et al.*, 1994) suggest a relaxation time of about 1.4 years, without presupposing any geophysical model (although the authors later interpret the surface displacements as the result of aseismic buried slip). A combined GPS–terrestrial surveying study of post-rifting relaxation after the 1975 Krafla, Iceland dyke intrusion events (Foulger *et al.*, 1992) has shown relaxation to take place over some tens of years.

1.3.5 Applications in the Aegean region

Both terrestrial and space geodetic techniques have been used to study the Aegean region (Figure 1.4) in recent years. Greece is fortunate in having a history of geodetic surveying going back over 100 years, and the 1^{st}-order triangulation network established in 1889 – 1900 (Figure 1.5) has been reoccupied with GPS (Billiris *et al.*, 1991; Davies *et al.*, 1996; Davies, 1996) to yield regional estimates of long-term strain, although the scale control from the early survey is sufficiently unreliable that it has proved better to make scale assumptions based on the relative lack of seismic activity in the Peleponnessos. Stiros (1993)

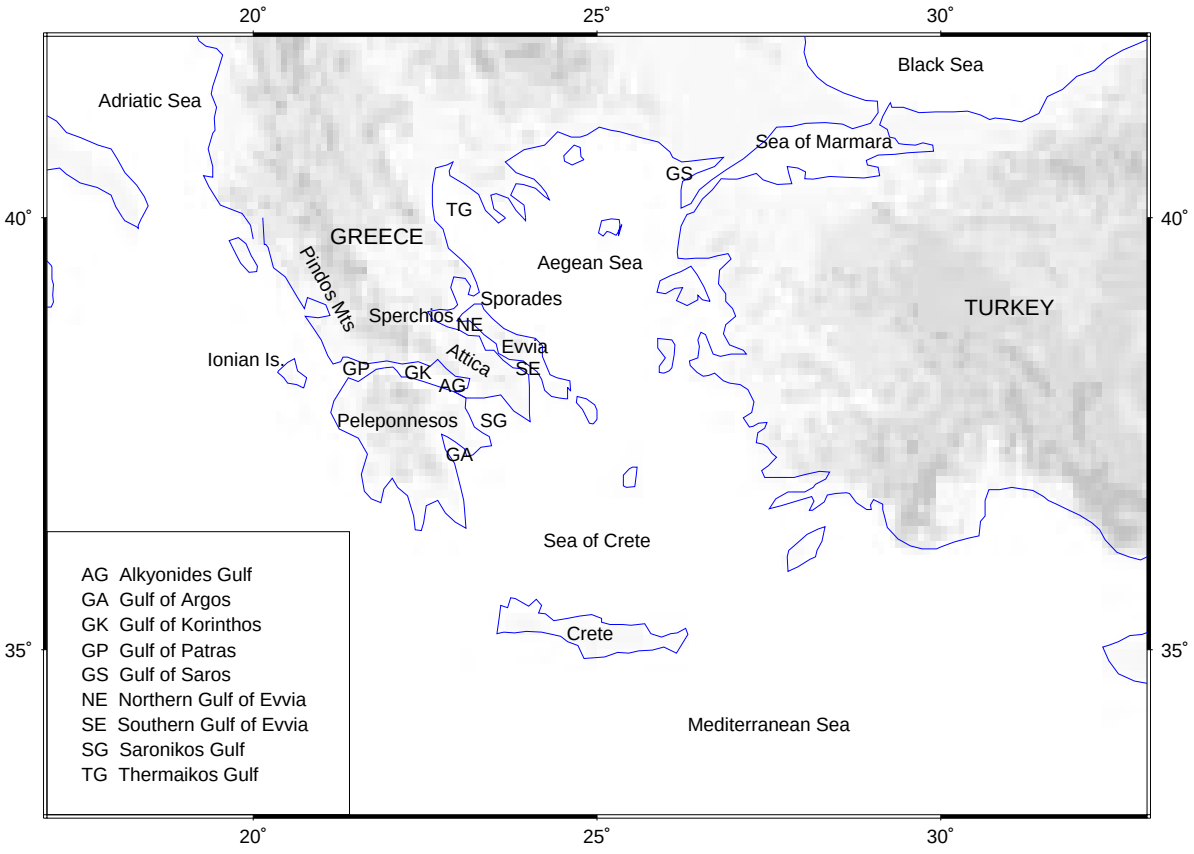

Figure 1.4: Geographical locations within the Aegean region. Topographic shading increases at 500 m vertical intervals.

has analysed 1^{st}-order terrestrial measurements made in 1889 – 1905, 1927 – 1930 and 1950 – 1970 and concludes that there has been a marked (although marginally statistically significant given the imprecision of triangulation data) change between the strain rate before and after 1930. One aspect of this study will be to compare the short-term deformation of central Greece with the hundred-year deformation observed by Davies (1996).

SLR measurements have been made since 1986 at six sites (Figure 1.5) within the Aegean region as part of the WEGENER/MEDLAS project (Noomen *et al.*, 1993; Robbins *et al.*, 1993). The velocity results have been compared with seismic strain rates by Jackson *et al.* (1994), who concluded that while strain orientations were in agreement, geodetic strain rates were significantly higher than seismic strain rates. The SLR and hundred-year triangulation–GPS results have also been studied by Le Pichon *et al.* (1995) who deduce the existence of various deforming and rigid zones. Recent GPS experiments in the Aegean Sea have been conducted since 1988 by Kastens *et al.* (1989), but conclusive results are not yet available.

Local, high-resolution geodetic studies have so far been limited to the GPS measurements of displacement

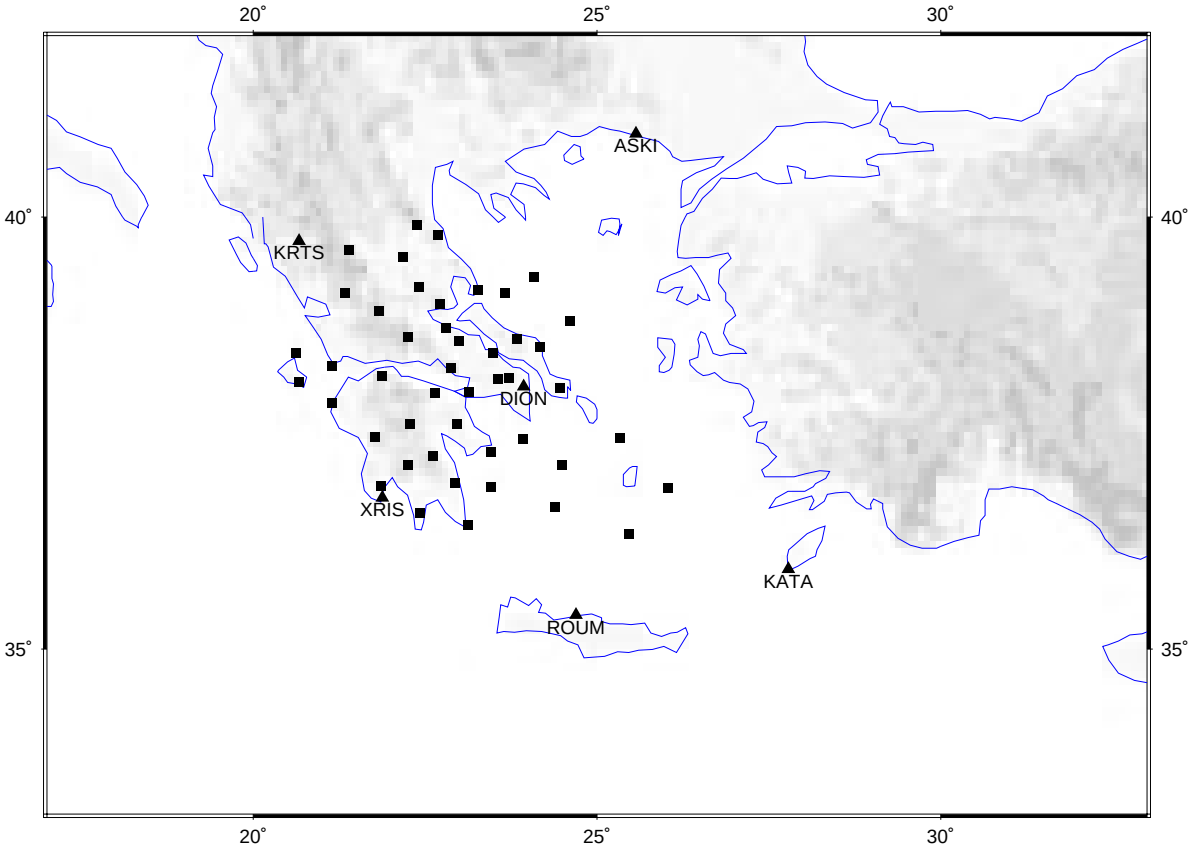

Figure 1.5: Reoccupied geodetic networks in Greece. Sites from the 1890s 1^{st} order triangulation occupied in 1889 – 1905 and again in 1988 / 1992 are shown as squares, and SLR sites occupied since 1986 are shown as named triangles. Local networks (GPS – GPS or triangulation – GPS) also exist around the Gulf of Argos and the Ionian Islands. Topographic shading increases at 500 m vertical intervals.

in the Western Hellenic Arc (Kahle *et al.*, 1995; Kahle *et al.*, 1996), and preliminary results from GPS measurements across the Gulf of Korinthos (Agatza-Balodimou *et al.*, 1995). In addition, Curtis (1994) used triangulation data from local surveys in the Alkyonides Gulf to estimate the source parameters of the 1981 Platea–Kaparelli (M_s 6.2) earthquake (see Subsection 3.5.4).

Combined GPS and SLR studies in Turkey (Oral *et al.*, 1994; Smith *et al.*, 1994; Lenk, 1995) indicate that much of the Anatolian plate is rotating as a rigid block about a pole of rotation situated in northeast Egypt. Oral *et al.* (1994) conclude that this block also includes the southern Aegean Sea (at variance with their data for SLR sites in Greece!), whereas Le Pichon *et al.* (1995) postulate a zone of southward extension in the southern Aegean in order to explain the deviation of the SLR sites at Kattavia, Roumelli and Xrisokellaria from their predicted rotational displacements.

1.4 Tectonic setting of the Aegean

The Aegean region, here taken to include mainland Greece and western Turkey as well as the area of the Aegean Sea itself, is one of the most rapidly deforming continental regions of the globe (Taymaz *et al.*, 1991), as demonstrated by recent seismicity (Figure 1.6). It is also an area of considerable complexity (Dewey & Şengör, 1979). Westward motion of thickened Anatolian continental crust is accommodated on the right-lateral strike-slip North Anatolian Fault (Figure 1.7), but this feature splays out into several faults west of 31°E and finally terminates west of the Gulf of Saros. To the south and west, the region is bounded by the Hellenic trench under which Mediterranean sea floor is being subducted. In the west near the Ionian islands, subduction is replaced by right-lateral strike-slip motion as the boundary swings to a north – south direction, in line with the relative plate motion. Within the northern Aegean Sea, northeast – southwest right-lateral strike-slip motion dominates, but on mainland Greece to the west, WNW–ESE-striking extensional faulting is the norm, and the southern Aegean Sea and Sea of Crete are seismically relatively quiet (Taymaz *et al.*, 1991). It has been suggested (Jackson & McKenzie, 1988a) that this is because deformation in this area is occurring aseismically due to the presence of soft sediments and evaporites in the crust. As yet, geodetic measurements in this region have not been sufficiently detailed to verify this.

1.4.1 Tectonic history

During late Mesozoic and early Tertiary time Greece underwent a number of collisional events that shortened the crust in a northeasterly direction, imparting a pronounced northwest – southeast structural grain to the geology of the area. It is possible that this structural grain has influenced the response of the crust during more recent tectonic episodes by favouring deformation along pre-existing lines of weakness (Jack-

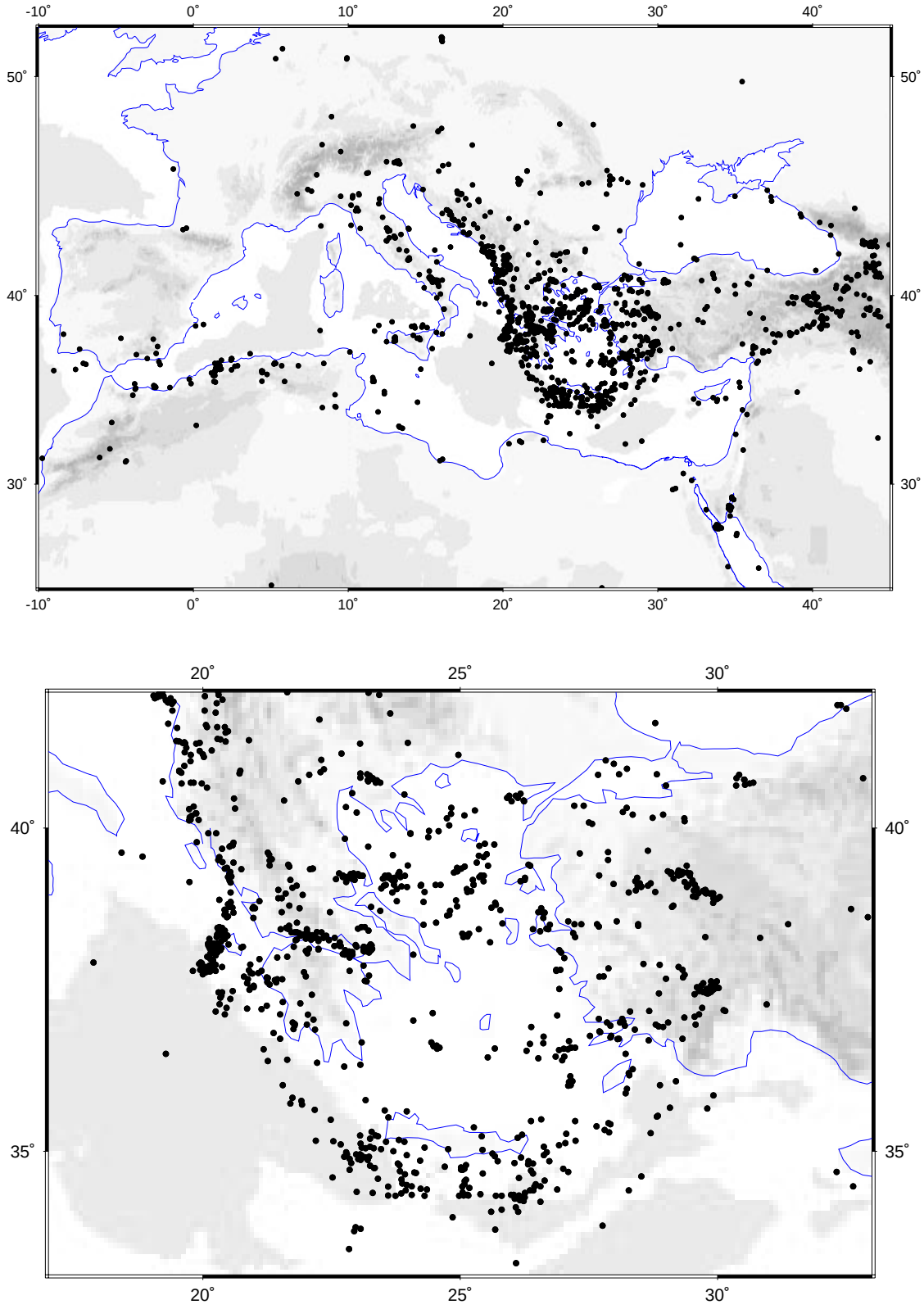

Figure 1.6: Locations of shallow earthquakes (above 40 km depth) of magnitude $M_b \geq 4.5$ from the ISC catalogue 1964 – 1993 for the Mediterranean (above) and Aegean (below) regions. Topographic shading increases at 500 m vertical intervals, and in addition seafloor below 2500 m is shaded.

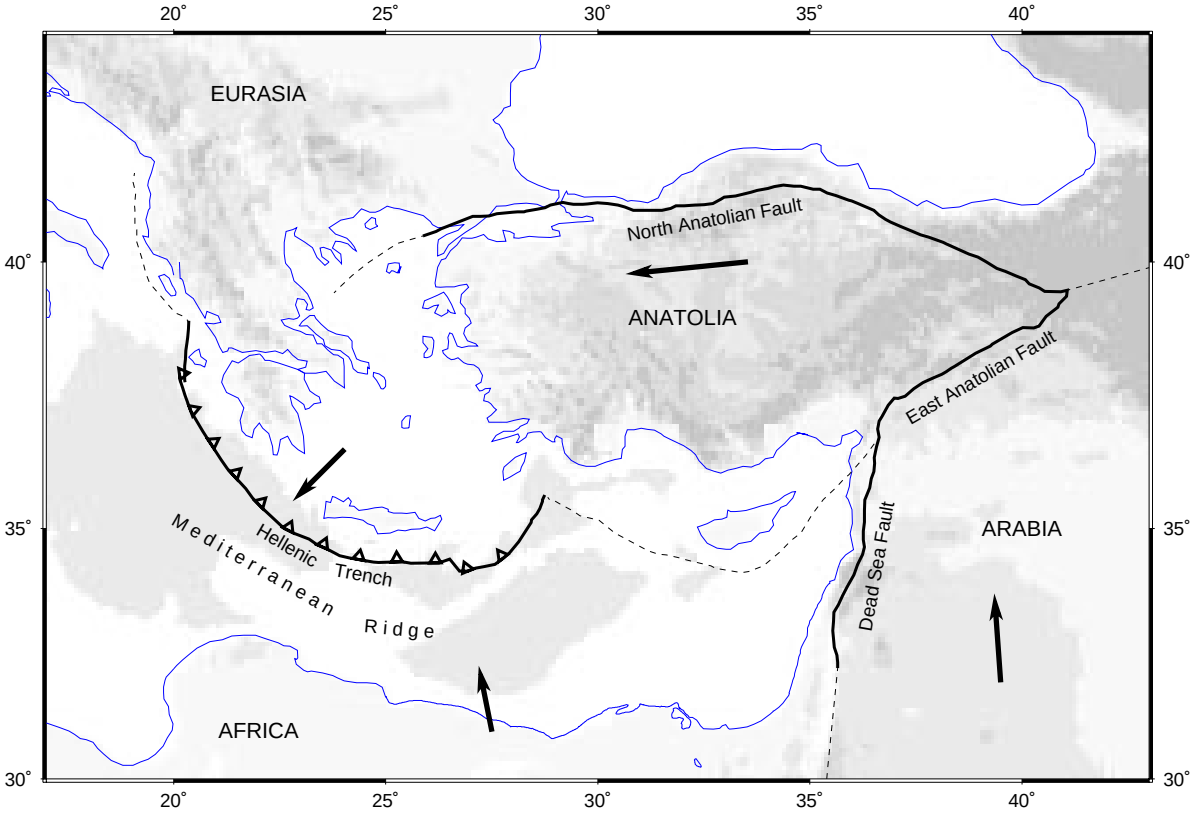

Figure 1.7: Sketch map of large-scale neotectonic features in the eastern Mediterranean region. Underthrusting boundaries are shown as solid lines with teeth on the overriding side, transform boundaries are shown as solid lines, and less active boundaries are shown as dotted lines. Arrows (not to scale) indicate the approximate tectonic motions relative to 'stable' Eurasia. For clarity, only the largest-scale features are shown; further detail is given in Figure 1.12.

son, 1994). The associated thickening of the relatively strong crystalline part of the crust (as opposed to just the softer surficial sediments) has served to thicken the seismogenic layer, but also to carry quartzose crustal materials to depths below which creep is likely to occur. The thickened crust is gravitationally unstable and therefore liable to spread, and this too may have promoted northeast – southwest extensional tectonics in more recent times.

The presence of metamorphic core complexes dated to early Miocene time (25 – 23 Ma) indicates that the northern Aegean region was already undergoing extension at this point in its history (*e.g.* Sokoutis *et al.*, 1993). The driving force of this deformation is presumed to be gravitational spreading of the thick Anatolian crust towards the less thickened Aegean crust, and also retreat of the subducting African slab encouraging southward migration of the Hellenic trench (Dewey & Şengör, 1979; Jackson, 1994; Le Pichon *et al.*, 1995) which forms a boundary of low resistance to spreading in the south.

The North Anatolian Fault began strike-slip motion in late Miocene time (13 – 10 Ma), although motion may not have propagated as far west as the Aegean Sea until around 5 Ma before present (Armijo *et al.*, 1996). The effect of this motion was to increase the rate of extension in the Aegean and also the rates of rotation of crustal units (Kissel & Laj, 1988) and hence increase the curvature of the Hellenic trench and change the direction of shortening across it from north – south to northeast – southwest. Since its inception, roughly 85 km of motion has occurred across the main North Anatolian Fault, and roughly 60 km and 20 km on its northern and southern main branches west of the splay in the Sea of Marmara (Armijo *et al.*, 1996). It is a matter of contention as to whether recent strike-slip motion is partitioned entirely between these branches or whether it is distributed throughout western Anatolia and the Aegean Sea by means of anti-clockwise rotation of the normal fault systems there (Taymaz *et al.*, 1991; Armijo *et al.*, 1996).

West of Greece, the rotation of the Adriatic microplate as it became detached from Africa since the late Miocene, coupled with the westward motion of Anatolia, caused the subduction to become transpression and later strike-slip motion (Dewey & Şengör, 1979). By late Pliocene time, volcanism in this region had ceased, although volcanism in the Cyclades and southwest Turkey continues. The Adriatic microplate continues to move northward with very little internal deformation (Anderson & Jackson, 1987).

Figures 1.8 and 1.9 show a summary of the tectonic evolution of the eastern Mediterranean as a whole since middle Miocene time after Dewey & Şengőr (1979), and Figures 1.10 and 1.11 show the tectonic evolution of the Aegean region in cartoon form after Armijo *et al.* (1996).

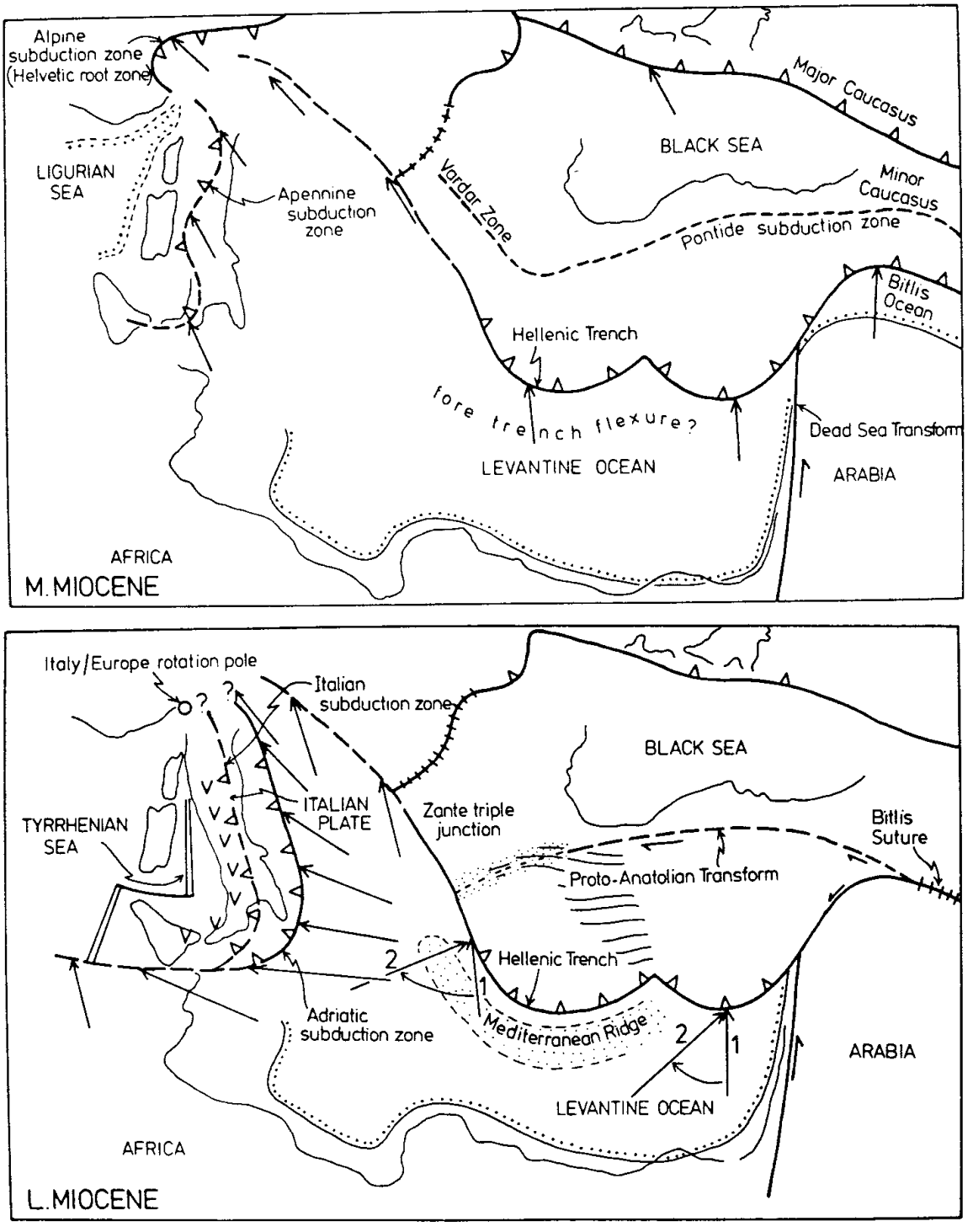

Figure 1.8: Tectonic evolution of the eastern Mediterranean (1) middle to late Miocene, from Dewey & Şengőr (1979). Westerly motion of Anatolia begins as the Dead Sea transform activates and subduction in northern Anatolia gives way to right-lateral strike-slip movement. Extension later begins to develop in western Anatolia. The Aegean Sea may have been land at this stage, with the ensuing crustal thinning leading to inundation.

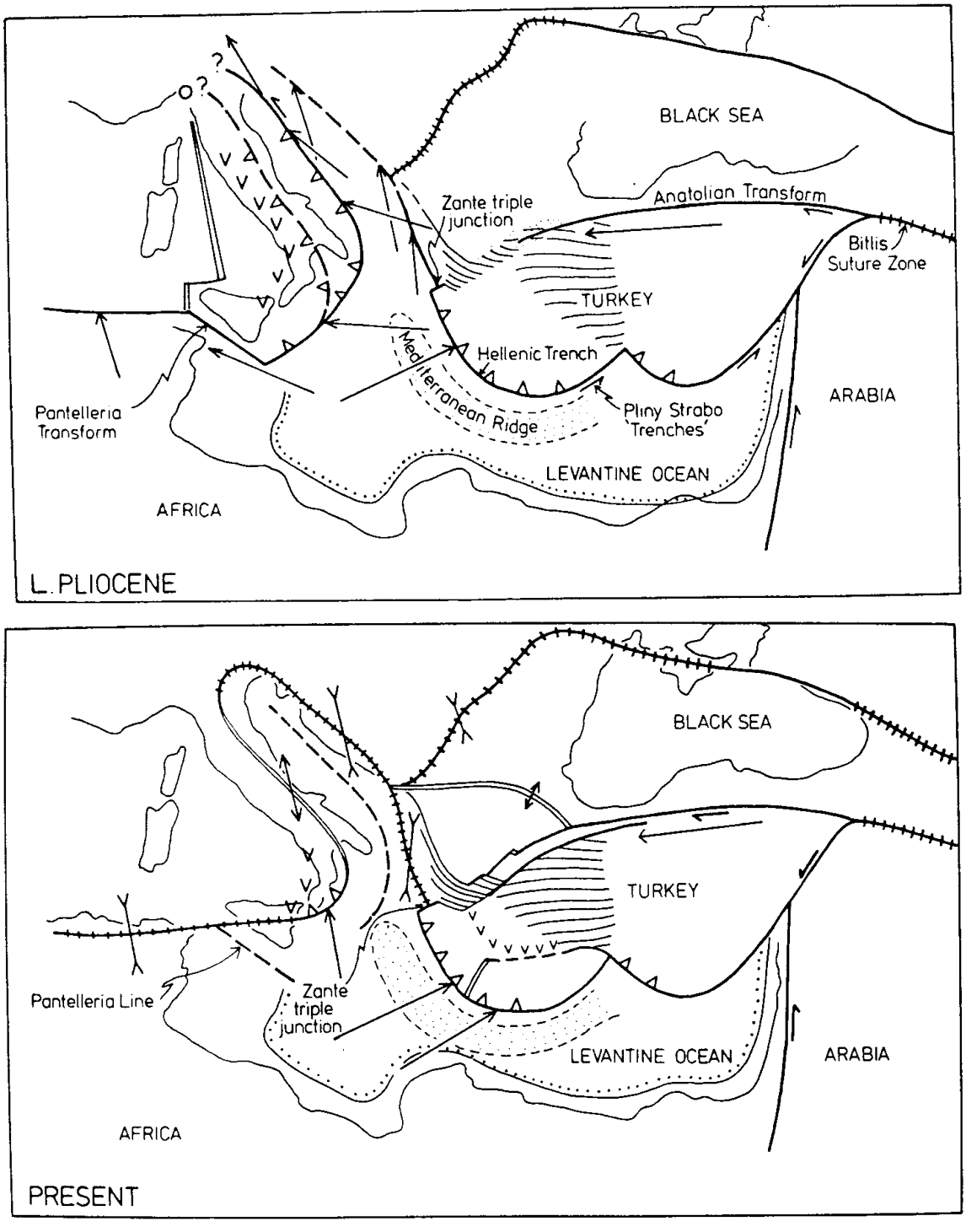

Figure 1.9: Tectonic evolution of the eastern Mediterranean (2) late Pliocene to present day, from Dewey & Şengőr (1979). The Anatolian transform faults are now established, and the Adriatic microplate ceases southwest-directed thrusting as it rotates to its present-day position. Volcanism develops in the southern Aegean Sea.

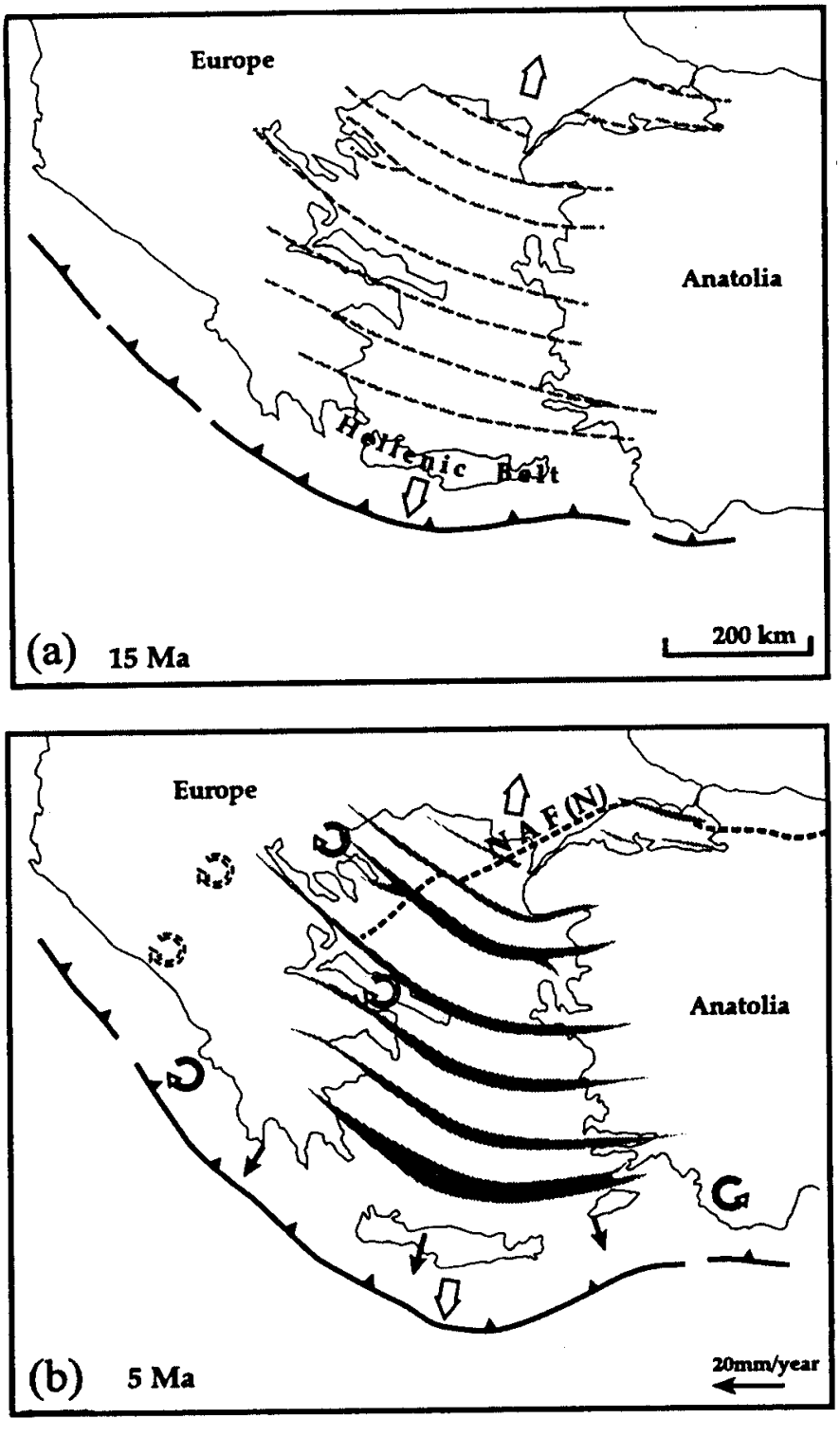

Figure 1.10: Tectonic evolution of the Aegean region (1) 15 Ma – 5 Ma, from Armijo *et al.* (1996). NAF = North Anatolian Fault, dashed lines and grey-shaded areas represent developing grabens, dotted lines represent strike-slip shear zones. Thickened crust north of the Hellenic belt begins to extend north – south as the Hellenic trench provides weak resistance to the south. Rotations of the grabens caused by extension in the centre outpacing that to the east and west causes curvature of the trench. Since about 10 Ma, the NAF propagates westward toward the northern Aegean Sea.

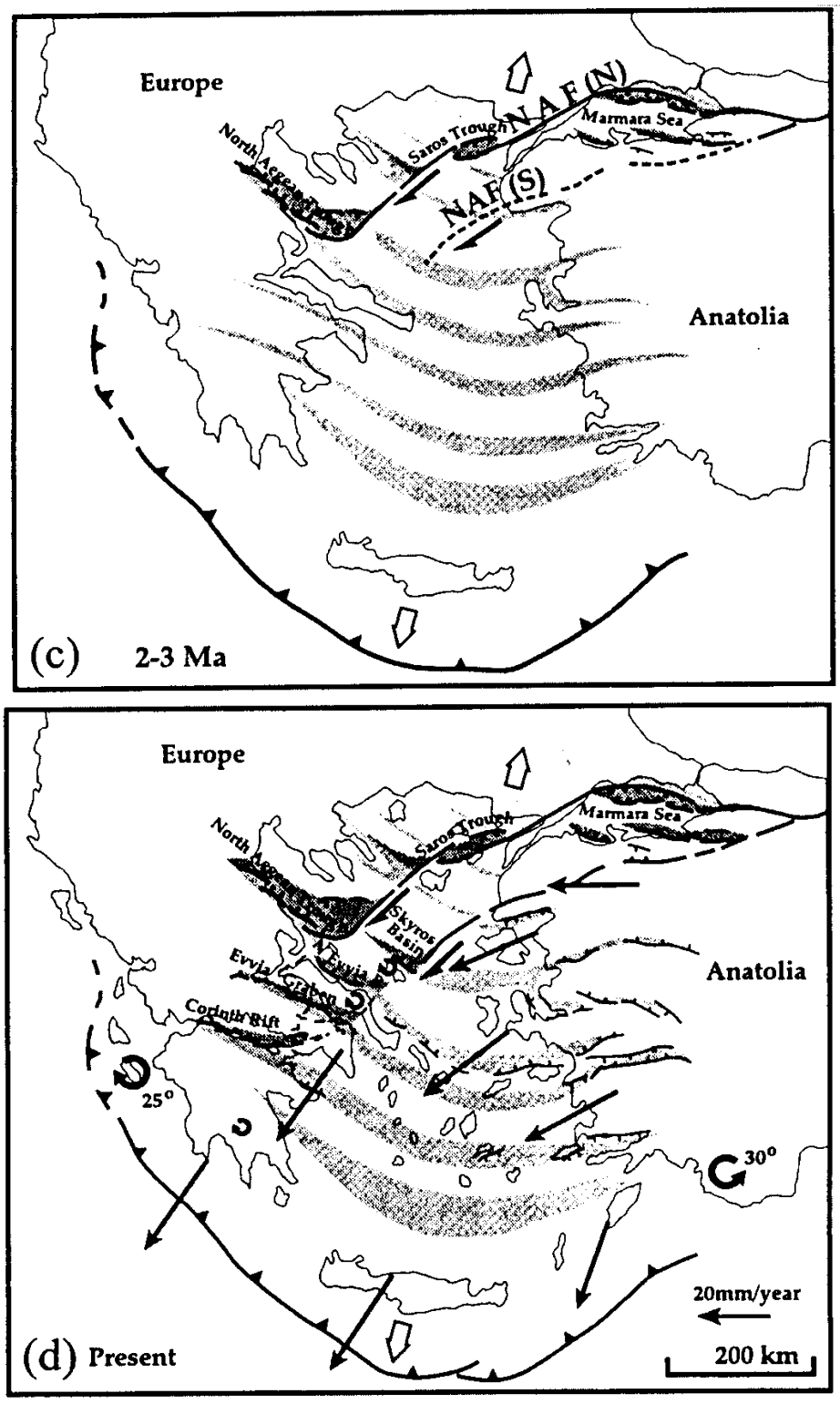

Figure 1.11: Tectonic evolution of the Aegean region (2) 3 Ma – present, from Armijo *et al.* (1996). NAF = North Anatolian Fault, grey-shaded areas represent developing grabens. Further westward propagation of the NAF encourages faster extension in nearby basins, and introduces asymmetry to the system, and splaying of the NAF becomes more pronounced.

1.4.2 Neotectonics in mainland Greece

The Gulfs of Korinthos and Evvia are characterised by recent and historical normal faulting (Ambraseys & Jackson, 1990; Taymaz *et al.*, 1991) and associated north – south to northeast – southwest extension, whereas the southern Peloponnesos is extending east – west at a much slower rate (Figure 1.12). In northern Greece, the level of seismicity is also much lower (Papazachos, 1990), but the direction of extension is again north – south, although the faults are aligned northwest – southeast. To achieve this direction of extension, the fault blocks must be rotating anti-clockwise. Faulting is generally segmented on a scale of 15 – 30 km, and drainage is strongly controlled by fault segmentation and fault block tilting (Roberts & Jackson, 1991).

Quaternary fault history in the Gulf of Korinthos is revealed by a succession of terraces on the south side, where shallow marine sediments have been deposited. These represent former high stands of sea level and have been dated using U/Th or ^{14}C methods on fossil organisms (Vita-Finzi & King, 1985; Collier *et al.*, 1992). Armijo *et al.* (1996) have modelled terrace elevation and age using a thick elastic sheet model, and obtain an estimate of slip rate on the major Xylokastro Fault of 11 ± 3 mm yr^{-1}, consistent with geodetic estimates of extension at this end of the Gulf (Billiris *et al.*, 1991).

1.5 Scope of this thesis

The main questions that I wish to address in this thesis are:

- How does inter-seismic deformation measured over a few years on the scale 10 – 100 km relate to deformation in the same area over medium-term time-scales of 100 years, which may include several earthquakes?
- Is inter-seismic deformation distributed or does it show signs of localisation?
- How can geodetic measurements of co-seismic deformation be related to simple models of fault movement, and how can these models be reconciled with seismological and other observations?

The measurement technique that I use is primarily GPS surveying, which is covered in Chapter 2, although some terrestrial geodetic surveying observations are also used. The raw site displacements are then interpreted in a variety of ways, detailed in Chapter 3. First, the errors in the realisation of the reference frame are dealt with by fitting a linear velocity at each site through multiple survey epochs, assuming that deviations from this trend are caused by a global translation of the network. Later, the regional velocity field is approximated by fitting a polynomial function to the displacement estimates, and this method is contrasted with the alternative technique of estimating uniform strain within polygonal regions.

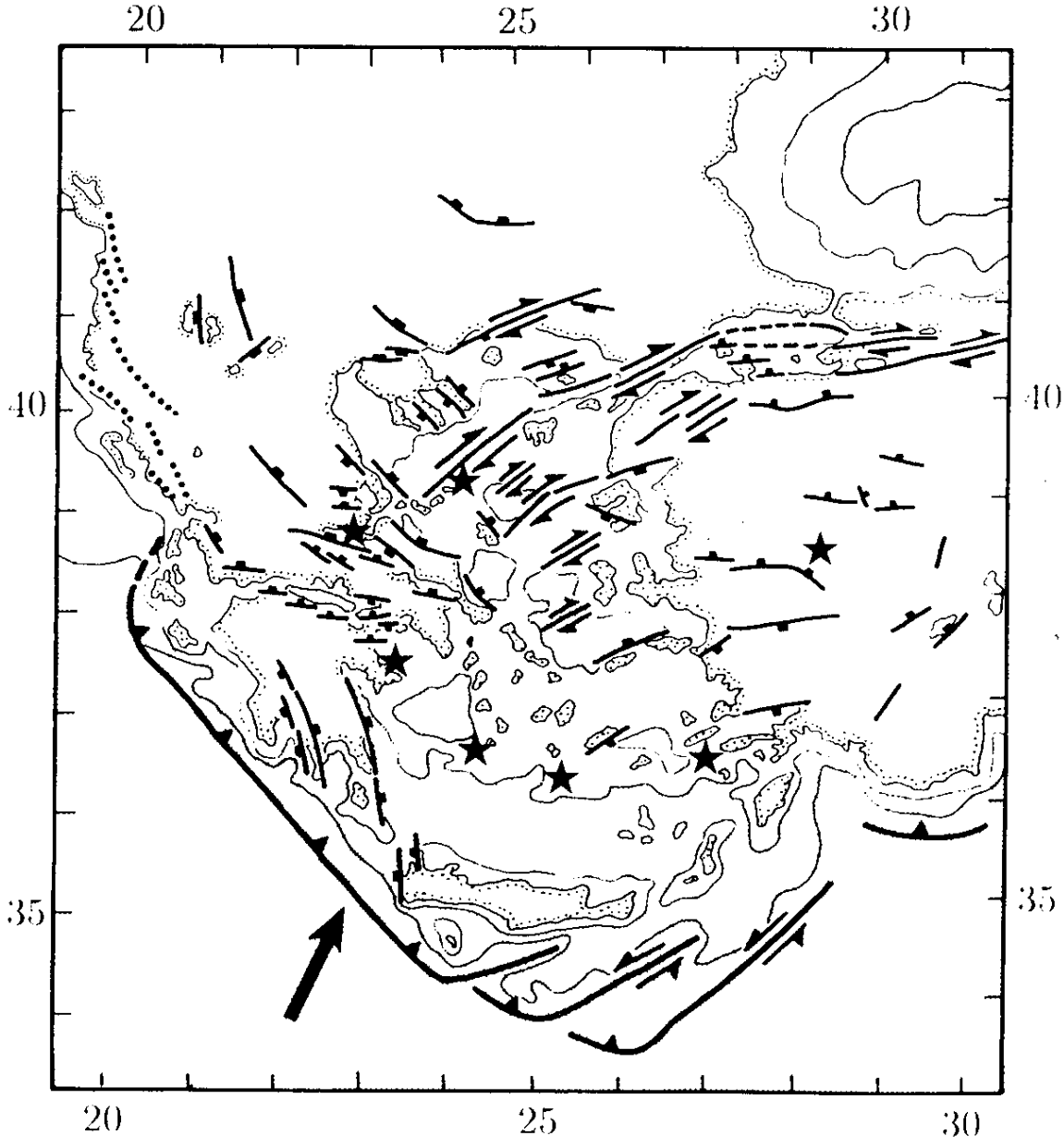

Figure 1.12: Active tectonic features of Greece and the Aegean Sea, from Jackson, 1994. The large black arrow shows the direction of convergence in the Hellenic trench (solid line with symbols on overriding plate), and Quaternary volcanoes are shown by stars. The dotted lines in Albania and northwest Greece represent anticlinal folding axes.

The second part of Chapter 3 deals with co-seismic deformation and its relation to observed displacements, and proposes a new algorithm that deals with the non-linearity of the inverse problem. The use of modelling to design networks for observing co-seismic and post-seismic deformation is dicussed, and examples are taken from the 1981 Alkyonides earthquake sequence.

Chapter 4 introduces the first dataset used in this thesis, the Central Greece Network, and covers the processing of GPS observations from this network, and interpretation of the regional strains using the methods in Chapter 3. A comparison is made with the 100-year data of Davies *et al.* (1996). Chapter 5 focuses on the Gulf of Korinthos, location of the 1995 Egion earthquake, and compares the results from this part of the Central Greece Network with the longer-term observations of strain and seismicity. The variation of extension rate along the Gulf is investigated, and its implications for seismic hazard are discussed.

The 1995 Grevena earthquake is the subject of Chapter 6. Issues of network design are discussed, followed by processing of the GPS datasets gathered in May and September 1995. Source parameters of the earthquake are then obtained by inversion from the geodetic displacements, and these are then compared with seismological and geomorphological observations. Limitations of the inversion scheme and constraints placed on our knowledge of the problem by the configuration of the GPS network are discussed.

In this thesis, no attempt is made to model the dynamics of the Aegean continental crust as a whole since this understanding can only come from observations made over wider areas that include the boundaries of the deforming zone. However, it will be seen that small- to medium-scale GPS networks can readily be used to constrain the kinematics of co-seismic and inter-seismic deformation, from which conclusions regarding the nature of the deformation can be drawn.

Chapter 2

GPS Surveying Methods

In this chapter I shall explain the principles behind the use of GPS measurements to obtain precise coordinates for points on the Earth's surface, and the practical aspects of applying these methods using the Bernese GPS processing software. The latter discussion will be in a general sense; precise details of survey-specific problems and their solutions will be left until Chapters 4 and 6.

2.1 Basic theory of relative GPS surveying

Only a brief outline of the theory underlying GPS positioning is given here, as comprehensive reviews (*e.g.* Beutler *et al.*, 1989; Dixon, 1991; Hager *et al.*, 1991; King *et al.*, 1985b) can be found elsewhere. Leick (1990) and Hofmann-Wellenhof *et al.* (1992) provide general texts on all aspects of the subject.

2.1.1 System overview

The Global Positioning System consists of (at present) 24 satellites orbiting in near-circular paths at altitudes of 19,000 – 22,000 km above the Earth's surface. The satellites are arranged in 6 planes inclined at 55° to the Earth's equator, 4 satellites per plane, and transmit radio signals on carrier frequencies 1.57542 GHz (known as L1, wavelength 0.19 m) and 1.22760 GHz (known as L2, wavelength 0.24 m). The carriers are modulated by pseudo-random noise (PRN) binary codes that are unique to each satellite, in order to provide timing information which can be used to provide positions with navigational accuracy (of order 100 m) in real time. In order to recover the carrier phase observable for geodetic positioning purposes, these PRN codes must be removed from the signal. The carrier waves are further modulated at the lower rate of 50 Hz with a binary code that provides information about satellite availability (almanac)

and position (broadcast ephemeris).

GPS receivers commonly measure the carrier beat phase which is obtained by heterodyning the incoming signal with a signal from the receiver's internal oscillator to obtain a lower-frequency (and hence more easily measured) signal. The cumulative carrier beat phase is recorded at intervals (epochs), usually 30 s apart. The satellite-to-receiver pseudoranges (see below) are also recorded at each epoch.

2.1.2 PRN codes and pseudorange

The L1 frequency is modulated with a code of chip rate 1.023 MHz known as the C/A (Coarse/Acquisition) code, which is freely available, and also with a code of chip rate 10.23 Mhz known as the P (Precise) code. The C/A code and P-code on L1 are offset by 90° of phase, so they can be separately identified. Also, the amplitude of the C/A code modulation is greater than that of the P-code, to facilitate initial lock-on, *i.e.*

$$S_1 = A_{P1}P(t)D(t)\cos(\omega_1 t) + A_{C/A}C(t)D(t)\sin(\omega_1 t) \tag{2.1}$$

where A_{P1} and $A_{C/A}$ are the amplitudes of the P and C/A codes $P(t)$ and $C(t)$; $D(t)$ is the almanac and broadcast orbit information code; and ω_1 is the angular frequency of the L1 carrier. $P(t)$, $C(t)$ and $D(t)$ can be regarded as square waves of amplitude $\pm$ 1. The L2 frequency is modulated only with the P-code and almanac/orbit information.

The P-code itself is generally available, but the US military reserve the right to encrypt the precise signal further at any time, under the system of Anti-Spoofing (AS). Under AS, the P-code is combined with a secret 500 kHz code known as the W-code, to create the secret Y-code. Only receivers with military code-keys can use this code.

Receivers obtain the unmodulated carrier phase by correlating the incoming signal with copies of the C/A and P-codes generated internally. Each satellite transmits a unique portion of the code with a repeat time of one week. Correlation yields a time offset within the week for each satellite's signal, and as a by-product, the original carrier phase. The time offset can then be converted to a delay time between transmission and reception of the signal, which in turn can be expressed as a range measurement (the pseudorange, as propagation delays and clock errors are not modelled) by multiplying by the speed of light. In real-time GPS, four pseudoranges to different satellites (and of course the satellite ephemerides) are needed to solve for the three components of the receiver's position plus a correction term for the receiver's (inaccurate) clock.

Pseudorange can be obtained from time offsets determined from either the C/A code (C/A pseudorange) or the P-code on the L1 or L2 carrier (P1 or P2 pseudorange). The P1 and P2 pseudoranges will be more precise than the C/A pseudorange because the chip rate of the P-code is higher than that of C/A code. Another advantage of having access to both P1 and P2 is that they can be combined in a similar

manner to the carrier phase observations (Subsection 2.1.7) in order to reduce the effects of ionospheric bias (Subsection 2.1.6).

Under AS, receivers may still obtain the L1 phase since the C/A code is always available. They may follow one of several strategies in order to obtain the L2 phase from the Y-code encrypted signal on L2 (see Hofmann-Wellenhof *et al.* (1992), Chapter 5):

- Older receivers (*e.g.* Ashtech LM-XII and P-XII) typically work by multiplying the incoming L2 signal by itself. As the PRN code is essentially a series of polarity changes, squaring the signal removes all traces of the code, and the resulting signal is known as L2-squared ($L2^2$). The disadvantages of this method are that the frequency of the $L2^2$ carrier is double that of L2 (*i.e.* the wavelength is halved, which makes ambiguity resolution more difficult), and the noise of the signal is increased by a factor of $\sqrt{2}$. Also, this method does not yield either P1 or P2 pseudoranges.
- Modern cross-correlating receivers utilise the fact that the P-code is present on both the L1 and L2 carriers. Cross correlating the P-code between L1 and L2 yields the lag between the P1 and P2 code, and the L2 phase. A proxy for P2 pseudorange can be reconstructed given the C/A code pseudorange and the P1–P2 lag, but it will only have the precision of C/A and not the usual increased precision of P-code.
- Other modern receivers track the P-code on L2 in short time windows between W-code polarity reversals, and use this tracking to get P2 pseudorange, then square the L2 signal to get $L2^2$ phase.
- The most advanced receivers correlate the P-code in short time windows between W-code polarity reversals on both L1 and L2, and thereby obtain a real-time estimate of the W-code which can be removed from the signal to recover true P1 and P2 pseudoranges, and full-wavelength L2 phase, with much less signal degradation than other techniques. This is the technique used in the Ashtech Z-XII receivers.

2.1.3 Mathematical model for carrier phase GPS

The basic phase equation for the carrier phase observable $\phi_k^p(t)$ (in units of cycles) for satellite p measured by receiver k at epoch t can be written

$$\phi_k^p(t) = \phi^p(t) - \phi_k(t) + N_k^p + \epsilon \tag{2.2}$$

where $\phi^p(t)$ is the received phase of satellite p, $\phi_k(t)$ is the receiver clock phase, N_k^p is the integer part of $\phi_k^p(t)$ when the receiver first locks on, and ϵ represents random measurement noise. The received phase is related to the transmitted phase $\phi_T^p(t)$ by

$$\phi^p(t) = \phi_T^p(t - \tau) \tag{2.3}$$

where τ is the travel time of the signal. This can be expanded as a Taylor series

$$\phi^p(t) = \phi^p_T(t) - \dot{\phi}^p_T(t)\tau + \frac{1}{2}\ddot{\phi}^p_T(t)\tau^2 + \cdots \tag{2.4}$$

$$= \phi^p_T(t) - f\tau \tag{2.5}$$

ignoring $\ddot{\phi}$ and higher derivatives, and neglecting the effects of clock errors on the frequency f. Thus,

$$\phi^p_k(t) = \phi^p_T(t) - \frac{f\rho^p_k(t)}{c} - \phi_k(t) + N^p_k + \epsilon \tag{2.6}$$

where $\rho^p_k(t)$ is the distance from the satellite at time of signal transmission to the receiver at time of signal arrival, and c is the speed of light. However in practice this expression is contaminated by error terms, especially drift of the satellite and receiver clocks, and refraction and delay of the signal. Leick (1990, chapter 8) and King *et al.* (1985b) give rigorous derivations including treatment of the clock error terms.

The success of GPS positioning thus depends on being able to estimate or eliminate the error terms, initial fractional parts of the phases and the integer ambiguity so as to achieve satellite–receiver ranges as functions of time, which can then be converted to receiver positions. The integer ambiguity is usually eliminated by first estimating it as a (non-integer) nuisance parameter, and then fixing it to an integer if the correct integer can be determined. To do this, either the receiver and satellite coordinates must be known very precisely (cm-level) *a priori*, or observations must be taken a sufficiently long time apart so that they are uncorrelated, and can thus be used to solve for the parameters as a system of simultaneous equations. About 30 minutes separation is the minimum time to achieve this for relative positioning of two receivers by the double-difference method (Subsection 2.1.5).

2.1.4 Sources of error and bias

Carrier phase GPS relies ultimately on the stability of the transmitted carrier frequencies derived from the satellite 'clocks'. The 'clocks' in GPS satellites (actually stable high-frequency rubidium or caesium oscillators) are accurate to 1 part in 10^{13}. As part of the broadcast ephemeris, each satellite transmits corrections to its clock phase determined by the satellite ground control stations (which have even more accurate hydrogen maser clocks). However, as a result of the security measure SA (Selective Availability) imposed by the US military, the satellite clocks are dithered by an undisclosed amount (limited to 1 p.p.b. of frequency drift). Post-processed (precise) orbits may include post-processed clock phase corrections which will reduce the effect of this.

GPS receivers usually use lower-quality quartz oscillators (accurate to about 1 part in 10^{10}). However, they can be synchronized to GPS time using the PRN code measurements. Despite this, there is still a degree of non-simultaneity of the times at which phase measurements are made in different receivers and this has an implication for double-differencing methods (see below). An even more serious timing error can

arise as different designs of receiver may sample the phase at slightly different offsets to the true epoch. This can be eliminated by using receivers of only one type in a survey.

Errors in the satellite positions will also cause errors in GPS positioning, so the most accurate *a priori* orbit parameters available must be used (Subsection 2.2.4). Alternatively, the orbit parameters may be estimated along with the receiver positions ('Fiducial' GPS, Subsection 2.2.9).

The ionosphere and troposphere both slow down and refract the carrier wave (relative to its propagation in a vacuum) and so bias the phase measurement, leading to errors in positioning. The biases can be minimised explicitly by using models of tropospheric and ionospheric behaviour (Subsection 2.1.6) to correct the observations, or by double-differencing the observations (Subsection 2.1.5).

2.1.5 Error limitation using the double-difference observable

If two receivers k, m observe a satellite simultaneously, then we can form two equations of the form (2.6) and subtract them to get the single-difference phase observation $\Delta^p_{km}(t)$

$$\Delta^p_{km}(t) \;=\; -\frac{f}{c}\left[\rho^p_k(t) - \rho^p_m(t)\right] - \left[\phi_k(t) - \phi_m(t)\right] + N^p_{km} + \epsilon' \tag{2.7}$$

In this expression, the initial satellite clock phase $\phi^p_T(t)$ has been eliminated, and satellite clock phase errors have been eliminated to first order. Some error in the satellite phase will remain, because the signals arriving at the two receivers at an epoch will not have been transmitted simultaneously unless the receivers are vanishingly close, and equation 2.5 does not account for second- and higher-order terms. The quantity N^p_{km} is the difference of two integers, so is itself integer.

A further property of the single-difference observation is that if the receivers are fairly close, the satellite–receiver signal paths will be almost the same, so most of the signal propagation delays (Subsection 2.1.6) will cancel. Similarly, errors in the satellite positions will be largely common to both receivers and so will be significantly reduced.

The disadvantage of receiver-to-receiver differencing is that the phase observable is now defined in terms of the relative position of the two receivers, not the positions with respect to the geocentre (although the solution has some sensitivity to the geocentric position because of the satellite orbits). However, relative positions are usually all that are required, and a geocentric position for one end of a baseline can be obtained by averaging undifferenced code observations over a period of time, or more accurately by long-term single-point phase positioning or from combining global networks of GPS and / or SLR and VLBI observations.

If each receiver observes the same two satellites p, q simultaneously, then two expressions like (2.7) can be

differenced to yield the double-difference phase observation $\Delta^{pq}_{km}(t)$

$$\Delta^{pq}_{km}(t) \;=\; -\frac{f}{c}\left\{[\rho^p_k(t) - \rho^p_m(t)] - [\rho^q_k(t) - \rho^q_m(t)]\right\} + N^{pq}_{km} + \epsilon'' \tag{2.8}$$

Now the initial receiver clock phases $\phi_k(t), \phi_m(t)$ have cancelled, and receiver clock errors have been eliminated (except for second- and higher-order errors dependent on the differential velocities of the satellites). The phase ambiguity N^{pq}_{km} remains integer. Unfortunately, the noise ϵ'' is increased with respect to ϵ' (and ϵ).

As it is constant with respect to time, the integer ambiguity N^{pq}_{km} can be eliminated by subtracting two equations like (2.8) for different epochs to give the triple-difference observable. However this observable is even more noisy than the double-difference, so it is not often used for positioning, but can be used to identify cycle slip epochs (Subsection 2.1.8) as they will appear as outliers to the triple-difference solution.

A general disadvantage of differenced observations is that sets of simultaneous observations for several satellites and receivers will be correlated. Accurate processing of the observations requires that the correlations be modelled, and this is a computer-intensive operation (Beutler *et al.*, 1989).

For more rigorous derivations of differencing methods, including analysis of the higher-order error terms, see Leick (1990, chapter 8) and King *et al.* (1985b).

2.1.6 Atmospheric models

GPS carrier waves are slowed down and refracted on passing through the atmosphere. The most significant effects are dispersive delays in the charged ionosphere (50 – 500 km altitude) and non-dispersive delays in the troposphere (0 – 10 km altitude).

As the ionospheric effect is dispersive, it can be eliminated up to first order effects by using the L3 phase combination (Subsection 2.1.7). However for short baselines this may not be worthwhile because of the extra noise introduced, as most of the ionospheric effect will be removed by double-differencing (Subsection 2.1.5). For single frequency receivers in a network of single- and dual-frequency receivers, the ionospheric effect can be modelled from the dual-frequency measurements using the L4 observable (Subsection 2.1.7). The ionosphere can be modelled approximately as a thin shell of electrons at altitude (the actual altitude used is relatively unimportant as the ionosphere is well above the receiver network). The delay on the signal is proportional to the Total Electron Content (TEC) of the shell, and unless modelled can introduce scale errors of several p.p.m. (Hager *et al.*, 1991). A TEC model cannot account for short-period changes in the ionospheric effect, but will reduce the scale error that applies to a whole session of observations.

The effects due to the troposphere (and also to some extent the mesosphere) are caused by the presence

of water vapour and are non-dispersive, so cannot be eliminated using dual-frequency observations. The path delay can be regarded as two components: a 'dry' (induced dipole effect) component of typically 2 m at zenith, and a 'wet' (permanent dipole effect) component of around 0.2 m at zenith. Both components vary approximately with the cosecant of the elevation angle of the path at the antenna, and so only the zenith delay needs to be modelled. The dry zenith delay varies with surface pressure and can be accurately estimated using a range of models (*e.g.* Hopfield, 1969; Saastamoinen, 1972). The surface pressure can either be measured directly or estimated from a standard atmosphere extrapolated from sea level to the station height. The wet zenith delay is considerably harder to measure as this requires the use of a Water Vapour Radiometer (WVR) which measures microwave black-body radiation caused by water vapour. However the wet delay can be estimated from GPS measurements using a stochastic model as it varies slowly with time.

2.1.7 Combinations of L1 and L2 phase measurements

The ionospheric effect is dispersive and can be approximated to first order by a phase advance of magnitude $\frac{a}{f}$, where a varies with the TEC of the ionosphere. Thus Equation 2.6 must be rewritten (for the L1 phase)

$$\phi^p_{k,L1} = \phi^p_{T,L1} - \frac{f_{L1}}{c}\rho^p_k - \phi_{k,L1} + N^p_{k,L1} - \frac{a}{f_{L1}} + \epsilon \tag{2.9}$$

and similarly for the L2 phase. As the L1 and L2 signals are derived from the same base oscillator, they are coherent, so

$$\phi_{L2} = \frac{f_{L2}}{f_{L1}}\phi_{L1} \tag{2.10}$$

which can be substituted into the L2 version of (2.9) to give

$$\phi^p_{k,L2} = \frac{f_{L2}}{f_{L1}}\phi^p_{T,L1} - \frac{f_{L2}}{c}\rho^p_k - \frac{f_{L2}}{f_{L1}}\phi_{k,L1} + N^p_{k,L2} - \frac{a}{f_{L2}} + \epsilon' \tag{2.11}$$

Multiplying (2.9) by α and (2.11) by β, then adding, where

$$\alpha = \frac{f^2_{L1}}{f^2_{L1} - f^2_{L2}} \quad ; \quad \beta = -\frac{f_{L1}f_{L2}}{f^2_{L1} - f^2_{L2}} \tag{2.12}$$

gives the L3 phase observable (after Leick 1990 and Beutler *et al.* (1989); other notations for L3, L4, L5 differ)

$$\phi^p_{k,L3} = \alpha\phi^p_{k,L1} + \beta\phi^p_{k,L2} = \phi^p_{T,L1} - \frac{f_{L1}}{c}\rho^p_k - \phi_{k,L1} + \alpha N^p_{k,L1} + \beta N^p_{k,L2} + \epsilon'' \tag{2.13}$$

which is independent of the first-order ionospheric effect. Unfortunately the ambiguities $\alpha N^p_{k,L1}$ and $\beta N^p_{k,L2}$ are no longer integers so cannot be fixed directly, and the measurement noise ϵ'' is roughly three times that of L1 or L2.

Alternatively, we can use (2.9) and (2.10) to form the L4 phase observable

$$\phi^p_{k,L4} = \phi^p_{k,L1} - \frac{f_{L1}}{f_{L2}}\phi^p_{k,L2} = N^p_{k,L1} - \frac{f_{L1}}{f_{L2}}N^p_{k,L2} - \frac{a}{f_{L1}}\left(\frac{f^2_{L2} - f^2_{L1}}{f^2_{L2}}\right) + \epsilon''' \tag{2.14}$$

which is independent of the satellite–receiver distance and of clock errors. Several L4 observations can be used to solve for a and hence the TEC, with the ambiguities (non-integer) being estimated as nuisance parameters. The measurement noise of L4 is approximately $1\frac{1}{2}$ times that of L1 or L2.

It is possible to overcome the disadvantage of non-integer ambiguities in the L3 combination, if the integer-valued L5 linear combination of the ambiguities can be resolved. The L1 and L2 ambiguities can then be fixed using the non-integer L3 ambiguity and the integer L5 ambiguity. The L5 phase observable (also know as the wide-lane phase observable) is given by

$$\begin{aligned}\phi^p_{k,L5} &= \phi^p_{k,L1} - \phi^p_{k,L2} \\ &= \left(\phi^p_{T,L1} - \phi^p_{T,L2}\right) + (f_{L1} - f_{L2})\,\frac{\rho^p_k}{c} - (\phi_{k,L1} - \phi_{k,L2}) \\ &\quad + \left(N^p_{k,L1} - N^p_{k,L2}\right) - a\left(\frac{1}{f_{L1}} - \frac{1}{f_{L2}}\right) + \epsilon''''\end{aligned} \tag{2.15}$$

Clearly the ambiguity term $(N^p_{k,L1} - N^p_{k,L2})$ is integer. The measurement noise on L5 is roughly six times that of L1, and the ionospheric effect is larger. However, these disadvantages are offset by the fact that the effective wavelength of L5 is 0.86 m (0.43 m if the receiver is L2-squaring) so it is much easier to resolve the ambiguity on longer baselines, even with poor orbit information. The ionospheric effect can be reduced if a TEC model is used. For short baselines (20 km or less, depending on solar activity), the ionospheric effect is almost completely removed by double-differencing so it is better to use the less noisy L1 and L2 observables.

2.1.8 Data cleaning

If a receiver momentarily loses lock on the signal from a satellite (*e.g.* because the signal path is temporarily blocked by an obstruction), then during the break it attempts to extrapolate the signal using its internal clock. However, because the internal clock is relatively unstable, the projected phase measurement may differ from the true value by some amount. When phase lock is achieved again, the projected phase measurement is 'snapped' back to the actual phase, but the number of complete cycles projected during the outage may differ from the true number by an integer amount. This is known as a cycle slip and must be eliminated for carrier phase GPS to work properly. The majority of cycle slips can be detected automatically, but manual inspection of the phase measurements (not available in all software packages) may be required to ensure that all slips have been removed.

2.1.9 Reference frames and satellite orbits

In order to compute the satellite orbits, it is necessary to model them in an inertial reference frame. The closest approximation we have to this is the celestial reference frame in which the distant stars are fixed points. However, station coordinates are required in an Earth-fixed (terrestrial) reference frame and so the two reference frames must be related by a model of the Earth's orientation and motion in space.

Several mechanisms operate to change the Earth's spin vector:

- Precession of the spin axis caused by lunar/solar gravitational attraction of the equatorial bulge, with a period of 26,000 years and an amplitude of 23.5°. Precession is a well-modelled process (*e.g.* Kaplan, 1981).
- Nutations of the axis, superimposed on the precession. These have periods in the range from a few days to 18.6 years and amplitudes of up to 9″, and are caused partly by lunar/solar attraction and partly by mass redistribution within the solid Earth (*e.g.* solid Earth tides). Nutations have also been modelled (*e.g.* Kaplan, 1981).
- Polar motions are less well-understood fluctuations thought to be caused by the free and forced responses to external torques, large earthquakes, core–mantle coupling, and motions of the hydrosphere (*e.g.* ocean tides, weather systems). The dominant motion is the Chandler wobble with a period of 14 months, but longer- and shorter-period terms exist. The amplitude of polar motion is of order 1″, and time series of polar motion (including short-term extrapolations) can be obtained from VLBI, SLR or global GPS analyses.
- The rate of rotation of the Earth varies due to the above mechanisms and is hard to model. It is measured and forecast in a similar way to polar motion, and with the aid of atomic clocks.

The celestial reference frame can be related to the terrestrial frame by a series of rotation matrices describing each of the above perturbations.

Given the celestial reference frame and a spherical harmonic expansion of the Earth's gravitational field (only low-order terms are needed, as the satellites are some distance from the Earth, and the Sun and Moon can be treated as point masses), it is possible to solve the dynamical equations of motion for the satellites and express each satellite's motion instantaneously in terms of six parameters (three components of position plus three of velocity, or the six parameters of a Keplerian orbit). To obtain a model that is valid over a longer period, the effects of direct radiation pressure on the satellite must be included, which requires a further two parameters (the components of acceleration resulting from the radiation pressure in the direction of the Sun, and in the direction perpendicular to this and along the satellite solar panel axis, known as direct radiation pressure and Y-bias).

Satellite orbits are computed *a posteriori* using global GPS solutions with observations from tracking stations spread around the Earth. The coordinates of these reference stations may have been determined by SLR, VLBI or global GPS network solutions. In addition, the reference stations' velocities with respect to each other must be specified so that the reference frame can be tied to the celestial frame at different epochs. Two classes of velocity model can be used:

- The NUVEL–1, NUVEL–1NNR and NUVEL–1A plate tectonic models (DeMets *et al.*, 1990; Argus & Gordon, 1991; DeMets *et al.*, 1994). These incorporate magnetic anomaly spreading rates, transform fault azimuths and recent earthquake slip vectors to give motions for the major rigid plates averaged over the last 3.0 Ma. The original model NUVEL–1 (and its no-net-rotation version NUVEL–1NNR) was revised to model NUVEL–1A in the light of an improved geomagnetic time scale (the actual change is to multiply the angular velocities of NUVEL–1 or NUVEL–1NNR by a constant factor of 0.9562 to achieve NUVEL–1A). The advantage of these models is the large amount of data used and the long time-span of the data, resulting in stable models. The models are in only two dimensions *i.e.* vertical motions are not included. In regions of post-glacial rebound or rapid tectonic deformation this may be significant, and so the models should not be used. Also, the models are only valid for the rigid plates, so do not apply to horizontal motions of stations in zones of continental deformation (*e.g.* Dionysos, Greece).

- The ITRF velocity models (*e.g.* Boucher *et al.*, 1993; Boucher *et al.*, 1994). These are based on long-term VLBI, SLR and GPS global solutions dating back as far as 1976 (although the majority of stations were not occupied until the mid-1980s and GPS measurements were not included before 1989). They include both horizontal and vertical velocities and cover regions of distributed deformation as well as the rigid plates. For stations lying on the rigid plates, the agreement with NUVEL–1A is very good (Robbins *et al.*, 1993), except that there is a small rotational difference between the NUVEL–1A reference frame (which is based on the assumption of no net rotation of the Earth's lithosphere) and the VLBI reference frame (which is based on the 'fixed' stars). Global GPS solutions used by the ITRF have a reference frame defined dynamically through the satellite orbital models. There is in turn a rotational difference between this reference frame and the VLBI and NUVEL–1A reference frames, but the differences in definition of these datums are well known (Heflin *et al.*, 1992).

The broadcast ephemerides transmitted by the GPS satellites are extrapolations of the *a posteriori* orbits, and degrade with time because of limitations in radiation pressure modelling. Also, the broadcast orbits are intentionally degraded as part of SA. However, the *a posteriori* orbits, known as 'precise' orbits, are undegraded and available from several global analysis centres.

2.2 Processing of GPS measurements

Other workers (*e.g.* Blewitt *et al.*, 1992; Dong & Bock, 1989; Lichten & Border, 1987) have already published GPS processing strategies for use with various software packages. The strategy I have adopted is based on these, but the choice of GPS processing method remains a somewhat subjective operation and varies from campaign to campaign. Further details specific to individual campaigns are given in later chapters.

2.2.1 GPS processing software

The software used is the Bernese GPS processing software, version 3.4 (Rothacher *et al.*, 1993) developed at the Astronomical Institute of the University of Berne, Switzerland. Originally, I used version 3.3 of the software, but all surveys processed with version 3.3 were reprocessed with version 3.4. The software is written in FORTRAN-77 and has been adapted to run on the Sun workstations at Oxford. Table C.1 summarises the functionality and derivation of all programs referred to in this thesis.

All combinations of phase and code observables can be used, for position estimation and ionospheric modelling. The 'dry' troposphere effect is calculated using the Saastamoinen model (Saastamoinen, 1972) under the assumption of standard atmospheric conditions.

2.2.2 Pre-processing

Before measurements can be input to the Bernese software, they must be converted to the RINEX (Receiver INdependent EXchange) format (Gurtner *et al.*, 1989; Gurtner & Mader, 1989). This is achieved using the receiver manufacturer's software (*e.g.* `ashtorin` for Ashtech receivers). RINEX is in ASCII character format, so at this point it is easy to check that the marker name and the offset between the ground mark and the phase centre of the antenna are stored correctly in the file. The file is also checked for the continuous presence of data throughout the observation session.

2.2.3 Zero-difference processing

Observations are then converted to the Bernese software internal format (using program `rxobv3`). The code observations (P1 and P2 in the ionosphere-free combination if available, or otherwise C/A) are checked for outliers and bad data (in program `codchk`) and then used (in program `codspp`) to solve for receiver clock offsets (Subsection 2.1.4) at each epoch of observation, simultaneously with the marker position for the entire session. If no better *a priori* coordinates exist (*e.g.* from a previous geodetic survey) for the

marker, then the code-derived coordinates can be used as *a priori* coordinates for subsequent processing.

2.2.4 Satellite orbit choice

For cleaning the data, the most accurate orbits available must be used, otherwise, particularly for 'long' baselines over 100 km, cycle slips can be hidden by orbital errors. For shorter baselines, the broadcast ephemerides can be used without ill effects.

The reference frame of the final network solution (Subsection 2.2.7) is determined by the choice of orbits. If the orbits are to be constrained using the fiducial method (Subsection 2.2.9) the initial choice of orbit is unimportant, as the reference frame implied by the fiducial station coordinates will be imposed on the orbits during orbit improvement.

Instead of applying the fiducial method explicitly, 'precise' ephemerides can be used. As these rely on global fiducial networks, the global accuracy of the orbits will be better than for the regional fiducial technique, although the accuracy as satellites pass over the region of interest may be lower. To get the most accurate positions for ground markers in a region, the precise orbits are chosen that have the greatest density of fiducial stations about the region of interest, because this improves the constraints on short-period orbit perturbations that affect the region in question. Other considerations are: the time-span of orbital arcs computed, the software used, and the suitability of the fiducial constraints imposed and the resulting reference frame (see also Subsection 2.2.9).

The orbits produced by CODE (Centre for Orbit Determination in Europe) since June 1994 are based around 70 global fiducial stations, including up to 25 in or near Europe, using the Bernese GPS software version 3.5. This proportion of European stations used is higher than that for other IGS analysis centres, and also there is advantage in using the same satellite dynamic models in the orbit generation as in the processing software. The orbits are computed as 3-day arcs, of which the middle day is released as the precise ephemeris for that day. Satellite clock correction parameters are also produced. I have used CODE orbits to process GPS observations from Greece, on occasions when I have not used the fiducial method (except for the June 1991 Alkyonides network (Subsection 3.5.4) which pre-dates CODE orbit determination). Stations on or near the European plate used in CODE orbit determinations are shown in Figure 2.1.

Figure 2.2 shows sets of double-difference L3 residuals obtained using broadcast and precise orbits, for a 145 km baseline observed in October 1991. The reduction of residuals (*i.e.* improvement of the solution) made by the use of precise orbits can be clearly seen.

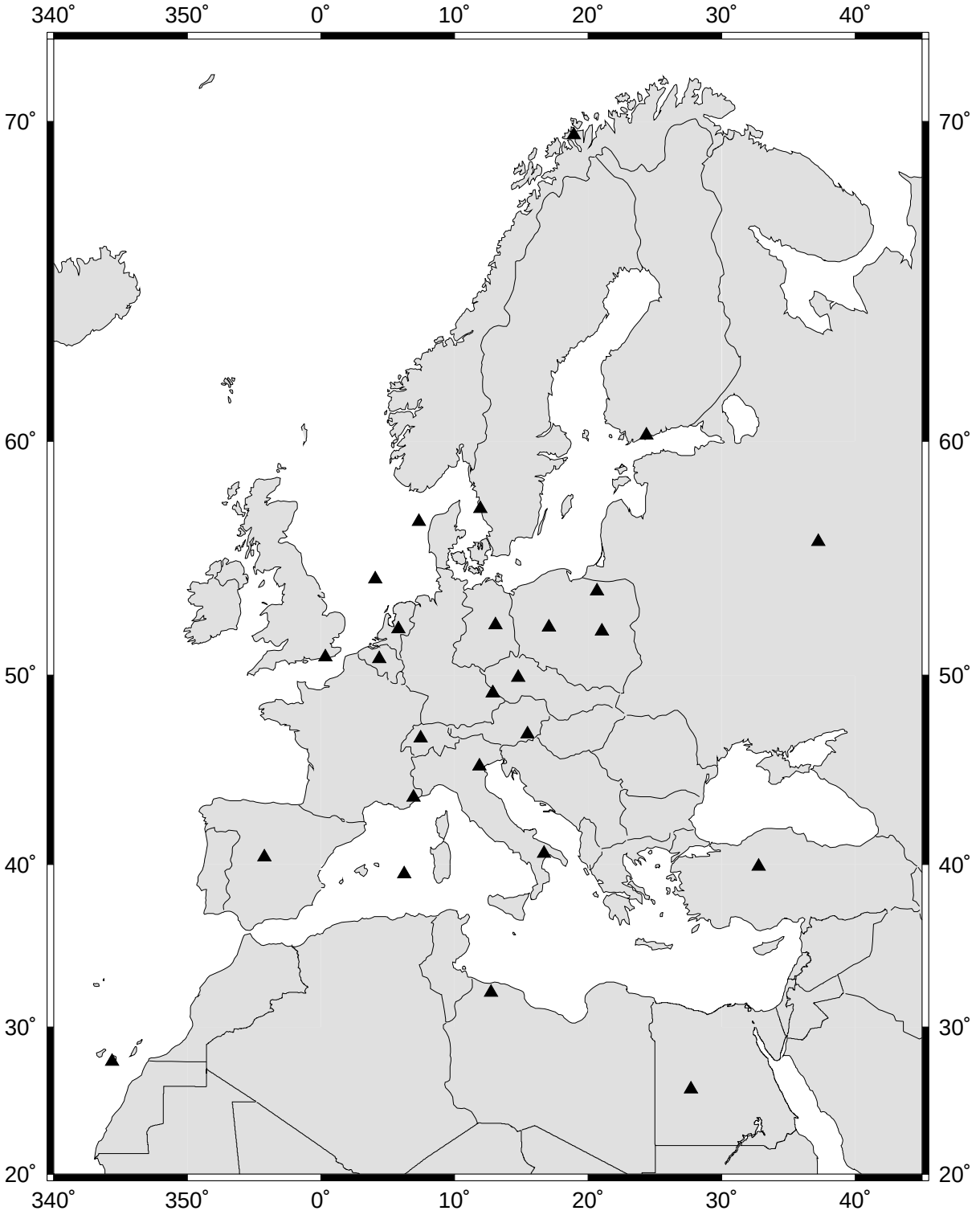

Figure 2.1: Fiducial sites on or near the European plate used by CODE, October 1995.

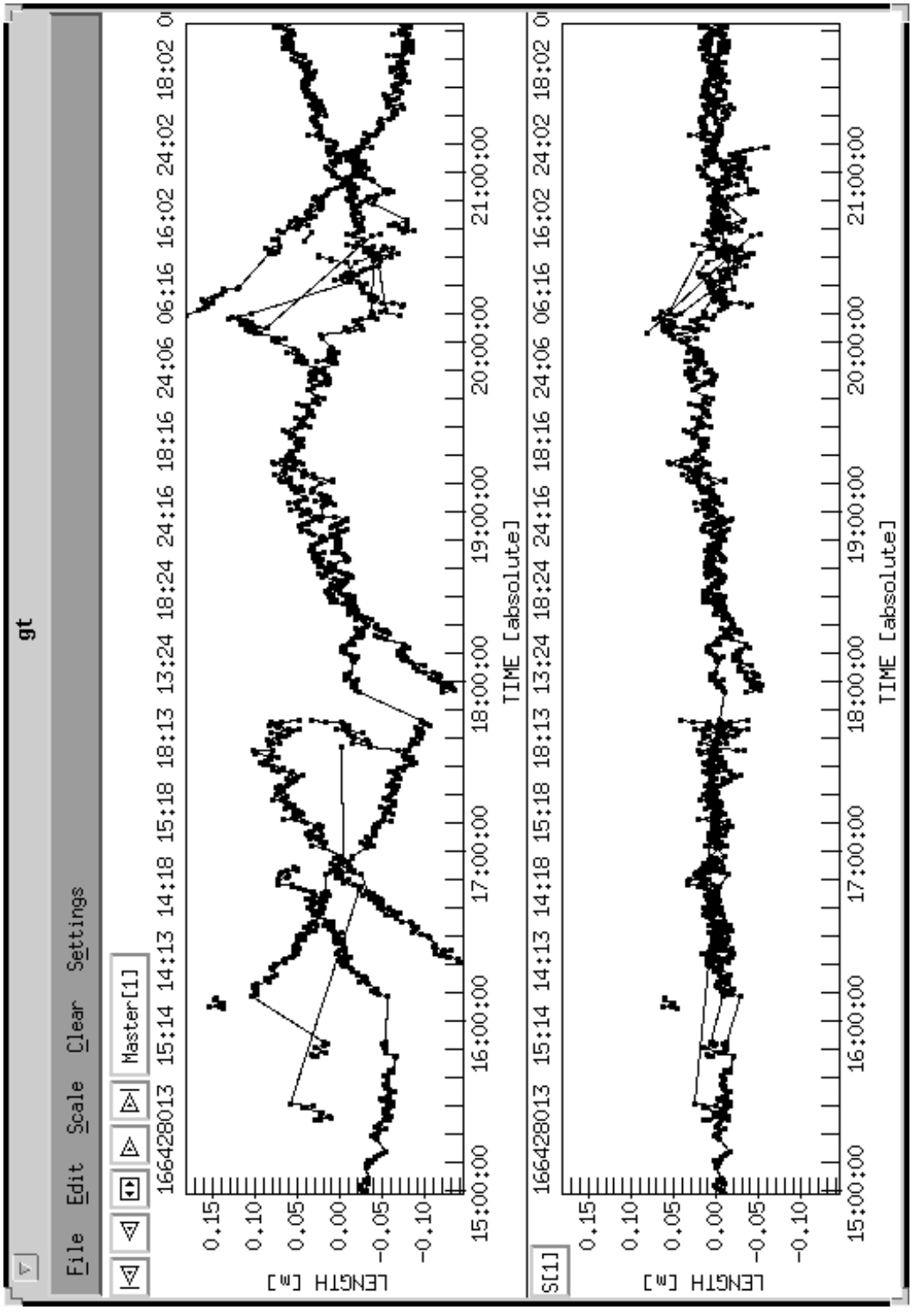

Figure 2.2: L3 double-difference phase residuals obtained using broadcast (above) and precise (below) orbits, shown at the same scale (using the UNAVCO program `gt`), for a 145 km baseline observed in October 1991. Orbit errors in the broadcast ephemerides show up as arcs in the residuals. Note that sometimes these arcs are large enough to be mistaken for cycle slips, or to mask possible cycle slips.

2.2.5 Baseline selection strategy

Although all cominations of baselines are mathematically equivalent, in order to maximise the benefit of using the double-difference observable (Subsection 2.1.5) and facilitate data cleaning, the shortest possible baselines (in length) are chosen. However, in surveys where there is no previously-agreed observation schedule, longer than optimal baselines must be used if one is to make use of all the data collected, because observations are only of use when they are made simultaneously at both ends of the baseline, and very short periods of simultaneous observations are of no use. Also, in a survey involving more than one type of GPS receiver or antenna, baselines involving similar receivers are chosen where possible, although this latter consideration is given less weight than the previous two. The choice of baselines to process is therefore somewhat subjective as there are no hard-and-fast rules to account for the trade-off between long periods of simultaneous observations and short baseline lengths.

Station-to-station single difference observations are formed in program `sngdif`. The task of forming the satellite-to-satellite difference (resulting in the double-difference observation) is performed 'on the fly' by later programs in the Bernese system, as is formation of the triple-difference observation for data cleaning purposes (see below).

2.2.6 Baseline cleaning strategy

Automatic cleaning is performed by the program `mauprp` which works on the triple-difference (station-to-station, satellite-to-satellite, epoch-to-epoch) measurements on L1, L2 or L3 frequency. `mauprp` filters the single-difference observations to identify 'clean' regions and then attempts to remove outlier observations and repair cycle slips. However, it is not always successful and manual inspection of the baseline data is required. In particular, when operating on particularly 'dirty' data with the L3 frequency `mauprp` can sometimes transfer cycle clips from one frequency to the other or even insert new slips.

A baseline relative position solution is generated using the program `gpsest` and double-difference residuals to this solution can then be inspected with the graphical utility `gt` (J. Johnson, UNAVCO, 1993). `gt` allows outlier points to be marked and cycle slips to be fixed manually. As `gt` shows double-difference residuals (*i.e.* for pairs of satellites), care must be taken that cycle slips and outliers are attributed to the correct satellite of the pair, by considering all combinations of satellites at each epoch. Repeated application of the `gpsest`–`gt` cycle will result in clean single-difference observations. This must be done for both L1 and L2 frequencies. Figure 2.3 shows a set of double-difference L1 residuals before and after the removal of a cycle slip.

The L3 frequency must then be checked for outliers (which may not be apparent on L1 or L2). Sometimes (particularly with L2-squaring receivers when half-cycle slips can occur on L2) it is easier to clean L1 and

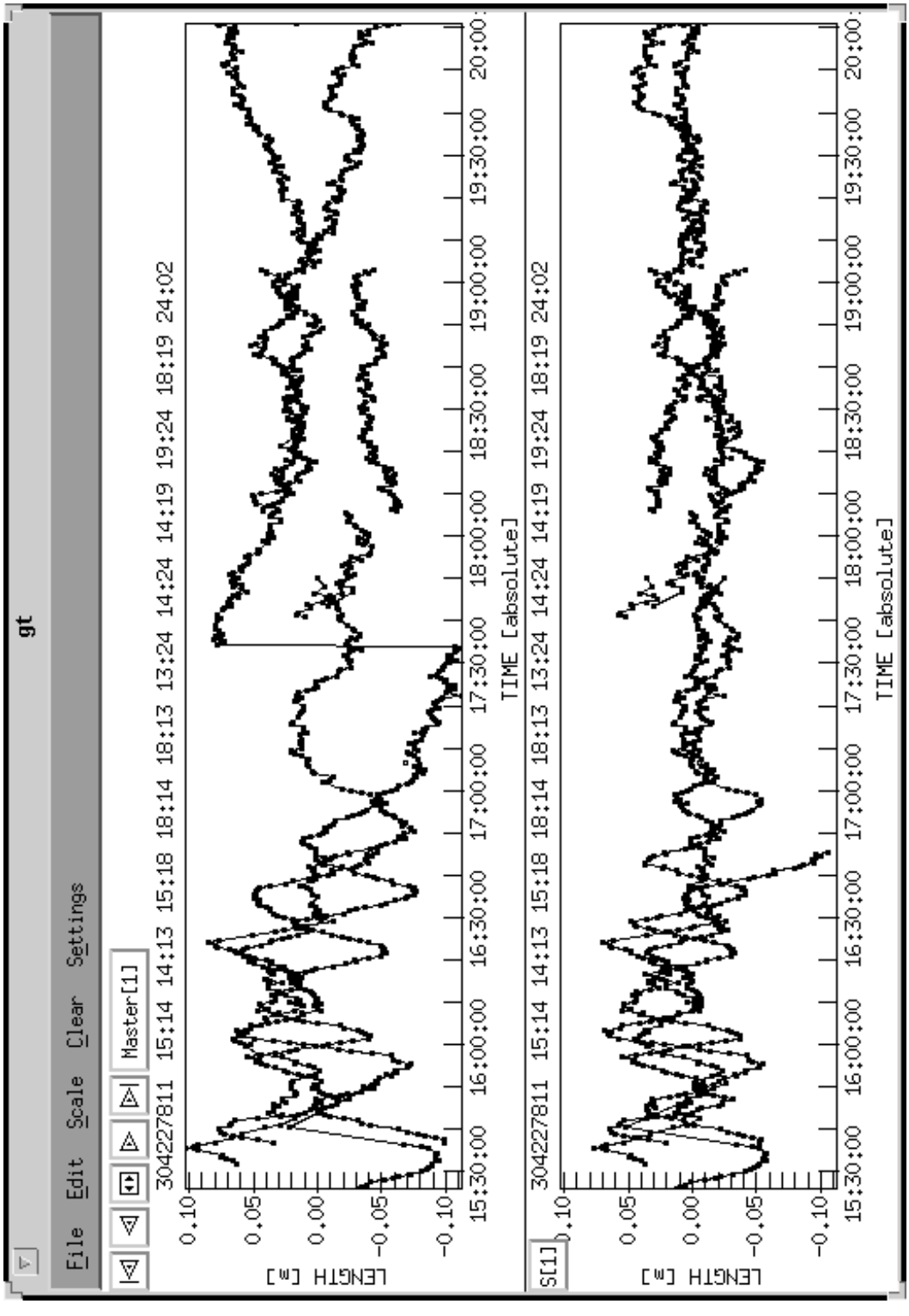

Figure 2.3: L1 double-difference phase residuals before (above) and after (below) removal of a cycle slip, shown at the same scale (using the UNAVCO program `gt`).

then fix the L2 slips using the L3 solution residuals. It is not possible to clean L3 directly as several different combinations of L1 and L2 cycle slips can look very similar to each other (or to no slip at all). Cleaning is a laborious process, especially for longer baselines where cycle slips can be hidden among short-term ionospheric fluctuations and orbital errors.

2.2.7 Network solutions

Once all the baselines in a day are cleaned, they are combined in a daily network solution (again using the program `gpsest`). In the network solution, in addition to the marker positions and ambiguity parameters, a tropospheric zenith wet delay at each site is estimated using a stochastic model (*i.e.* with parameters allowed to vary randomly within *a priori* limits). A functional model (mapping function) is used to describe the variation of delay with elevation angle as the satellite moves away from its zenith position. Several delay parameters per receiver per session can be estimated, with successive parameters constrained to vary by small amounts only. Also, ionospheric TEC variation models can be introduced. The correlations between the double-differenced phase observations are modelled correctly at this stage, to improve error estimation. At least one station (but usually only one, except in fiducial GPS) must be held fixed in a network solution as the relative GPS method only gives estimates of relative positions. If only one station is fixed, the scale and orientation of the network are implied by the choice of orbital parameters, whereas if three or more stations are fixed, the scale and orientation are constrained by the coordinates adopted for the fixed stations. It is possible to relax the *a priori* constraints on the orbital parameters in this case to reduce the distortion.

First, an L3 solution is performed, with no attempt to resolve the non-integer L3 ambiguity parameters. This provides a set of *a priori* coordinates much better than that obtained from `codspp`. Then, an attempt is made to resolve the ambiguity parameters to integers. Two methods are possible:

- For larger networks (more than 30 km in extent) or if there are large short-term ionospheric fluctuations, the L5 ambiguities are resolved first using an L5 network solution. The L1 and L2 ambiguities can then be resolved using an L3 network solution, which will also provide the final daily coordinate set. For very large networks (100 km or more in extent) it may be necessary to use an ionospheric model in the L5 solution. Ionospheric models are derived using the program `ionest` which uses the zero-difference L4 phase or code observable. L4 code is preferable to L4 phase as it is not vulnerable to cycle slips and requires no ambiguity terms to be estimated, but is only obtainable if both P1 and P2 code measurements are available.

- For smaller networks in times of ionospheric calm, it is better to resolve L1 and L2 ambiguities using L1 and L2 network solutions respectively, as these are less noisy than L5 solutions. The resolved L1

and L2 ambiguities can then be incorporated in an L3 network solution to produce the final daily coordinate set.

Often it is impossible to resolve all the ambiguities so the final daily solution will include several non-integer ambiguity estimates as 'nuisance' parameters. In general, it seems that a 'good' solution involves 80 – 90% ambiguity resolution. If the percentage of ambiguities resolved is much smaller than this, it may be better to use the wholly-unresolved ('float') L3 solution as the final daily solution.

2.2.8 Campaign solutions and quality assessment

Daily geocentric Cartesian coordinate sets and their covariance matrices are combined in a weighted least-squares adjustment using the Bernese program `compar` or the program `L3D` developed by the Department of Surveying, University of Newcastle upon Tyne (Cross, 1990). The advantages of `L3D` over `compar` are that `L3D` uses Pope's tau-test to flag outlying coordinates which may require attention (Cross, 1983), and provides better statistical information.

A measure of the quality of the campaign solution is the root mean square residual (in the local north, east and up coordinate frame) between the daily coordinate sets and the campaign coordinate set. The precision of GPS in the local horizontal plane is typically three to four times that in the local vertical direction, as the effects of tropospheric and ionospheric delay act mostly on the vertical, and as all the satellites are observed above the horizon, there is less redundancy of measurement in terms of the vertical component than in the horizontal.

2.2.9 Fiducial orbit improvement

If the satellite orbits are inaccurate or the reference frame of the orbits is not sufficiently precisely determined, the orbits may be 'improved' with the aid of GPS measurements made at well-constrained reference stations in or near the region of the survey. For the central Greece network occupations in 1989, 1991 and 1993, the satellite orbits have been improved by this method. This because prior to 1992, the 'precise' orbits available did not have the required degree of precision (about 5 p.p.b.) and it was important to compare the later 1993 survey with the earlier two surveys using the same reference frame. In practice, differences between the 1993 'precise' and 'fiducial' positions were negligible (P.A. Cruddace, pers. comm.).

The fiducial sites available within the region are Herstmonceux (Hers), Madrid (Madr), Onsala (Onsa) and Wettzell (Wett) (Figure 2.4), all of which lie on the stable European plate. In addition, the mobile SLR site at Dionysos (Dion) was occupied as a temporary GPS fiducial site during the campaigns. The station at Matera (Mate) was rejected as a fiducial site because of uncertainty as to whether it moves with

ITRF Site	X/m	Y/m	Z/m
DION SLR 7515	4595216.4190	2039435.3200	3912629.5200
HERS GPS	4033470.3180	23672.6330	4924301.1260
MADR GPS	4849202.5450	-360329.2940	4114912.9740
ONSA GPS	3370658.8100	711876.9010	5349786.7870
WETT GPS	4075578.7450	931852.5330	4801569.9600

ITRF Site	V_x/mm yr^{-1}	V_y/mm yr^{-1}	V_z/mm yr^{-1}
DION SLR 7515	0.0036	0.0069	-0.0114
HERS GPS	-0.0140	0.0175	0.0102
MADR GPS	-0.0072	0.0203	0.0147
ONSA GPS	-0.0136	0.0136	0.0118
WETT GPS	-0.0169	0.0161	0.0092

Table 2.1: Fiducial station coordinates at epoch 1988.00 (above) and velocities (below) in the ITRF 92 system.

the stable European plate velocity, or with the Adriatic 'microplate'. Similarly, Ankara (ANKA) was not used, as it lies within the Anatolian deforming zone (and was unavailable in 1989 anyway). The Trőmso site (TROM) was not used because of uncertainties in the vertical motion due to post-glacial rebound. Other stations at Kootwijk (KOSG) and Zimmerwald (ZIMM) were not used because the data were redundant (the other sites surround them), and not available in 1989.

Coordinates were obtained in the ITRF 92 reference frame (Boucher *et al.*, 1993) for all fiducial sites. Because of the somewhat uncertain dependency of the ITRF velocity fields and the short time-span of the measurements on which they were based at the time of GPS processing, the NUVEL–1NNR velocity model was used for fiducial station velocities, with the exception of DION, for which the ITRF 92 velocity was used as DION lies within a deforming zone. The difference between the NUVEL–1NNR and ITRF 92 site velocities is at the level of 1–2 mm yr^{-1}, so the difference between NUVEL–1NNR-derived coordinates and ITRF 92-derived cordinates will be sub-centrimetric at any time during the period 1989–1993 and hence not significant for orbit determination. However it was felt the time of GPS processing that the NUVEL–1NNR model represented a more stable reference frame should it be necessary to re-create the frame at a later epoch.

As ITRF coordinates are quoted as extrapolated values at epoch 1988.00 (1993.00 for more recent ITRF solutions), the coordinates must be restored to the mean date of the ITRF observations (taken to be 1992.50 because the frequency of observations increased rapidly around this date) using the ITRF 92 velocity model, and then projected to the campaign epochs using the NUVEL–1NNR velocities (except DION). Table 2.1 gives the coordinates of the fiducial stations in the ITRF 92 system at epoch 1988.00 and their ITRF 92 velocities, Table 2.2 gives their NUVEL–1NNR velocities, and Table 2.3 gives their epoch coordinates calculated as above.

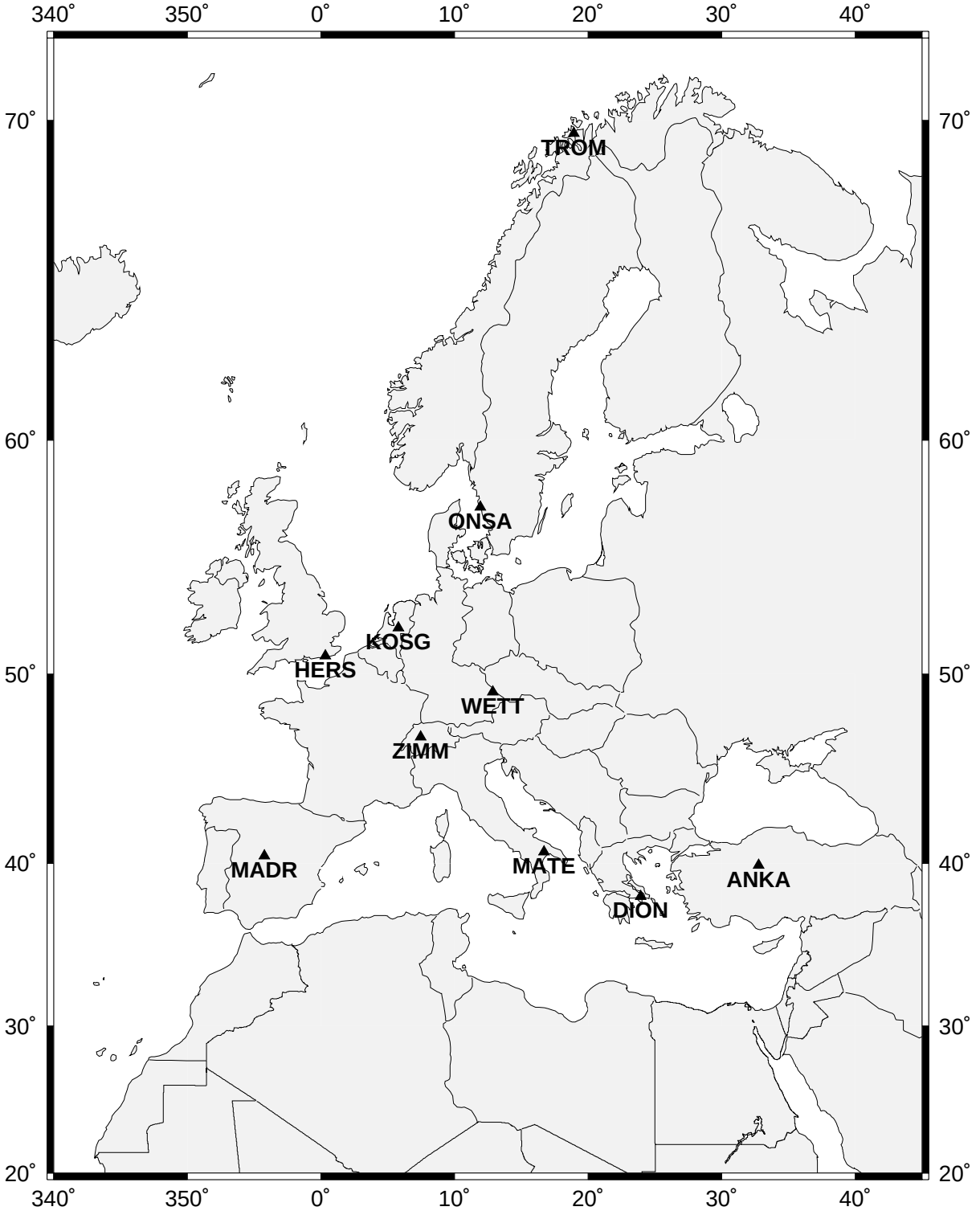

Figure 2.4: Fiducial sites on or near the European plate operating since 1991. Only Hers, Onsa, Wett and Trom were available in 1989. Dion was only available on a campaign basis until 1995.

ITRF Site	V_x/mm yr^{-1}	V_y/mm yr^{-1}	V_z/mm yr^{-1}
DION GPS	-0.0165	0.0192	0.0094
HERS GPS	-0.0140	0.0175	0.0102
MADR GPS	-0.0091	0.0202	0.0125
ONSA GPS	-0.0157	0.0166	0.0077
WETT GPS	-0.0151	0.0184	0.0092

Table 2.2: Fiducial station velocities according to the NUVEL–1NNR model (assuming vertical site velocities are zero).

ITRF Site	X/m	Y/m	Z/m	Epoch
DION SLR 7515	4595216.4242	2039435.3299	3912629.5036	1989.44
	4595216.4326	2039435.3461	3912629.4769	1991.78
	4595216.4384	2039435.3572	3912629.4586	1993.39
	4595216.4456	2039435.3709	3912629.4359	1995.38
	4595216.4458	2039435.3713	3912629.4352	1995.44
	4595216.4469	2039435.3735	3912629.4315	1995.76
	4595216.4492	2039435.3779	3912629.4244	1996.39
HERS GPS	4033470.3180	23672.6326	4924301.1261	1989.44
	4033470.2889	23672.6756	4924301.1496	1991.78
	4033470.2690	23672.7051	4924301.1659	1993.39
MADR GPS	4849202.5405	-360329.2644	4114913.0018	1989.44
	4849202.5192	-360329.2171	4114913.0311	1991.78
	4849202.5045	-360329.1846	4114913.0512	1993.39
ONSA GPS	3370658.7969	711876.9114	5349786.8165	1989.44
	3370658.7601	711876.9502	5349786.8346	1991.78
	3370658.7348	711876.9770	5349786.8470	1993.39
WETT GPS	4075578.7152	931852.5493	4801569.9731	1989.44
	4075578.6799	931852.5923	4801569.9947	1991.78
	4075578.6556	931852.6218	4801570.0096	1993.39

Table 2.3: Fiducial station coordinates at campaign epochs, calculated according to the method described in the text.

Fiducial orbit improvement was performed using L3 `gpsest` solutions over 2- to 4-day arcs to ensure stability, holding the fiducial stations fixed and including observations from the stations in the central Greece network. Six initial Keplerian elements per satellite were estimated, plus the direct radiation pressure parameters and Y-biases throughout the arc. The process was repeated until successive orbits converged at the 1 m level, corresponding to a precision of 5 p.p.b..

2.3 Summary

Relative GPS observations can be processed to yield very precise relative positions, as many of the errors affecting GPS receivers are common to all simultaneously-operating receivers in a network. Removal of cycle slips and bad data points from the observations is a laborious but necessary process. The accuracy and precision of the coordinates can be improved by using generally-available post-processed 'precise' satellite orbits, or by generating such orbits explicitly using the fiducial method. The reference frame of the coordinates is determined by the satellite ephemerides, which in turn depend on the coordinates adopted for fixed sites during fiducial orbit improvement. Signal propagation effects (dispersive ionospheric delay and non-dispersive tropospheric delay) and clock errors (especially for receiver clocks) must be considered in order to obtain the most accurate coordinates. If integer ambiguity parameters can be fixed, this will improve the constraints on the final coordinate solution.

Chapter 3

Displacement Analysis Methods

This chapter concerns the techniques that I have used in order to interpret coordinate sets derived from GPS and other geodetic observations. The analysis methods fall into two classes:

- data-driven methods, which are simple representations of coordinate differences in an appropriate reference frame, or strain parameters expressed as smoothly varying fields or for uniformly-straining finite regions. These approaches do not presuppose any particular physical model for the deformation and are essentially means of representing and / or interpolating the observed coordinate differences.
- model-driven methods, in which an attempt is made to relate the observed deformation to a particular dynamic or kinematic model. In this case, if the fit between model and observations is good, inferences can be made about physical properties of the Earth. As such models of necessity comprise simplifications of the real Earth (*e.g.* planar fault surfaces, uniform fault displacement, isotropic elastic materials) it is important not to over-interpret the results of such modelling.

In both cases, care must be taken to ensure that distortions of the reference frames in which coordinates have been measured are not mistaken for tectonic deformation. Also, true deformation of the crust must not be confused with local site deformation, *e.g.* if the marker is unstably situated on poorly-consolidated sediments.

Deformations arising from the 1981 Alkyonides earthquake sequence are used as an example; application of the methods to the Central Greece and Grevena datasets is left until later chapters.

3.1 Displacement vectors and reference frames

Subtracting one set of geocentric Cartesian epoch coordinates from another yields a set of site displacement vectors in the chosen geocentric Cartesian reference frame. For ease of geophysical analysis, these vectors must be projected onto local axes (east, north, up). It is important that both sets of coordinates are in the same reference frame, otherwise the apparent deformation will include a component caused solely by the difference between the two reference frames.

If the coordinates of the fixed station at each epoch have been calculated using the ITRF or NUVEL–1NNR velocity model for that station, then the reference frame for the displacements will in principle be the global no-net-rotation reference frame. However, in practice, there will be errors in the realisation of the reference frame which are equivalent to infinitesimal rotations and translations (and less significantly, strains) of the network from epoch to epoch. The effects of these errors can be removed either by considering only baseline length changes, or by adopting local azimuthal change constraints (*i.e.* to assume that a certain baseline does not rotate) and applying them directly to the displacements (by constraining the positions or velocities of two or more stations at two or more epochs).

If more than two epochs have been observed, the reference frame errors at further epochs can be estimated if the assumption is made that the sites in the network have maintained constant velocities during the total interval of observation. Global translations (and if necessary, rotations) can be applied to the network solution at each epoch such that the deviation of each site coordinate at each epoch from the smoothed value is minimised (using a least-squares or other criterion). Consider a set of local coordinates in the x and y directions $(u_j^{(i)}, v_j^{(i)})$, at epochs $t^{(i)}$, relative to a geodetic position $(u_j^{(0)}, v_j^{(0)})$ at arbitrary epoch $t^{(0)}$, for stations j with geographic coordinates (x_j, y_j) (relative to an arbitrary origin (x_0, y_0)), moving at constant velocities $(\dot{u}_j, \dot{v}_j)$. If the network as a whole is subject to translations $u_0^{(i)}$ and $v_0^{(i)}$, and an infinitesimal rotation $\omega^{(i)}$ at each epoch, then each position observation will obey

$$\begin{pmatrix} u_j^{(i)} \\ v_j^{(i)} \end{pmatrix} = \begin{pmatrix} u_j^{(0)} \\ v_j^{(0)} \end{pmatrix} + \begin{pmatrix} \dot{u}_j \\ \dot{v}_j \end{pmatrix} \cdot \left(t^{(i)} - t^{(0)} \right) + \begin{pmatrix} u_0^{(i)} \\ v_0^{(i)} \end{pmatrix} + \begin{pmatrix} 0 & \omega^{(i)} \\ \omega^{(i)} & 0 \end{pmatrix} \begin{pmatrix} x_j - x_0 \\ y_j - y_0 \end{pmatrix} + \epsilon \qquad (3.1)$$

where ϵ is an error term. The zero-epoch coordinates $(u_j^{(0)}, v_j^{(0)})$ and the site velocities $(\dot{u}_j, \dot{v}_j)$ can be determined simply by linear regression on a site-by-site basis, but the network translation and rotation parameters affect all sites. The model is implemented in 2-D because the inaccuracy of GPS height information for epoch surveys prevents the use of these data. The estimation of network parameters reduces the degree of freedom of the solution by an amount equivalent to the loss of three coordinate data, which is .

I have written the C program `velsmooth` (Appendix C.1) to solve this problem. The network translations and rotations are estimated using a simplex algorithm (Subsection 3.5.2) with a least-squares criterion

for goodness of fit to the observed coordinates, with the site velocities and zero-epoch coordinates being estimated by linear regression within each simplex movement.

3.2 Interpolated smooth velocity and strain rate fields

The velocities measured using GPS are merely samples of the true velocity field (here regarded as a function of horizontal position only) for the region. This velocity field may be smooth or may contain discontinuities. In the absence of large local earthquakes during the period of the experiment, it may be reasonable to suppose that strain in the upper crust has not been localised and the velocity field can therefore be represented over a sufficiently small region by a smooth, continuous function such as a low-order polynomial in latitude and longitude. Slip at depth on faults will increase the order of polynomial required, and so this method provides a model-independent test for this behaviour. The polynomial representations of the strain, rotation and dilatation rate fields can be obtained by analytic differentiation of the polynomial describing the velocity field.

Consider a set of local coordinates in the x and y directions $(u_j^{(i)}, v_j^{(i)})$, at epochs $t^{(i)}$, relative to a geodetic position $(u_j^{(0)}, v_j^{(0)})$ at arbitrary epoch $t^{(0)}$, for stations j with geographic coordinates (x_j, y_j) (relative to an arbitrary origin (x_0, y_0)). If the velocity field remains constant with respect to time, the relationship between the observed coordinates and the coefficients of the polynomials can be expressed as follows:

$$\begin{pmatrix} u_j^{(i)} \\ v_j^{(i)} \end{pmatrix} = \begin{pmatrix} u_j^{(0)} \\ v_j^{(0)} \end{pmatrix} + \begin{pmatrix} F(x_j - x_0, y_j - y_0) \\ G(x_j - x_0, y_j - y_0) \end{pmatrix} \cdot \left(t^{(i)} - t^{(0)} \right) + \epsilon \tag{3.2}$$

where $F(x, y)$ and $G(x, y)$ are given by

$$F(x, y) = a_1 \cdot f_1(x, y) + a_2 \cdot f_2(x, y) + \cdots + a_N \cdot f_N(x, y) \tag{3.3}$$

$$G(x, y) = b_1 \cdot g_1(x, y) + b_2 \cdot g_2(x, y) + \cdots + b_M \cdot g_M(x, y) \tag{3.4}$$

with $f_1 \ldots f_N$ being unique polynomial terms in x and y and $a_1 \ldots a_N$ the unknown coefficients of these terms (and similarly for $g_1 \ldots g_M$ and $b_1 \ldots b_M$), and ϵ is an error term.

Each site must be occupied at least twice, otherwise the problem is under-determined with respect to the 'nuisance' parameters $(u_j^{(0)}, v_j^{(0)})$. However, the observations at different sites need not be simultaneous. As the epochs of observation are known 'perfectly' and the displacements involved are infinitesimal, the problem is linear and can be cast into the matrix form $\mathbf{A}\mathbf{x} = \mathbf{y}$ where $\mathbf{x}$ is the vector of unknown coefficients and nuisance parameters, $\mathbf{y}$ is the vector of observations, and $\mathbf{A}$ is the design matrix of known constants (dependent on the epochs of observation and the geographic coordinates of the sites).

The number of coefficients N (and similarly M) can be less than, equal to or greater than the number of sites. If it is less than or equal to the number of sites, then the problem is overdetermined or even-

determined and can always be solved, given a well-constructed network. However, some coefficients may be better constrained than others, and in the extreme case of N greater than the number of sites, some coefficients may be completely undetermined. Singular value decomposition (SVD) of the design matrix $\mathbf{A}$ will determine which coefficients are well determined by a particular network and which are poorly determined (*e.g.* Press *et al.*, 1992). Poorly-determined coefficients can then be eliminated from the solution, and the rest (now constituting an overdetermined set) obtained by back-substitution. The singular values can also be used to determine which (usually higher-order term) parameters are degrading rather than improving the fit.

To simplify the problem, the $a_1 \ldots a_N$ and $b_1 \ldots b_M$ can be regarded separately as SVD will only work with a diagonal observation covariance matrix (Menke, 1984), so the north and east displacements will be uncorrelated. I have written the C program `svdvel2` (Appendix C.2) to implement this algorithm.

3.3 Multi-epoch finite-element strain analysis

Alternatively, the strain rate can be regarded as uniform within polygonal regions bounded by geodetic monuments, and constant with respect to time. In this case, no assumptions are made regarding the continuity of displacement between regions (*c.f.* Frank, 1966; Welsch, 1983).

Consider a set of local coordinates in the x and y directions $(u_j^{(i)}, v_j^{(i)})$, at epochs $t^{(i)}$, relative to a displacement $(u_j^{(0)}, v_j^{(0)})$ at arbitrary epoch $t^{(0)}$, for stations j with geographic coordinates (x_j, y_j) (relative to an arbitrary origin (x_0, y_0)), bounding a polygonal region. Each coordinate must obey the equation

$$\begin{pmatrix} u_j^{(i)} \\ v_j^{(i)} \end{pmatrix} = \begin{pmatrix} u_j^{(0)} \\ v_j^{(0)} \end{pmatrix} + \begin{pmatrix} \dot{u} \\ \dot{v} \end{pmatrix} \cdot \left(t^{(i)} - t^{(0)} \right) + \begin{pmatrix} \frac{\partial \dot{u}}{\partial x} & \frac{\partial \dot{u}}{\partial y} \\ \frac{\partial \dot{v}}{\partial x} & \frac{\partial \dot{v}}{\partial y} \end{pmatrix} \begin{pmatrix} x_j - x_0 \\ y_j - y_0 \end{pmatrix} \cdot \left(t^{(i)} - t^{(0)} \right) + \epsilon \tag{3.5}$$

or in vector notation

$$\mathbf{u}_j^{(i)} = \mathbf{u}_j^{(0)} + \dot{\mathbf{u}} \cdot t + \mathbf{L}\mathbf{x} \cdot t + \epsilon \tag{3.6}$$

where $(\dot{u}, \dot{v})$ represents the rate of translation of the region as a whole, with the dot denoting differentiation with respect to time throughout. ϵ represents an error term. There are six unknown parameters in this equation: two representing translation of the region as a rigid body ($\dot{u}$ and $\dot{v}$), four components of the velocity gradient tensor $\mathbf{L}$ ($\frac{\partial \dot{u}}{\partial x}$, $\frac{\partial \dot{u}}{\partial y}$, $\frac{\partial \dot{v}}{\partial x}$ and $\frac{\partial \dot{v}}{\partial y}$). In addition, two 'nuisance' parameters per station ($u_j^{(0)}$ and $v_j^{(0)}$), which represent the initial position of the region, must be estimated. Each occupation of a site yields two observations (horizontal coordinates), and $t^{(0)}$, x_0 and y_0 can be chosen arbitrarily. Thus a minimum of three sites bounding a non-degenerate triangle, with repeated occupations at each site, is necessary to solve for the six strain parameters. Further repeat occupations at any of the three sites, or repeat occupations at other sites extending the polygon, will introduce redundancy and the equations can

then be solved using weighted linear least squares with the full covariance matrix of the observations, by the method of normal equations (*e.g.* Cross, 1983).

This formulation of the velocity gradient – displacement relationship does not require any of the site occupations to be simultaneous, so results from different surveys can be combined at will. In practice, however, problems may arise from the reference frame definition if care is not taken when combining several survey epochs.

If a linear (matrix) relationship $\mathbf{A}$ exists between parameters $\mathbf{x}$ and observations $\mathbf{y}$ such that $\mathbf{y} = \mathbf{A} \cdot \mathbf{x}$, then the covariance matrix $\mathbf{C}_x$ of the parameters will be related to the covariance matrix of the observations $\mathbf{C}_y$ by Gauss' law of error propagation $\mathbf{C}_x = \mathbf{A}^{-1} \cdot \mathbf{C}_y \cdot \left(\mathbf{A}^T\right)^{-1}$ (*e.g.* Cross, 1983). This is the case for the above velocity gradient parameters. Further strain parameters can be derived (Section 3.4) which do not have a linear relationship with the velocity gradient tensor, but the relationship can be linearised in the region of the solution (Welsch, 1983), so error estimates can be made for all parameters, although linearisation may bias the error estimates of these parameters.

I have written the C program `polystr2` (Appendix C.3) to implement the above algorithm.

3.4 Derived products of the velocity gradient tensor

The velocity gradient tensor $\mathbf{L}$ derived in Equation 3.5 or by differentiation of the smoothed velocity field in Equation 3.2 can be separated into a symmetric part $\mathbf{S}$ and an antisymmetric part $\mathbf{A}$, where

$$\mathbf{S} = \frac{1}{2}\left(\mathbf{L} + \mathbf{L}^T\right) = \begin{pmatrix} \frac{\partial \dot{u}}{\partial x} & \frac{1}{2}\left(\frac{\partial \dot{u}}{\partial y} + \frac{\partial \dot{v}}{\partial x}\right) \\ \frac{1}{2}\left(\frac{\partial \dot{u}}{\partial y} + \frac{\partial \dot{v}}{\partial x}\right) & \frac{\partial \dot{v}}{\partial y} \end{pmatrix} = \begin{pmatrix} \dot{e}_{xx} & \dot{e}_{xy} \\ \dot{e}_{xy} & \dot{e}_{yy} \end{pmatrix} \tag{3.7}$$

$$\mathbf{A} = \frac{1}{2}\left(\mathbf{L} - \mathbf{L}^T\right) = \begin{pmatrix} 0 & \frac{1}{2}\left(\frac{\partial \dot{u}}{\partial y} - \frac{\partial \dot{v}}{\partial x}\right) \\ -\frac{1}{2}\left(\frac{\partial \dot{u}}{\partial y} - \frac{\partial \dot{v}}{\partial x}\right) & 0 \end{pmatrix} = \begin{pmatrix} 0 & \dot{\omega} \\ -\dot{\omega} & 0 \end{pmatrix} \tag{3.8}$$

In these equations, $\dot{e}_{xx}$, $\dot{e}_{yy}$ and $\dot{e}_{xy}$ are independent of the absolute orientation (and rotation) of the network (for infinitesimal strains) but depend on the absolute length scale of the network, whereas $\dot{\omega}$ represents the rigid-body rotation rate of the region and is independent of scale but depends on absolute orientation (which is commonly lacking for terrestrial geodetic networks). $\mathbf{S}$ can be related to the moment tensors of earthquakes using the result of Kostrov (1974), but $\mathbf{A}$ (*i.e.* $\dot{\omega}$) cannot, so comparisons of regional seismicity with GPS displacements must be made using $\mathbf{S}$. However, the relative rates of rotation between neighbouring parts of the region can be obtained from the variation of $\mathbf{S}$ within the region using the relationship of Haines & Holt (1993).

$\mathbf{S}$ can be further decomposed into two components of shear strain rate ($\dot{\gamma}_1$, $\dot{\gamma}_2$) and one of dilatation rate

$(\dot{\Delta})$:

$$\mathbf{S} = \frac{1}{2}\begin{pmatrix} \dot{\Delta} - \dot{\gamma}_1 & \dot{\gamma}_2 \\ \dot{\gamma}_2 & \dot{\Delta} + \dot{\gamma}_1 \end{pmatrix} \tag{3.9}$$

where

$$\dot{\gamma}_1 = \dot{e}_{yy} - \dot{e}_{xx} \quad ; \quad \dot{\gamma}_2 = 2\dot{e}_{xy} \quad ; \quad \dot{\Delta} = \dot{e}_{xx} + \dot{e}_{yy} \tag{3.10}$$

$\dot{\gamma}_1$ and $\dot{\gamma}_2$ are referred to as the pure and engineering shear strain rates respectively, and are independent of both scale and orientation of the network. They are thus the only quantities that can be used when analysing triangulation-only networks unless geological knowledge can be used to make scale and orientation assumptions. The rate of total shear, $\dot{,}$ can be derived from $\dot{\gamma}_1$ and $\dot{\gamma}_2$ by

$$\dot{,}^2 = \dot{\gamma}_1^2 + \dot{\gamma}_2^2 \tag{3.11}$$

and is easily related to the eigenvalues of $\mathbf{S}$, $\dot{e}_1$ and $\dot{e}_2$ (the principal strain rates), which correspond to the directions of greatest and least extensional strain rate respectively (in which no shear strain takes place).

$$e_1 = \frac{1}{2}(\dot{\Delta} + \dot{,}) \quad ; \quad e_2 = \frac{1}{2}(\dot{\Delta} - \dot{,}) \tag{3.12}$$

The azimuth of $\dot{e}_1$, θ, is given by

$$\theta = \frac{1}{2}\tan^{-1}\left(\frac{\dot{\gamma}_2}{\dot{\gamma}_1}\right) \tag{3.13}$$

The direction of greatest shear strain rate is mid-way between that of $\dot{e}_1$ and that of $\dot{e}_2$ (which are perpendicular).

3.5 Co-seismic elastic deformation

In an earthquake, strain is localised from a region into a fault, and much of the resulting energy is released as teleseismic waves, thus demonstrating that the Earth behaves elastically, at least over short time-scales. It is therefore reasonable to model the permanent co-seismic deformation in the region of the fault in terms of static-elastic dislocation theory.

3.5.1 Theory

The original application of the Cauchy and Poisson elasticity equations to dislocation theory was done by Volterra (1907), and the integral equations he obtained have since been simplified (*e.g.* Steketee, 1958; Press, 1965; Okada, 1985). I use here the formulation of Okada (1985) which gives explicit solutions

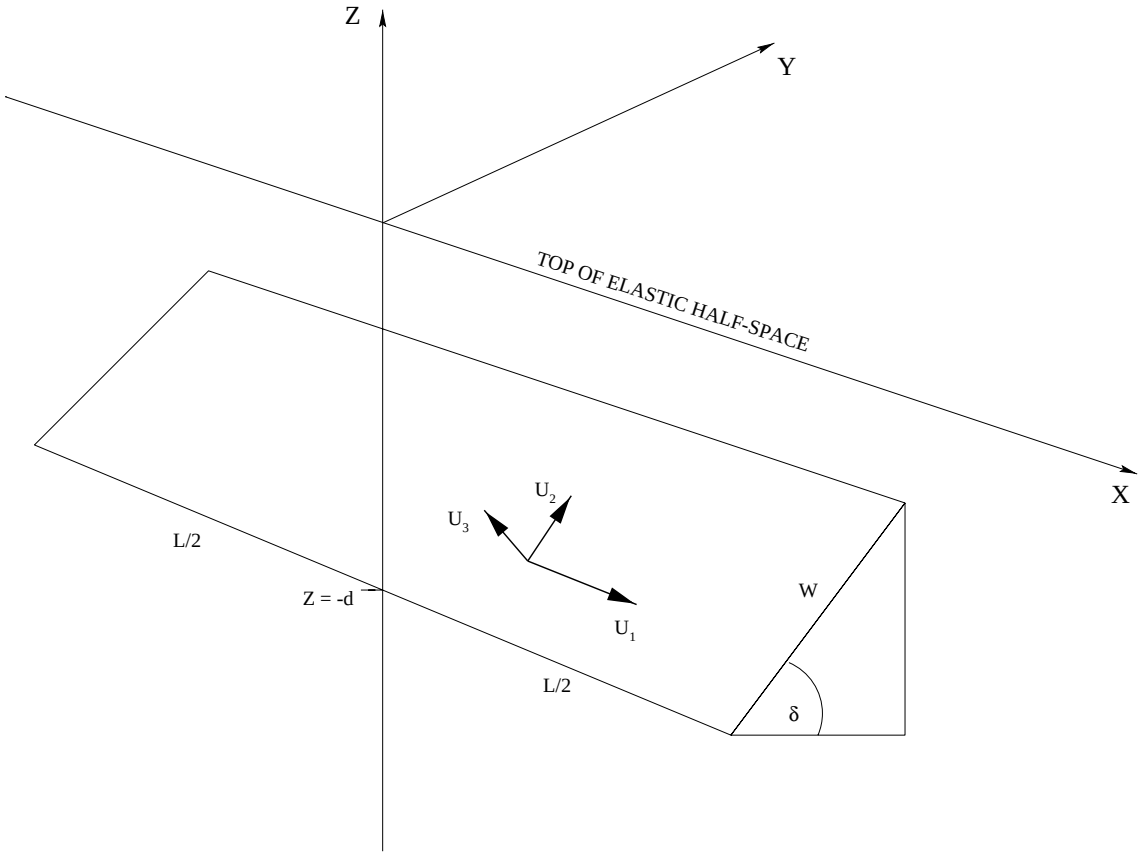

Figure 3.1: Cartoon showing the reference frame and notation used by Okada (1985).

of Steketee's (1958) integral formula for the displacement field Green's function $u_i^j(x_1, x_2, x_3)$ due to a dislocation $\Delta u_j(\xi_1, \xi_2, \xi_3)$ across a plane rectangular surface Σ in an isotropic half-space, caused by a unit point force in direction j at (ξ_1, ξ_2, ξ_3). Summing over components of the dislocation:

$$u_i = \iint_{\Sigma} \Delta u_j \left[\lambda \, \delta_{jk} \, \frac{\partial u_i^n}{\partial \xi_n} \; + \; \mu \left(\frac{\partial u_i^j}{\partial \xi_k} + \frac{\partial u_i^k}{\partial \xi_j} \right) \right] \, v_k \, d\Sigma \tag{3.14}$$

where λ and μ are Lame's constants, δ_{jk} is the Kronecker delta and v_k are the direction cosines of the normal to $d\Sigma$. The coordinate system is illustrated in Figure 3.1. For the full explicitly-integrated expressions see Okada (1985).

The reference frame implied by Okada's formulæ is one in which the average displacement on the fault plane (both upper and lower surfaces) is zero. If the effects of multiple dislocations within a region are summed, then the reference frame will no longer have any fault plane with zero average displacement. However this can be achieved by an appropriate translation, as the strains involved are infinitesimal so the principle of superposition applies. In reality, displacements will be observed in a reference frame which differs from this by translation and rigid-body rotation of the region in which the dislocations occur, and

which may have errors of scale.

3.5.2 Inversion for earthquake source parameters from geodetic data

It is clear from Okada's formulæ that the displacements due to an event do not depend linearly on the source parameters of that event (here taken to be strike, dip, rake, magnitude of slip, position and length of fault scarp, minimum and maximum vertical depth of the dislocation surface). Further nonlinearity arises when we must also solve for the translation, rotation and scale change of the reference frame.

The 'goodness' of a particular solution can be described by a penalty function which quantifies the difference between the observed deformation and that predicted by the model. For example, for N sites with measured displacements (u_1^i, u_2^i, u_3^i) and computed displacements $(u_1^{i\prime}, u_2^{i\prime}, u_3^{i\prime})$, possible penalty functions f include:

$$f = \frac{1}{N} \sum_{i,j} \frac{|u_j^i - u_j^{i\prime}|}{\sigma_j^i} \qquad L_1 \text{ norm} \tag{3.15}$$

$$f = \sqrt{\frac{1}{N} \sum_{i,j} \frac{(u_j^i - u_j^{i\prime})^2}{\sigma_j^{i\,2}}} \qquad L_2 \text{ norm ('least squares')} \tag{3.16}$$

$$f = \max \frac{|u_j^i - u_j^{i\prime}|}{\sigma_j^i} \qquad L_\infty \text{ norm} \tag{3.17}$$

where $1 \leq i \leq N, 1 \leq j \leq 3$ throughout, and σ_j^i is the weight associated with the measurement u_j^i. The L_2 norm can also be expressed using the full variance-covariance matrix of the observations. The choice of penalty function will depend to some extent on the quality of the dataset with respect to 'outlier' measurements. The L_∞ norm is highly sensitive to even a single outlier, whereas the L_1 norm is relatively insensitive to a few large outliers. The sensitivity of the L_2 norm lies somewhere in between.

The 'best' solution is that which has the smallest value for the penalty function (although this solution may be rejected as physically meaningless). For a linear or mildly non-linear problem, there will only be one minimum of the penalty function within parameter space, but for a strongly nonlinear problem there may be several minima. Normal least-squares methods can be adapted to mildly nonlinear problems by using iteration, but for more serious cases another method must be sought. In their work on the 1989 Loma Prieta earthquake data, Williams *et al.* (1993) use a quasi-Newtonian method which relies on the ability to calculate the second differentials of the penalty function. Likewise, Briole *et al.* (1986), in their study of the 1976 – 1977 Friuli earthquake sequence, use the Tarantola – Valette formalism (Tarantola & Valette, 1982), which is a generalisation of this approach using *a priori* covariance information. Ward and Barrientos (1986) use a basic downhill approach which overcomes this limitation, but is still vulnerable to local minima. This study follows Curtis (1994) in the use of a downhill simplex method (Nelder & Mead, 1965; from Press *et al.*, 1992), which does not require any derivatives, and is more robust as it samples parameter space more widely during the course of the inversion and so is more likely to find a

global minimum eventually.

A simplex is a multi-dimensional polyhedron in parameter space, with one more vertex than the number of parameters to be estimated. The initial vertices are chosen by adopting a suitable starting point (*e.g.* the best known *a priori* values of the parameters) as one vertex, then perturbing each parameter in turn to generate another vertex. Each vertex has a penalty associated with it, and the process of inversion consists of a series of moves whereby the 'worst' vertex is replaced by a new vertex calculated according to certain rules:

- The simplex is reflected by projecting the worst vertex through the opposite face.
- If the projected vertex is worse than the old worst vertex, an attempt is made to enlarge the simplex further in the direction of the projected vertex.
- If this projected vertex is worse than the old worst vertex, the simplex is uniaxially contracted by projecting the old worst vertex toward (but not through) the opposing face.
- If the latter step does not yield a better vertex, the simplex is contracted by projecting all vertices towards the best vertex.

See Press *et al.* (1992), chapter 10 for a fuller explanation.

When the penalties of all the vertices converge within a predefined fractional tolerance of each other, a minimum has been found (the vertices will be virtually coincident at this point because the chance of convergence of penalties for widely separated vertices is infinitesimal). If the *a priori* estimates of the parameters were sufficiently close to their 'true' values, and the observations are unbiased, this minimum should be the global minimum desired. If either of these conditions is not met, however, the minimum may only be local, unless the simplex has 'explored' enough of parameter space to be able to locate the global minimum. Curtis (1994) attempts to obtain good *a priori* parameters by performing a linearised least squares inversion before initialising the simplex. In this study, I adopt a different approach, which is to perform many simplex inversions with different starting simplexes, and thus to sample parameter space more extensively. In Subsection 3.5.5 below, I will make a comparison between methods involving large and small starting simplexes.

During the simplex inversion, checks must be made so that the simplex vertices remain in physically reasonable parts of parameter space (*e.g.* the fault must lie wholly within the elastic half-space). I have written the C program `okinv` (Appendix C.4) to implement the above algorithm for L_1, L_2 and L_∞ norms, with the option of weighted or unweighted penalty functions in all cases, and full covariance or variance-only weighting in the case of the L_2 norm.

3.5.3 Example: Inversion using a synthetic dataset

The program `okinv` has been tested using a synthetic dataset based on the CMT solution (J.H. Woodhouse, pers. comm.) for the 13 May 1995 Grevena earthquake. The station distribution was chosen to match that of the Grevena co-seismic GPS network (Chapter 6). Gaussian noise with a standard deviation of 0.05 m (comparable with the estimates of the actual accuracy) was added to the site displacements, and in addition, a few large 'outliers' were inserted.

Inversion was performed with starting parameters chosen from within very wide limits of the known values. At first, only horizontal observations were used; r.m.s. residuals were slightly lower than the level of noise added to the dataset. Strike, dip and rake were recovered to within 5°, moment to within 15%, and scarp position and length to within 1 km. Minimum depth was recovered to within 0.25 km, and maximum depth to within 1 km. Also, spurious translations of a few mm and scale and rotation parameters of less than 0.5 p.p.m. were generated.

When vertical displacements were added to the dataset, control on fault dip and minimum and maximum depths were greatly improved, and the spurious Helmert parameters were reduced in magnitude. Other parameters were also recovered better.

Further discussion of inversions from synthetic datasets (in the context of solution stability) will be made in Chapter 6

3.5.4 Example: The 04 March 1981 Platea – Kaparelli event

On 24 and 25 February and 04 March 1981, three large earthquakes of M_s 6.7, 6.4, 6.2 (Ambraseys & Jackson, 1990) struck the Alkyonides Gulf in the eastern Gulf of Korinthos (Figure 3.2). Jackson *et al.* (1982b) made an initial study of the seismic and geomorphological data, including details of the fault scarps that moved in the events, and King *et al.* (1985a) interpreted the aftershock sequence. Their conclusions were later revised (Taymaz *et al.*, 1991) in the light of more accurate relocations of the events, relative to the reference aftershocks of 05 March and 07 March which were observed by temporary local seismic networks.

Taymaz *et al.* (1991) conclude that the 24 and 25 February events moved faults to the north of Mt. Gerania and on the coast between Skinos and Alepohori respectively (Figure 3.3), although, because the events occurred within the same night (and so fault breaks were not observed occurring individually) and may also have moved offshore faults, these conclusions are not definite. The 04 March (daytime) event generated movements observed by local inhabitants on two neighbouring surface scarps to the north of the Gulf. For the latter event, the slip vectors observed at the surface (Jackson *et al.*, 1982b) are in disagreement

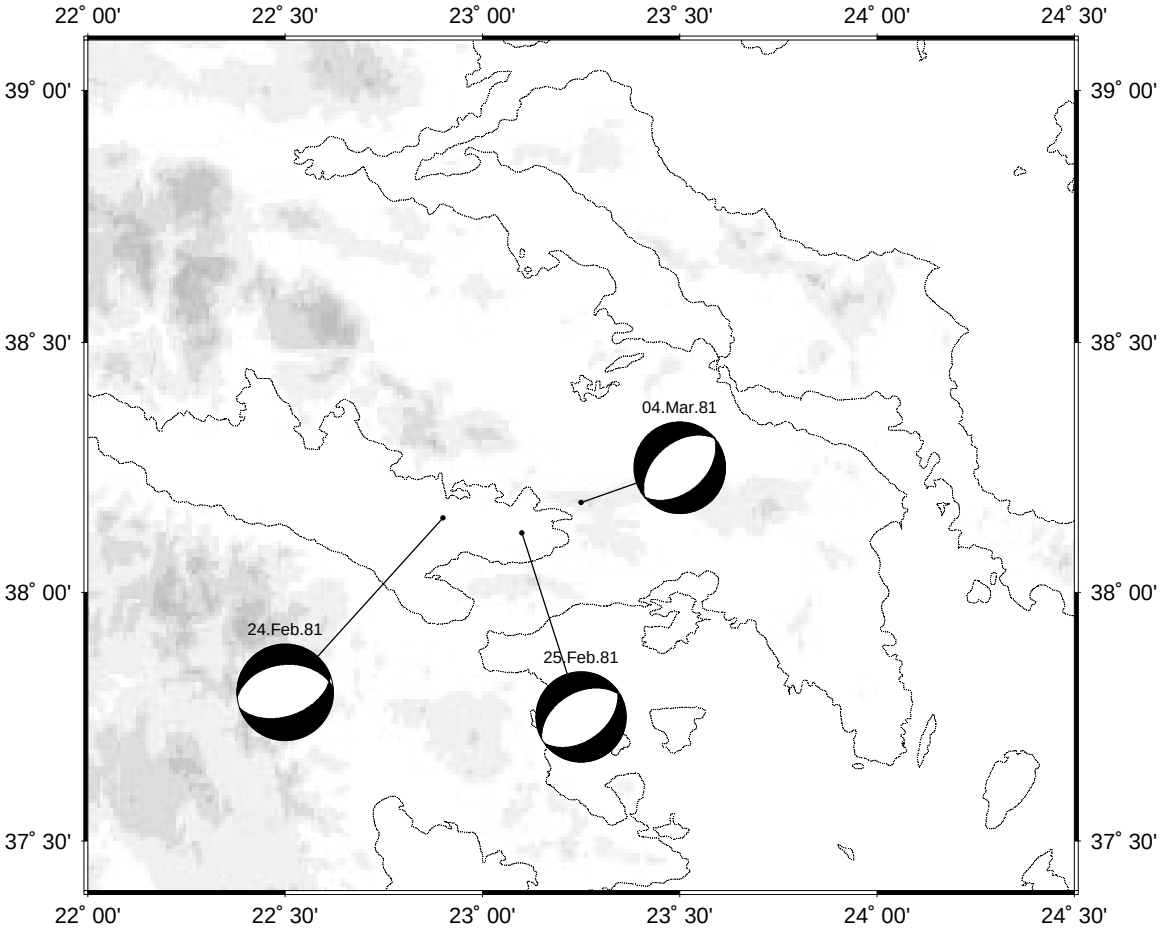

Figure 3.2: Epicentral locations of the 1981 Alkyonides earthquakes. Topographic shading increases at 500 m vertical intervals.

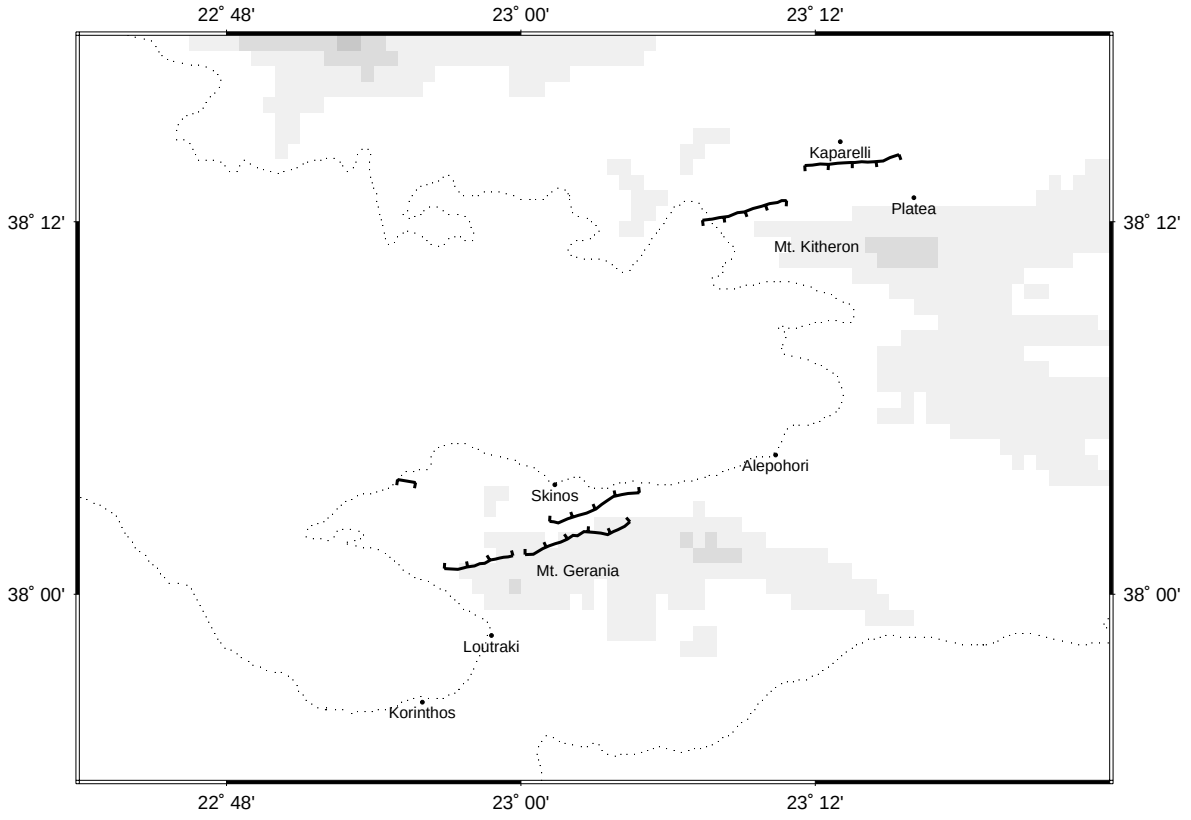

Figure 3.3: Fault scarp locations of of the 1981 Alkyonides earthquakes (after Jackson *et al.*, 1982b and Taymaz *et al.*, 1991), and geographic locations referred to in the text. Ticks are shown on the downthrown side of the fault. Topographic shading increases at 500 m vertical intervals.

with those implied by the focal mechanism, and the CMT solution (Dziewonski & Woodhouse, 1983; Dziewonski *et al.*, 1988) includes a large (42%) non-double-couple component (Ekström & England, 1989), although this may not be resolved for such a small event. Both of these observations imply that the faulting mechanism was complex, but long-period seismology cannot provide the resolution necessary to constrain the exact mechanism. Abercrombie *et al.* (1995) have attempted to resolve the problem using teleseismic broad-band data recorded by seismometer arrays, and prefer a twin-source mechanism.

The region of the Gulf was surveyed in 1969 by the Hellenic Army Geographic Service (HAGS) as part of a general re-survey of Greece. In 1981, a few months after the Alkyonides earthquakes, the locality of the 04 March event, and also a few points to the south and east of the Gulf, were resurveyed by HAGS. From the first survey, only angle measurements are available (there is no nearby scale control), but in the second survey, 15 Geodimeter distance measurements were also taken (Galanis, 1993; Kagiadakis, 1995). However, for the purposes of this example, and in order to be able to compare results with Curtis (1994), I have used only the angle measurements from 1981 and made assumptions with regard to scale.

Date	Scarp		M_0	Strike	Dip	Rake	Length	Displacement /m	
	Lat.	Lon.	/Nm$\times 10^{18}$				/km	Dip-slip	Strike-slip
24 Feb	38.03	23.00	8.75	264	42	-80	12	1.5	-0.3
25 Feb	38.07	23.10	3.97	260	44	-85	12	0.7	-0.1
04 March	38.22	23.19	2.70	75	45	-90	12	0.5	0.0

Table 3.1: *A priori* parameters for the Alkyonides earthquake sequence (from Taymaz *et al.*, 1991).

In June 1991, eight stations from the 1981 network of 29 sites were reoccupied with Ashtech dual-frequency GPS instruments. The measurements were processed using NGS precise ephemerides, with the station at Makri Plagi (MKPG, station CG48 of the central Greece network) held fixed at its October 1991 value (Subsection 4.1.2). In order to calculate site displacements from 1981 – 1991, an assumption about the orientation and scale of the network in 1981 must be made. I have assumed that post-seismic displacements in the interval 1981 – 1991 are small, so that the baseline MKPG – PGNZ can be constrained to its 1991 azimuth, and MKPG – MYTK to its 1991 length.

In order to process the 1969 triangulation measurements, both scale and orientation assumptions are required. As the networks do not include any stations in the far field of the 1981 earthquakes, all stations will have been affected by the events. Also, all sites will have been displaced by regional crustal strain in the 12-year period, but for small networks close to the fault this is relatively unimportant relative to the co-seismic strain. However, it is possible to determine the baseline lengths and azimuths that will have been least affected by the earthquakes using a forward model. In this the teleseismic focal mechanisms (Taymaz *et al.*, 1991) and scarp locations for the 24 and 25 February and 04 March earthquakes were used (Table 3.1). The baseline length MKPG–MYTK and azimuth MKPG–PGNZ were held fixed at the 1991 GPS values.

The displacements obtained are shown in Figure 3.4. In practice, the program `okinv` will invert for global scale change and rotation in the 1969–81 displacements, so the assumptions have minimal effect. The 24 and 25 February events have the largest effects, especially on the stations to the south, so the forward model for these events must be included in any inversion. The parameters estimated are: strike, dip, rake, magnitude of slip, fault scarp latitude and longitude, fault length, and two translation parameters (to relate the reference frame of the geodetic observations, in which MKPG is fixed, to that of the Okada model, in which the displacements on the fault plane average to zero and displacements are zero at infinite distance from the fault). It is assumed that the fault extends from the surface to 10 km depth. The L_2 norm is used, weighted with the diagonal elements of the covariance matrix (*i.e.* variances). Only those observations in the neighbourhood of the 04 March fault break (24 out of 29 sites) were used.

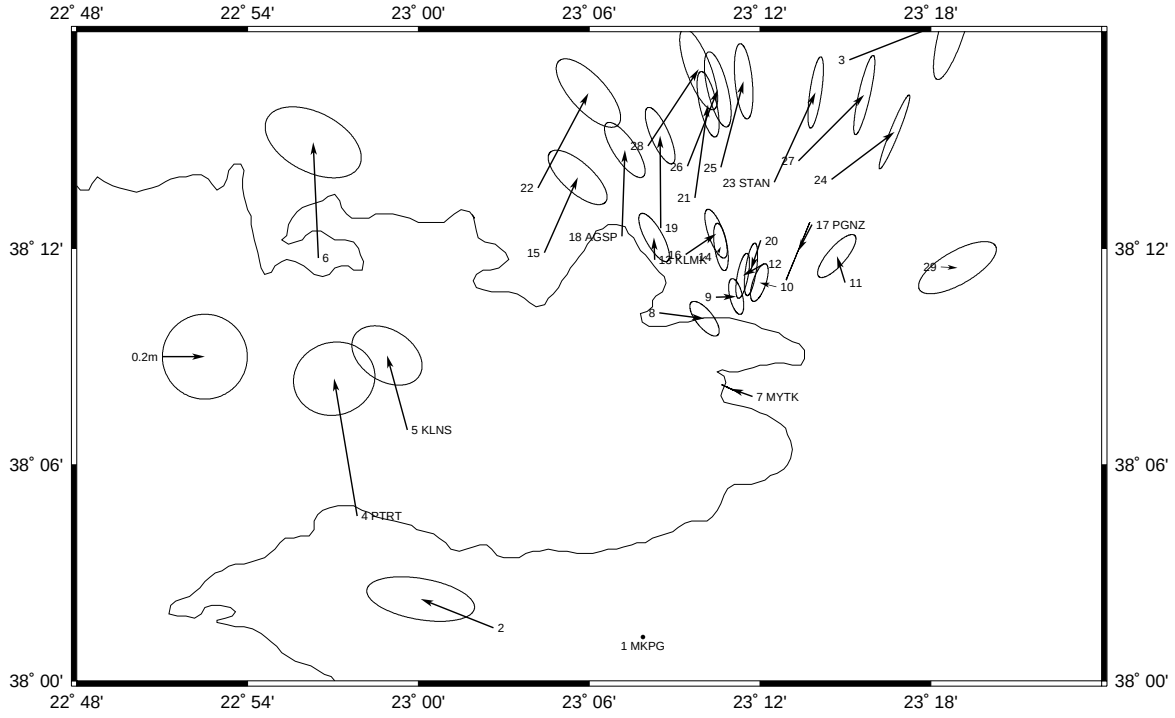

Figure 3.4: Geodetic displacements of monuments in the Alkyonides Gulf, 1969–81 (black arrows), calculated subject to the scale and rotation assumptions described in the text. Ellipses show 68% confidence limits of displacement observations.

Date	Scarp		M_0	Strike	Dip	Rake	Length	Displacement /m	
	Lat.	Lon.	/Nm$\times 10^{18}$				/km	Dip-slip	Strike-slip
04 March	38.20	23.17	5.4	77	62	−99	9.3	1.6	0.2

Total penalty 52.1, r.m.s. weighted residual 1.04.

Table 3.2: Single-event model for the 04 March 1981 Platea–Kaparelli earthquake, without scale or rotation parameters estimated.

Date	Scarp		M_0	Strike	Dip	Rake	Length	Displacement /m	
	Lat.	Lon.	/Nm$\times 10^{18}$				/km	Dip-slip	Strike-slip
04 March	38.20	23.18	4.6	73	59	-100	7.8	1.5	0.3

Scale change 4 p.p.m., rotation -14 p.p.m. (*i.e.* clockwise).
Total penalty 31.7, r.m.s. weighted residual 0.81.

Table 3.3: Single-event model for the 04 March 1981 Platea–Kaparelli earthquake, with scale and rotation parameters estimated.

Curtis (1994) has already inverted for the 04 March earthquake parameters from this geodetic dataset, and puts forward a plausible twin-source mechanism which matches the data to within 1.39σ r.m.s., significantly better than his best-fitting single-source mechanism (1.88σ r.m.s.). However, because of the repeated 'Monte-Carlo' restarting algorithm, `okinv` has identified a single-source solution (Table 3.2) that matches the data within 1.04σ, using exactly the same dataset and without solving for scale or rotation parameters. If scale and rotation parameters are included in the inversion, the best single-source solution found by `okinv` (Table 3.3, Figure 3.5) matches the data to within 0.81σ. This solution matches the surface faulting fairly well, although the fault length (8 km) is somewhat less than that mapped by Jackson *et al.* (1982b), and the dip is steeper. It is not suprising that the 'geodetic' fault length is shorter, because the Okada model assumes constant displacement on the fault, whereas in reality the displacement will decrease towards the ends of the fault. The resulting slip on the fault agrees well in direction with the mapped slip vectors, but the model displacements are slightly larger.

An attempt was made to perform a two-fault solution with `okinv`. A full inversion only matched the data to within 0.71σ, not significantly better than the single-source solution, and the fault scarp locations bear no relation to the mapped faults. Fixing the fault strike, location or length only resulted in worse solutions, again with the resulting estimated parameters being incompatible with surface observations.

3.5.5 Sensitivity to initial parameters

As the inverse problem is highly nonlinear, there are many minima into which the simplex could collapse. Some of these may be very close to the *a priori* parameters, but others will be some distance away and will not be detected unless either a large starting simplex is used, or the simplex is started somewhere else. In trials with different simplex sizes and different ranges of starting places, using the Alkyonides dataset, I have found that large starting simplexes are likely to find good minima even when the starting parameters are far from the true values, but small starting simplexes may easily get stuck far from the

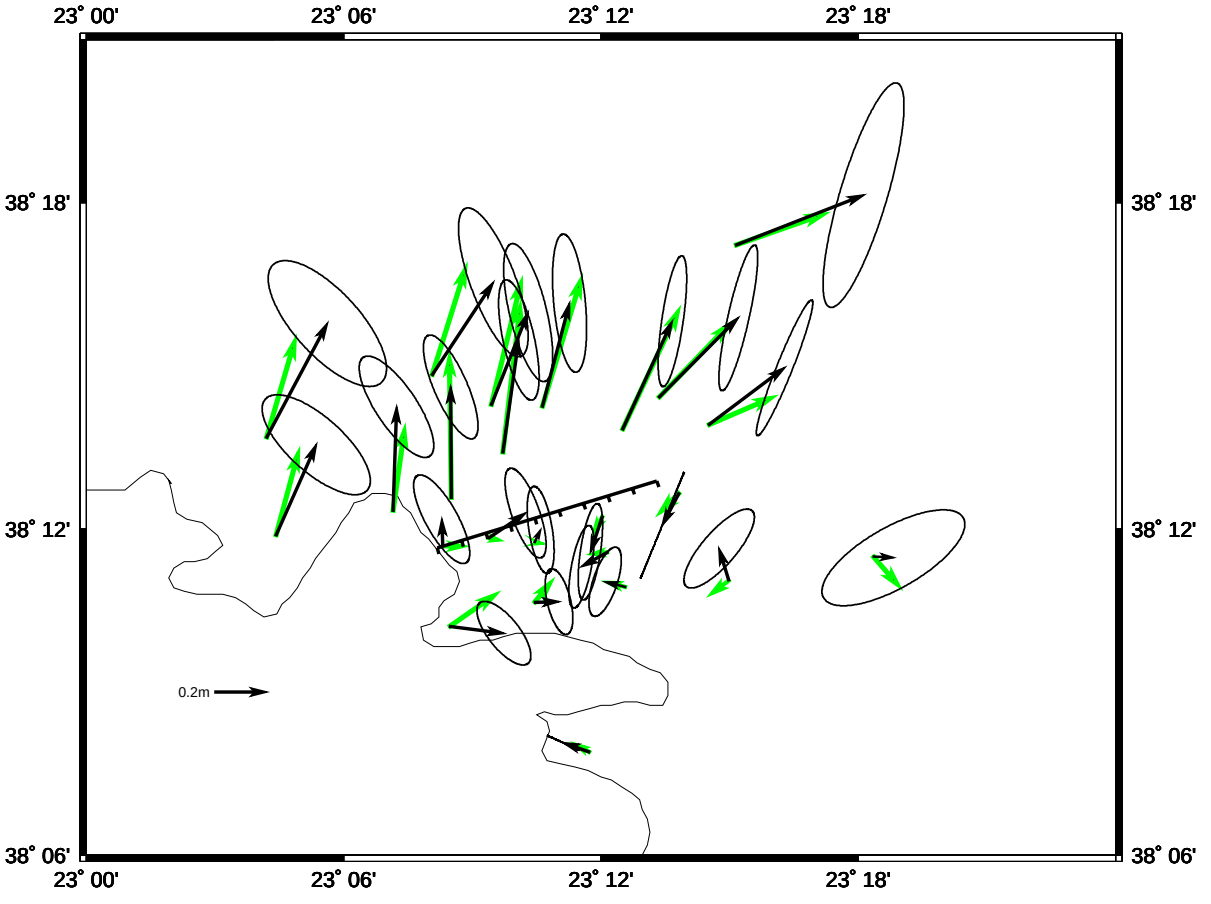

Figure 3.5: Single-event model for the 04 March 1981 Platea–Kaparelli earthquake, with scale and rotation parameters estimated. Observed geodetic displacements are shown in black with error ellipses, and model displacements are shown in green. The fault scarp is shown as a thick line with ticks on the downthrown side.

global minimum. However, many of the local minima close to the global minimum have penalty values far higher than the global minimum, which will lead to misplaced lack of confidence about the solution, so in order to find the global minimum the problem must be restarted many (hundreds) of times. As `okinv` is very fast, this can easily be achieved. Otherwise, the false high 'minimum' found will encourage inversion for extra parameters (*e.g.* more fault segments, variable-slip models) which will result in over-fitting of the data.

3.6 Post-seismic buried slip

Aseismic creep on a buried fault or narrow shear zone is a mechanism by which stresses may relax after an earthquake, particularly in the short term (Savage *et al.*, 1994; Massonnet *et al.*, 1996). As the strain at depth is localised, if the shear zone is near the surface, the surface deformation will be more localised than for the visco-elastic half-space model (Section 3.7). However, for a deeper shear zone the blanketing effect of the material above will delocalise the deformation so the half-space and shear zone models cannot be distinguished.

Shear zone slip can be modelled using the Okada (1985) fault model described in Subsection 3.5. However, this model is static rather than dynamic, and so no material properties can be deduced from the time-dependence of slip when it is modelled in this way. Also, as there is no independent estimate of the amount of slip or location of the buried fault, the problem is under-determined and observed displacements can be precisely modelled by any number of plausible fault–slip combinations.

3.7 Post-seismic visco-elastic deformation

The instantaneous release of stress in the elastic upper crust by means of an earthquake will load the lower crust and upper mantle with elastic stresses, and will redistribute stress within the upper crust. Over a long time, the stresses may be to some extent relaxed by creep in the ductile lower crust. If the creep is distributed within the lower crust, it can be modelled as visco-elastic behaviour of the lower crust, while the upper crust (and possibly upper mantle) remain elastic. When dealing with initial disturbance over a length-scale similar to the thickness of the upper crust, the response will occur over a similar length-scale, and the behaviour of the upper mantle and other remote parts will be less important. Thus, the system can be approximated as an elastic layer overlying a visco-elastic half-space.

3.7.1 Theory

Rundle (1978) developed the original analytic expressions for surface displacements due to relaxation of a visco-elastic half-space overlain by an elastic layer, when perturbed by a point source, and the model has later been extended to include finite fault planes and the effects of gravity (Rundle, 1980; Rundle, 1981; Rundle, 1982; Fernandez *et al.*, 1996; Yu *et al.*, 1996). Other similar models have also been developed (Cohen, 1980; Pollitz, 1992; Ma & Kusznir, 1995).

The elastic-gravitational equations (Love, 1911)

$$\nabla^2\mathbf{u} + \frac{1}{1-2\sigma}\nabla\nabla\cdot\mathbf{u} + \frac{\rho_0 g}{\mu}\nabla\left(\mathbf{u}\cdot\mathbf{e}_z\right) - \frac{\rho_0}{\mu}\nabla\phi - \frac{\rho_0 g}{\mu}\mathbf{e}_z\nabla\cdot\mathbf{u} = 0 \tag{3.18}$$

$$\nabla^2\phi = -4\pi\rho_0 G_0\nabla\cdot\mathbf{u} \tag{3.19}$$

are solved for the case of a finite dislocation within a layered elastic half-space, by a Green's function method (Singh, 1970). The effect of self-gravitation is small (Rundle, 1981) and can be ignored (*i.e.* $G_0 = 0, \phi = \text{const}$). The displacements are propagated between layers using the Thomson-Haskell layer matrix approach (Singh, 1970), with the dislocation lying within the uppermost layer.

The correspondence principle of linear visco-elasticity (see Fung, 1965) is then used to introduce time-dependent elasticity to the lower layer to make it visco-elastic, *i.e.* λ and μ are replaced in the elastic expression for $\mathbf{u}$ by $s\bar{\lambda}(s)$ and $s\bar{\mu}(s)$ to give $\bar{\mathbf{u}}(s)$ where the bar signifies the Laplace transform and $\mathbf{u}(t)$ is the time-varying displacement in the visco-elastic case. Maxwell rheology is used for the shear modulus μ, whereas λ is kept constant, *i.e.*

$$s\bar{\mu}(s) = \frac{s\mu_0}{s + 2/\tau_0} \quad ; \quad s\bar{\lambda}(s) = s\lambda_0 = s\mu_0 \tag{3.20}$$

where $\tau_0 = 2\eta/\mu_0$ is the Maxwell time of the system and η is the effective viscosity. Variation of $s\bar{\lambda}(s)$ such that the bulk modulus is kept constant gives virtually identical results (Thatcher & Rundle, 1979), so is not implemented. As $\lambda_0 = \mu_0$, Poisson's ratio for the elastic system is 0.25.

The inverse Laplace transform is then computed by means of a Prony series (Cost, 1964) in which $\mathbf{u}(t)$ is approximated by

$$\mathbf{u}(t) \cong \mathbf{u}'(t) = \sum_i^N A_i\tau_i\left(1 - e^{-t/\tau_i}\right) \tag{3.21}$$

in a least-squares sense, where τ_i are arbitrary dimensionless relaxation times and A_i are parameters to be determined. This is easily achieved as the Laplace transform of (3.21) is well known. The displacements at arbitrary times can then be calculated.

I use the FORTRAN-77 codes developed by Rundle and co-workers to calculate post-seismic displacements due to a given fault, for a single elastic layer (plus visco-elastic half-space) model. For numerical stability,

Date	Scarp		M_0	Strike	Dip	Rake	Length	Displacement /m	
	Lat.	Lon.	/Nm$\times 10^{18}$				/km	Dip-slip	Strike-slip
04 March	38.212	23.213	2.0	90	46	-108	6.0	0.71	0.23
04 March	38.200	23.163	1.9	80	62	-100	6.0	0.86	0.15

Table 3.4: Double-event model for the 04 March 1981 Platea–Kaparelli earthquake used for post-seismic modelling, (A. Curtis, pers. comm.).

the solution is calculated as a perturbation to the fully-elastic half-space solution of Okada (1985). Despite this, numerical integration of the Green's functions remains a very computer-intensive process.

3.7.2 Comparison with actual deformation

Comparison of theoretical post-seismic displacements with known displacements or strains at a given time after an earthquake, or a time series of displacements or strains, will yield an estimate for the Maxwell time τ_0 from which the effective viscosity η can be derived. As the theoretical calculations are so time-consuming, it is not feasible to invert for the source parameters of the earthquake from the post-seismic displacements. Also, the post-seismic signal is small compared with the co-seismic signal and so resolution will be poor if too many parameters are estimated. Thus the method of choice is to compare observed post-seismic deformation (displacements, or baseline length changes) with the theoretical time series calculated from a given set of source parameters. To limit the non-uniqueness of the problem, care must be taken that these source parameters are as close as possible to the 'true' values, particularly if only a single epoch of observed post-seismic displacements is available.

3.7.3 Example: forward modelling of deformation after the 1981 Alkyonides events

Post-seismic visco-elastic relaxation has been modelled using the Rundle model. The major contribution to the signal is from the 24 February main shock, although the contribution from the 25 February shock is also significant. Parameters for these events are taken from Taymaz *et al.* (1991) (see Table 3.1). The 04 Mar event has a less significant effect (approximately 20% of the total seismic moment) and so the choice of parameters is less critical. For this forward model, I used parameters of the dual fault plane mechanism obtained by Andrew Curtis of Oxford University (A. Curtis, pers. comm.) and given in Table 3.4, as these were the the best estimates at the time. The elastic layer thickness has been taken to be 10 km throughout, with all faults penetrating the entire elastic layer.

Predicted post-seismic displacements after one Maxwell time are shown in Figure 3.6, and model dilatation and shear strain (total gamma) are shown in Figures 3.7 and 3.8. The model signal is dominated by that due to the 24 February mainshock, and to a lesser extent, that due to the large 25 February aftershock. The displacement is characterised by movement of material down-dip away from the fault planes and movement of material along strike towards the centre of the main fault plane. The highest total shear strain occurs within the immediate hanging wall of the main fault, a region which unfortunately is under water. However, the area of greatest positive dilatation lies close to the fault scarp and so this signal should be possible to measure, unlike that in the area of greatest negative dilatation which lies further down-dip, in the centre of the Alkyonides Gulf. Maximum shear strains are of order 5 p.p.m. which is barely detectable by classical triangulation with the existing network geometry, particularly as the post-seismic signal is likely to be contaminated by continued secular opening of the Alkyonides Gulf.

Figure 3.9 shows computed vertical post-seismic displacements along a down-dip profile as a function of increasing Maxwell time, for a single 20 km fault with 1 m of displacement across a plane dipping 30° throughout the entire 10 km elastic upper layer (similar to the 1995 Grevena earthquake). The displacement pattern changes very little with time other than to increase in magnitude at a decaying exponential rate, although the region of strain in the foot-wall also broadens slightly. The implications of this behaviour for network design are that a good knowledge of the earthquake magnitude, coupled with a dense network surrounding the fault, is necessary to be able to distinguish increased strain rates resulting from short Maxwell time from increased strain rates resulting from higher than expected co-seismic displacements. As reported relaxation times range from 1 month (Peltzer *et al.*, 1996) to 10 years (Reilinger, 1986), this issue is clearly important.

3.8 Summary

Epoch-to-epoch coordinate differences can be regarded as samples of smooth velocity fields or as the result of discrete straining crustal blocks. In all cases, care must be taken over the definition of the reference frames in which the coordinates have been measured, or reference frame-independent observations (baseline lengths, principal strains or shear strains) must be used. It is possible to estimate errors induced by setup problems at the GPS base station if site velocities are assumed constant and three or more epochs of data exist.

Co-seismic displacements can be modelled using Okada's (1985) fault slip model, given the parameters of the earthquake. The inverse problem is highly non-linear, but can be solved effectively using a combined Monte-Carlo / downhill simplex approach, even when the *a priori* values of the parameters are poor. This method also yields information regarding the stability and confidence limits of the solution. Coordinates

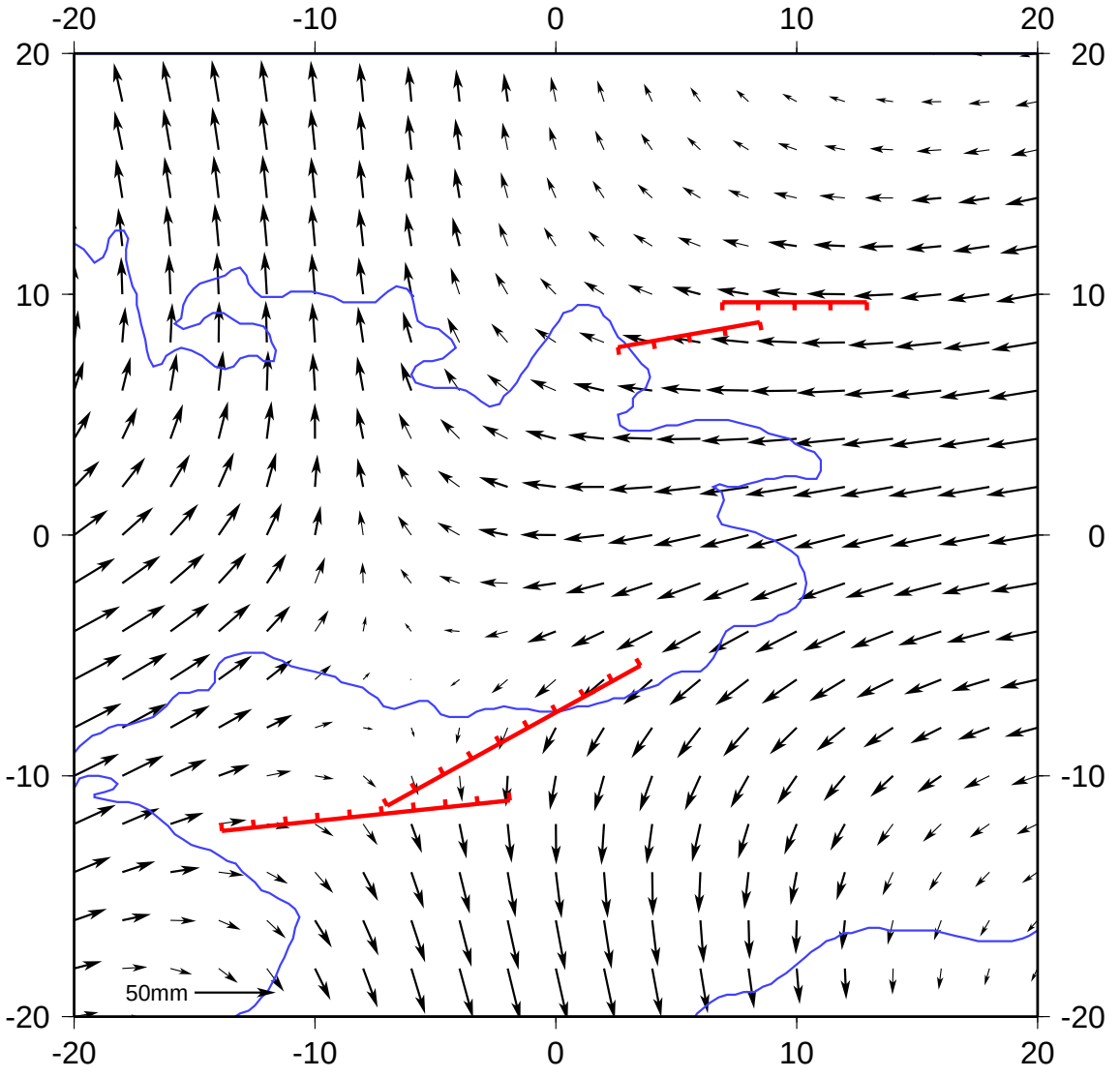

Figure 3.6: Predicted post-seismic horizontal displacements for the 1981 Alkyonides earthquakes, after one Maxwell time. The tick marks on the margins are in units of km.

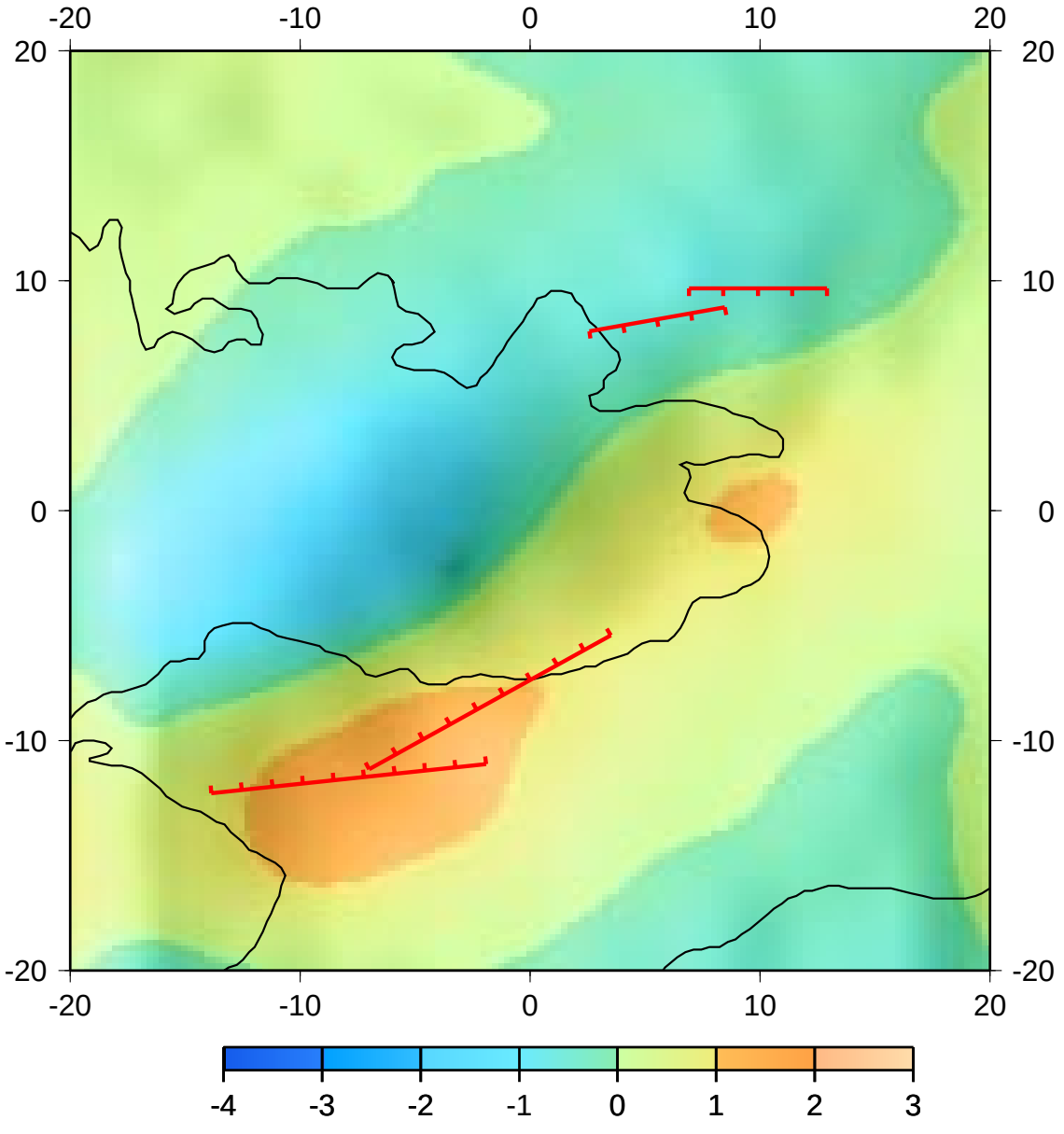

Figure 3.7: Predicted post-seismic dilatations for the 1981 Alkyonides earthquakes, after one Maxwell time. The tick marks on the margins are in units of km and the scale bar is in units of p.p.m..

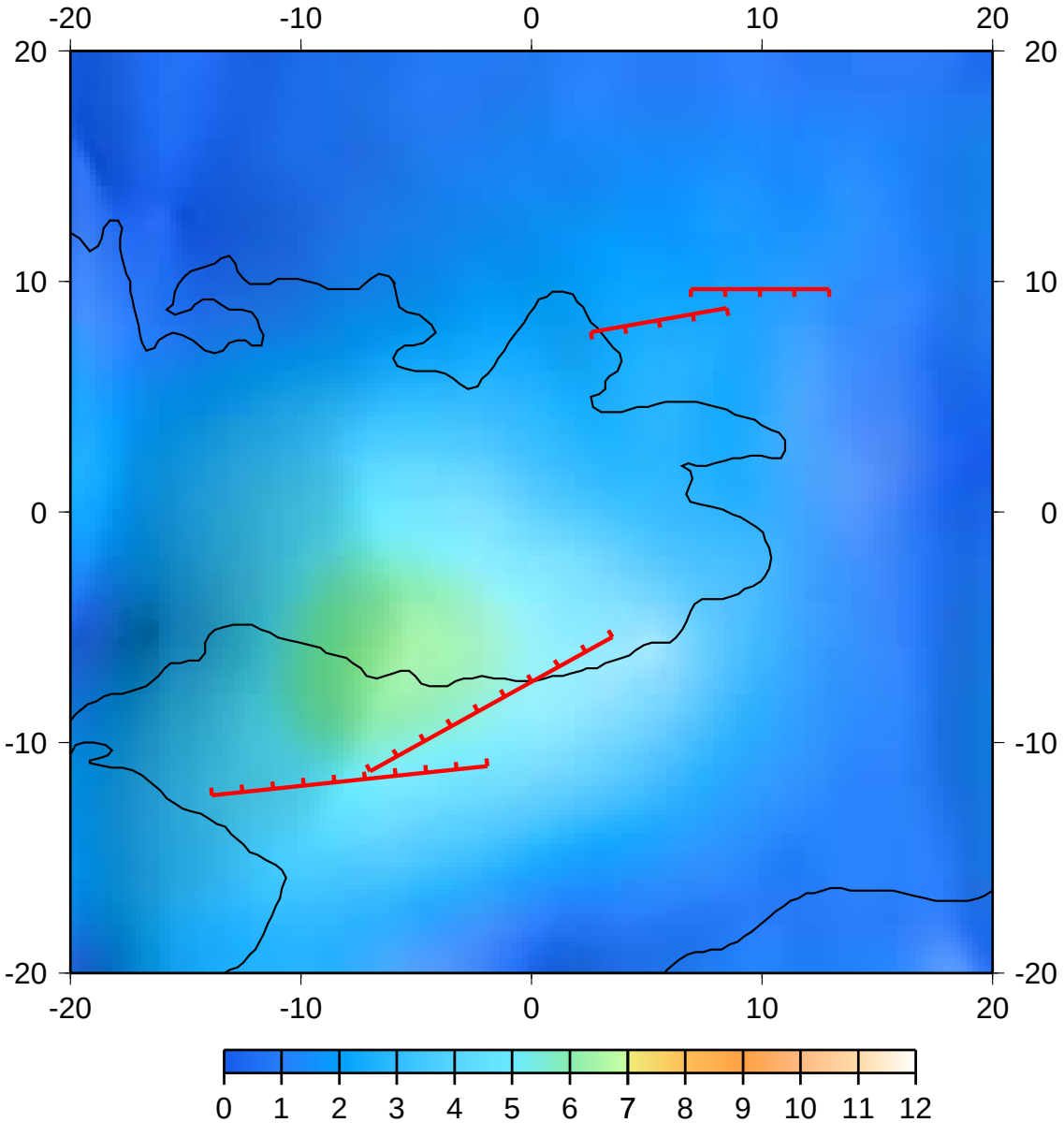

Figure 3.8: Predicted post-seismic shear strains (total gamma) for the 1981 Alkyonides earthquakes, after one Maxwell time. The tick marks on the margins are in units of km and the scale bar is in units of p.p.m..

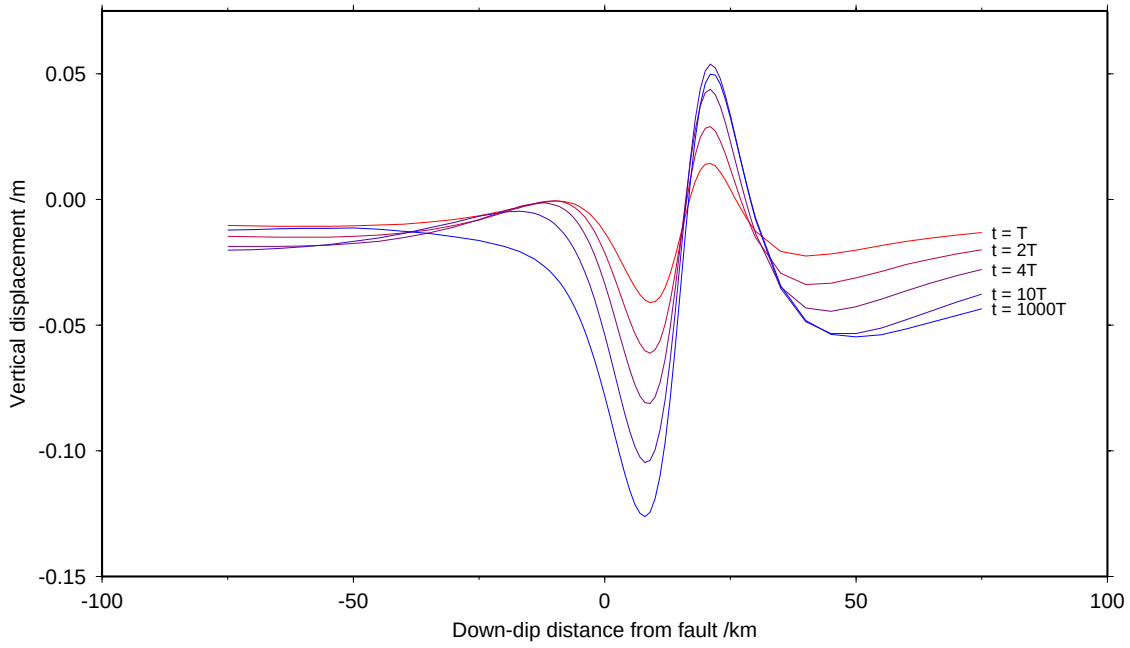

Figure 3.9: Down-dip profile of predicted post-seismic vertical displacements as a function of Maxwell time, for a simple fault (see text).

obtained from triangulation and trilateration networks are less precise than those obtained from GPS, but are sufficiently accurate to detect moderately high co-seismic strains, *e.g.* in the locality of the 1981 Alkyonides earthquakes.

Post-seismic displacements may arise as a result of viscous relaxation within the more ductile lower crust. They can be modelled using the two-layer half-space model of Fernandez *et al.* (1996), but computation times are long so that inverse modelling is impracticable. The time-scale of deformation is directly related to the viscosity of the lower layer which for shallow crustal earthquakes corresponds to the lower crust.

Alternatively, post-seismic deformation may take the form of slip on a narrow lower crustal shear zone. In certain circumstances it may be possible to distinguish this case from the half-space relaxation model. Terrestrial surveying techniques are too imprecise to distinguish between models of post-seismic deformation after crustal normal-faulting earthquakes such as the 1981 Alkyonides sequence, but GPS techniques may make this possible.

Chapter 4

Central Greece, 1989 – 1996

The Central Greece Network consists of 66 sites located in central Greece, Evvia and the Peleponessos between latitudes 37°N and 40°N (Figure 4.1). The network was established in 1988 as a collaborative venture by the Universities of Oxford, Newcastle upon Tyne and Nottingham in the UK, and the National Technical University of Athens, Greece (Denys *et al.*, 1995). The majority of the sites are brass marker pins set in stable limestone outcrops, but seven stations (11, 13, 22, 44, 48, 49, 59) take the form of concrete triangulation pillars established by earlier surveys. Three stations (16, Xrisokellaria, XRIS; 18, Karitsa, KRTS; 54, Dionysos, DION) have also been observed with mobile Satellite Laser Ranging (SLR) apparatus (Boucher *et al.*, 1994 and references therein; Noomen *et al.*, 1993; Smith *et al.*, 1994).

The network bounds the Gulfs of Korinthos and Evvia, two of the most active areas of recent seismicity. The station spacing is of order 20 km which is similar to the length-scale of the large normal faults in the region (Leeder & Jackson, 1993). The purpose of this chapter is to present the GPS observations of the network that have been made, and to provide a preliminary analysis of the site displacements of the whole network. A detailed discussion of that part of the network surrounding the Gulf of Korinthos, which has been occupied more often than the rest, is left until Chapter 5.

4.1 Network occupations

The network has been occupied fully three times (June 1989, October 1991 and May 1993) as a joint venture between the Universities of Oxford, Newcastle upon Tyne and Nottingham, the National Technical University of Athens (NTUA) and ETH Zurich. In addition, several stations around the Gulf of Korinthos were reoccupied by the University of Oxford, NTUA and others immediately after the June 1995 Egion earthquake, and again in October 1995 and May 1996. Table 4.1 shows the occupation history of Central

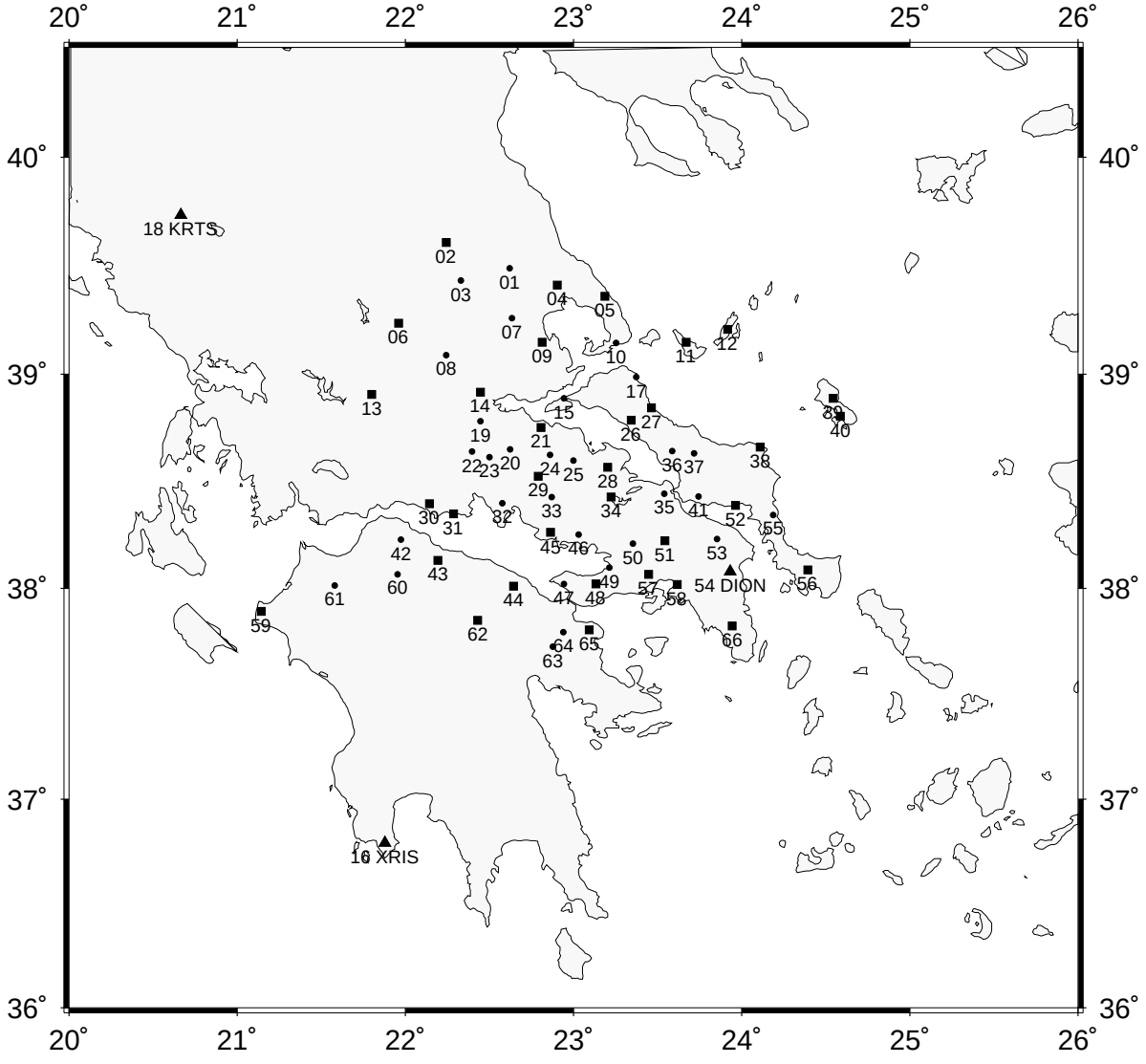

Figure 4.1: Stations in the Central Greece Network, occupied in 1989, 1991 and 1993. Stations Dion, Krts and Xris, shown as triangles, have also been occupied with SLR. These and the stations shown as squares were occupied with dual-frequency receivers at all three epochs; other stations were only occupied with dual-frequency receivers in 1991 and 1993.

Greece Network sites.

The first three occupations have all been processed using fiducial methodology (Subsection 2.2.9), whereas the latter occupations have been processed using CODE precise orbits (Subsection 2.2.4). Each survey epoch is now reviewed.

4.1.1 Epoch 1989.44 (May 1989)

Data

In June 1989, six dual-frequency and six single-frequency receivers (Table A.1) were deployed in Greece. Fiducial stations operating at the time were HERS (temporarily), ONSA and WETT (Figure 2.4). DION was occupied by one of the six dual-frequency receivers in Greece. Table A.1 gives the observation schedule and antenna heights, and Table A.2 gives offsets from the ITRF fiducial site coordinates to those sites occupied in 1989, where necessary. Each Greek station was occupied for a five-hour window 0900 – 1400 UT, on two days. Fiducial sites were occupied daily, but the data are patchy and do not span the whole of each day.

GPS Processing

The data were preprocessed and cleaned as described in Section 2.2. Cleaning the fiducial data using the less accurate broadcast orbits was particularly frustrating, as cycle slips were frequently masked by systematic variations in the residuals caused by orbit errors. Baselines were chosen to link the dual-frequency receivers, then single-frequency baselines were added as spurs to complete the daily network. The data quality of the WM101 single-frequency receivers (groups 8 and 11) proved to be so poor that the baselines to these were uncleanable and had to be discarded. Thus stations 10, 22, 25, 32, 35 and 49 were lost.

Orbits were formed in 3- and 4-day arcs, selected so that times of fiducial station unavailability occur during the middle of the arc and thus extrapolation errors are limited. Fiducial orbit improvement was done in 3- and 4-day L3 network solutions to match the arcs, with all fiducial stations and dual-frequency baselines in Greece included.

For the regional daily networks, a local ionosphere model up to degree 5 in hour angle and degree 2 in latitude was formed using the L4 phase observable. L5 ambiguity resolution was then attempted, with 80–90% success. Finally, L1/L3 ambiguity resolution was performed, with typically 70% success.

Two campaign solutions were constructed, one from the daily dual-frequency (L3) solutions, and one from the daily mixed-frequency (L3 and L1) solutions.

	1989.44	1991.78	1993.39	1995.44	1995.76	1996.39
CG01		•	•			
CG02	×	•	•			
CG03		•	•			
CG04	•	•	•			
CG05	•	•	•			
CG06	•	•	•			
CG07		•	•			
CG08		•	•			
CG09	•	•	•			
CG10		•	•			
CG11	•	•	•			
CG12	•	•	•			
CG13	•	•	•			
CG14	×	•	•			
CG15		•	•			
CG16	•	•	•			
CG17		•	•			
CG18	•	•	•			
CG19		•	•	•	•	•
CG20		•	•	•	•	•
CG21	•	•	•			
CG22		•	•	•	•	•
CG23		•	•	•		•
CG24		•	•			
CG25		•	•			
CG26	•	•	•			
CG27	•	•	•			
CG28	•	•	•			
CG29	×	•	•			•
CG30	•	•	•	•	•	•
CG31	×	•	•	•	•	•
CG32		•	•	•	•	•
CG33		•	•		•	•
CG34	•	•	•			
CG35		•	•			
CG36		•	•			
CG37		•	•			
CG38	•	•	•			
CG39	•	•	•			
CG40	•	•	•			
CG41		•	•			
CG42		•	•	•	•	•
CG43	•	•	•	•	•	•
CG44	•	•	•	•		•
CG45	•	•	•	•		•
CG46		•	•	•		•
CG47		•	•			×
CG48	•	•	×			
CG49		•	•			•
CG50		•	•			•
CG51	•	•	•			
CG52	•	•	•			
CG53		•	•			
CG54	•	•	•	○	○	•
CG55		•	•			
CG56	•	•	•			
CG57	•	•	•			
CG58	•	•	•			
CG59	•	•	•			
CG60		•	•	•	○	•
CG61		•	•	•	•	•
CG62	•	•	•		•	•
CG63		•	•			•
CG64		•	•			•
CG65	•	•	•			•
CG66	•	•	•			

Table 4.1: Central Greece Network site occupations, 1989 – 1996. Primary marker occupations are denoted by •, eccentric marker occupations by ○, and occupations known to be unreliable *a priori* by ×.

Results

The mixed campaign solution has an r.m.s. station repeatability of (0.089 m, 0.092 m, 0.224 m) in east, north and up components, whereas the L3-only solution has an r.m.s. repeatability of (0.034 m, 0.050 m, 0.053 m). Histograms of residuals to the campaign solutions are shown in Figure 4.2. The mixed-frequency solution is far too noisy to be used to detect crustal motions over a five-year time-scale, so I have used only the L3 solution, although the quality of this is somewhat marginal. The reasons for the poor quality of the 1989 solution are:

- Short observation times (only 5 hours).
- Poor quality and amount (only 4 sites, with large data gaps) of fiducial station data.
- Low numbers of satellites observed (maximum of 8 during 5-hour sessions).
- Long (60 – 120 km) dual-frequency baselines, with poor network geometry.

The coordinates determined in the L3 campaign solution are given in Table A.3. Only 36 sites (Figure 4.1) had dual-frequency occupations (including DION), and of these, 12 sites had only a single occupation so no blunder checking can be performed, and the data must therefore be viewed with circumspection. The measurements at sites 02, 14, 29 and 31 are suspect on the grounds of very poor day-to-day repeatability. As there were only two measurements, it is impossible to tell which is the outlier and so both days must be discarded, leaving 32 'reliable' stations.

4.1.2 Epoch 1991.78 (October 1991)

Data

The October 1991 campaign utilised eleven dual-frequency receivers (Table A.4) in Greece. Fiducial site observations were available from HERS, MADR, ONSA and WETT. Each Greek site was occupied on two days for seven-hour sessions 1500 – 2200 UT, whereas the fiducial sites were occupied daily on a nominal 24-hour basis. However, there are several large gaps in the fiducial data, although the situation is better than that of 1989. Table A.4 gives the observation schedule and antenna heights, and Table A.5 gives offsets from the ITRF fiducial site coordinates to those sites occupied in 1991, where necessary.

GPS Processing

The data were preprocessed and cleaned as described in Section 2.2. Baselines were chosen so as to minimise receiver type mixing, and where possible to maintain a sub-network 'backbone' of full-wavelength

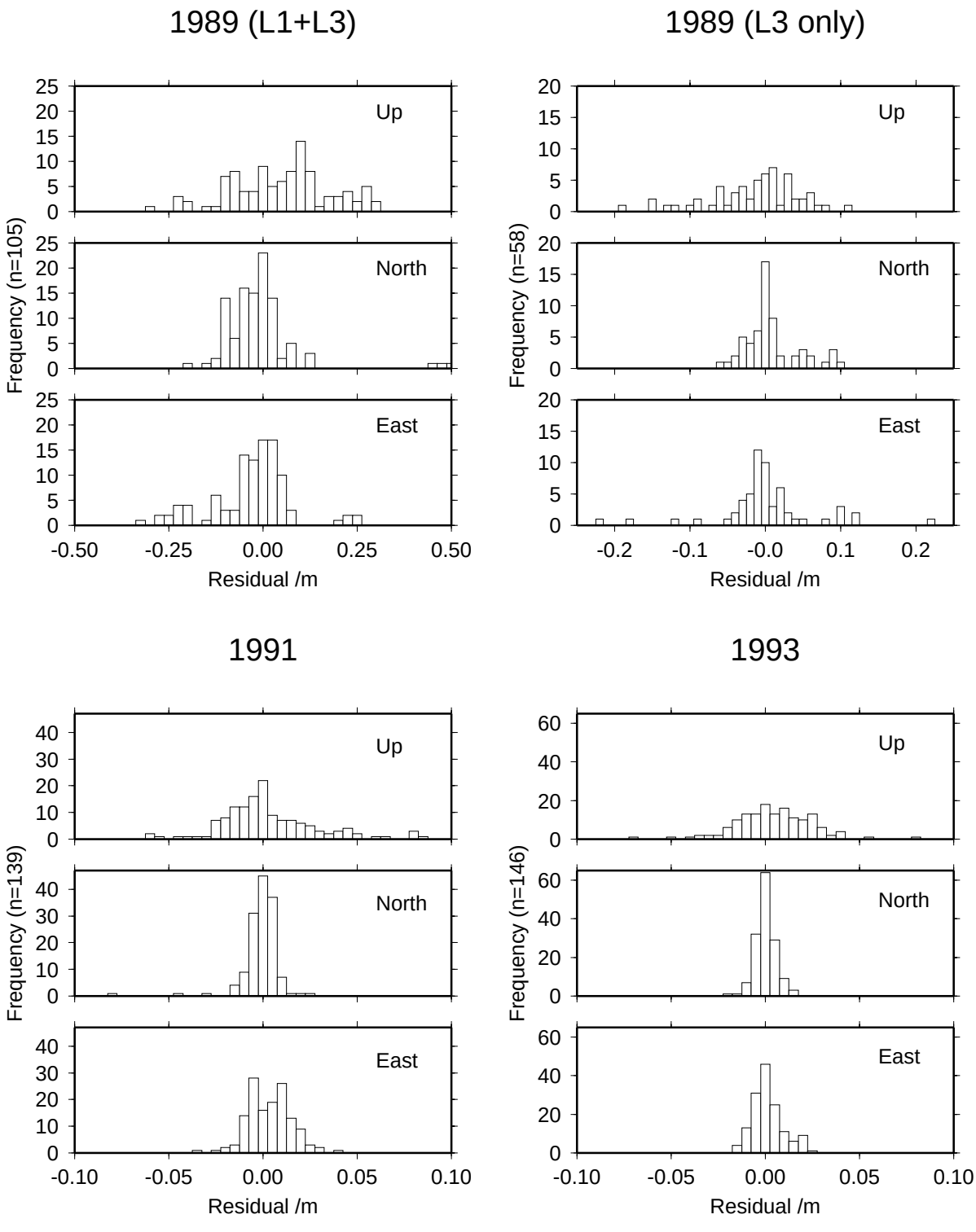

Figure 4.2: Residuals to Central Greece Network full network campaign solutions (1989, 1991 and 1993) in north, east and up components. For 1989, both the mixed-frequency solution and the L3-only solution residuals are shown, whereas for 1991 and 1993, there were no mixed-frequency solutions (see text). Horizontal and vertical scales vary from epoch to epoch, but are constant within each epoch. For statistics of residuals see text.

L2 receivers (Trimble and Wild). Orbits were formed in 3- and 4-day arcs from the NGS precise ephemeris, and then iteratively improved using L3 solutions including all data from the relevant days. An ionosphere model up to order 5 in hour angle and order 2 in latitude was formed for each day from the L4 phase observable, and used to assist baseline cleaning. The level of ionospheric noise in the 1991 campaign was much higher than for all other campaigns, and this caused problems in ambiguity resolution because the majority of the receivers only observed half-wavelength (squared) L2. Ambiguity resolution was attempted using daily L5 then L3 solutions. However, the daily level of L3 ambiguity resolution achieved was typically only 60%, indicating that considerable errors in resolution may have occurred, so the daily L3 ambiguities-unresolved solutions were used to form the campaign solution.

Results

The solution is of higher quality than that of the 1989 survey, because of increased numbers of satellites, increased observation session lengths, and the use of dual-frequency receivers throughout. The r.m.s. residuals to the network solution are (0.010 m, 0.015 m, 0.028 m). Histograms of the residuals are shown in Figure 4.2, and the coordinates are given in Table A.6.

4.1.3 Epoch 1993.39 (May 1993)

Data

In May 1993, 18 dual-frequency P-code receivers were used to reoccupy the network. Three receivers (one each of Ashtech, Leica, Trimble) were deployed 24 hours a day at DION, and the other SLR sites, KRTS and XRIS, were occupied every day with single receivers. Other Greek sites were occupied twice, with the exception of site 48 (a concrete pillar) which was found to have been destroyed on the first visit, and so the remains were only occupied for one short session. The observation schedule was well-designed so that in general, baselines were quite short (20 – 30 km). As a result, the network geometry was much stronger than in previous years and baselines were easier to clean. The observation time was increased to 9 hours, 1100 – 2000 UT. Fiducial site observations were available 24 hours a day from HERS, MADR, ONSA and WETT, with very few data gaps. Table A.7 gives the observation schedule and antenna heights, and Table A.8 gives offsets from the ITRF fiducial site coordinates to those sites occupied in 1993, where necessary.

GPS Processing

Orbits were formed in 2- and 3-day arcs from the JPL (NASA Jet Propulsion Laboratory) precise ephemeris, and improved using repeated L3 fiducial network solutions, including all the Greek observations as well as the data from fiducial sites. Baselines in the daily networks were chosen to minimise baseline lengths, but also to limit receiver type mixing by maintaining a 'backbone' of Ashtech – Ashtech baselines. An ionospheric model up to degree 5 in hour angle and 2 in latitude was generated using the L4 phase measurements and used to assist L5 ambiguity resolution in daily network solutions. Ambiguity resolution was very successful: over 99% of L5 and L3 ambiguities were resolved.

Results

The r.m.s. residuals to the campaign solution are (0.005 m, 0.008 m, 0.020 m) in east, north and up components. This considerable improvement on previous campaigns is the result of having both more and better data. Histograms of the residuals are shown in Figure 4.2 and the campaign coordinates are given in Table A.9.

The May 1993 survey offers the opportunity to compare the residuals for sites observed on only two days with those observed on a larger number of days, because sites 16 and 18 (Xris and Krts) were occupied on all ten days of the campaign. The r.m.s. residuals for these two sites only were (0.005 m, 0.010 m, 0.022 m) in east, north and up components. This implies that the underestimate in error caused by only having two occupations per site is not serious, perhaps 25%, although larger systematic errors may occur as a result of seasonal (atmospheric pressure and humidity) and other long term (*e.g.* tidal) effects. Also, if two measurements differ significantly, it may be impossible to decide which of the two is in error, so third and subsequent occupations are desirable to eliminate such blunders.

On comparison with the 1989 and 1991 surveys, site 51 showed a large (approximately 1 m) anomalous displacement. It was at first thought that this might imply that the 1993 setup was over one of the witness marks, but no witness marks correspond to the displacement and so it seems likely that site 51 is not on true bedrock but on a boulder that has shifted (a 1 m GPS horizontal error repeated on two separate days is exceptionally unlikely).

4.1.4 Epoch 1995.44 (June 1995)

On 15 June 1995 a large earthquake of magnitude $M_0 = 6.0 \times 10^{18}$ Nm struck the Gulf of Korinthos close to Egion, within the area of the Central Greece Network (Figure 4.3). In order to enable post-seismic motions to be studied, GPS measurements were made immediately after the event, from 20 – 26 June.

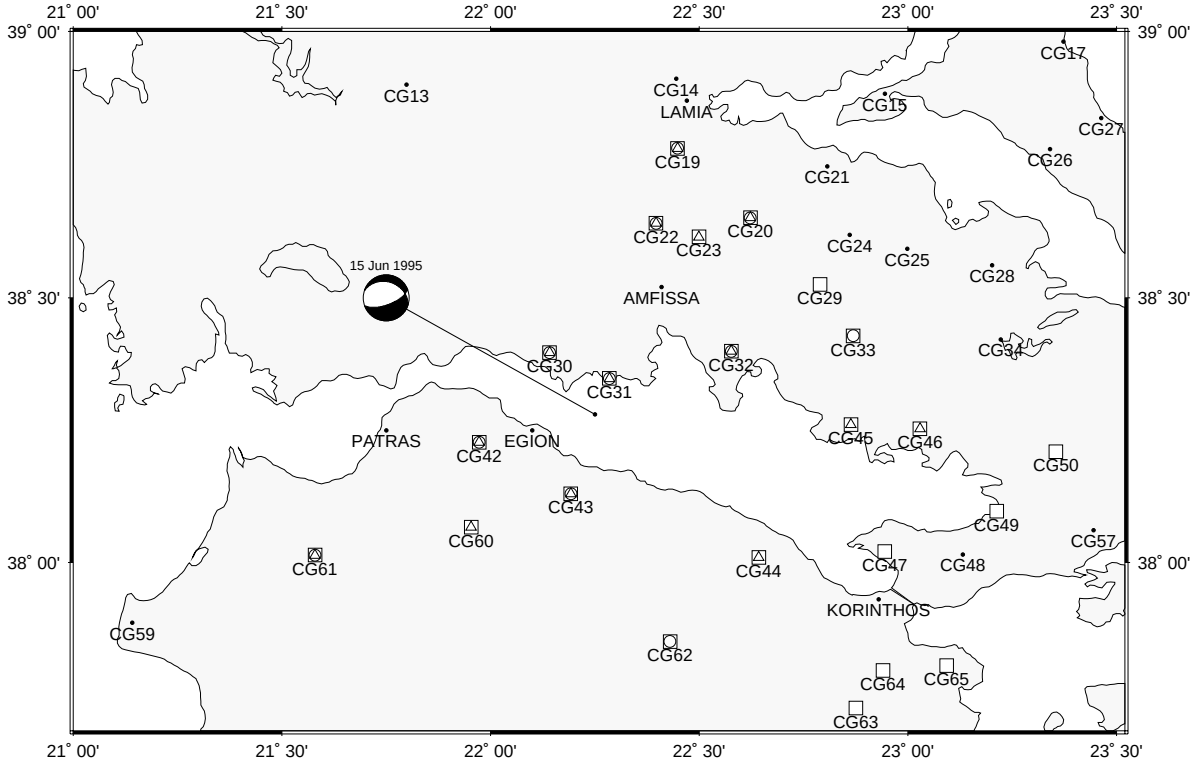

Figure 4.3: Location of the 15 June 1995 ($M_0 = 5.5 \times 10^{18}$ Nm) Egion earthquake. Nearby stations in the Central Greece Network are shown. Triangles denote sites occupied in June 1995, circles denote sites occupied in October 1995, and squares denote sites occupied in May 1996.

Fourteen Central Greece sites were reoccupied, and in addition, the network was densified to the north and south of the Gulf in the region of the earthquake (Subsection 5.2.3).

Stations close to the epicentre of the earthquake were displaced during the co-seismic release of strain, although there are too few sites to enable an inversion for the source parameters to be carried out. The co-seismic motion will bias measurements of long-term inter-seismic strain made over the epoch of the earthquake, but this bias can be removed using a forward model of the co-seismic displacements (Subsection 5.2.2).

Data

Central Greece sites 19, 20, 22, 23, 30, 31, 32, 42, 43, 44, 45, 46, 60, 61 were included in the Egion network (Figure 4.3). There were no fixed observation windows, but stations to the north and south of the Gulf of Korinthos were occupied continuously during the campaign, so relatively short baselines could be formed radiating from these fixed stations. Stations were occupied for a minimum of five hours on at

least two occasions. A mixture of Ashtech Z-XII and P-code receivers was used. Z-XII receivers record full-wavelength L2, whereas P-code receivers can currently only record $L2^2$, because AS (Subsection 2.1.2) has been operational since January 1994 when the GPS satellite constellation became complete.

The permanent GPS pillar at DION (Table A.10) was occupied continuously during the campaign by ETH (Zurich) and NTUA and this data was in principle made available to us (D. Paradissis, pers. comm.). However, due to a computer malfunction at Dionysos only data for days 171 – 173 were in practice available, and due to a receiver malfunction only two six-hour sessions within this period contained processable data. Thus the network is only weakly tied to the reference point as DION. However this misfortune will not affect the relative position vectors of stations either side of the Gulf.

Figures A.1 – A.4 show the observation schedule and antenna heights, and Table A.10 gives offsets from the ITRF site coordinates at DION to the GPS pillar occupied during the campaign.

GPS Processing

The data (with the exception of the DION site) were pre-processed and cleaned by Robert Davies of Oxford University, using CODE precise ephemerides. Later, I added the DION data and reprocessed the daily network and campaign solutions. Ambiguities were resolved using daily L5 solutions followed by daily L3 solutions. Ionospheric models were not required becasue the network is relatively small in extent (100 km end-to-end, with typical baselines only 10 km) and ionospheric conditions in June 1995 appear to have been very quiet. Over 97% ambiguity resolution was achieved.

Results

The r.m.s. residuals to the campaign solution (0.005 m, 0.004 m, 0.020 m) are of the same order as those from 1993. Although the occupation times were halved (with respect to 1993) and the majority of receivers could only observe half-wavelength L2, the benefits of further improvements in the satellite constellation and the use of much shorter baselines outweighed these problems. However, these r.m.s. residuals may be artificially low because most stations were only occupied twice and only one continuous site on one or other side of the Gulf was operational at any one time. Histograms of the residuals are shown in Figure 4.4 and campaign coordinates are given in Table A.11.

4.1.5 Epoch 1995.76 (October 1995)

In October 1995 fieldwork was carried out around the Gulf of Korinthos in collaboration with NTUA, the Institut de Physique du Globe de Paris (IPGP) and the Hellenic Army Geographic Service (HAGS). The

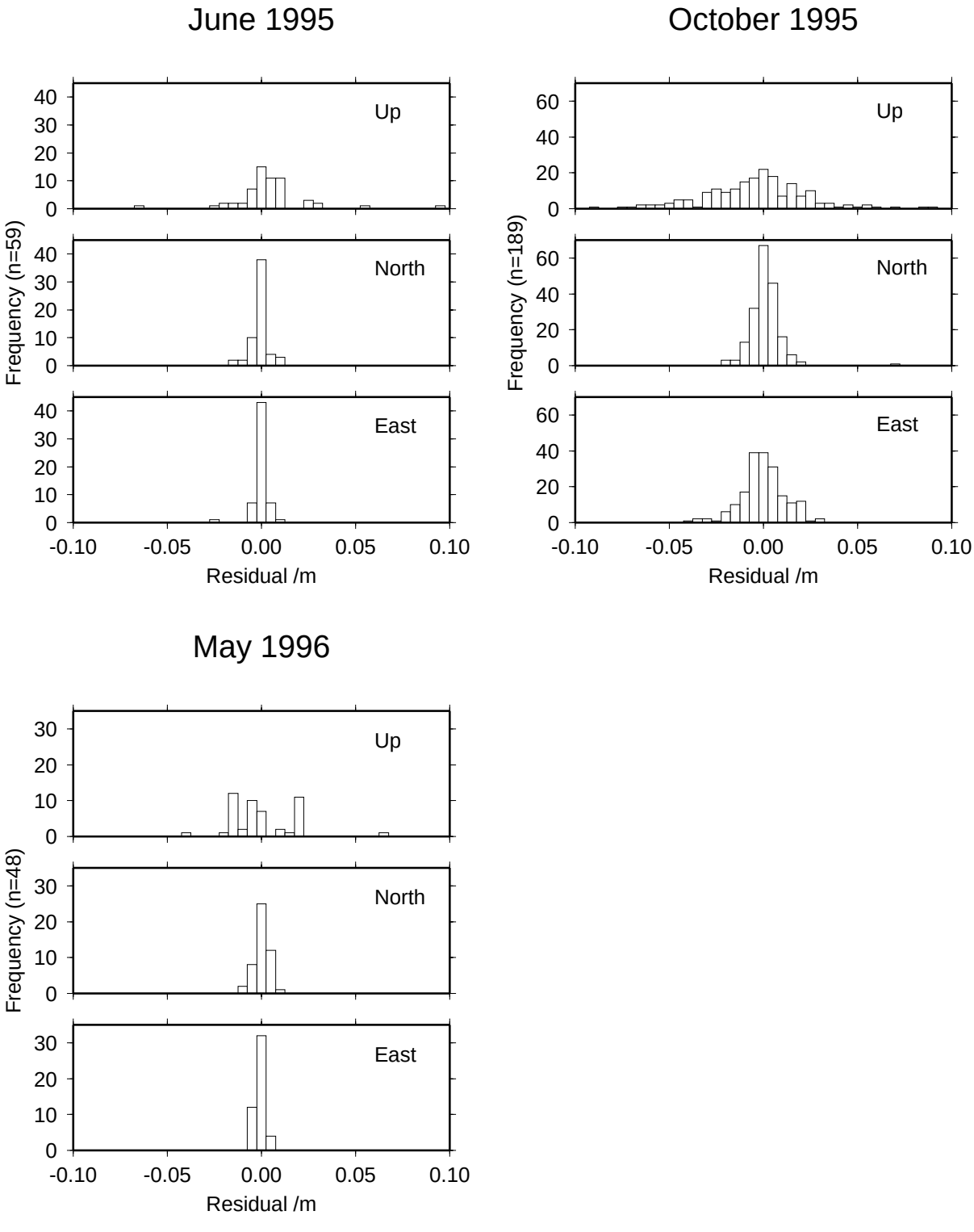

Figure 4.4: Residuals to Central Greece Network / Gulf of Korinthos sub-network campaign solutions (June 1995, October 1995 and May 1996) in north, east and up components. Vertical scale varies from epoch to epoch, but is constant within each epoch. For statistics of residuals see text.

network for post-seismic deformation studies established in June 1995 was densified and some additional Central Greece sites were occupied.

Data

Sites 19, 20, 22, 30, 31, 32, 33, 42, 43, 60, 61, 62 were included in the network (Figure 4.3). In general, sites were occupied either overnight from approximately 1600 – 0700 UT, or during the day from approximately 0700 – 1500 UT, on at least two occasions, using either Ashtech or Trimble receivers. A total of 11 Ashtech Z-XII receivers (L1,L2), 16 Ashtech P-code receivers (L1, L2 or $L2^2$) and 3 Trimble 4000SSE receivers (L1,L2) was used. Continuous stations either side of the Gulf (CT00 and T000 from the Egion post-seismic network) and at DION (GPS pillar) provided the basis for the network. Figures A.5 – A.10 show site occupation times and antenna heights.

GPS Processing

The data were pre-processed and cleaned by Margaret Moore of Oxford University, with the exception of the HAGS and Dion Trimble data which I added later. Ionospheric conditions were quiet and the daily networks were small in geographic extent, so over 99% L5 ambiguity resolution was achieved using daily L5 network solutions with no ionospheric model. However, daily L3 solutions only achieved 84% L3 ambiguity resolution, possibly because baselines were slightly longer than in June 1995 (without necessarily having longer occupation times).

Results

R.m.s. residuals to the whole-campaign network solution (0.009 m, 0.012 m, 0.028 m) are higher than those in June 1995 or May 1993, but similar to those in October 1991. The causes of this may be partly the lack of total L3 ambiguity resolution and also partly the presence of several short baselines to witness markers which were only occupied for 1- or 2-hour sessions and therefore have higher (and more realistic) formal errors in the GPS solutions, although their presence does not affect the positions of other sites. Histograms of residuals to the campaign network solution are shown in Figure 4.4 and campaign coordinates are given in Table A.12.

4.1.6 Epoch 1996.39

In May 1996, that part of the Central Greece Network bounding the Gulf of Korinthos was reoccupied by Oxford University and NTUA as part of an experiment involving the Egion post-seismic network which

had been set up in June and October 1995. All relevant Central Greece sites (Figure 4.3) were occupied once during the final four days of the campaign, and also several local site ties and witness marks. Up to 15 Ashtech Z-XII receivers (L1, L2) were used simultaneously. Only the subset of the network observed on the last four days of the campaign has been processed to date.

Data

Stations shown in Figure 4.3 were occupied for a minimum of six hours, usually on only one occasion, but stations also in the Egion post-seismic network had repeat occupations within the four-day period. The observation schedule and antenna heights are given in Figures A.11 and A.12. The antenna at site 47 on day 152 was disturbed at an unknown time during the session and so may be unreliable, but was processed anyway. Continuous stations either side of the Gulf (CT00 and T000 from the Egion post-seismic network) and at DION (point C) provided the basis for the network.

GPS Processing

Orbits were formed from CODE precise ephemerides in 1-day arcs. Baselines were chosen so as to maximise data use and minimise baseline lengths (receiver mixing was not an issue in this campaign). The residuals to the baseline solution to site 47 showed no evidence of jumps or bias, so the antenna disturbance must have occurred either very early or very late in the session.

100% L5 ambiguity resolution was achieved in daily network solutions without the need for an ionospheric model, and subsequently over 95% of the L3 ambiguities were resolved in daily network solutions.

Results

The r.m.s. residuals to the campaign solution (last four days) are (0.003 m, 0.002 m, 0.017 m). These artificially low figures reflect the fact that the majority of stations were only occupied once, so the day-to-day repeatabilities are only meaningful for the continuous sites CT00, T000 and DION. For these sites, the r.m.s. residuals are (0.006 m, 0.003 m, 0.014 m), which are slightly less unrealistic, but as discussed above for the SLR sites occupied in 1993, occupation for only a few sessions will underestimate actual errors.

Histograms of the residuals are shown in Figure 4.4 and the campaign coordinates are given in Table A.14. On comparison with the previous occupation, site 47 showed a large (approximately 0.1 m) anomalous motion when compared with neighbouring sites and so I conclude that the antenna disturbance occurred early on in the session, and discard the 1996 coordinate from future study.

4.2 Site displacement analysis

The most complete and precise survey of the network was the May 1993 campaign (sites 48 and 51 missing, but far better precision than 1991), so this has been used as the basis for later comparison. During the GPS computations, the ITRF position and velocity has been used for DION as this describes the site within the chosen global reference frame. However, for geological and geophysical interpretation, it is of more use to know the motion of the network with respect to the stable part of the European plate. This can be calculated by subtracting the NUVEL–1NNR site velocity (which describes its motion if it were moving as part of the European plate) from each site in the network. In practice, the pole of rotation of the European plate in the NUVEL–1NNR frame is sufficiently far away that the rotational velocities differ by less than 0.7 mm yr^{-1} between the most extreme sites, so the Europe-fixed reference frame can be realised well within experimental error simply by subtracting the NUVEL–1NNR velocity of DION from all sites.

Figure 4.5 shows the velocities of sites in the network based on each of the intervals 19xx.xx – 1993.39. The 1991 – 1993 velocities show a general trend of southwestward velocity (with the exception of sites 17 and 24 which are clearly anomalous, possibly caused by poor setups or site instabilities) decreasing towards the northwest (but not to zero, implying that further deformation must take place between the northwest edge of the network and 'stable' Europe), and increasing across the Gulf of Korinthos (particularly in the west). The 1989 – 1993 velocities display a similar pattern, but as expected from the formal error estimates, the level of scatter in the data is much higher and so the 1989 survey does not add any information to the problem.

Velocities for later intervals also show trends consistent with 1991 – 1993 (except in the western Gulf of Korinthos where local co-seismic deformation in the 1995 earthquake has affected several sites) but with slight apparent systematic offsets. It is possible that these offsets are caused by errors in the realisation of the reference frame which could be the result of GPS satellite orbit errors or errors in the setup over the base station mark (DION). As DION was only set up once at each epoch, the latter errors are not averaged out over the campaign, but will result in an apparent translation of the network as a whole at that epoch, unlike the other sites for which independent setups were performed on each day of occupation so that setup errors could be mitigated by averaging. Also, the position of DION is subject to errors in the local terrestrial surveys between SLR point A (the ITRF site), SLR point C which was occupied in the 1989 – 1993 and 1996 campaigns, and the GPS pillar which was occupied in the two 1995 campaigns (and only weakly linked to the rest of the network in June 1995).

If the site velocities are in fact smooth over time (within error), it will be possible to estimate a network translation at each epoch so as to minimise the roughness in velocity for sites that have more than two epochs of occupation (Section 3.1). In order to do this it is necessary first to model any known phenomena that may have caused a temporal discontinuity of displacement. The 1995 Egion earthquake is such a

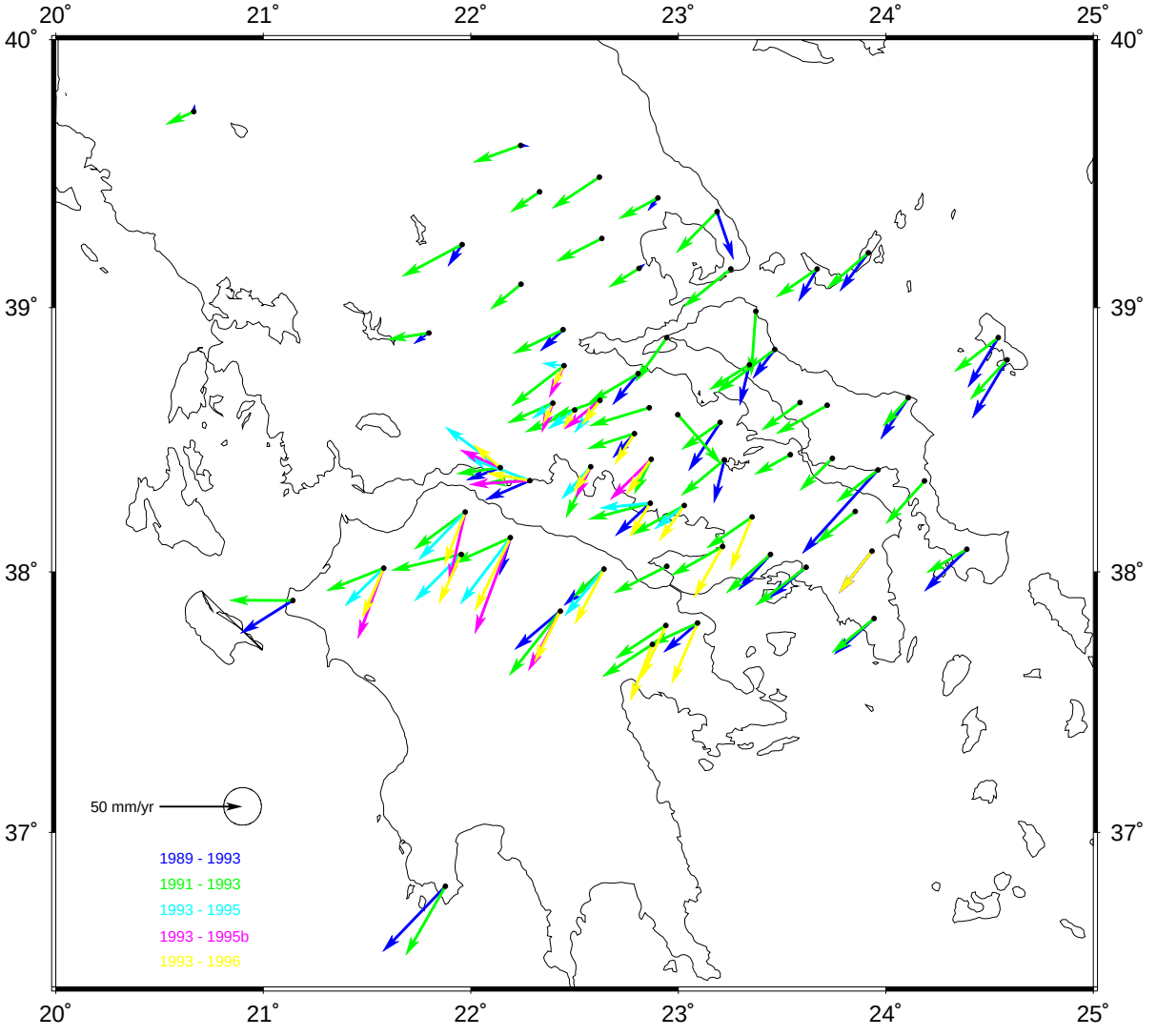

Figure 4.5: Raw velocities of Central Greece Network sites, based on each of the intervals 1989.44 – 1993.39 (blue), 1991.78 – 1993.39 (green), 1993.39 – 1995.44 (cyan), 1993.39 – 1995.76 (magenta), 1993.39 – 1996.39 (yellow). For clarity, errors are not shown, but are of the order shown in the scale bar (68% confidence) for intervals other than 1989 – 1993, for which they are approximately twice as large.

phenomenon and its effects were removed for epochs after the earthquake (June 1995, October 1995 and May 1996) before smooth velocities were estimated (see Subsection 5.2.2 for method and relevant figures).

Evidence for the smoothness of site velocities comes from the baseline lengths (corrected for co-seismic effects) which show remarkably smooth variation with time. If the baseline length changes vary smoothly, apparent discontinuities in the site displacements are likely to be caused by translation (and possibly rotation, but not scale) of the network as a whole from epoch to epoch. Figure 4.6 shows changes in baseline length from 1991 to 1996, relative to an arbitrary zero-offset, for baseline measurements across the Gulf of Korinthos, which have been repeated most often. The close fit of these baseline length measurements to a constant extension rate validates the hypothesis of temporally smooth deformation over the five-year time-scale of the surveys, once local co-seismic effects have been removed.

The 1993 campaign was taken as the fixed (base) epoch and whole-network translations were estimated at the other four epochs using the `velsmooth` algorithm (Section 3.1). The 1989 data were not included because of their unreliability, and the sites named above as outliers were also excluded. Figure 4.7 shows the cumulative site displacements for sites around the Gulf of Korinthos which have been occupied in 1991, 1993 and 1996 (and in many cases also once or twice in 1995), before and after the network translation at each epoch has been applied. The best-fitting set of epoch translations of the network have an r.m.s. value of 6.1 mm in the north component and 9.4 mm in the east component, which is within the probable bounds of error for a single-setup (tripod) GPS position, and compares with r.m.s. residuals of 7.2 mm per coordinate for the fit to time-invariant velocities of all stations with three or more occupations. If the assumption of uniform velocities of all sites is valid, and the modelling of the 1995 Egion earthquake is sufficiently good, then this latter value provides an estimate of the *a posteriori* reliability of the GPS positions allowing for the effects of local site instability.

Figure 4.8 shows the smoothed velocities of all the Central Greece Network sites. Even after the effects of the 1995 Egion earthquake have been modelled, the extension in the western Gulf of Korinthos is still seen to be higher than that in the east (see Chapter 5 for further discussion). However, there is little or no extension apparent across either the northern or southern Gulf of Evvia, the other major extensional feature of the region. The main effect of solving for whole-network translation parameters is to swing the velocities of sites in the northern part of the network (which only had reliable occupations in 1991 and 1993) towards a more southwesterly direction, orthogonal to the strike of the normal faults in the region.

It is possible that some of the deviations from smoothness of the site velocities are caused by rotations of the network as a whole caused by changes from the reference frame imposed by fiducial GPS up to 1993 to that imposed by the precise orbits thereafter, although the reference frames should be the same as they are based on compatible global velocity models. Smoothed velocity solutions in which network rotations were estimated were also attempted, but resulted in no improvement of fit despite the increase in the number of

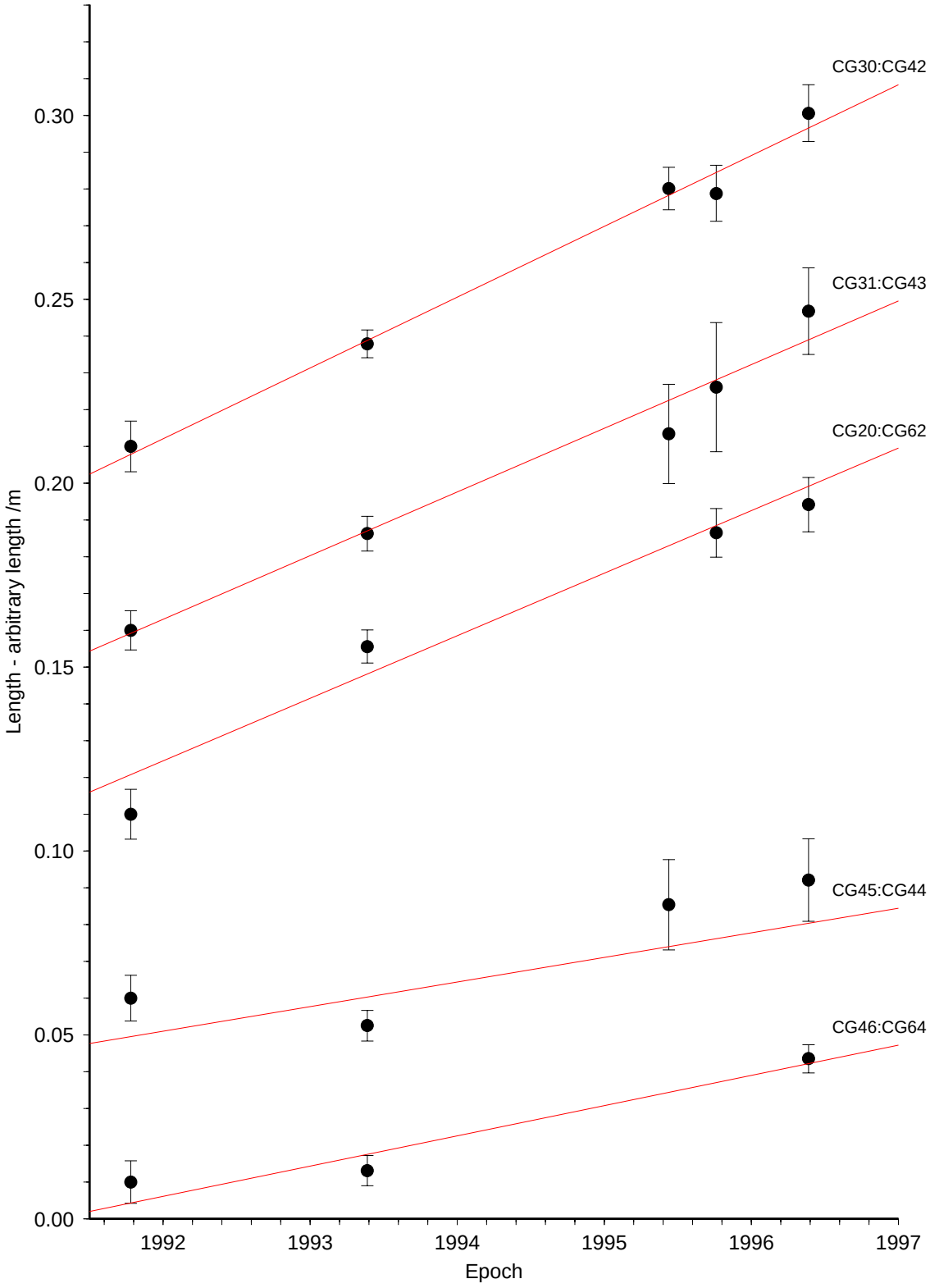

Figure 4.6: Baseline length changes between sites around the Gulf of Korinthos, 1991 – 1996, relative to an arbitrary zero-offset. Site coordinates at epochs later than 15 June 1995 have been corrected for co-seismic displacements in the 1995 Egion earthquake. The best weighted linear fit to each set of baseline length measurements is shown in red.

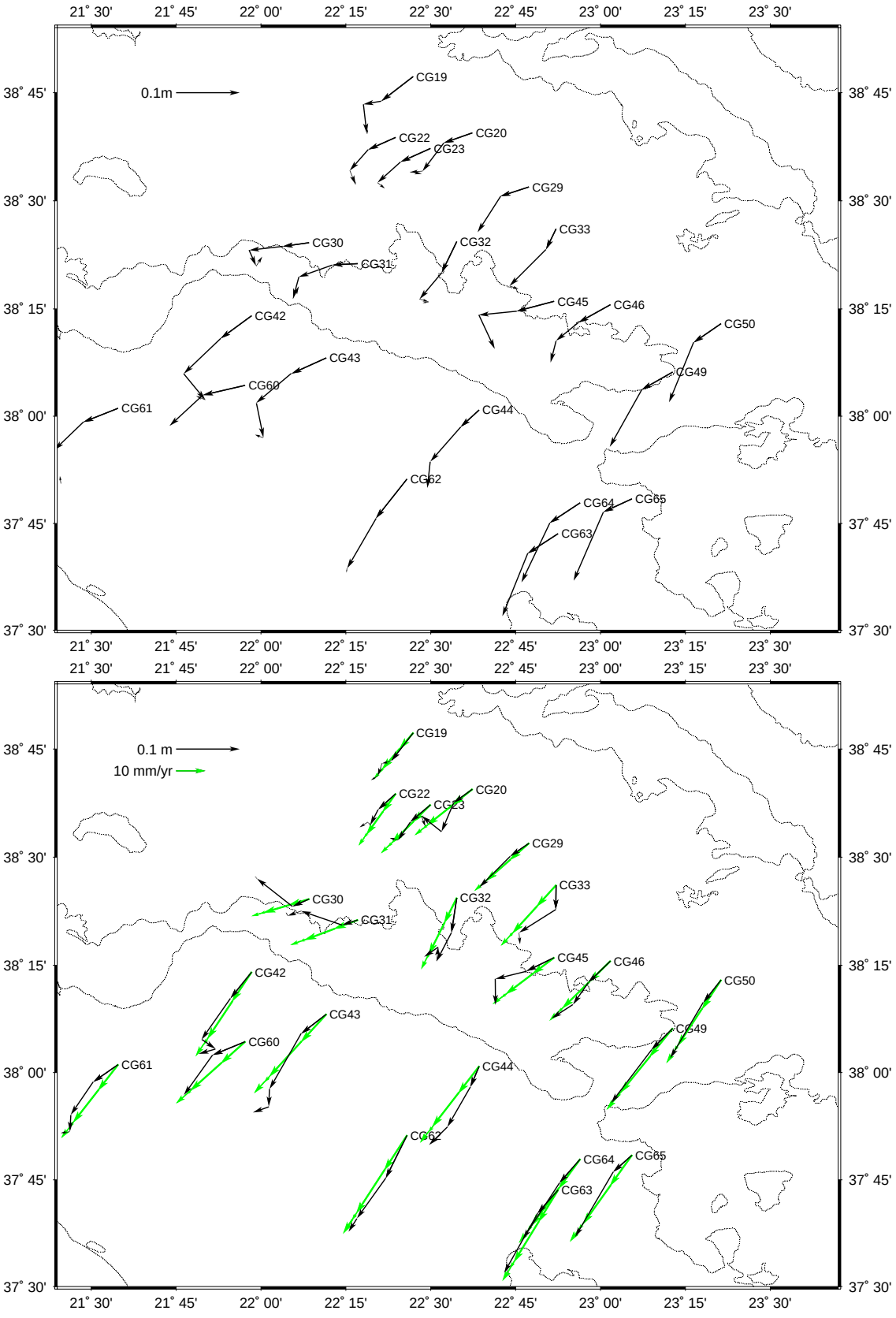

Figure 4.7: Cumulative displacements of Central Greece Network sites around the Gulf of Korinthos which have been occupied on at least three occasions 1991 – 1996, before (above) and after (below) network epoch translations have been applied. Black arrows in each plot show raw and epoch-translated cumulative displacements respectively; green arrows in the lower plot show the averaged site velocities.

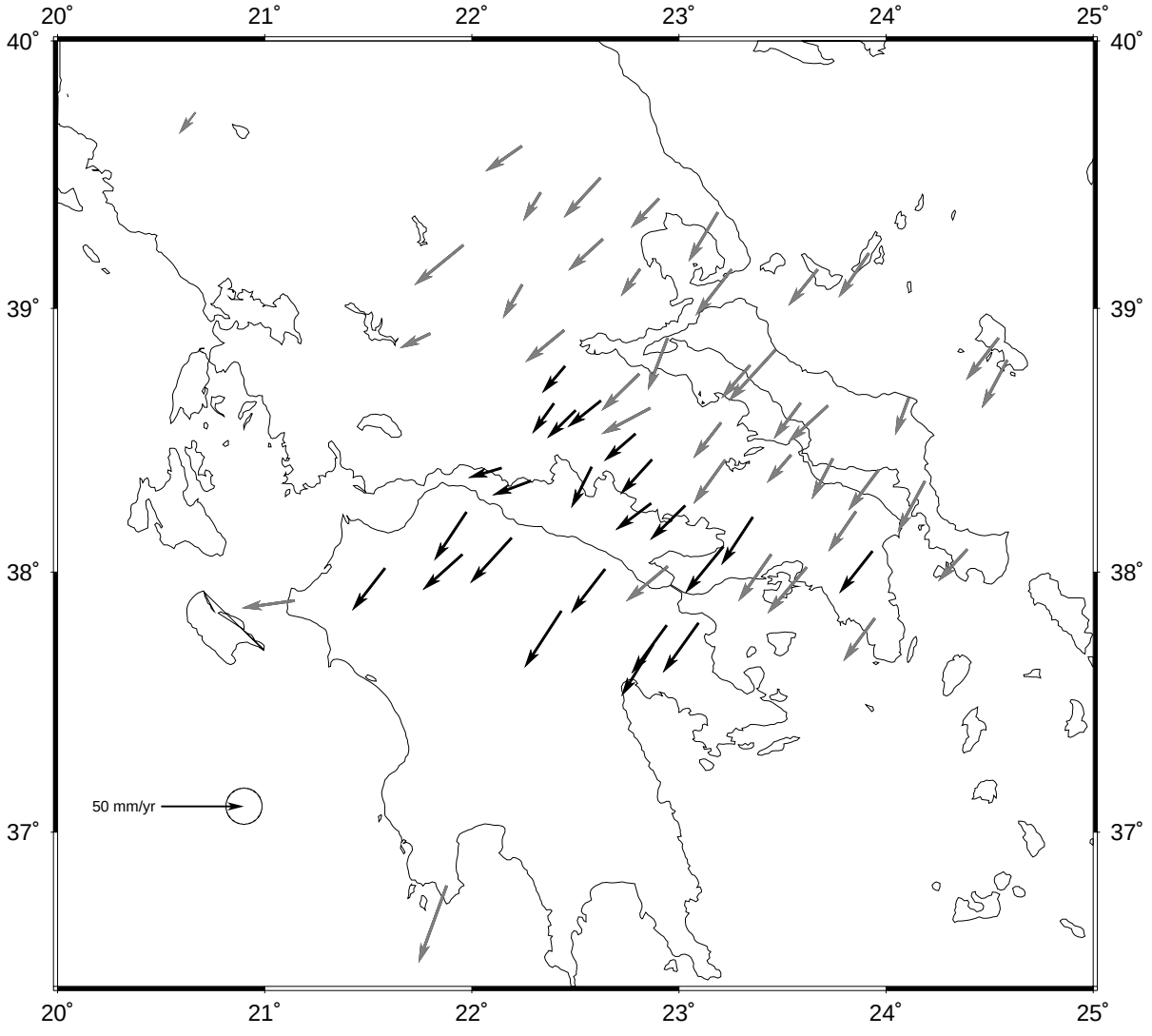

Figure 4.8: Velocities of Central Greece Network sites, averaged over the interval 1991.78 – 1996.39, with whole-network translations at each epoch removed. For clarity, error ellipses are not shown, but the error ellipse on the scale arrow represents 68% confidence for a standard error of 7.2 mm (the r.m.s. of fit to the smoothed velocities). In practice, site velocities around the Gulf of Korinthos (which are determined from three or more occupations, black arrows) are known to within much better limits, and other site velocities (determined from only two occupations, grey arrows) are correspondingly less well resolved.

estimated parameters. It therefore seems that the most significant error in the reference frame realisation is that caused by errors in the setup over the base station, as atmospheric and other errors will tend to be averaged out over the span of each campaign.

4.3 Regional velocity fields

The first method of analysis that I have used is to attempt to fit low-order polynomials to the velocity field using the program `svdvel2` described in Section 3.2. Coordinate sets from all epochs within the interval 1991 – 1996 were used, with displacements for the 1995 Egion earthquake (Subsection 5.2.2) subtracted from the coordinates for 1995 and 1996 so as to leave only the inter-seismic regional strain. If the displacements can be fitted within their errors by a low-order polynomial, it may indicate that the dynamics of the crust may be controlled by processes acting at depth, with the upper crust merely a 'passenger' on the surface, but if the displacements cannot be fit in such a way, it may indicate that the heterogeneities of stress and strength in the upper crust or strain localisation in the lower crust or lithospheric mantle have an important effect on the surface deformation.

The computed displacement field for a 1^{st} order polynomial fit to the east ($\dot{u}$) and north ($\dot{v}$) velocities (3 parameters each) is shown in Figure 4.9. The r.m.s. weighted residuals in $\dot{u}$ and $\dot{v}$ are 1.20σ and 1.90σ respectively, the worse fit for $\dot{v}$ reflecting greater variability of north-south strain. As both r.m.s. weighted residuals are greater than unity, the hypothesis that the region is experiencing uniform strain (linearly varying velocity) can be rejected with 68% but not 95% confidence. In particular, the larger and more systematic residuals around the Gulf of Korinthos indicate that greater north-south strain may take place in this area. Figure 4.9 also shows the results of a 2^{nd} order polynomial fit to the data. The r.m.s. weighted residual in $\dot{u}$ is only marginally reduced (to 1.19σ), but that in $\dot{v}$ is reduced to 1.71σ. The velocities near the boundaries of the region are significantly better fit, but there remains significant unmodelled strain in the western Gulf of Korinthos.

Figure 4.10 shows 3^{rd} and 4^{th} order fits to the velocity field. R.m.s. weighted residuals in $\dot{u}$ and $\dot{v}$ are 1.15σ and 1.65σ for 3^{rd} order and 1.14σ and 1.57σ for 4^{th} order. For the latter fit, systematic residuals have been eliminated in all areas except the Gulf of Korinthos (where marginally significant residuals remain), but spurious velocities (which will only increase at higher orders of polynomial) are generated near the edges of the grid, where the velocity field is extrapolated beyond the region of GPS observations. For this reason fits of higher order were not attempted, even though the 4^{th} order fit can be rejected at the 68% (but not 95%)confidence level.

From these results it appears that a low-order polynomial smooth velocity field cannot fit the GPS observations, although this conclusion cannot be drawn with 95% confidence. The implication of this is

that inter-seismic strain within central Greece must be localised in the crust in some way. In order to analyse crustal deformation on a smaller scale, preventing regions of localised strain from influencing the measurement of strain in other areas, I next compared strains estimated independently for small polygonal regions.

4.4 Finite-element strain analysis

The method that I have used to analyse the Central Greece network coordinates in terms of strain and rotation of the region is that of finite-element uniform strain tensor estimation (Section 3.3). In this approach, the network is divided up into polygonal regions and the (uniform) velocity gradient tensor within each region is estimated from the time series of site coordinates and their variance-covariance matrices. The choice of polygons is of necessity somewhat arbitrary, but reasonable criteria can be adopted. Here, I have chosen polygons so that in general they either bound the whole of a major feature (*e.g.* a segment of a gulf or uplifted landmass) or omit it entirely. Also, where possible the polygon is chosen such that the uniform strain is a good fit to all the coordinate observations that pertain to it (*i.e.* the r.m.s. weighted residual is close to or less than unity).

Coordinate sets from all epochs within the interval 1991 – 1996 were used, with displacements for the 1995 Egion earthquake subtracted from the coordinates for 1995 and 1996 (Subsection 5.2.2). The principal strain rates computed using the program `polystr2` for each of the chosen polygons are shown in Figure 4.11. Errors in the magnitude of the principal strain rates are estimated to be of order 0.1 ppm yr^{-1} throughout the network, but uncertainties in the directions of the axes are high, particularly in the northern part of the network (which has only been occupied 1991 – 1993), so the only regions in which strains and their directions are well resolved are the Gulf of Korinthos (extensional north – south strain, increasing from east to west), the Peleponnessos (northwest – southeast extension) and to the northwest of Evvia (northeast – southwest shear). For this reason, analysis of the dataset (Chapter 5) will focus on the Gulf of Korinthos, although the similarity between the overall pattern of strain and that derived from the hundred-year data of Davies *et al.* (1996) implies that the errors in the principal axis directions may be overestimated by a factor of 2–3.

The situation for rigid-body rotations (Figure 4.12) is similar. In many polygons, the uncertainties are sufficiently large that no significant rotation can be detected. However, significant clockwise rotations of up to 10° Myr^{-1} in the centre-west of the network, with lesser degrees of rotation to the east and west. This finding is in agreement with the results of Davies *et al.* (1996) who used classical triangulation and GPS over a 100-year period.

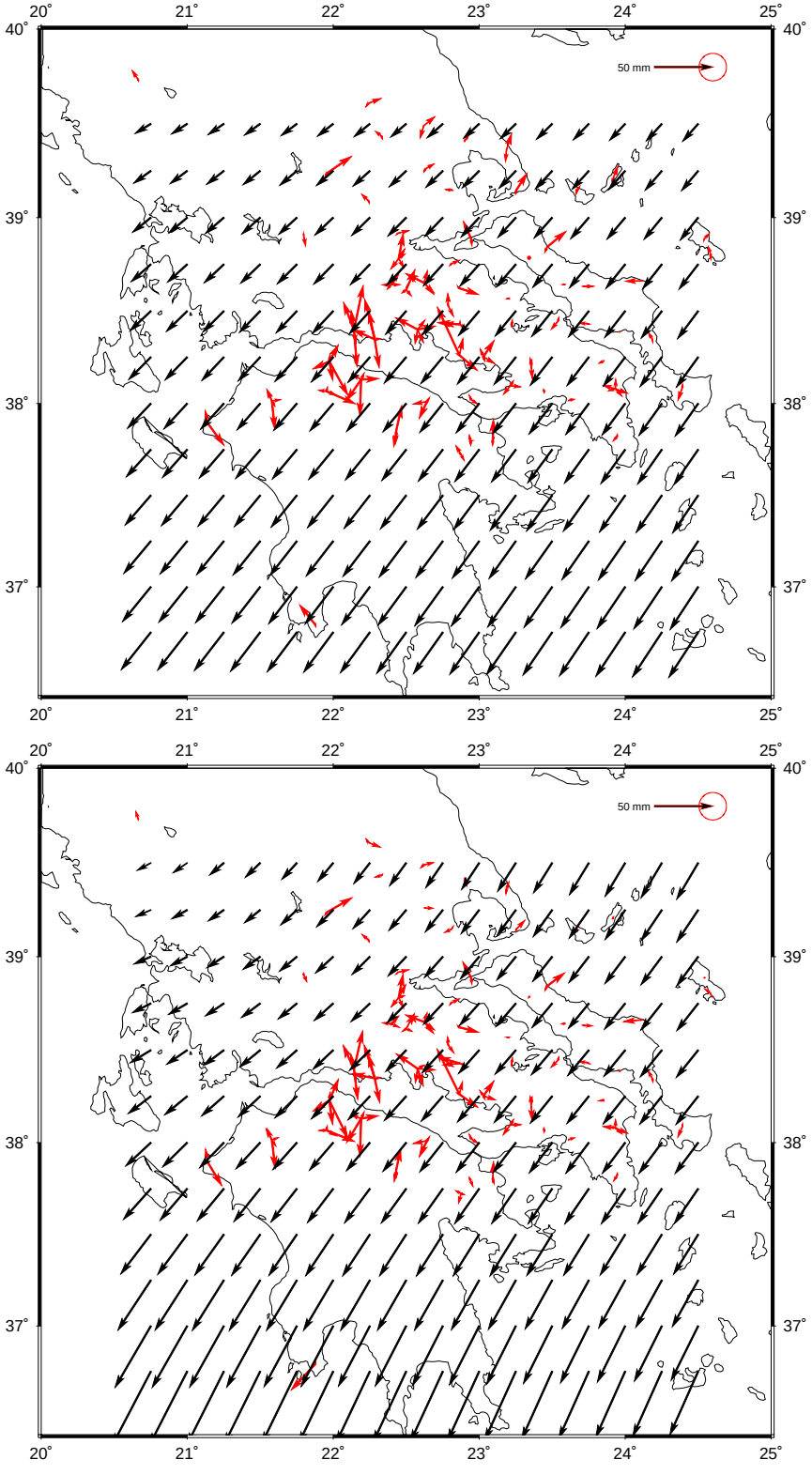

Figure 4.9: Velocity fields approximated as polynomials of order 1 (above) and 2 (below). Black arrows (mm yr^{-1}) show the velocity field sampled at points on a regular grid; red arrows (mm, red error ellipse on scalebar shows typical standard error of a coordinate of 7.2 mm) show residuals at each site for all occupied epochs.

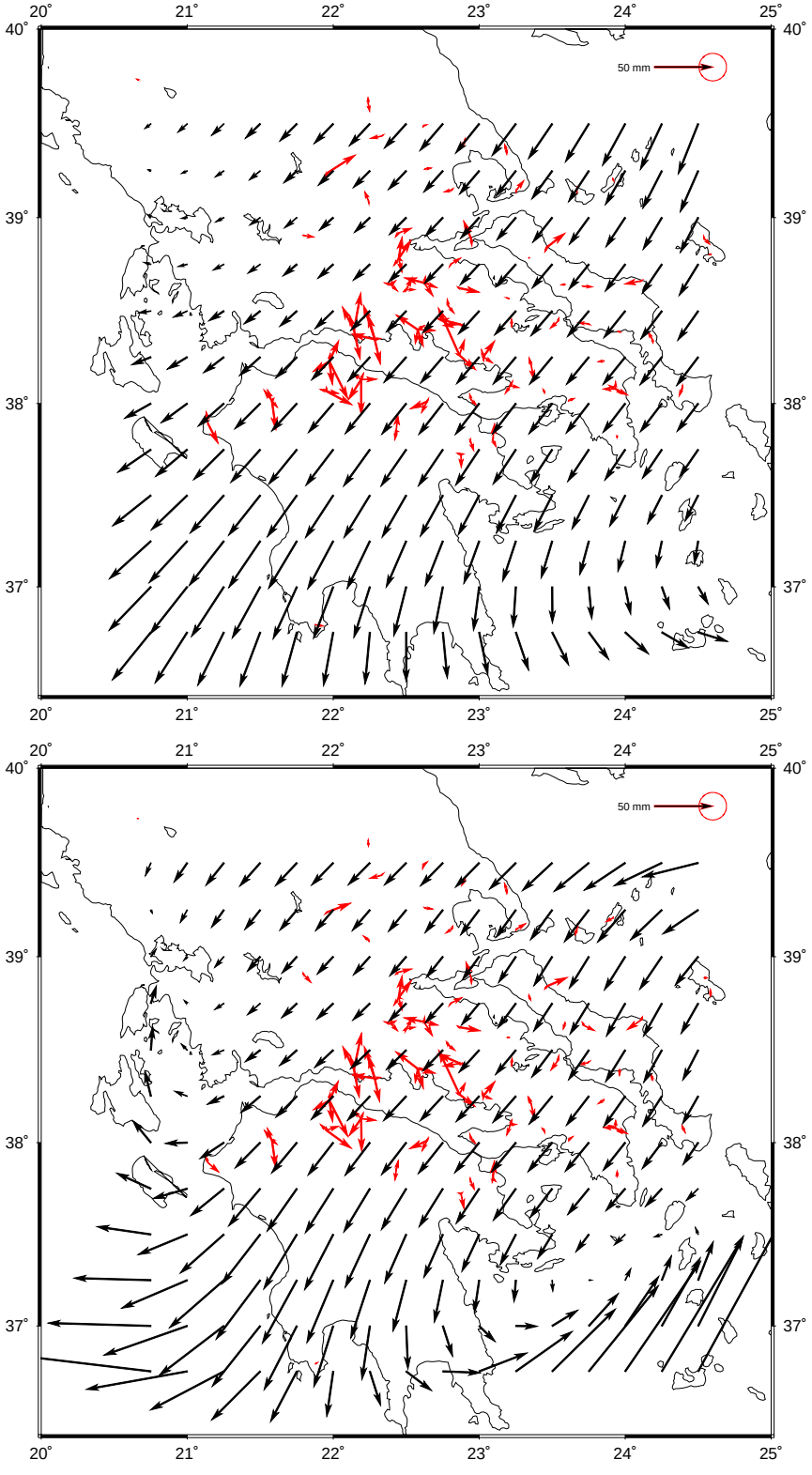

Figure 4.10: Velocity fields approximated as polynomials of order 3 (above) and 4 (below). Black arrows (mm yr^{-1}) show the velocity field sampled at points on a regular grid; red arrows (mm, red error ellipse on scalebar shows typical standard error of a coordinate of 7.2 mm) show residuals at each site for all occupied epochs. Large velocities at the corners of the lower plot are spurious results caused by extrapolation of the high-order velocity field outside of the region of constraining data.

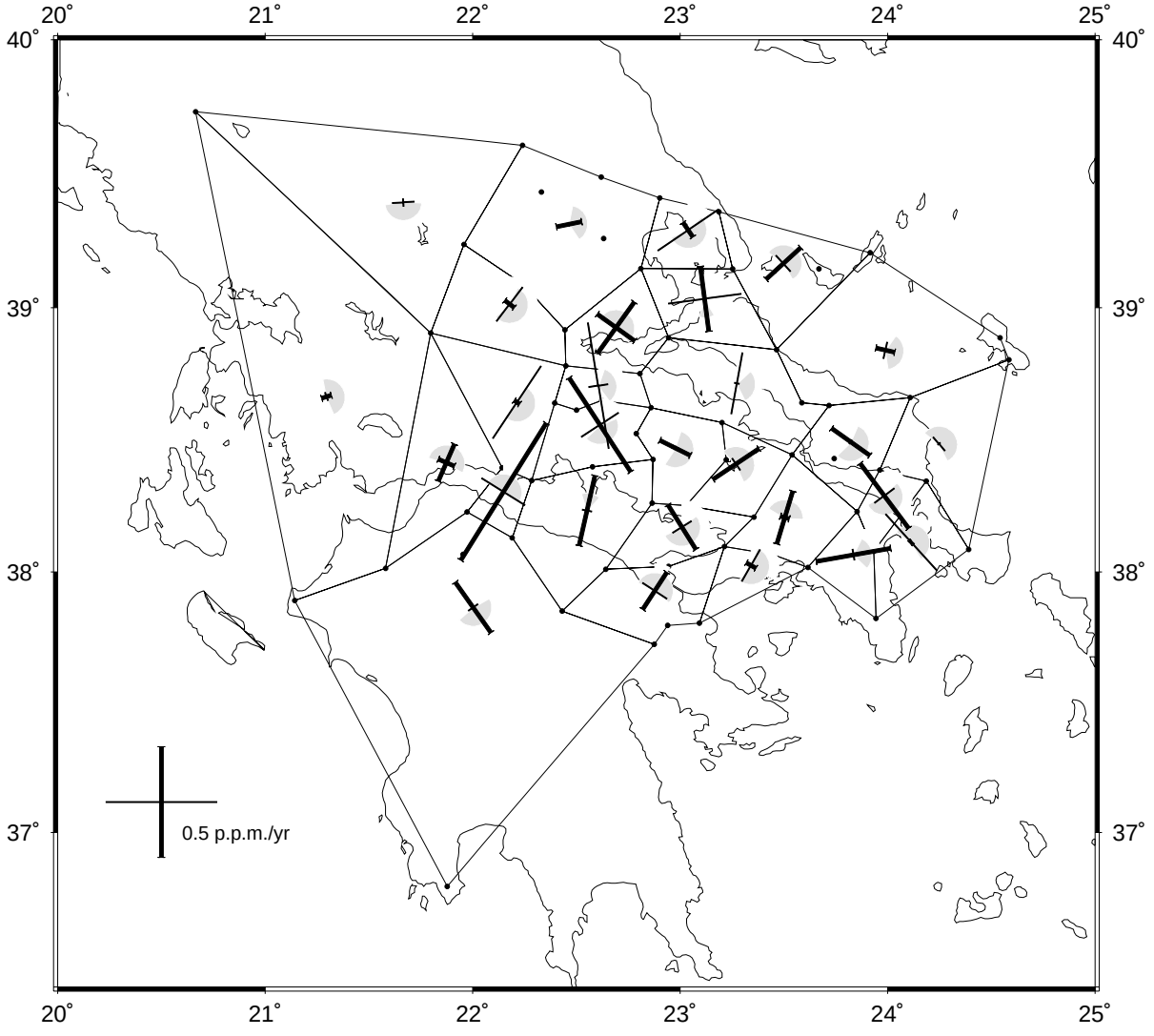

Figure 4.11: Principal strain rates computed for polygonal regions, Central Greece Network 1991 – 1996. Sites used in the computation are shown as black dots. Extensional axes are shown as thick lines, and compressional axes as thin lines. Uncertainties in the azimuth of the most extensional principal strain axis are shown by grey wedges ($\pm 1\sigma$ errors), but these estimates rely on a linearised relationship and comparison with the results of Davies *et al.* (1996) demonstrates that they are overestimated.

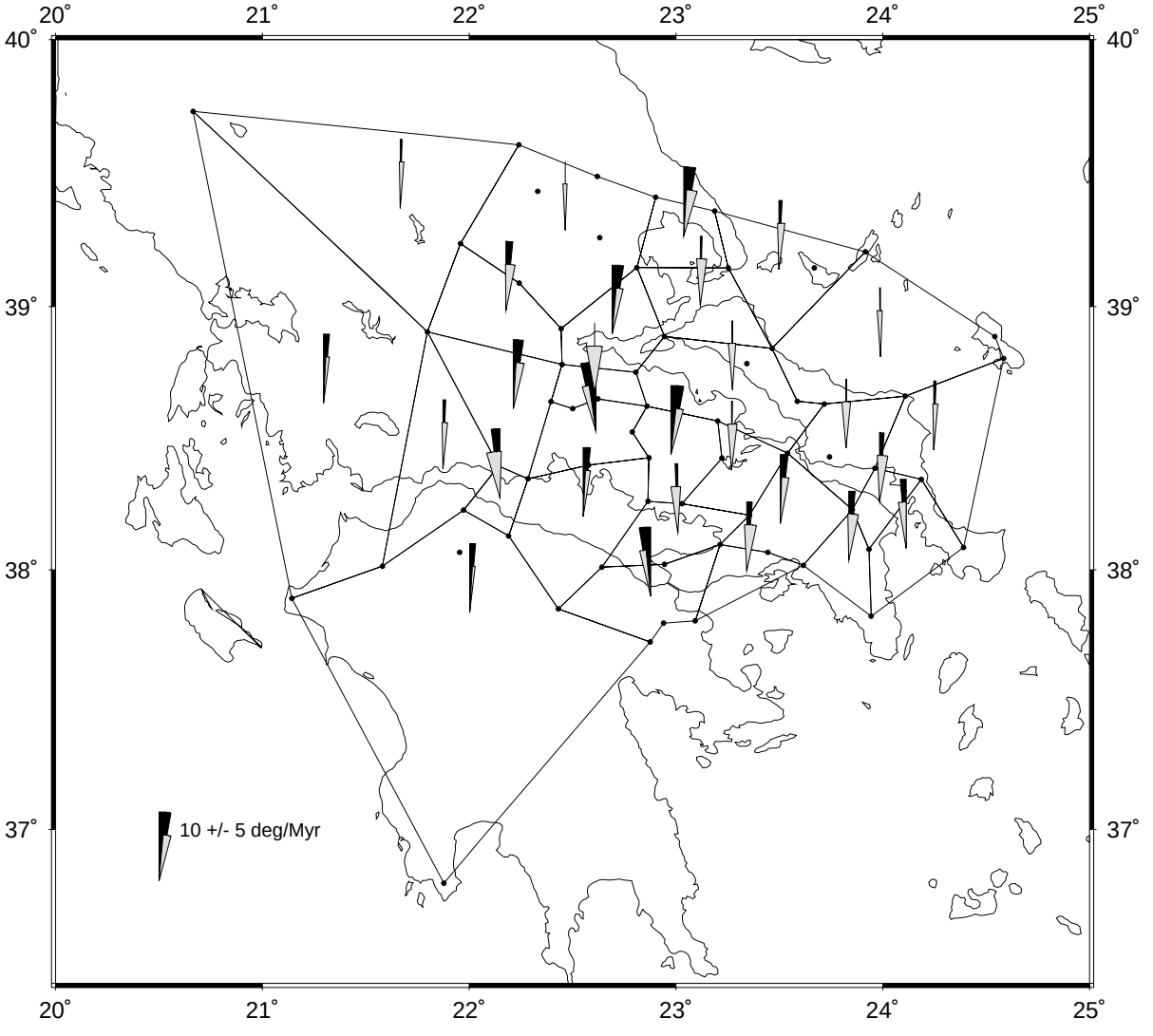

Figure 4.12: Geodetic rigid-body rotation rates computed for polygonal regions, Central Greece Network 1991 – 1996. Sites used in the computation are shown as black dots. Uncertainties are shown by grey wedges ($\pm 1\sigma$ errors).

4.5 Summary

Six sets of GPS measurements have been made of sites in the Central Greece Network in the interval 1989 – 1996. The first three sets cover the entire network, but the 1989 observations are of poor quality because dual-frequency observations were lacking for many sites, and because the GPS satellite constellation was incomplete, so only two epochs of measurement (separated by 1.6 years) of the complete network exist. The second three sets of observations include only those sites bordering the Gulf of Korinthos which were occupied after the 1995 Egion earthquake. The co-seismic displacements of this event significantly affect several sites, but can be modelled and subtracted leaving only the long-term (inter-seismic) strain.

The fixing of a base station at each epoch, necessary for regional relative GPS, can cause a global translation of the network if (as is likely) setup errors exist at the base station. When three or more epochs of observation exist, this network translation can be estimated if the assumption is made that site velocities are constant over time (*i.e.* co-seismic and other transient effects must be eliminated). It is demonstrated that reasonable network translations at each epoch can be applied to the Central Greece Network coordinate sets in order to yield time-invariant site velocities that fit the data to within expected error bounds.

Polynomial fits of up to 4^{th} order were attempted to the coordinate sets, though this continuum description of the velocity field can be rejected at the 68% (but not 95%) level of confidence. Significant, systematic residuals remain, indicating higher rates of localised strain near the (western) Gulf of Korinthos which is the best-sampled part of the network. Analysis of strain computed over independent polygonal regions reveals significant deformation in the Gulf of Korinthos, but elsewhere the time interval between GPS observations is so short that errors are too large for strain to be resolvable. For this reason, only a cursory analysis of the entire dataset has been given here, and the next chapter contains a full analysis of the subset of the network bounding the Gulf of Korinthos.

Chapter 5

The Gulf of Korinthos, 1989 – 1996

The Gulf of Korinthos is the largest and most active neotectonic feature within the Central Greece Network. It is an asymmetric graben, with large, normal faults on the southern side accounting for the majority of the Quaternary deformation, resulting in a sharp coastline interspersed with river deltas, and smaller, less active normal faults on the northern side which are subject to hanging-wall subsidence caused by the southern faults, generating a 'drowned' coastline. The extensional strain rate in the Gulf may be as much as one order of magnitude higher than that elsewhere in the Aegean (Armijo *et al.*, 1996). Particular interest in the Gulf arises partly because of this, but also because the eastern Gulf was the location of the well-documented 1981 Alkyonides large earthquake sequence (Subsections 3.5.4 and 3.7.3) and the western Gulf (site of the large 1995 Egion earthquake) has been identified as a region of high seismic hazard (Davies *et al.*, 1996). A problem which must be addressed is to relate the geodetic extension revealed by the Central Greece Network observations (Chapter 4), which includes both co-seismic and inter-seismic deformation, to the recent and long-term seismicity of the Gulf.

The western Gulf of Korinthos (Figure 5.1) is bounded by higher mountains than the eastern Gulf of Korinthos, implying that the crust in the west may be thicker than that in the east, for isostatic compensation to be in force. Armijo *et al.* (1996) argue, on the basis of greater width and depth of the eastern Gulf, that it has undergone a greater amount of extension, and that rifting has propagated westward. However, this line of argument relies on the supposition that crustal thickness was uniform at the start of rifting, which may be untrue as the previous structural grain of the region is aligned north – south, so east – west variations might be expected.

On the basis of marine terrace uplift rate modelling around the Xylokastro fault (Figure 5.1), Armijo *et al.* (1996) deduce a probable opening rate of 7 mm yr^{-1} (and at least 5 mm yr^{-1}) for the eastern Gulf of Korinthos. Similar work has been done on the Eliki fault to the west, yielding opening rates of

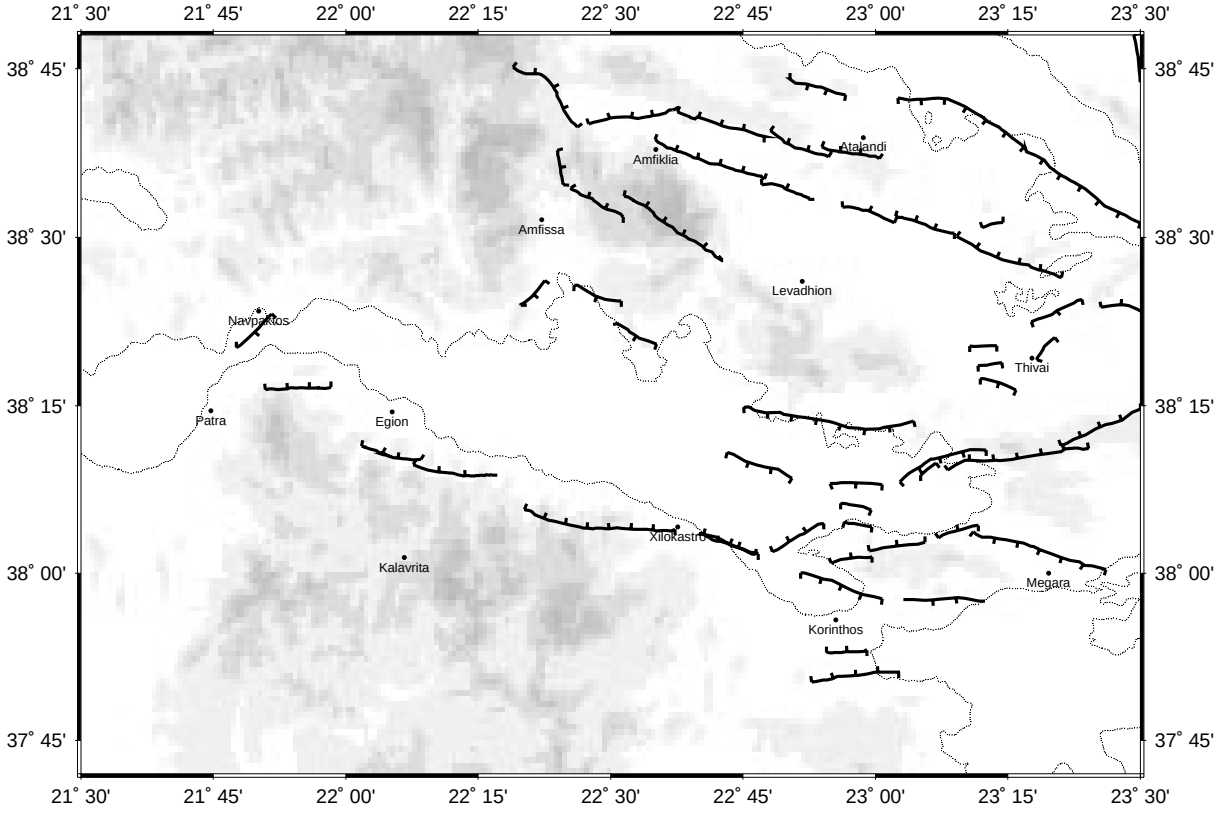

Figure 5.1: Major tectonic and other features of the Gulf of Korinthos. Recently active faults are shown as heavy lines with ticks on the downthrown side. Land over 500 m is shaded pale grey; land over 1000 m is shaded darker grey. Significant centres of population are named.

3 – 8 mm yr^{-1}, but the terraces are not so well constrained. Maximum ages of marine sediments uplifted to the flanks of the eastern Gulf of Korinthos are greater than those in the west and in the Gulf of Patras (Doutsos & Poulimenos, 1992), implying the eastern Gulf of Korinthos was the first part to subside below sea level. If initial crustal thickness was uniform this implies that the onset of extension was earlier in the east, but this does not enable an estimate of total extension to be made as the relative rates are not well constrained.

The eastern end of the Gulf is tectonically more complex than the west. The Perachora peninsula represents an uplifted horst block within the eastern Gulf, bounded by faults to the north and south. North-east of this, the antithetic Platea-Kaparelli Fault (see Subsection 3.5.4) appears to be segmented on a smaller scale than most faults in the Gulf, and its morphology has been cited as evidence of eastward propagation of rifting (King *et al.*, 1985a). However, other authors (Doutsos & Poulimenos, 1992; Le Pichon *et al.*, 1995; Armijo *et al.*, 1996) hold that the rift is propagating westward, based on the evidence of sediment ages

and apparent cumulative extension. These views are not necessarily incompatible: rifting may propagate in both directions from the centre. This thesis deals only with present-day deformation rates and cannot address this issue without additional information.

5.1 Seismicity in the region of the Gulf

Historically, the Gulf is the most seismically active part of mainland Greece. Both intermediate and shallow earthquakes occur under the region; the intermediate earthquakes (Figure 5.2) are related to the subduction of African seafloor underneath the European plate (in its loosest sense) and are sufficiently few and far away that they do not directly affect the surface deformation. Shallow earthquakes (less than 40 km deep) are clearly distinguishable, although these depths from the ISC catalogue are not as reliable as relocated or waveform-modelled solutions. The thickness of the seismogenic layer in Greece appears to be 10 – 15 km from relocated depths of large earthquakes (*e.g.* Taymaz *et al.*, 1991) and microseismicity studies (*e.g.* Rigo *et al.*, 1996), with the majority of earthquakes nucleating at 6 – 10 km depth.

5.1.1 Historical seismicity

Ambraseys & Jackson (1990) have compiled a catalogue of earthquakes in central Greece between 1890 and 1988 that is complete for events of $M_s \geq 5.8$, and this can be used to estimate the long-term strain rate using the method of Kostrov (1974). Catalogues of earlier seismicity exist (Papazachos & Papazachos, 1989; N.N. Ambraseys & J.A. Jackson, pers. comm.), but are based on less reliable information so are not used for quantitative comparison here.

The largest uncertainty in older focal mechanisms is that in scalar moment M_0, because this is deduced via the M_s – M_0 relationship from the surface-wave magnitude M_s which in turn is estimated from felt intensities. The M_s – M_0 relationship is itself somewhat uncertain, and this will affect scalar moments of more recent events also. Use of the global relationship leads to an overestimate of up to 30% in scalar moment M_0 when compared with that derived via the local relationship (Ekström & Dziewonski, 1988; Ekström & England, 1989; Jackson & McKenzie, 1988a). Even events constrained by waveform modelling are still subject to error in M_0, and the combined effect of earthquakes of $M_s \leq 5.8$ may amount to 50% of the summed moment of larger events (Ambraseys & Jackson, 1990).

Focal mechanisms of events in the Gulf of Korinthos from the Ambraseys & Jackson (1990) catalogue are listed in Table 5.1 and shown in Figure 5.3. It can be seen that the majority of large events prior to 1989 have occurred in the eastern Gulf, in particular during the 1981 Alkyonides earthquake sequence (see also Subsections 3.5.4 and 3.7.3). This pattern may not be representative of the longer-term seismicity.

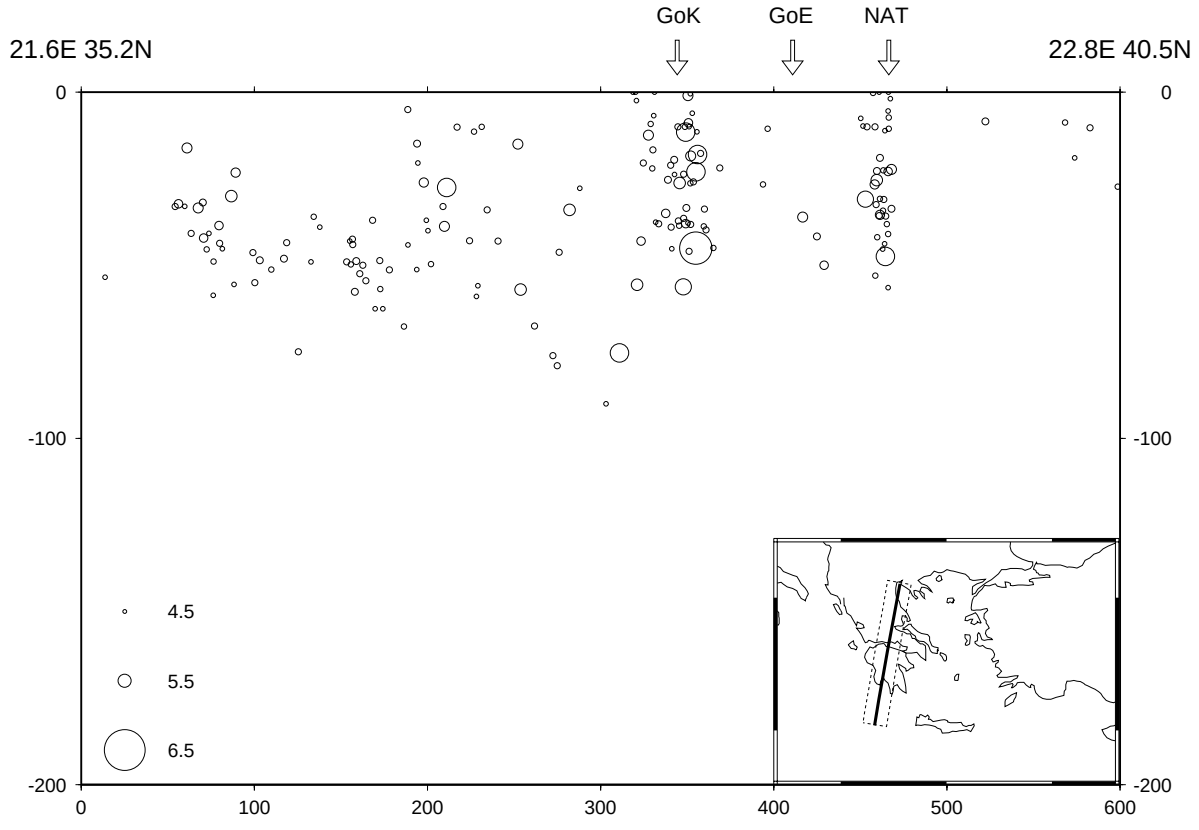

Figure 5.2: Vertical cross-section perpendicular to the Gulf of Korinthos (inset shows line of section and region from which earthquakes are projected onto profile). Earthquakes ($M_s \geq 4.0$, from ISC catalogue 1967 – 1994) are shown as circles; symbol area is proportional to scalar moment. Depths are not constrained by relocation or waveform modelling. GoK = Gulf of Korinthos; GoE = Gulf of Evvia; NAT = North Aegean Trough. Depth is in km; horizontal scale shows distance northward along section in km.

Name	Date	Location		M_s	$M_0/10^{18}$ N m		Strike	Dip	Rake
		Lat. ° N	Lon. ° E		Local	Global			
Fokis	30 May 1909	38.44	22.14	6.3	3.2	4.2	090°	74°	−115°
Nafpaktos	24 Dec 1917	38.45	21.75	5.8	1.2	1.3	090°	74°	−115°
Korinthos	22 Apr 1928	38.03	22.83	6.3	3.6	4.9	285°	40°	−70°
Eratini	06 Jul 1965	38.27	22.37	6.4	4.2	5.8	090°	74°	−115°
Antikira	08 Apr 1970	38.23	22.60	6.2	2.6	3.4	075°	67°	−94°
Nafpaktos	31 Dec 1975	38.50	21.65	5.7	0.9	1.0	090°	45°	−90°
Alkyonides	24 Feb 1981	38.10	22.84	6.7	8.8		264°	42°	−80°
Alkyonides	25 Feb 1981	38.13	23.05	6.4	4.0		241°	44°	−85°
Alkyonides	04 Mar 1981	38.18	23.17	6.2	2.7		230°	45°	−90°
Egion	15 Jun 1995	38.10	22.46	6.2	6.0		265°	43°	−103°

Table 5.1: Shallow earthquakes in the Gulf of Korinthos of magnitude $M_s \geq 5.8$ (for which the catalogue is believed to be complete) 1890 – 1996, taken from Ambraseys & Jackson (1990) and Jackson *et al.* (1992), except for the 1995 Egion earthquake (Harvard CMT solution). The 1975 Nafpaktos earthquake (M_s 5.7) is also included. Where M_0 is computed from M_s, values using both global and local relationships are given; where M_0 is computed directly from waveform modelling, only this value is given.

Because the Gulf of Korinthos is one of the more populated parts of Greece, historical evidence for earlier earthquakes is relatively easily available (although its interpretation is far from straightforward), and from this it would appear that long-term seismicity (since 1690, J.A. Jackson, pers. comm.) has been similar in the eastern and western Gulf, with lower seismicity in the west only observed this last century.

5.1.2 Seismicity 1989 – 1996

Earthquakes occurring between epochs of observation of the Central Greece Network may have caused co-seismic displacement of sites which will interfere with the observation of inter-seismic strain rates. Shallow seismicity (above 40 km) since 1989 from the ISC catalogue is shown in Figure 5.4. The largest event during this period (apart from the 1995 Egion earthquake, Section 5.2) was the 18 November 1992 M_b 5.9 Galaxidi earthquake. If it can be demonstrated that this had insignificant effect on site displacments, then the same will be true of all lesser events.

The co-seismic effect of the Galaxidi earthquake on the network can be calculated using the Okada elastic model (Section 3.5). Figure 5.5 shows the co-seismic displacements predicted using the forward model. The largest displacement is of order 10 mm (site 32), comparable with GPS displacement measurement

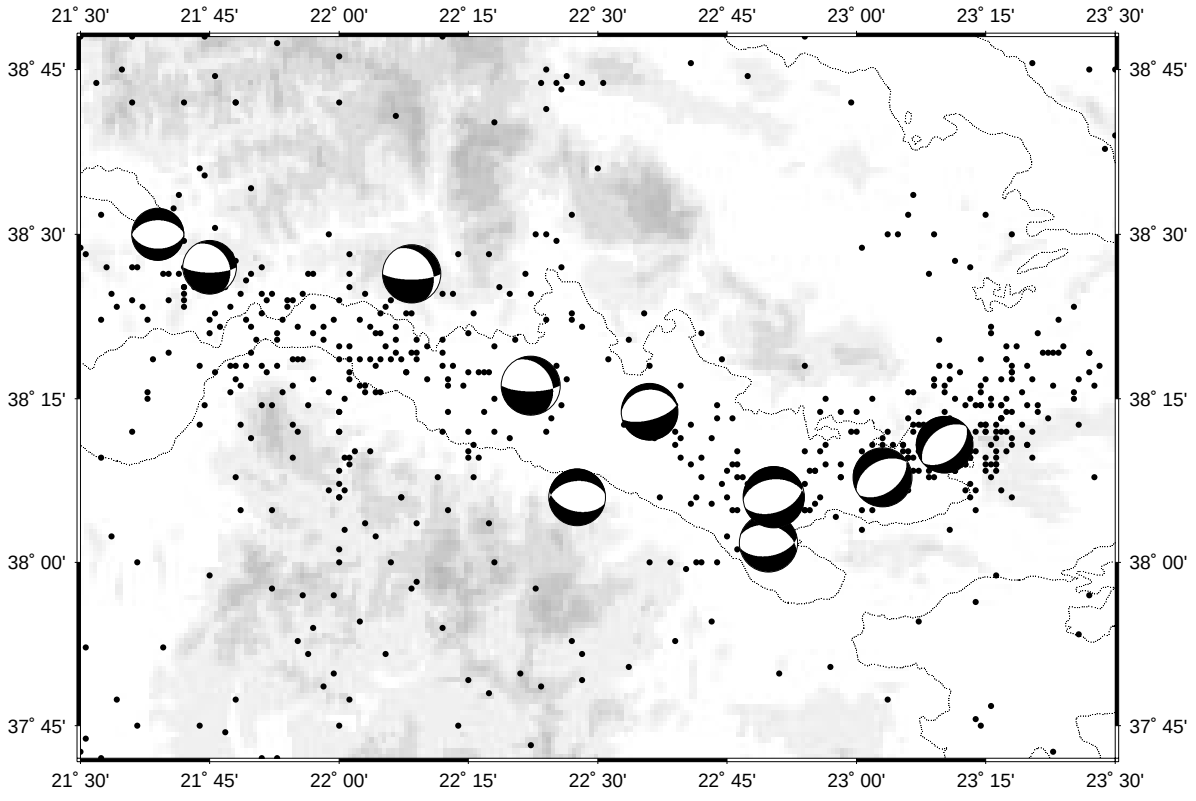

Figure 5.3: Source mechanisms of large earthquakes in the Gulf of Korinthos, 1890 – 1988, after Ambraseys & Jackson (1990). Shallow earthquakes (depth $\leq$ 40 km, $M_s \geq 3.5$) from the ISC catalogue 1964 – 1989 are also shown (black dots, 573 events).

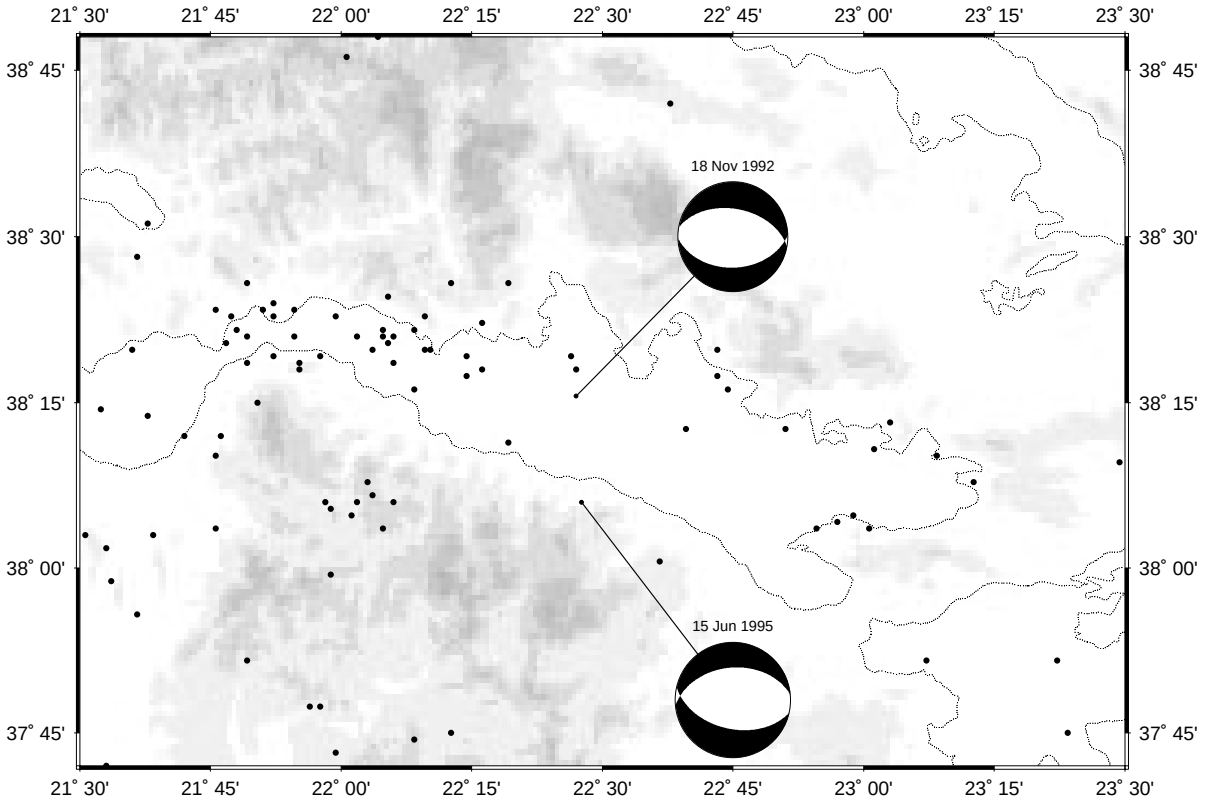

Figure 5.4: Seismicity in the Gulf of Korinthos, 1989 – 1996. All $M_s \geq 3.5$ earthquakes shallower than 40 km in the ISC catalogue for the period 1989 – 1994 are shown as black dots (89 events), and in addition the focal mechanisms of the 1992 $M_0 = 0.85 \times 10^{18}$ Galaxidi earthquake and 1995 $M_0 = 6.0 \times 10^{18}$ Egion earthquake are shown.

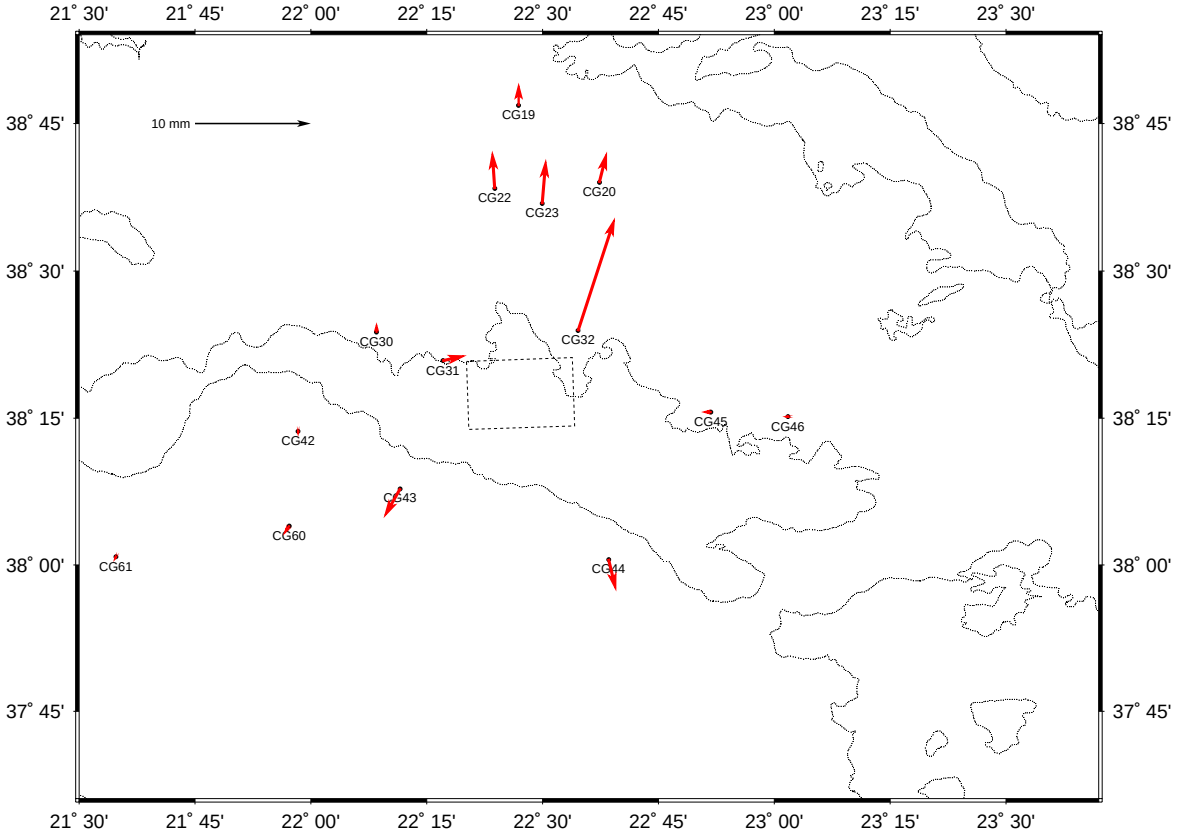

Figure 5.5: Forward model of the 1992 Galaxidi earthquake displacements computed for stations in the Central Greece Network The surface projection of the (south-dipping) model fault plane is shown by the dotted rectangle.

errors, with the majority of nearby displacements only of order 2 – 3 mm, well within the margin of observational geodetic error. Only if a smaller earthquake were very much closer to a site would it disturb the site by as much as this. Therefore, it can be stated that no earthquake during the interval of Central Greece Network observations has significantly affected site displacements, with the possible exception of the 1995 Egion earthquake which is now discussed.

5.2 The 1995 Egion earthquake

On Thursday 15 June 1995 a large (M_s 6.2) earthquake struck the western Gulf of Korinthos at 0015 UT (0315 local time), causing damage to buildings in and near the town of Egion (Figures 5.1 and 5.6) and killing 26 people (including 15 tourists when a hotel collapsed). Although the event was smaller in moment ($M_0 = 6.0 \times 10^{18}$ N m according to the Harvard CMT solution) than the 1995 Grevena earthquake (M_0

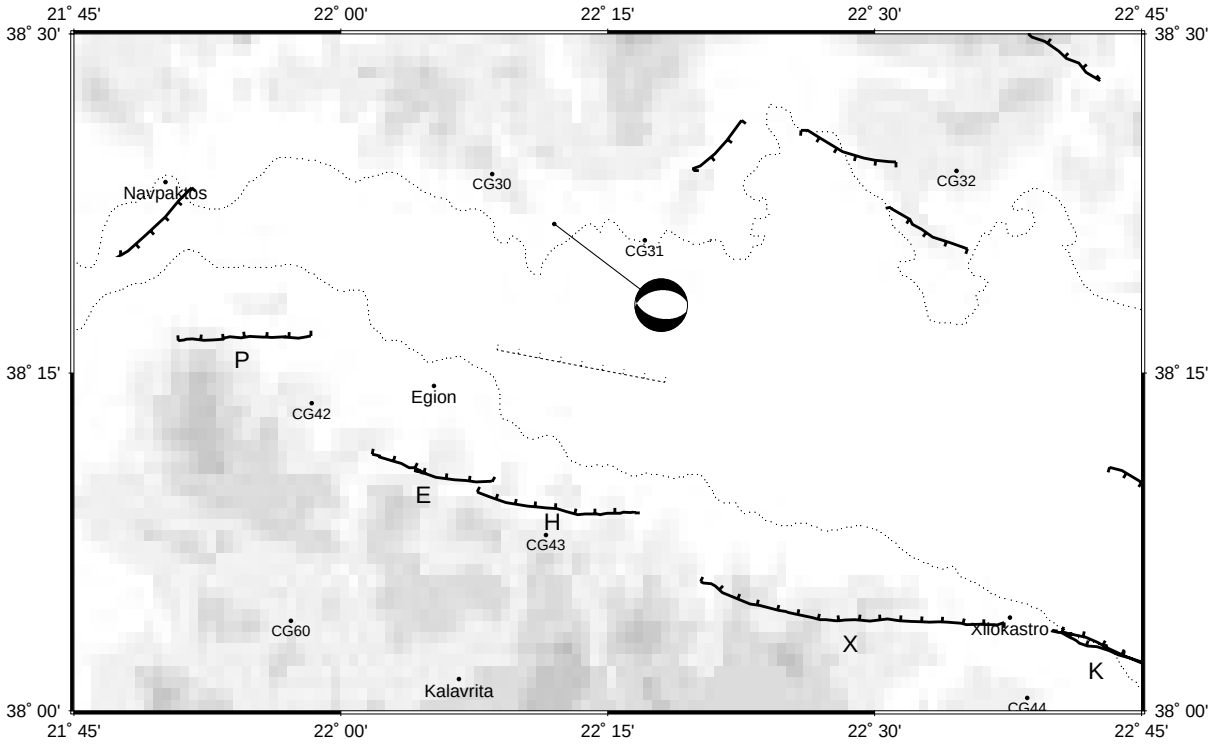

Figure 5.6: Location of the 15 June 1995 Egion earthquake (after Bernard *et al.*, 1996) in relation to nearby neotectonic features (fault scarps have ticks on the downthrown side). The location of the geodetically-modelled fault scarp of the Egion earthquake is shown by the dotted line. Topographic shading increases at 500 m vertical intervals. Nearby points in the Central Greece Network are also shown. P = Psathopyrgos Fault; E = Egion Fault; H = Eliki Fault; X = Xilokastro Fault; K = Kiato Fault.

$= 7.6 \times 10^{18}$ N m likewise) and damage to property was less extensive, there were no large foreshocks (unlike Grevena) to alert the local population to the danger and this increased the human cost.

A team from Oxford University was in Egion four days after the event in order to densify the Central Greece GPS network in the area (Subsection 4.1.4) to enable future study of post-seismic deformation, and to conduct seismotectonic observations. No clear fault break was found on land, either on the likely candidate faults (Psathopyrgos, Egion, Eliki) to the south of the Gulf or elsewhere. In view of the greater amount of damage on the south side it seemed likely that the earthquake had ruptured a north-dipping submarine fault just offshore of Egion.

Epicentre		Moment	Strike	Dip	Rake	Depth
Lat.	Lon.	$/10^{18}$ N m				/km
38.49°	22.25°	5.5	288°	24°	−56°	10

Table 5.2: European – Mediterranean Seismological Centre 'quick' solution (F. Ramon, pers. comm.) for the 15 June 1995 Egion earthquake, used in initial post-seismic visco-elastic modelling.

5.2.1 Location and focal mechanism

Seismological analysis of the event is complicated by a major (M_b 5.5) aftershock occurring 15 s after the mainshock and therefore lying within the seismic coda (J.H. Woodhouse, pers. comm.). The epicentral location of the earthquake after the relocation of Bernard *et al.* (1996) is shown in Figure 5.6. Estimated uncertainties in horizontal position are of order 2 km, and hypocentral depth is estimated to be 10.5± 3 km.

The nodal planes of the Harvard CMT solution strike 265° and 102°, dipping 43° and 48° with rake −103° and −78° respectively, whereas the body-wave solution of Bernard *et al.* (1996) strikes 281°, with the shallower dip 23° and rake -72° (north-dipping plane only), similar to the European – Mediterranean Seismological Centre 'quick' solution (F. Ramon, pers. comm., Table 5.2). This north-dipping plane seems very shallow compared with previous understanding of the behaviour of normal faults in Greece (Jackson & White, 1989; Roberts & Jackson, 1991), which is that faults are steeper at the surface and do not flatten with depth, with deformation in the lower crust being accommodated in a distributed manner. However, microseismicity studies in the western Gulf of Korinthos (Rigo *et al.*, 1996; Rietbrock *et al.*, 1996) have offered other evidence for low-angle (15± 10°) north-dipping seismogenic detachments in this region.

Bernard *et al.* (1996) have also combined GPS and SAR data to generate a detailed model of the 1995 earthquake that is compatible with the seismological data, consisting of a shallow, steeply-dipping (55°) fault plane connected to the top of a deeper, gently-dipping (23°) fault plane (Table 5.3). The total moment of the event is 4.5×10^{18} N m, between the body-wave moment (2.4×10^{18} N m) and the Harvard CMT moment (6.0×10^{18} N m). The upper fault plane does not correspond to the onshore Eliki Fault (Figure 5.6) but outcrops offshore, 9 km to the north. Bernard *et al.* (1996) have suggested that it may be connected at its western end to the Egion fault.

5.2.2 Co-seismic effect on the network

Raw displacements of Central Greece Network sites from May 1993 to June 1995 (just after the earthquake) are displayed in Figure 5.7 (black arrows). Anomalous northward displacements of sites 30 and 31 relative to the Peleponnessos are evident, and a slight increase in the southward displacement of site 43 relative

Scarp		Total slip	Strike	Dip	Rake	Length	Dmin	Dmax	Displacement /m	
Lat.	Lon.	/m				/km	/km	/km	Dip-slip	Strike-slip
38.26°	22.23°	0.54	281°	55°	−90°	14	0.5	5.7	0.54	0.00
38.18°	22.17°	0.54	281°	23°	−72°	18	5.7	9.8	0.51	−0.17

Table 5.3: Two-segment faulting model for the 15 June 1995 Egion earthquake, after Model A of Bernard *et al.* (1996). Scarp coordinates refer to the up-dip projection of the centre of the fault plane.

to neighbouring sites can also be seen. However, the density and distribution of observations are not sufficient to enable an inversion for the earthquake parameters to be carried out as has been done for the 1995 Grevena earthquake (Chapter 6).

Instead, the model of Bernard *et al.* (1996), which is based on much denser GPS displacement observations from which the secular strain has been removed, and SAR interferometry, is used to remove the co-seismic displacement from the Central Greece Network observations, so that only the inter-seismic displacement remains. The displacements computed by the forward model (Table 5.3) are shown in Figure 5.7 (red arrows). The forward model matches the anomalous motions of Central Greece Network sites, at least qualitatively, which is independent confirmation of its verisimilitude. If anything, the model seems to over-correct for co-seismic displacement, since model displacements close to the fault are on the whole larger than the actual displacements, which must also include inter-seismic strain.

5.2.3 Post-seismic network design

The main purpose of the June and October 1995 Central Greece Network occupations was to establish a densified network so that post-seismic deformation could be investigated. For this, it was necessary to choose additional points with the aid of a model of the post-seismic displacement field. Using the Rundle model (Section 3.7) for visco-elastic relaxation in a half-space underlying a 10 km elastic layer, the computed surface displacements shown in Figure 5.8 were obtained. The European – Mediterranean Seismological Centre 'quick' solution (F. Ramon, pers. comm., Table 5.2) was used as the basis for the model (with a fault of length 20 km penetrating to 10 km depth assumed) because it was the best solution based on local observations available at the times of initial network design (June and September 1995).

The sites in the new network are shown in Figure 5.8. Initially (June 1995), post-seismic displacements following dip-slip co-seismic motion only were modelled to save time, but later (September 1995), post-seismic displacements arising from the strike-slip component of co-seismic motion were added. The early modelling predicted a displacement field symmetric along strike, with the line of maximum displacements

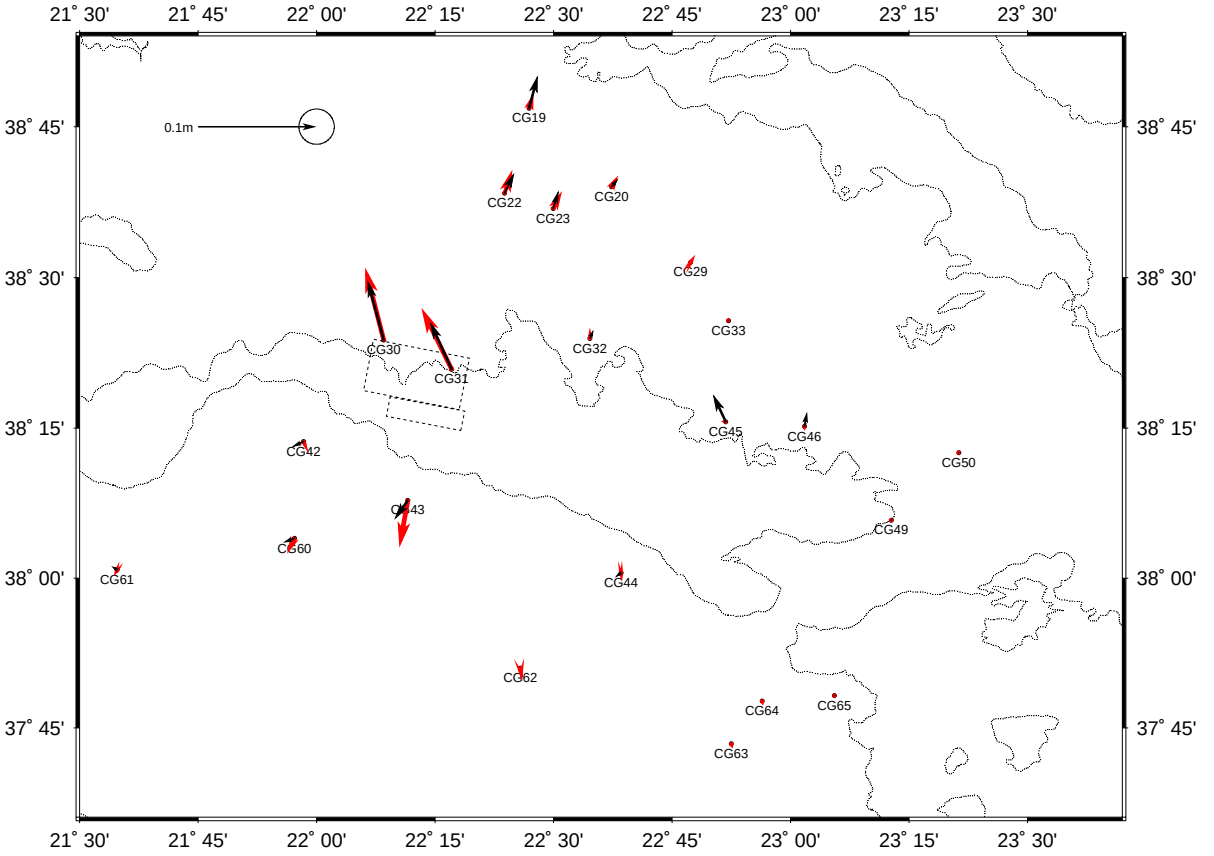

Figure 5.7: Central Greece site displacements in the Gulf of Korinthos, May 1993 – June 1995 (black arrows, not all sites have observations). Site 54 (DION, not in area of plot) is constrained to be stationary. A typical error ellipse of 10 mm in a displacement observation is shown on the scalebar. Also shown are computed displacements (red arrows), using a forward model after Model A of Bernard *et al.* (1996). The surface projections of the (north-dipping) model fault planes are shown by the dotted rectangles.

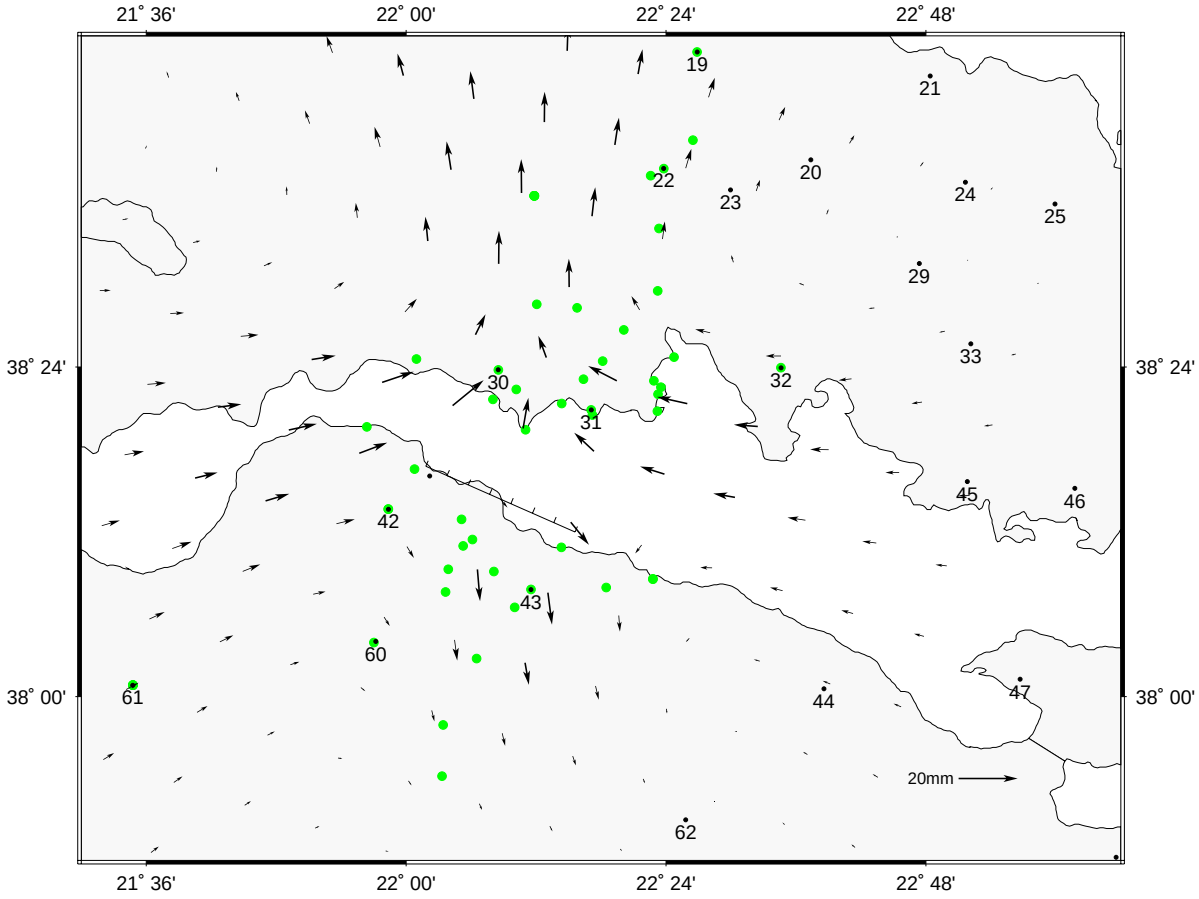

Figure 5.8: Forward model of post-seismic displacements after the 15 June 1995 Egion earthquake (European – Mediterranean Seismological Centre 'quick' solution, for fault 20 km long), after one Maxwell time. All sites in the Egion Post-seismic Network are shown as dark grey dots; Central Greece Network sites are numbered. The model fault scarp is shown as a black line with ticks on the downthrown side.

running north-northeast towards the vicinity of Central Greece Network site 19, and so several sites along this transect were included in the Egion Post-seismic Network. The addition of the response to strike-slip motion had the effect of shearing the displacement field left-laterally, so that the line of maximum displacements runs approximately north – south. In September 1995, sites along this section were added to the network, but site choice was limited by steep valley sides that limit satellite visibility at many locations, and by poor road communications.

Since the September 1995 campaign, the model of Bernard *et al.* (1996) has become available. Post-seismic displacements based on this co-seismic solution are shown in Figure 5.9. Displacements in this model are larger the north side of the Gulf, and show a greater contrast between the maximum displacements down-dip of the fault plane and the displacements to either side. The line of maximum displacements again runs

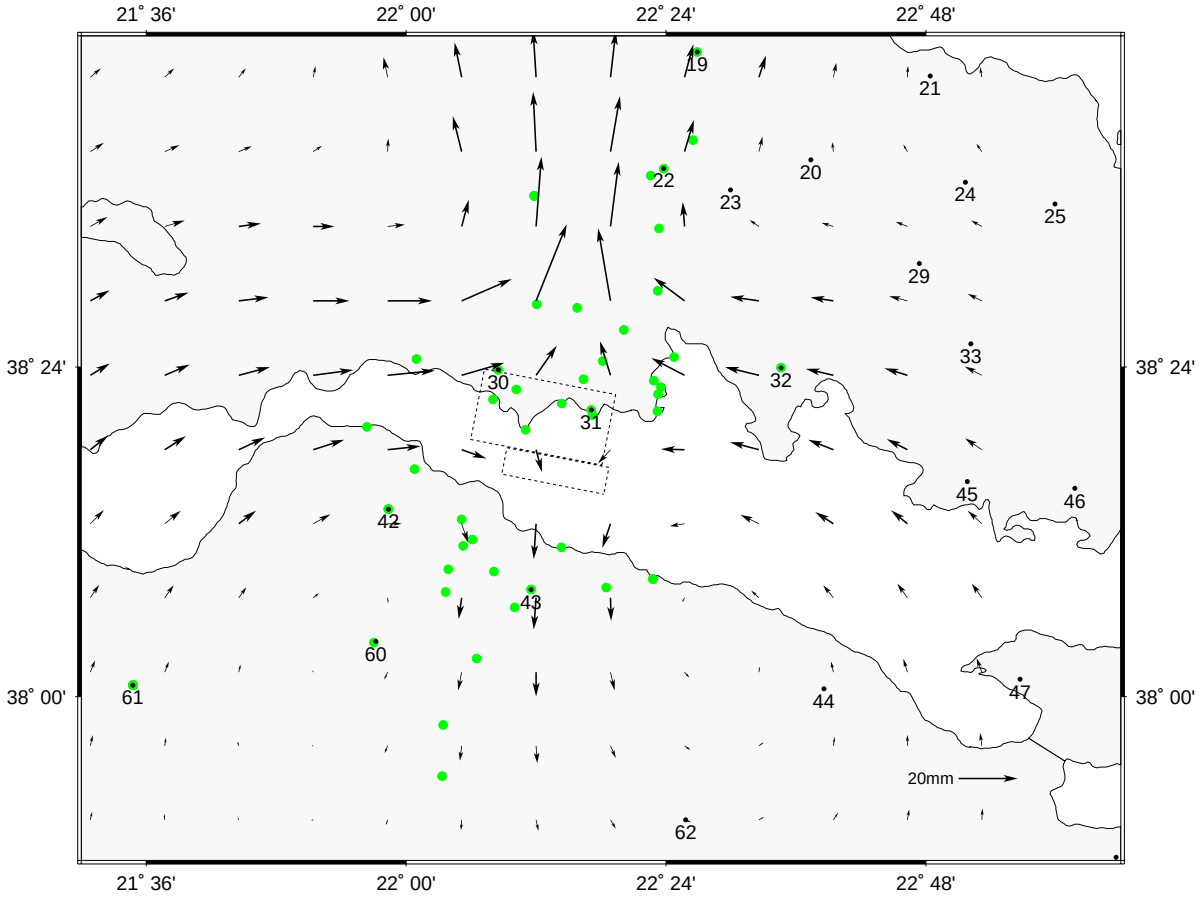

Figure 5.9: Forward model of post-seismic displacements after the 15 June 1995 Egion earthquake (Bernard *et al.* (1996) Model A solution), after one Maxwell time. All sites in the Egion Post-seismic Network are shown as dark grey dots; Central Greece Network sites are numbered. The surface projections of the model fault planes are shown by dotted rectangles.

north – south, and so it may be necessary to install yet more sites along this section.

Unlike the situation in Grevena (Subsection 6.1.2), site choice in this network is limited by the presence of a large body of water which precludes the use of stations close to the fault (and may also affect the validity of the model because the model assumes a horizontally-uniform geometry). However, the sediments on land to the north of the Gulf are more consolidated than those near Grevena, so site stability is not such a problem in this area, although it is of concern to the south of the Gulf.

Another problem relating to this experiment is that current deformation rates (as revealed by seismicity) are higher in the Gulf of Korinthos than in the Grevena region, and this background signal must be separated from the post-seismic signal to enable analysis of the latter. The inter-seismic strain in the Gulf of Korinthos can be constrained by Central Greece Network observations in the area. Post-seismic

deformation in Egion will be smaller than in Grevena because the size of the initial forcing perturbation is smaller, so it is necessary that background deformation in the Gulf of Korinthos is constrained in this way.

5.3 Variation in geodetic extension rate along the Gulf

From Figure 4.8 and the results of Section 4.4 it would appear that the rate of extension varies along the length of the Gulf, being higher in the west than the east. As the seismological record over the last hundred years suggests that the reverse should be the case, I have conducted a more detailed investigation of this phenomenon.

5.3.1 Computation of smoothed velocities

To a first order approximation, the sites on the Peleponnessos are not moving with respect to each other (Figure 4.8), and this observation provides the basis for a reference frame in which the extension in the Gulf of Korinthos can be easily visualised. Using the `velsmooth` algorithm (Section 3.1), the smoothed velocities of sites on the north side of the Gulf are computed from coordinate observations during the interval 1991 – 1996 (corrected for the co-seismic effects of the 1995 Egion earthquake). Sites in the northern Peleponnessos (42, 43, 44, 60, 61, 62, 63, 64, 65) have their velocities constrained to be zero, and two translation parameters are estimated for each epoch except May 1993, the reference epoch..

The results of this fit are displayed in Figure 5.10. The r.m.s. residual (for all sites) is 7.5 mm; for constrained sites it is 8.3 mm and for unconstrained sites it is 7.0 mm, indicating that the assumptions of zero and smooth velocities respectively are both valid to within close to 1σ error. The smoothed velocities of the sites north of the Gulf in this new reference frame are generally in the direction 020°, perpendicular to the strike of the Gulf. The next step is to rotate the reference frame so that velocity in this direction can be related to distance along strike in the Gulf.

The components of velocity along 020° of the northern sites are shown as a function of distance along the Gulf in Figure 5.11. The data exhibit a reasonable approximation to a linear trend (red line in Figure 5.11), and clearly exhibit significantly greater velocities (*i.e.* greater extension) in the western Gulf of Korinthos. Sites further away from the Gulf do not show a significantly different pattern of motion from those closer to the shoreline, which implies that the scale of localisation of the deformation is comparable with or smaller than the width of the Gulf.

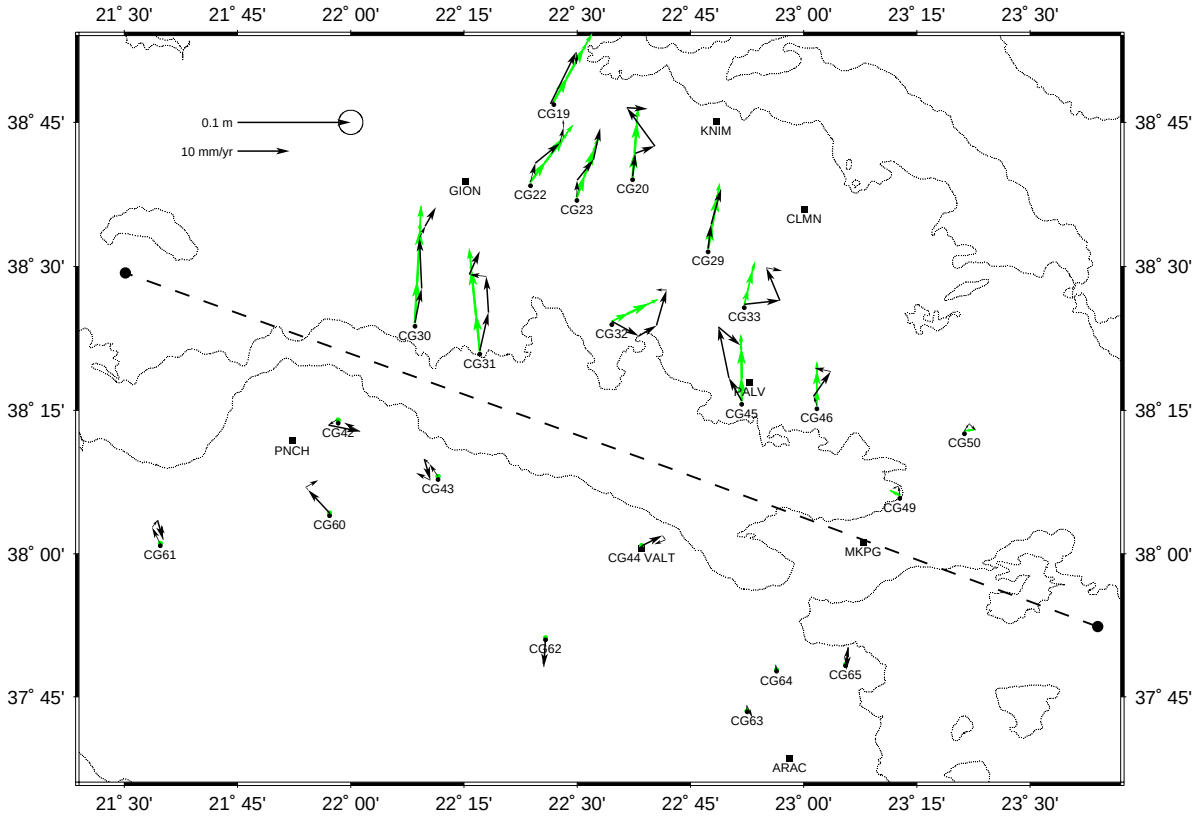

Figure 5.10: Smoothed displacements over the interval 1991.78 – 1996.39 (green arrows) of Central Greece sites relative to the Peleponnessos. Actual epoch-to-epoch displacements (after the network translation at each epoch has been removed) are also shown (black arrows, number of arrow segments may vary as not all sites were occupied at each epoch, but all sites shown were occupied in 1991, 1993 and 1996), and show insignificant motion of Peleponnessos sites; whereas sites to the north of the Gulf show steady movement within error. The scalebars show both displacement (with a typical error ellipse of a single coordinate, 7.2 mm) and velocity scales. Sites from the 1890s network later referenced are shown as squares, and the approximate line of strike of the Gulf (020°) is shown by black dashes.

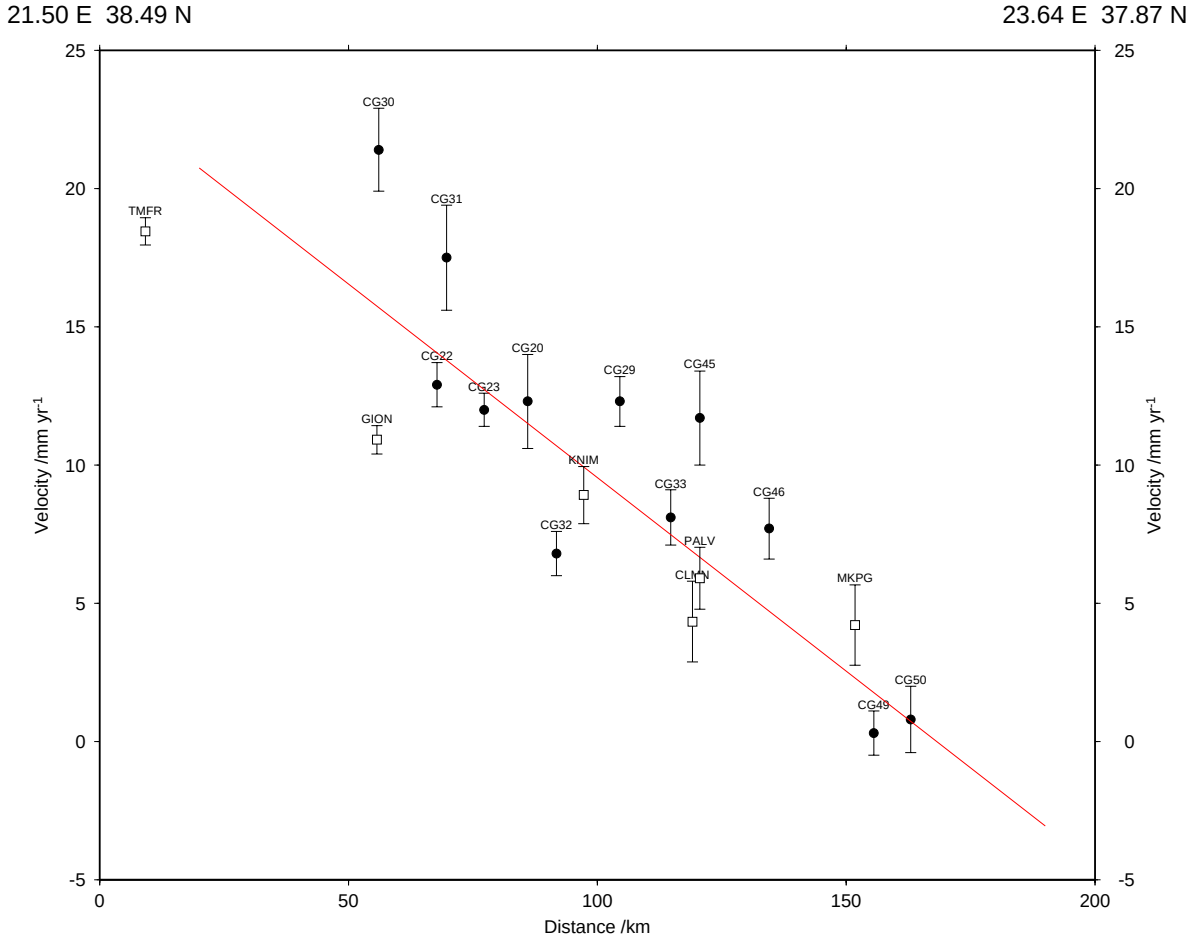

Figure 5.11: Velocity components along 020° smoothed over the interval 1991.78 – 1996.39 of Central Greece sites (filled circles), relative to the Peleponnessos. The x-axis shows eastward distance along the line of the section indicated in Figure 5.10, and the red line shows the best-fitting weighted linear trend to the Central Greece data. Also shown are the velocity components estimated over the interval 1892 – 1992 along 020° for 1890s network sites (open squares). Note that there is no significant difference between the behaviour of sites close to and far from the Gulf, or between five-year and hundred-year estimates.

5.3.2 Comparison with longer-term geodetic studies

The region of the Gulf of Korinthos is also covered (although not as densely) by the 1^{st}-order triangulation network established in 1890 – 1900 ("the 1890s network") which has been recently (September 1988 and June 1992) reoccupied with GPS (Billiris *et al.*, 1991; Davies *et al.*, 1996; Davies, 1996). Davies *et al.* (1996) have defined the reference frame of the triangulation survey by assuming that a north – south baseline in the Peleponnessos has not strained over the hundred-year interval. Their justifications for this approach are firstly the low seismicity of the central Peloponnessos (which in any case would be expected to extend east – west, perpendicular to their fixed baseline, on the basis of the orientation of normal faults in the region), and secondly the insignificant relative geodetic motion of other sites in the Peleponnessos once this assumption has been made.. The orientation of their reference frame is obtained by constraining triangulation sites close to the SLR sites DION and XRIS (Figure 4.1) to have the same azimuth of relative motion as the SLR sites. This generates displacement vectors in good agreement with the slip vectors of thrust-faulting earthquakes in the neighbouring Hellenic Arc.

Figure 5.12 shows the displacements obtained and the principal strains for triangular regions inferred from these displacements. For the purposes of this comparison I apply a translation and rotation to the vectors obtained by Davies *et al.* (1996) such that stations PNCH, VALT and ARAC (on the Peleponnessos, Figure 5.10) do not move significantly (they have no significant relative motion in any case). Thus the hundred-year velocity estimates are brought into a reference frame comparable with the Central Greece Network velocities.

The components along 020° of the hundred-year velocities are shown in Figure 5.11 (open symbols). As with the Central Greece Network velocities, there appears to be no significant difference between the trend exhibited by sites further away from the Gulf and that exhibited by sites close to the shoreline, implying that the strain is localised within the Gulf. The two sets of velocity estimates are in very good agreement, although the increase in northward velocity from east to west is slightly less pronounced for the hundred-year than for the Central Greece Network data. This might be partly the result of residual unmodelled co-seismic motion from the 1995 Egion earthquake affecting sites CG30 and CG31, but if anything the forward model (Subsection 5.2.2) overcompensates for this, and the trend is still shown by the remainder of the data and so this effect seems unlikely.

Le Pichon *et al.* (1995) have combined triangulation data from the 1890s network with modern triangulation data from 1975 and recent SLR measurements to compute velocities for sites throughout the Aegean region. They suggest that to first order, the relative motion of sites either side of the Gulf of Korinthos may be described by differential rotation of 4.4° Myr^{-1} about a pole situated just west of Athens, at 38.04° N, 23.57° E, with resulting maximum relative velocities of 15 mm yr^{-1} in the western Gulf. This also is in broad agreement with the Central Greece Network results, although the latter would suggest

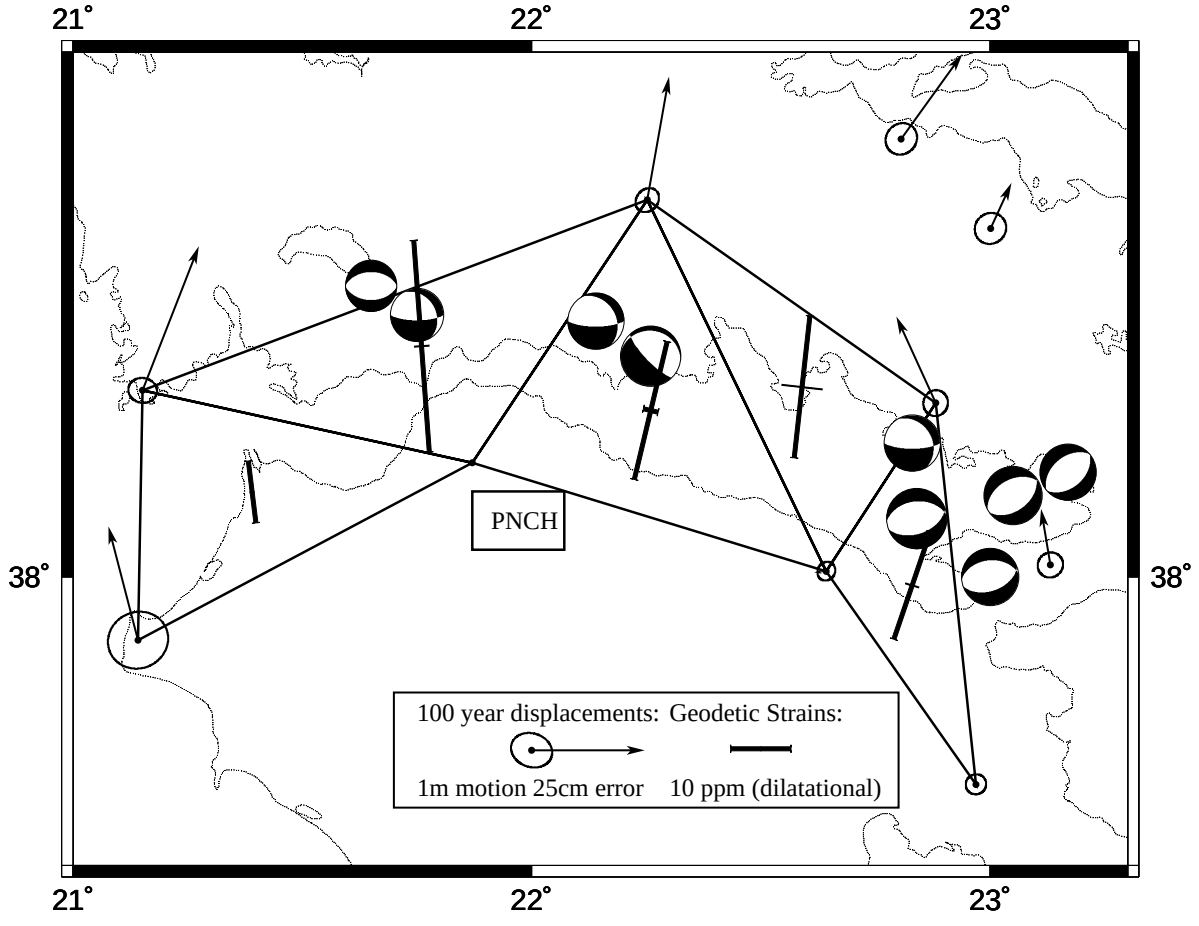

Figure 5.12: Geodetic displacements across the Gulf of Korinthos, 1892 – 1992, from Davies (1996). Site PNCH is held fixed. Crosses indicate principal strains for triangular regions; thick lines denote extensional strain and thin lines compressional strain. Also shown are focal mechanisms earthquakes of $M_s \geq 5.8$ occurring during the same period. (Figure prepared by Rob Davies).

that the pole of rotation be slightly further west and the rotation rate be of order 75% higher.

5.3.3 Implications for kinematics of the Gulf

The rigid-body rotation approach adopted by Le Pichon *et al.* can only describe first-order features of the deformation across the Gulf, because significant strains throughout Greece have been demonstrated by Davies *et al.* (1996). However the method of Le Pichon *et al.* cannot be refuted by observations from the Central Greece Network, because although strains are observed throughout central Greece, the greatest strains are in the Gulf of Korinthos and strains elsewhere may not be significant, although further measurements would constrain them more tightly.

Armijo *et al.* (1996) observe greater apparent Quaternary finite extension in the eastern Gulf than in the west, and conclude that the rifting is propagating westwards. This study shows that present-day rates are higher in the west. These statements are not incompatible: if the locus of maximum extension rate is indeed shifting westward, it is to be expected that the greatest present-day extension rates is to the west of the zone of greatest previous extension. If, however, as argued above, the amount of finite extension in the east is not known to be greater than in the west, no conclusion can be drawn as to the direction of propagation of rifting.

The eastern Gulf of Korinthos might be expected to show some evidence of post-seismic relaxation following the 1981 Alkyonides earthquake sequence (Subsection 3.7.3). Unfortunately, the Central Greece Network site distribution in the eastern Gulf of Korinthos is far from optimal for detection of this signal. Sites 46, 49 and 50 are the only ones close enough to be affected, and they do not show significant deviation from the general trend. In any case, post-seismic deformation over a five-year period, ten years after the event, is likely to be small.

5.4 Comparison of geodetic and seismic strain

As noted in Section 5.1, seismicity in the Gulf this century has been mostly concentrated at the eastern end, although longer-term seismicity (over several centuries) may be more evenly distributed between east and west. This observation may of itself indicate an increased risk of medium-term (years to tens of years) seismic activity in the western Gulf. However, this will not be the case if the true (geodetic) rate of extension in the west has slowed recently, or if significant aseismic strain has occurred in the western Gulf of Korinthos this century. To test these hypotheses it is necessary to compare both seismic and geodetic extension rates (strain rates) in the eastern and western Gulf.

5.4.1 Computation of seismic strain

If all the strain of a volume V of rock were released by N earthquakes in time t, then the strain rate tensor would be related to the moment tensors of the earthquakes by

$$\dot{\epsilon}_{ij} = \frac{1}{2\mu V t} \sum_{n=1}^{N} M_{ij}^{n} \tag{5.1}$$

where $\dot{\epsilon}_{ij}$ and M_{ij}^{n} are the ij^{th} components of the strain rate tensor $\dot{\epsilon}$ and the n^{th} earthquake moment tensor respectively (Kostrov, 1974). The latter is defined by

$$M_{ij} = \mu A s \left(u_i n_j + u_j n_i\right) \tag{5.2}$$

where μ is the modulus of rigidity of the rock (here taken to be 3.23×10^{10} Pa) and A is the area of the fault plane with unit normal vector $\mathbf{n}$ that slipped an average distance s in the direction of unit vector $\mathbf{u}$ during the earthquake. The quantity $M_0 = \mu A s$ is referred to as the scalar moment of the earthquake and can be related to the moment M_s obtained from surface-wave studies (Ekström & Dziewonski, 1988), although the relationship varies between continental and oceanic earthquakes, and between different regions. The difference between use of the global and local relationship is no more than 30% for earthquakes in the Gulf of Korinthos this century, with the global estimate of M_0 being higher. To be conservative about future seismic risk (*i.e.* to tend to overestimate the effect of past earthquakes), I use the global M_s – M_0 relationship where no direct estimates of M_0 exist, and following Ambraseys & Jackson (1990) add a further 40% to the scalar moments of all earthquakes to allow for the effect of events smaller than M_s 5.8.

To compare the seismic strain rate defined above with the geodetic strain rate, the seismic strain rate tensor can be expressed in terms of the equivalent rate of extension across the Gulf of Korinthos (considering only the strike-perpendicular component). The principal extensional eigenvalue $\dot{\varepsilon}$ of the seismic strain rate tensor $\dot{\epsilon}$ is related to the seismic extension rate $\dot{u}_s$ by

$$\dot{\varepsilon} = \dot{u}_s / a \tag{5.3}$$

where a is the width of the Gulf in the direction of extension. Strictly speaking, use of the eigenvalue results in a slight overestimate of seismic extension rate, as the strike-perpendicular component should be used instead, but since the strike-parallel component of extension is small, this effect is minor (and in any case leads to a more conservative estimate of seismic hazard).

The vertical dimension of the volume V is given by the thickness d of the brittle seismogenic crust. For the Gulf of Korinthos area, this is in the region of 10 – 15 km (Taymaz *et al.*, 1991; Rigo *et al.*, 1996). The horizontal dimensions of V are given by the length l of the portion of the Gulf in question, and the width a of that portion of the Gulf. Combining (5.1) and (5.3):

$$\dot{u}_s = \frac{M_e}{2\mu d l t} \tag{5.4}$$

	$\sum M_0$		$\sum M_0^{\max}$	M_e	$\dot{u}_s$ /mm yr^{-1}	$\dot{u}_s$ /mm yr^{-1}	$\dot{u}_g$	M_g
	Local	Global			(d = 15 km)	(d = 10 km)	/mm yr^{-1}	
West	15	18	25	16 ± 6	3.0 ± 1.2	4.5 ± 1.8	12.7 ± 1.0	47 (70)
East	22	24	29	21 ± 8	4.0 ± 1.6	6.0 ± 2.4	6.4 ± 1.0	24 (35)

Table 5.4: Seismic and geodetic extension rates across the eastern and western Gulf of Korinthos (either side of 22.5° E). All moments are in units of 10^{18} N m. $\sum M_0^{\max}$ is obtained by adding 40% to the value of $\sum M_0$ obtained from the global M_s – M_0 relationship. M_e is calculated from the global $\sum M_0$, and the uncertainty quoted represents ± 40%. $\dot{u}_g$ is the weighted mean of the Central Greece Network velocity components along 020° for sites west and east of 22.5° E appropriately. M_g is the moment release expected this century, if all the geodetic strain had been released by earthquakes, for a 10 km seismogenic layer (15 km layer figure in parentheses).

where M_e is the eigenvalue of the summed moment tensor (5.1) that corresponds to the principal extensional strain rate $\dot{\varepsilon}$. The quantity $\dot{u}_s$ can be compared directly with the geodetic strike-perpendicular extension rate $\dot{u}_g$ computed in Section 5.3.

To obtain a reliable estimate of seismic strain it is necessary to consider earthquakes from several decades over a sufficiently large region (Jackson & McKenzie, 1988a), so that a representative sample of seismicity is taken. For this reason I consider the earthquakes over the whole of the last century, for which good records exist, and divide the Gulf of Korinthos into only two sub-regions, between 21.9° E and 22.5° E, and between 22.5° E and 23.1° E, each of which has an along-strike length l of 56 km. Details of shallow earthquakes in the Gulf of Korinthos with magnitudes $M_s \geq 5.8$ are given in Table 5.1, and the calculation is carried out with these data.

The extension rates computed for the eastern and western Gulf based on the above assumptions are given in Table 5.4. In the eastern Gulf, the seismic and geodetic extension rates agree well for a 10 km seismogenic layer, and within the bounds of possible error for a 15 km layer, although the spread of observed geodetic extension rates (Figure 5.11) does not preclude higher strain within this area. However, in the western Gulf, the seismic extension rate is at most half the averaged geodetic extension rate, and is significantly smaller than even the lowest observed geodetic extension rate in this area.

5.4.2 Implications for future seismic hazard

Because the seismic extension rate agrees with both short-term and hundred-year estimates of the geodetic extension rate in the eastern Gulf of Korinthos, it seems likely that strain in this region continues to be

released seismically. Although aseismic strain has been documented elsewhere in the Aegean (Jackson & McKenzie, 1988b), there is no obvious reason why neighbouring areas so tectonically similar as the eastern and western Gulf should behave differently, and so it is reasonable to assume that the geodetic strain in the western Gulf will eventually be released by means of earthquakes. The higher rates of seismicity in the western Gulf over the interval 1690 – 1890 (N.N. Ambraseys & J.A. Jackson, pers. comm.) may thus be indicative of future rates of seismicity.

The hypothesis that the rate of extension in the western Gulf has recently slowed seems unlikely to hold; the short-term extension rates are if anything higher than the hundred-year estimates for this region. Another possibility that can be discounted is that the seismogenic layer in the west is significantly less than 10 km thick. The western Gulf of Korinthos is particularly well-studied in terms of microseismicity (Rigo *et al.*, 1996; Rietbrock *et al.*, 1996) and these studies indicate that the majority of microseismicity occurs between 6 and 11 km depth. In addition, it would seem unlikely that the western Gulf should have more ductile crust permitting aseismic strain, given that the frequency of recent small earthquakes is as high as in the east.

The deficit of seismic extension in the west need not all be released as large earthquakes; it is possible that many small earthquakes could do this (or already have done so). However, three large earthquakes ($M_0 \geq 5 \times 10^{18}$) have already occurred in the western Gulf this century, and the historical record (N.N. Ambraseys & J.A. Jackson, pers. comm.) shows that $M_s \geq 6.0$ earthquakes have been common in this area in the past, so it seems likely that large earthquakes will be the eventual mechanism of moment release.

Using (5.4) it is possible to calculate the total moment required to release the geodetic strain, if all the geodetic strain is released by normal-faulting earthquakes. Table 5.4 shows this "geodetic moment". For the western Gulf, the moment deficit is at least 22×10^{18} N m (for 10 km seismogenic layer, $\sum M_0^{\max}$), equivalent to four earthquakes the size of the 1995 Egion earthquake. At the extremes of possibility, it may be as large as 52×10^{18} N m (for 15 km layer, global $\sum M_0$), equivalent to six earthquakes the size of the 24 February 1981 Alkyonides mainshock, the largest event of the region this century.

5.5 Summary

Seismicity within the Gulf of Korinthos is well documented and indicates that it is the fastest-deforming part of mainland Greece. This is borne out by the Central Greece Network data, which are in agreement with the hundred-year data of Davies *et al.* (1996). The co-seismic displacement field of the 1995 Egion earthquake has affected several stations from the Central Greece Network, but there is insufficient information to allow an inversion for source parameters from this dataset. However, the co-seismic effects can be removed using a forward model with the source parameters of Bernard *et al.* (1996). The next largest

earthquake within the timespan of the Central Greece Network is the 1992 Galaxidi event, but this, and therefore all smaller earthquakes, has had negligible co-seismic effect on the network.

Once the co-seismic effects of the 1995 Egion earthquake have been removed, the inter-seismic extension rates across the Gulf can be computed. These show good agreement with the hundred-year rates, and indicate a significant increase of extension rate from east to west along the Gulf. This trend is independent of whether the observation is made near to or far from the Gulf, suggesting that strain is to first order localised within the Gulf. If the extension rates averaged over the Quaternary are higher in the eastern Gulf, as is suggested by Armijo *et al.* (1996), then the finding in this study of greater present-day extension rates in the western Gulf is compatible with the hypothesis of westward migration of the zone of most active rifting.

In contrast with present-day geodetic deformation, seismicity over the past hundred years has been higher in the eastern Gulf of Korinthos than in the west. The seismic moment release can be related to the geodetic extension rate, and indicates that seismic strain release matches geodetic strain in the east, but there is a large deficit of seismic strain release in the west. It seems likely from the history of seismicity in the Gulf that this deficit will be released by a number of large earthquakes in the medium term.

Chapter 6

The 1995 Grevena earthquake

On Saturday 13 May 1995, a large earthquake of M_s 6.6 struck northern Greece in the Kozani – Grevena region, an area of little historical seismicity (Figure 6.1). Fortunately, there was no loss of life, as a warning had been provided by five large foreshocks (M_s 3.5 – 4.5) in the half hour preceding the main shock, but extensive damage was inflicted on both modern and traditional buildings in the towns of Kozani and Grevena and surrounding villages. Figure 6.2 shows an enlarged map of the Kozani–Grevena region, with the locations of fault scarps, major towns and other important features.

Within five days of the earthquake a team from Oxford University was operating GPS receivers on pre-existing triangulation monuments in the region, in order to determine the co-seismic displacement and provide a basis for the study of both short- and long-term post-seismic displacements. This fieldwork was carried out in collaboration with the National Technical University of Athens and the Institut de Physique du Globe de Paris.

The CMT solution (J.H. Woodhouse, pers. comm.) is almost purely normal, with planes striking 240° and dipping 31° NW, and striking 070° and dipping 59° SE, and a centroid depth of 8.6 km (Table 6.1). No obvious surface break was found and so it was not possible to determine *a priori* which nodal plane formed the fault. However, the majority of the severe damage and ground cracking lay to the south of the epicentre, and the nearby Servia fault dips NW, so it was tentatively assumed that the 1995 fault dipped NW, and the geodetic network was planned and occupied on this basis. The dip and strike were later confirmed by the aftershock sequence (Hatzfeld *et al.*, 1996).

The region had previously been thought to have low seismic risk on the basis of recent seismicity (Papazachos, 1990), although the Servia and Palaeochori – Sarakina faults have been active during Holocene time (Pavlides *et al.*, 1995). There are no existing estimates of regional geodetic strain in the area, as it lies to the north of the Central Greece Network (this study) and the 1980s triangulation network (Billiris *et al.*,

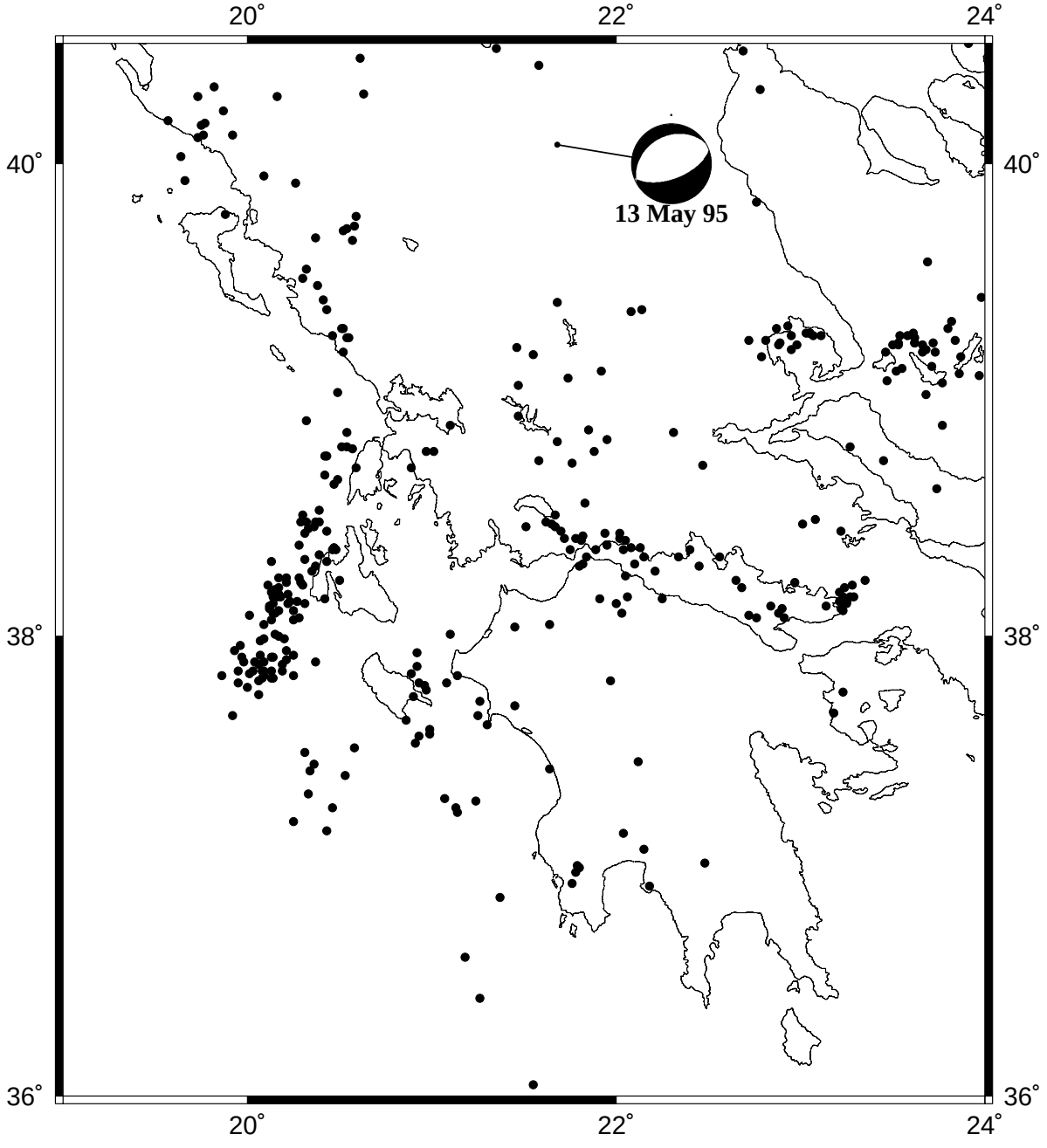

Figure 6.1: Location and CMT focal mechanism of the 13 May 1995 Grevena earthquake, in relation to recent shallow earthquakes $M_b > 4.5$ from the ISC catalogue 1964 – 1992 (dots).

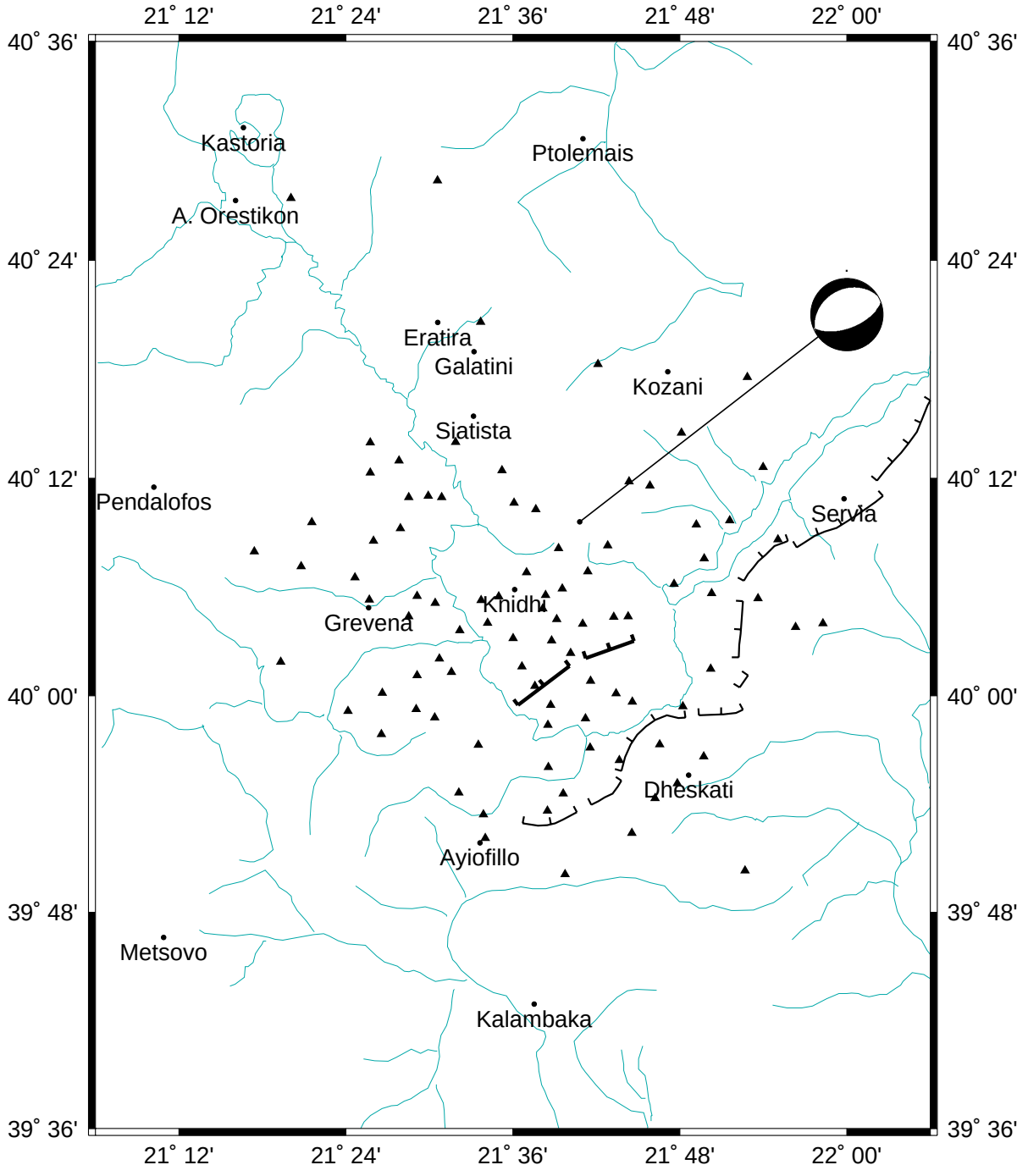

Figure 6.2: Map of the region of the 1995 Grevena earthquake, showing geologic fault scarps (light ticked lines), surface ruptures (heavy ticked lines), drainage and major towns (partly after Pavlides *et al.*, 1995 and Hatzfeld *et al.*, 1996). Sites observed in the co-seismic network are shown by small triangles.

Scarp		M_0	Strike	Dip	Rake	Length /km	Dmin	Dmax
Lat.	Lon.	/Nm$\times 10^{18}$					/km	/km
39.97	21.60	7.6	240°	31°	-98°	25	4	15

Table 6.1: Seismic parameters for the 13 May 1995 Grevena earthquake (focal plane parameters, location and scalar moment from CMT solution, J.H. Woodhouse, pers. comm.; fault depths and lengths from aftershock locations, Hatzfeld *et al.*, 1996).

1991; Davies, 1996; Davies *et al.*, 1996), so seismic hazard cannot be estimated on this basis.

6.1 Network design and occupation

The geodetic network was designed with the aid of forward models of co-seismic elastic surface deformation due to slip on a planar fault in a uniform half-space (Okada, 1985) and post-seismic surface deformation due to visco-elastic relaxation of a half-space underlying the elastic layer containing the fault (Rundle, 1982), based on the CMT solution. The requirements of the two experiments are somewhat different so they are described separately.

Most low-lying areas of the region are covered with poorly-consolidated fluvial and lacustrine sediments of mid-Pliocene and younger ages. Geodetic monuments sited on these sediments may be unstable, despite having concrete foundations, and so the measured monument displacements will not always match the true displacement of the upper crust as a whole. Where possible, triangulation stations sited on basement rock (usually forming higher ground) were used, or those on level ground where local motions (*e.g.* down-slope creep or rotation of boulders) should be less significant. This consideration is particularly important when trying to detect small post-seismic displacements.

The local GPS network was linked to the ITRF reference frame by a baseline to the continuously-recording GPS receiver at the Dionysos SLR site, and by the use of CODE precise satellite ephemerides.

6.1.1 Co-seismic network

The surface displacements predicted using the CMT solution are shown in Figure 6.3. The most recent pre-seismic survey of the area was the Hellenic Army Geographic Service (HAGS) 1984 – 1986 triangulation of Greece. Pillars from this survey were selected so as to sample all parts of the co-seismic displacement field, including the far field which should have no co-seismic strain (although there may be some secular strain).

Because of the uncertainty regarding site stability, particularly on the soft sediments in the locality of the fault, more stations were occupied than would be required to constrain the fault parameters given perfect observations. Also, redundancy of data is required because the triangulation survey was of relatively low precision compared with GPS. The precision of individual stations in the 1984 – 1986 survey is formally 15 mm horizontally and 30 mm vertically (D. Paradissis, pers. comm.), but will depend on the order of the triangulation pillar as second- and lower-order pillars were adjusted in local networks bounded by first-order pillars. These local networks have a horizontal station precision of approximately 30 mm (G. Veis, pers. comm.) and so this estimate was adopted for all triangulation-derived coordinates. Stations in the near field only needed to be occupied for 2 to 3 hours to achieve centimetric precision over 10 to 20 km baselines.

Sites forming the co-seismic network are shown in Figure 6.4.

6.1.2 Post-seismic network

Displacements predicted by the Rundle (1982) model (Section 3.7), using the CMT solution, are shown in Figure 6.5. Comparison with Figure 6.3 reveals that the post-seismic strain is distributed over a much wider area than the co-seismic strain, and is also much smaller in magnitude. Thus the post-seismic network needs to be both larger in extent and more precisely measured, with repeated occupations to ensure reliability. Notable features of the model displacement field that must be sampled include the down-dip strain maximum in the hanging-wall, the change in displacement direction down-dip of the ends of the fault plane, and strike-parallel displacements towards the fault from beyond its ends. Sites in the far field must be included to provide reference points for the post-seismic deformation and to measure the regional secular strain.

Post-seismic sites must have high stability, so many co-seismic pillar sites were rejected because either the pillar or its foundations were unstable. A selection of stable co-seismic sites forms most of the post-seismic network. Alternatively, brass pins have been set into bedrock in localities where pillar access is poor or non-existent, or where all pillars are unsuitable. The post-seismic network established in May and September 1995 is shown in Figure 6.6.

6.1.3 Epoch 1995.39

Data

Sites in the co-seismic network were occupied at least once for a period of 2 – 3 hours. No observation schedule was followed; instead, fixed sites at ACR0, KN03, and KN88 were occupied semi-continuously

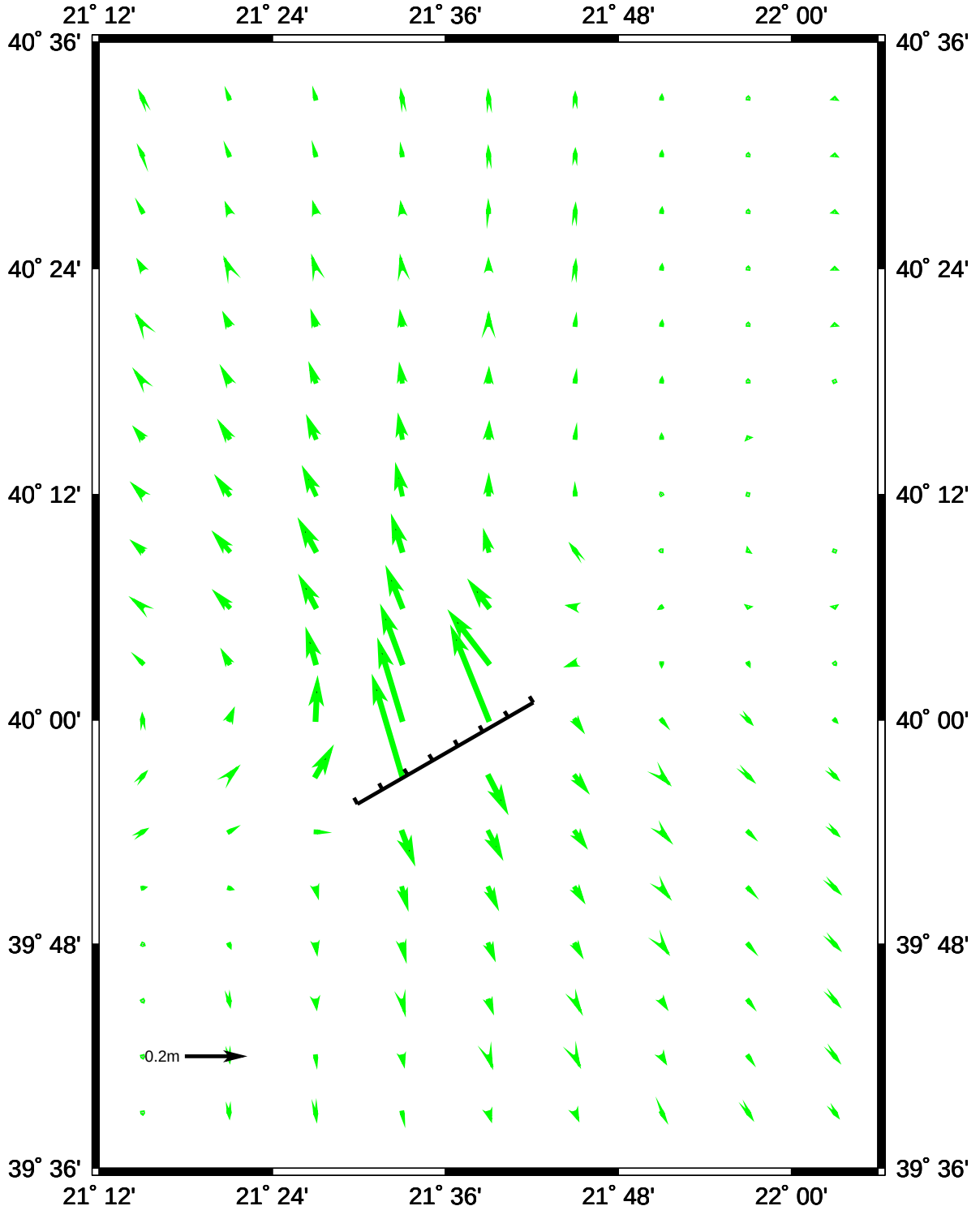

Figure 6.3: Forward model of co-seismic horizontal displacements in the 13 May 1995 Grevena earthquake. The map region is the same as for Figure 6.2.

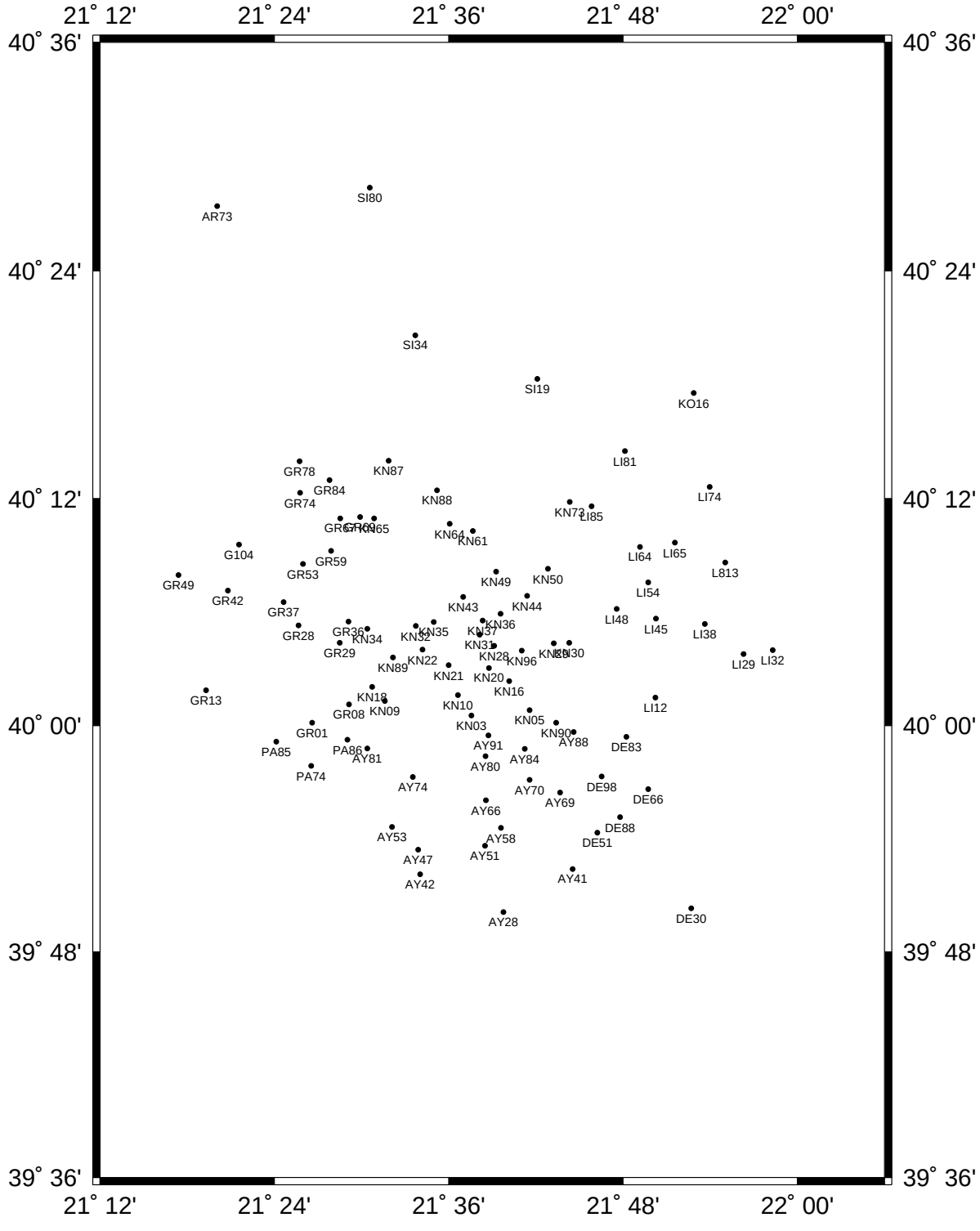

Figure 6.4: Sites observed in the May 1995 Grevena co-seismic network.

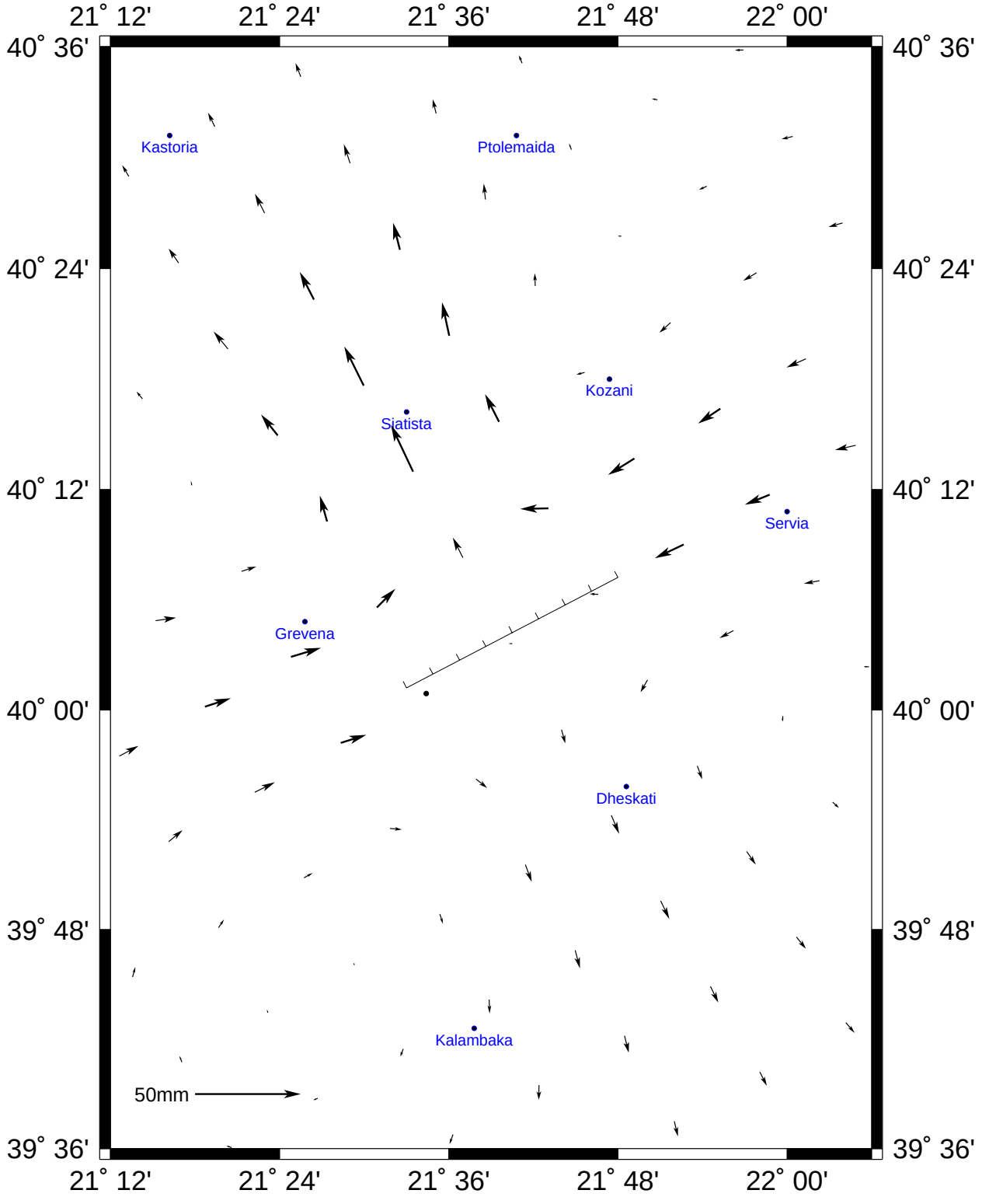

Figure 6.5: Forward model of post-seismic displacements after the 13 May 1995 Grevena earthquake, using the CMT solution for a 20 km fault penetrating the entire 10 km elastic layer.

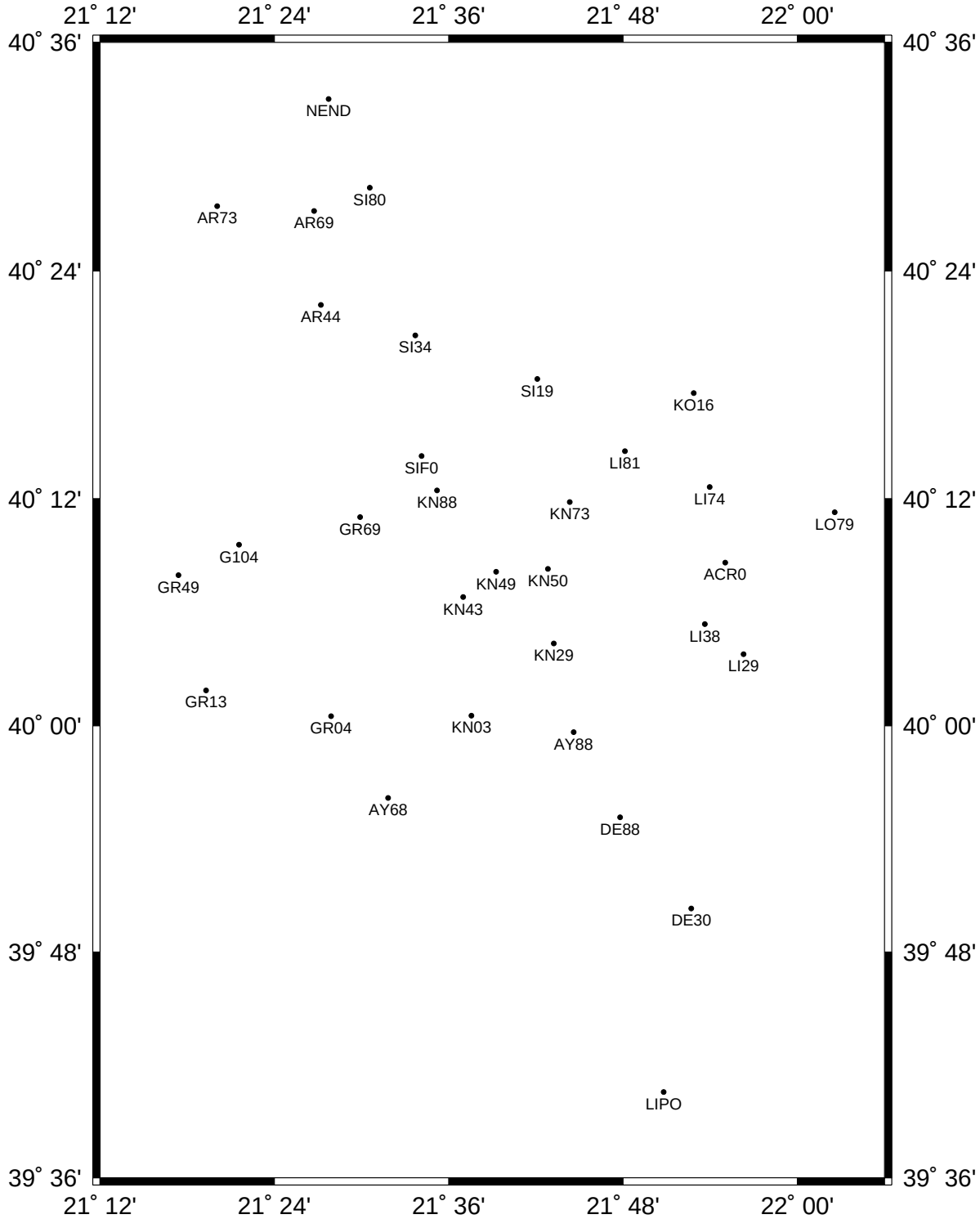

Figure 6.6: Sites observed in the May–September 1995 Grevena post-seismic networks.

during the campaign so baselines could be formed radiating from these stations.

During the post-seismic network occupation, sites ACR0, DE30, GR49 and SI80, which bound the region, were occupied continuously. Other sites were occupied for at least six hours, on several occasions if possible.

The receivers used were 9 Ashtech P-XII or LM-XII (L1, $L2^2$ under AS), 3 Ashtech Z-XII (L1,L2) and 2 Trimble 4000SST (L1, L2), plus the fixed Trimble 4000SSE (L1, $L2^2$) at DION. Figures B.1 – B.4 show the times of observations and antenna heights (to L1 phase centre) pertaining to the May 1995 campaign.

GPS Processing

Baselines were formed radially from the local 'fixed' sites, where possible maintaining a backbone of full-wavelength L2 baselines. All coordinates were computed relative to the SLR site at DION (see Table 2.3). CODE precise orbits were used throughout.

Despite the shortness of the baselines, ambiguity resolution was not successful using either L5 / L3 or L1 / L2 daily networks, even with an ionospheric TEC model. This was mostly due to the high level of short-period ionospheric fluctuations apparent in the double-difference residuals, and the shortness of the occupation times in the co-seismic network schedule. Daily ambiguities-unresolved L3 solutions were combined to make a campaign network solution.

Results

The r.m.s. residuals to the campaign solution were (0.010 m, 0.009 m, 0.021 m) in east, north and up components. Residuals are shown in Figure 6.7, and the coordinates are given in Tables B.1 and B.2. However, these statistics are slightly spurious as the majority of stations were only occupied once and so there is no direct measure of their repeatability. Despite this, as correlations were modelled correctly during the GPS network processing, the network variance-covariance matrix should accurately reflect the relative confidence of site coordinates.

6.1.4 Epoch 1995.74

Data

In September 1995, stations from the post-seismic network set up in June, plus several new sites, were occupied using 4 Ashtech Z-XII receivers (L1, L2) and 10 Ashtech P-XII or LM-XII receivers (L1, $L2^2$ under AS). Occupations lasted 6 – 8 hours during the day, or 12 hours or more overnight, and each site was occupied two or three times. In addition, observations were taken from the fixed Trimble 4000SSE

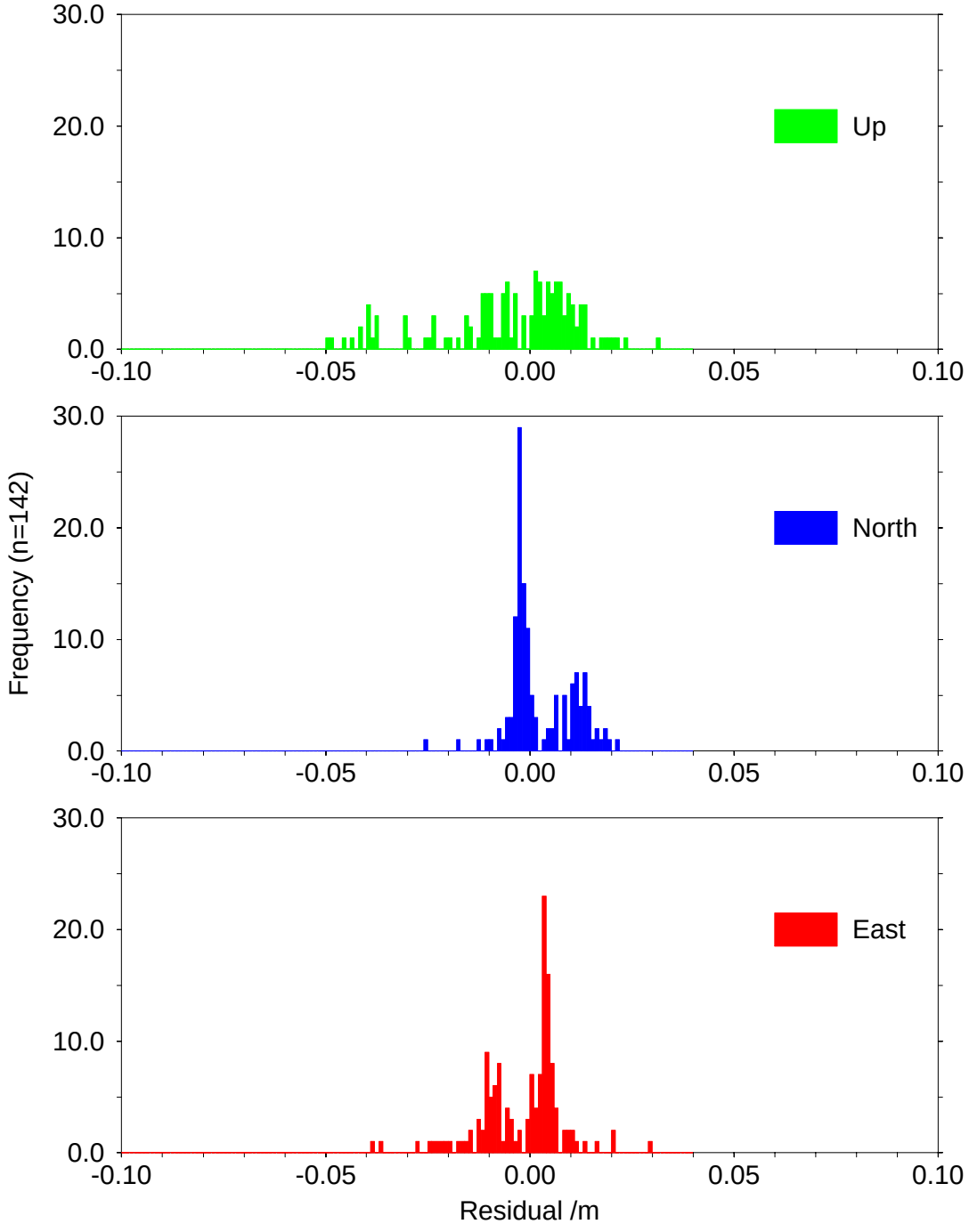

Figure 6.7: Residuals to the May 1995 campaign solution in east, north and up components. For statistics of residuals see text.

(L1, L2[2]) at DION. The local fixed site, SIF1, was occupied continuously with an Ashtech Z-XII, and its eccentric site SIF2 was occupied with a Trimble 4000SST (L1, L2) to enable processing of the long baseline to DION without mixing receiver types.

Occupation times and antenna heights (to L1 phase centre) for the September 1995 Grevena campaign are depicted in Figures B.5 – B.8.

GPS Processing

Where possible, baselines were chosen to link the full-wavelength L2 receivers, but the primary concerns were duration of observations and baseline length. Cleaning and processing were performed relative to the DION fixed site, using CODE precise orbits.

Ionospheric conditions were much less severe than in June 1995, so 90% ambiguity resolution was achieved for the local network using L1 and L2 daily network solutions, without an ionospheric TEC model. Daily L3 ambiguities-resolved networks were then combined to yield the campaign solution.

Results

The r.m.s. residuals to the campaign solution were (0.017 m, 0.008 m, 0.024 m) in east, north and up components. These values seem larger than those for the May 1995 campaign, but more accurately represent the precision of the network since all sites had multiple occupations in the October 1995 campaign.

The network residual histograms are shown in Figure 6.8 and coordinates are given in Table B.3.

6.2 Inversion for focal mechanism from geodetic data

I have estimated the source parameters of the Kozani – Grevena earthquake using the inversion method detailed in Section 3.5 and implemented in the C program `okinv` (Appendix C.4). This method has successfully been used to invert from both synthetic datasets (Subsection 3.5.3) and real data (Subsection 3.5.4); further synthetic experiments relating to the reliability of the Kozani – Grevena earthquake inversion are described below.

6.2.1 Co-seismic displacement field

The co-seismic displacement field was obtained by differencing the May 1995 pillar coordinates with coordinates from the 1984 – 1986 HAGS survey of Greece. The latter survey was conducted using triangulation,

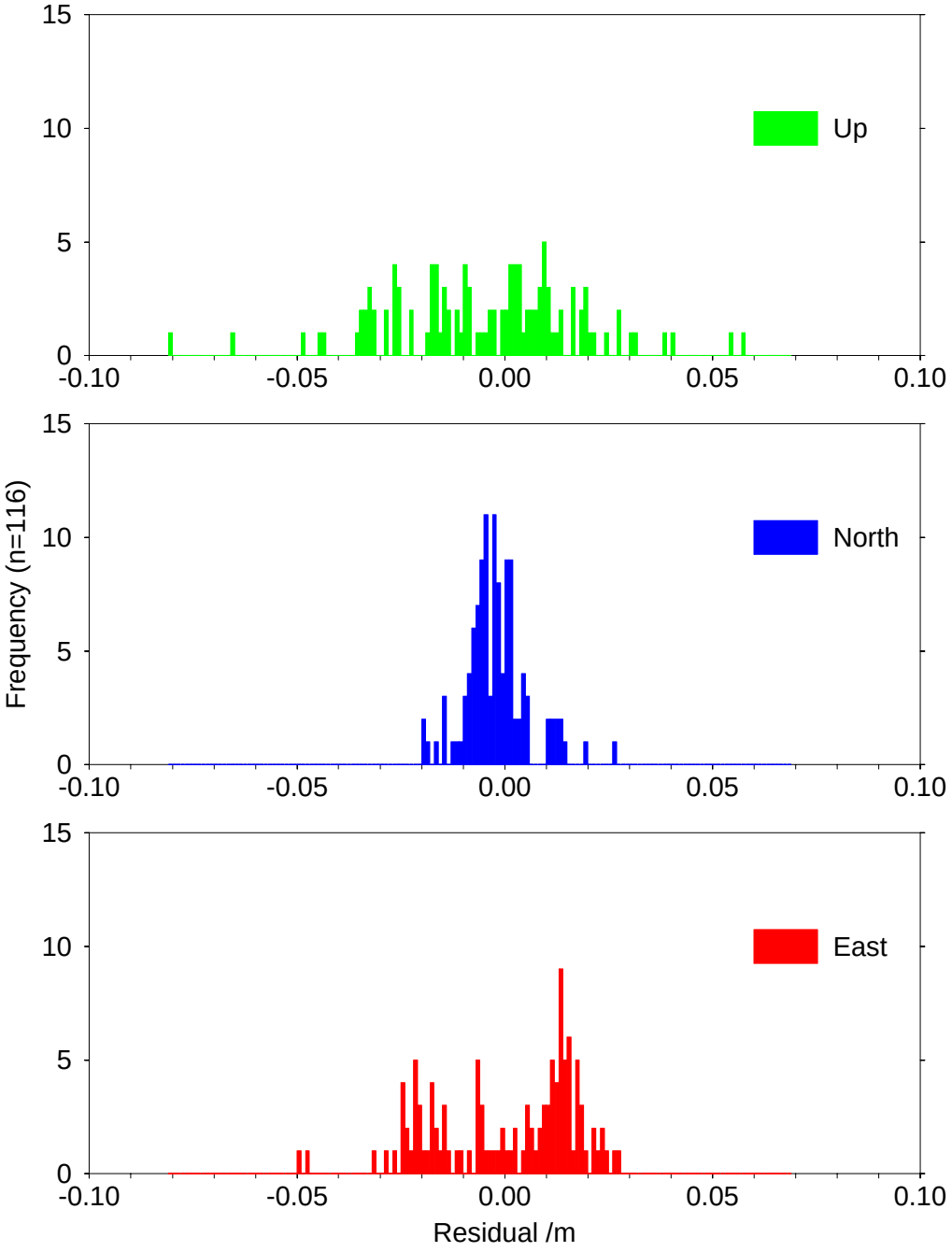

Figure 6.8: Residuals to the September 1995 campaign solution in east, north and up components. For statistics of residuals see text.

with limited electronic distance measurement for scale control. The HAGS coordinates are expressed in the GGRS87 reference frame, which within the limits of survey accuracy (approximately 1 p.p.m.) is parallel to and scales with the ITRF reference frame (G. Veis, pers. comm.). Thus an arbitrary offset must be added to the displacements to render them into the ITRF frame.

The site displacements obtained (D. Paradissis, pers. comm.) are shown in Figure 6.9. Sites KO16, LI12, AY58 and AY88 are revealed to be outliers (possibly caused by monument stability problems or GPS operator blunders) when compared with the trend of nearby stations and so were excluded from focal mechanism inversions. Pillars AR73, SI19, SI34 and SI80 were excluded because they are in the far field of the earthquake and so do not form part of the co-seismic signal, yet are not sufficiently well-distributed to enable estimation of regional strain.

Vertical displacements were also obtained, but as the vertical component of the HAGS coordinates is determined from triangulation only, they are too unreliable and so were not used.

6.2.2 Choice of inversion parameters

It is desirable to invert for as few parameters as possible so as to increase the redundancy of the solution, and to reduce the amount of trade-off that can occur between parameters. However, it is necessary to estimate sufficient parameters in order to be able to describe the observations adequately. No surface break was found, and the centroid depth of the CMT solution (8.6 km) is much shallower than the depth extent of the aftershocks (to 15 km, Hatzfeld *et al.*, 1996), so the maximum and minimum depth of faulting were included as parameters to be inverted, in addition to strike, dip, rake, fault length and scarp location, and slip magnitude.

As the HAGS reference frame is much less precisely defined (only to within 1 p.p.m.) than the ITRF reference frame used by GPS (defined to within a few p.p.b.), network scale and rotation parameters must also be estimated. Including these parameters will also ameliorate the effects of secular strain in the interval 1985 – 1995, although only the dilatational component (not shear). There are no independent estimates of the regional strain so its effects cannot be removed more rigorously.

The CMT solution is very close to a simple double-couple event, and so it was not felt necessary to invert for more than one fault plane at first (*c.f.* the 03 March 1981 Platea – Kaparelli earthquake, Section 3.5.4). However, the aftershock distribution (Hatzfeld *et al.*, 1996) has some indication of an antithetic fault to the NW of the main fault plane, so a few runs of two-fault inversions were attempted. No significant reduction of the residuals was observed, and the residuals to the one-fault solution are already close to the level of noise in the data (see below), so I conclude that the geodetic data do not require the presence of an antithetic fault or of any segmentation of the main fault plane.

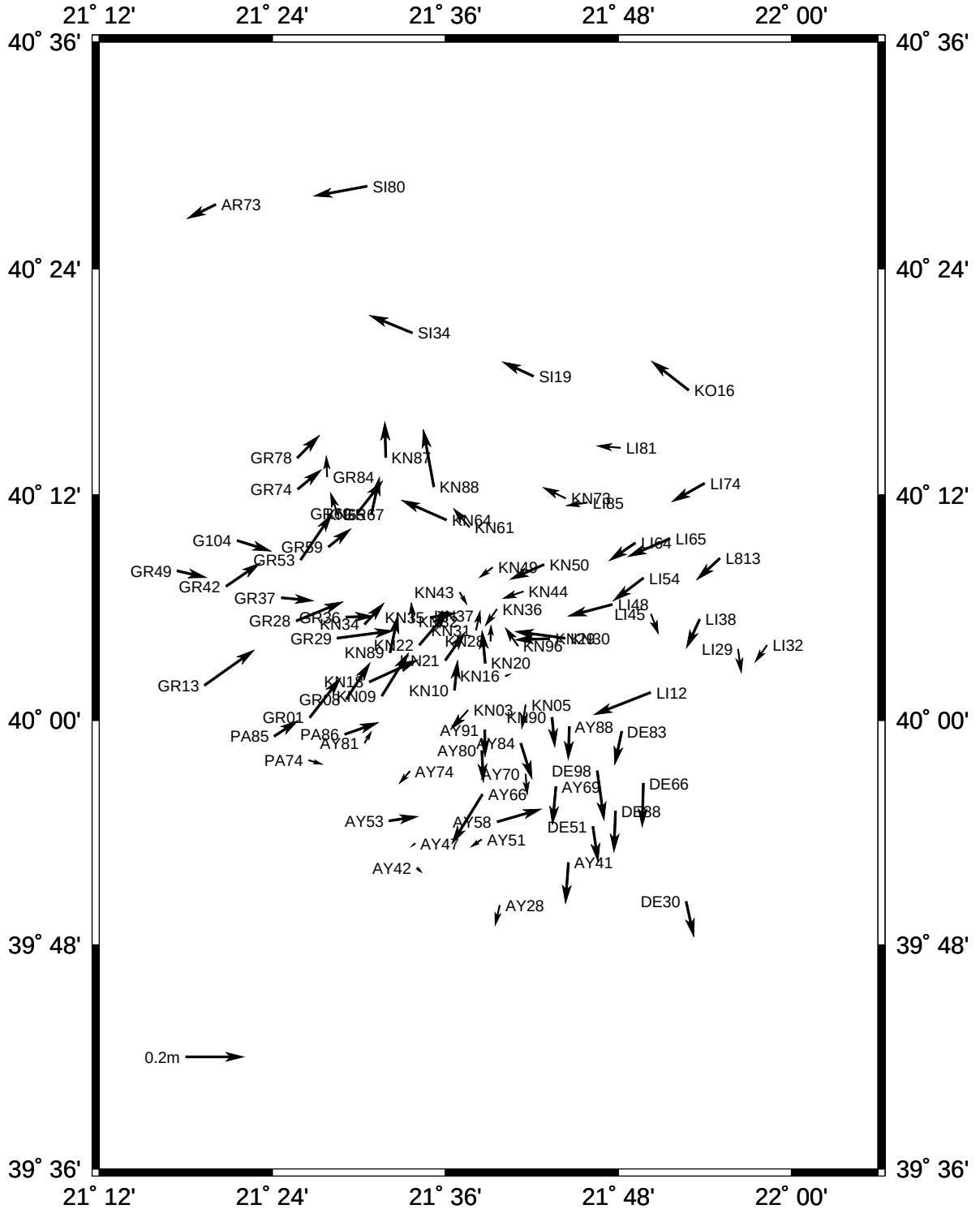

Figure 6.9: Site displacements for the Grevena network, 1985 – 1995.

Scarp		Total slip	Strike	Dip	Rake	Length	Dmin	Dmax	Displacement /m	
Lat.	Lon.	/m				/km	/km	/km	Dip-slip	Strike-slip
40.02°	21.63°	1.2	253°	43°	−95°	26.9	2.8	13.5	1.2	0.1

$M_0 = 16.3 \times 10^{18}$ N m.

Scale change: −1.9 p.p.m., rotation: −1.7 p.p.m. (*i.e.* clockwise).

Total penalty 0.372 m^2, r.m.s. residual 0.047 m.

Table 6.2: Model for the 13 May 1995 Grevena earthquake, with scale and rotation parameters estimated.

6.2.3 Results and discussion

The parameters of the best physically reasonable minimum found by `okinv` after 2000 restarts using unweighted horizontal displacement observations, with an r.m.s. residual of 47 mm, are given in Table 6.2. The residuals are displayed in histogram form in Figure 6.10, and show a roughly normal distribution, with the exception of the vertical residuals. This exception is not significant as the vertical measurements played no part in the inversion process, being too unreliable, and are merely shown for comparison.

A marginally smaller minimum (r.m.s. residual 44 mm) was discovered for a deeper fault (down to 115 km), but solutions requiring the fault to penetrate much deeper than the base of the lower crust (20 km cut-off) were rejected as physically unreasonable, as were solutions requiring faults longer than 40 km. The horizontal surface displacements are relatively insensitive to the maximum depth of the fault, for a normal fault, as shown by test inversions using synthetic datasets, and much better results could have been obtained if reliable vertical displacements were available (Subsection 3.5.3).

The model and observed displacements are compared in Figures 6.11 and 6.12. The surface projection of the model fault scarp lies within 2 km of the most prominent ground cracks observed after the earthquake. Model dip and rake are within 5° of the CMT values, and strike is within 15°. The fault length, and minimum and maximum depths are close to what would be expected given the aftershock distribution (Figures 6.13 and 6.14).

However, the slip magnitude of the model, combined with the length and down-dip extent of the fault (which agree well with the aftershock distribution) imply a scalar moment that is larger than that of the CMT by a factor of two, which is a notable discrepancy. The combined scalar moment of the foreshocks and aftershocks cannot add more than 25% to the moment of the main shock, and errors of the CMT are unlikely to be more than 30%, so there remains a further mismatch of roughly 50% of the scalar moment. This might be accounted for by immediate (or very short-term) post-seismic creep on the fault. At least 0.3 m of creeping fault displacement is required if creep occurs on the same fault plane that moved in the

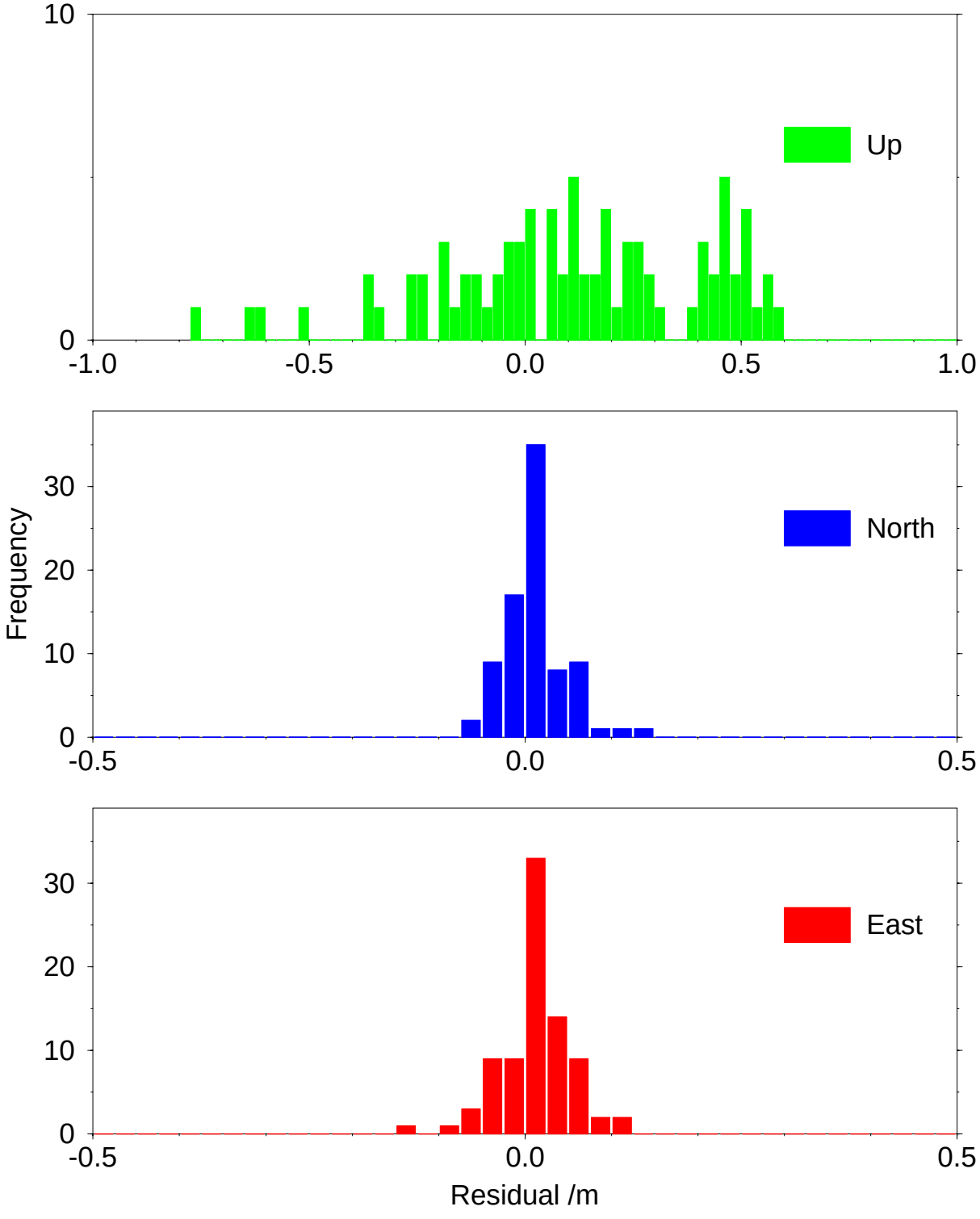

Figure 6.10: Histograms of residuals to the Grevena earthquake model, in north, east and up components. Note that the scale for the vertical residuals is different, and that vertical observations were not used for the inversion.

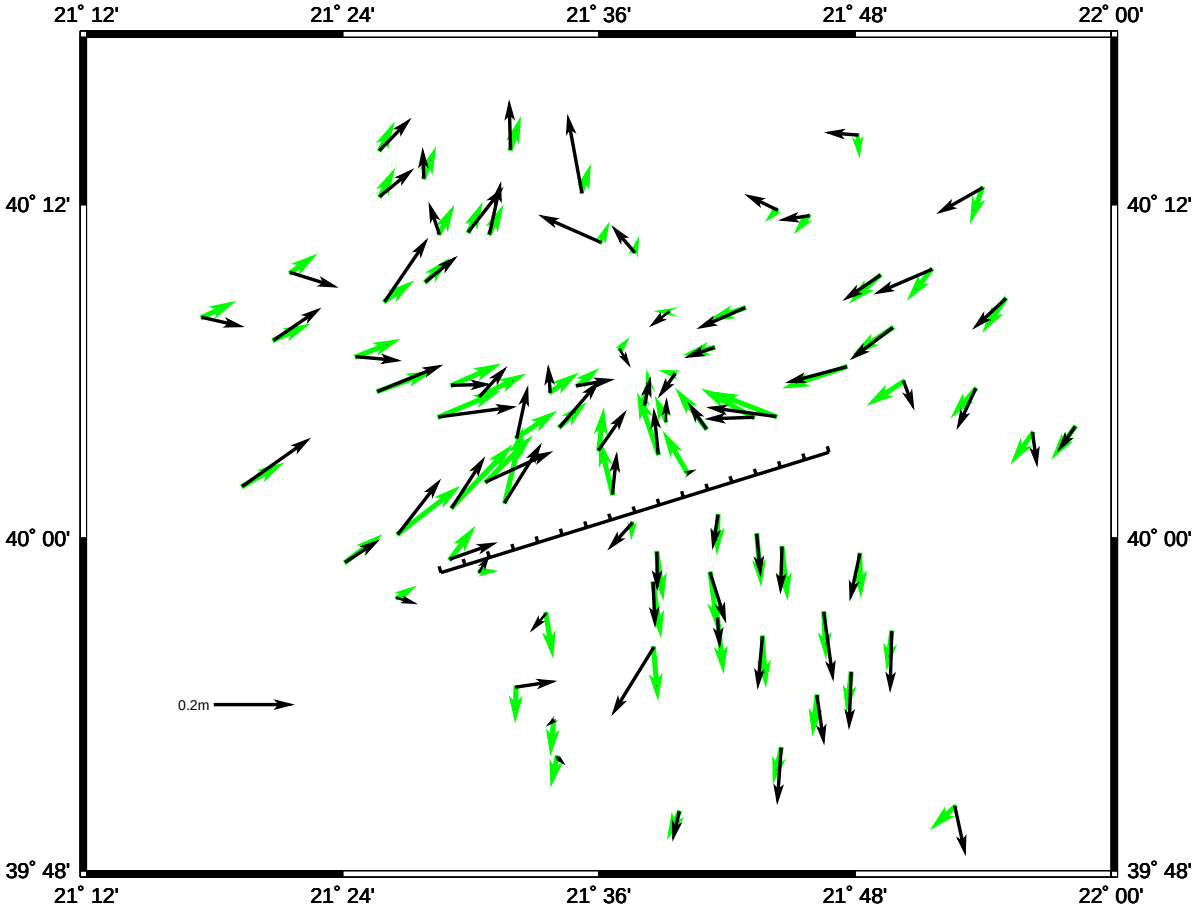

Figure 6.11: Model (green) and observed (black) co-seismic site displacements for the Grevena earthquake.

main event. It is conceivable that some of this creep may have been precursory or the result of longer-term inter-seismic creep, although the implied rate of 30 mm yr^{-1} for the latter is very high.

Other workers (*e.g.* Lundgren *et al.*, 1993) have also encountered geodetic moments that are much higher than the CMT scalar moment. Available examples from the literature are given in Table 6.3. Where two entries for the same event are given, it indicates that different datasets were used. It can be seen that geodetic moments are generally higher than seismic moments, but that the excess of geodetic moment bears no relation to the post-seismic or total survey interval. The average ratio of geodetic / seismic moment is 2.0, but both geodetic and seismic data from the earlier events may have large errors and should be treated with caution. However, if this ratio is representative of a general phenomenon it may help to explain some of the excess of geodetic over seismic strain that has been observed in regions with relatively long histories of both seismicity and geodetic measurement, although it cannot always do so entirely (*e.g.* Davies *et al.*, 1996; see also Chapter 5). More, detailed, studies of individual earthquakes are necessary before statistically valid conclusions can be drawn.

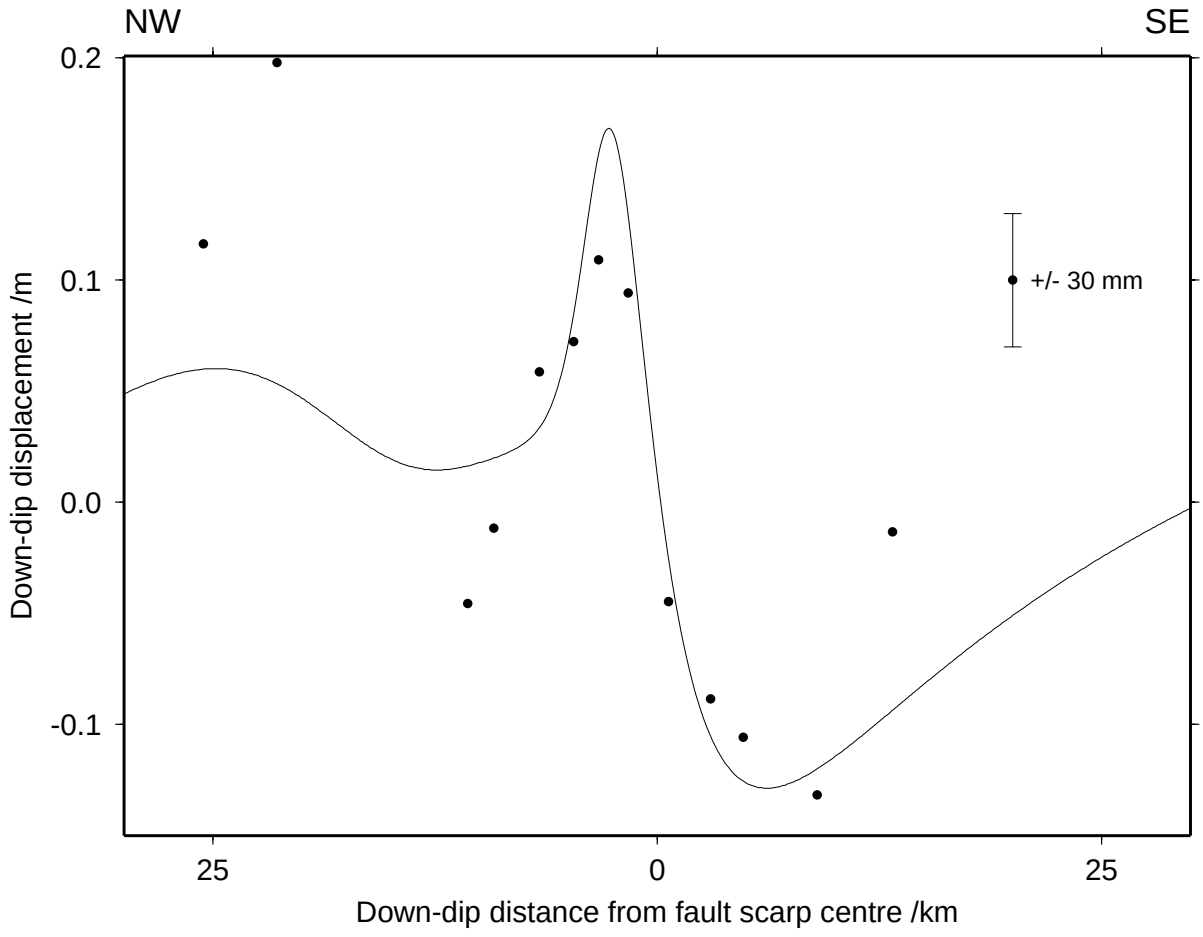

Figure 6.12: Model (solid curve) and observed (dots) displacements perpendicular to strike for the Grevena earthquake. The sample error bar shows the assumed error of a displacement observation of $\pm$ 30 mm.

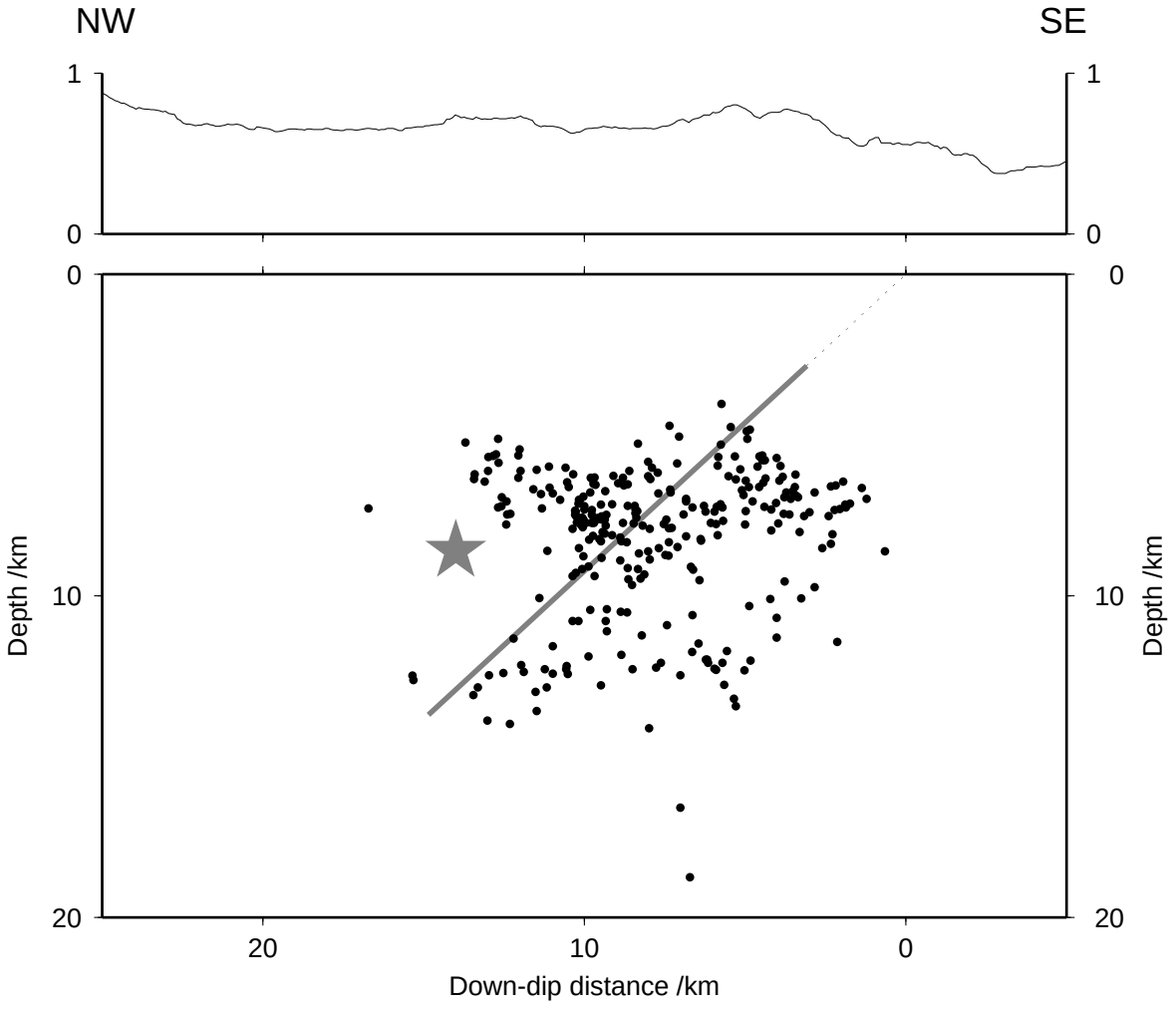

Figure 6.13: Profile of aftershocks within 3 km of a vertical plane perpendicular to the strike of the model fault plane and passing through the mid-point of the fault. The model rupture surface is shown as a thick grey line, and the centroid location of the main shock (actually to the east of the plane) is projected onto the plane and shown as a star. A topographic profile (vertical exaggeration $\times 5$) along the same section is shown above. Note the lack of surface expression of the fault scarp, implying that the fault is both young and blind.

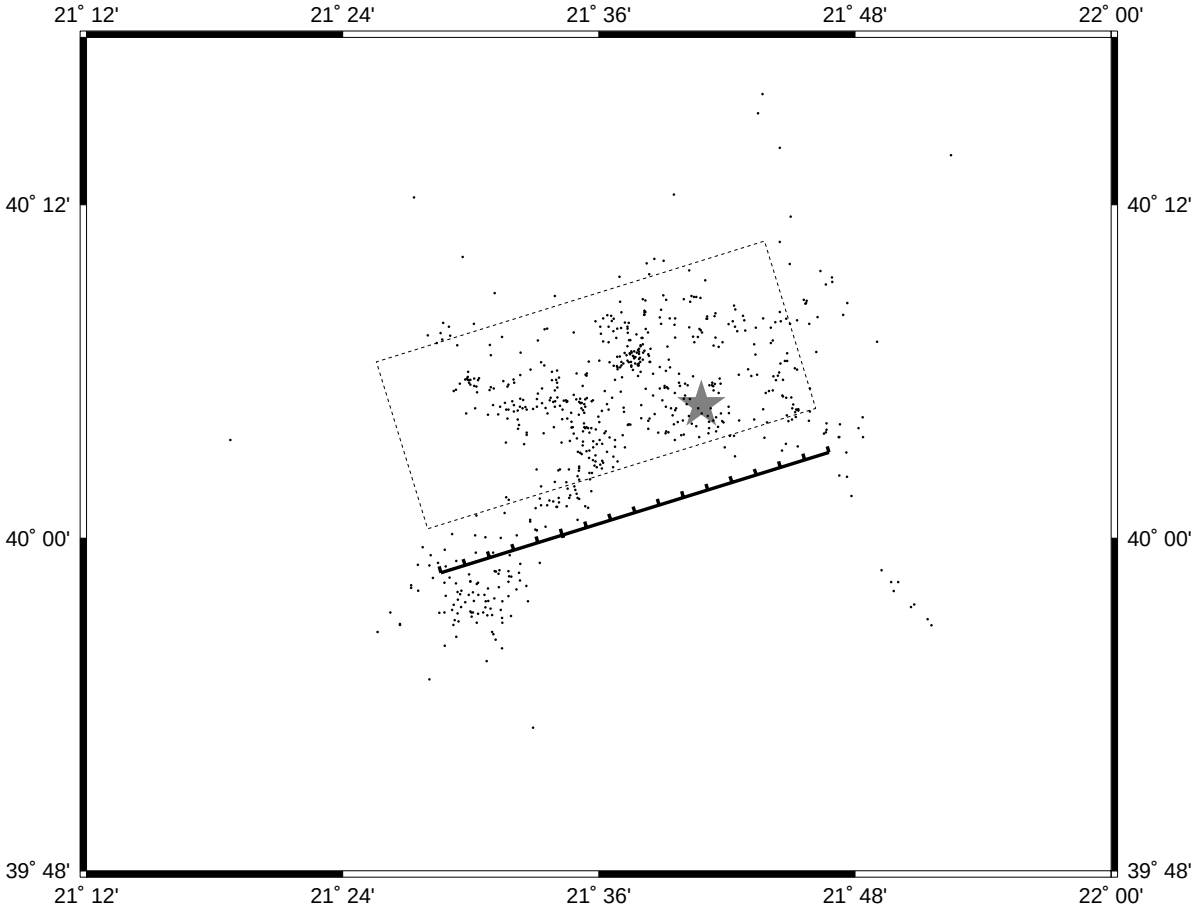

Figure 6.14: Locations of aftershocks to the Grevena earthquake, shown in relation to the main shock (grey star). The vertical projection of the model fault plane onto the surface is shown by the dotted rectangle, and its up-dip continuation to the surface is shown by the heavy line with ticks on the downthrown side.

Date	Location	Geodetic moment	Seismic moment	Geodetic / seismic	Observation interval /yr	Post-seismic interval /yr	Reference
1944 Dec 07	Tonankai	1200	980	1.2	?	?	Ando (1975)
1946 Dec 20	Nankaido	1400	980	1.4	?	?	Ando (1975)
1954 Dec 16	Fairview Peak	50	79	0.6	23	3	Savage & Hastie (1969)
1959 Aug 18	Hebgen Lake	120	110	1.1	?	?	Savage & Hastie (1966)
1964 Mar 28	Alaska	40000	5000	8	?	?	Savage & Hastie (1966)
1966 Jun 27	Parkfield	4.5	0.9–2.1	~ 3	< 2	< 0.5	King *et al.* (1987)
1976 May 06	Friuli	1.5	-	2.2	> 25	1	Briole *et al.* (1986)
1976 Sep 15	Friuli	0.5	-	3.2	> 25	1	Briole *et al.* (1986)
1981 Mar 04	Alkyonides	5.4	2.7	2.0	~ 9	0.5	This study
1983 Oct 28	Borah Peak	26	19	1.4	36–51	0.6	Stein & Barrientos (1985)
1989 Oct 17	Loma Prieta	29	32	0.9	1.0	0.4	Williams *et al.* (1993)
1989 Oct 17	Loma Prieta	30	32	0.9	0.1	0	Lisowski *et al.* (1990)
1991 Apr 22	Costa Rica	-	442	~ 2	0.4	0.2	Lundgren *et al.* (1993)
1992 Jun 28	Landers	90	110	0.8	0	0	Blewitt *et al.* (1993)
1992 Jun 28	Landers	-	-	1.0	0	0	Massonnet *et al.* (1993)
1995 May 13	Grevena	16.7	7.6	2.2	10	0	This study
1995 Jun 15	Egion	4.5	2.4	1.9	0	0	Bernard *et al.* (1996)

Table 6.3: Geodetic *versus* seismic moments for earthquakes observed by both methods (see text). All moments are in units of 10^{18} N m.

6.2.4 Confidence limits of results

One of the advantages of the combined simplex / Monte Carlo inversion method is that the distribution of minima found by the algorithm can be studied in order to estimate the confidence limits of the results qualitatively. Isolated minima that are very far from the expected values, and thus represent unrealistic source parameters are discarded. In addition, further constraints may be applied, *e.g.* to restrict the minima to those representing upper crustal fault planes.

Distributions of minima arising from 2000 restarts are given in Figures 6.15 – 6.17. The figures show scatter-plots of the L_2 norm penalty function χ^2 at each minimum against the chosen parameter. Mild constraints have been applied: maximum depth of faulting has been restricted to less than 20 km (*i.e.* upper crustal faults only, not faults extending to lower depths) and the maximum fault length has been set to 40 km. Both constraints are based on the aftershock distribution and reasonable physical expectations of the behaviour of a continental crustal fault.

The strike and dip of the event (Figure 6.15) show well-defined minima as defined by the bottom envelopes of the scatter-plots. Points lying above the bottom envelope may represent 'false' minima where the simplex algorithm has failed, or 'true' local minima lying out of the plane of the plot, although the former seems

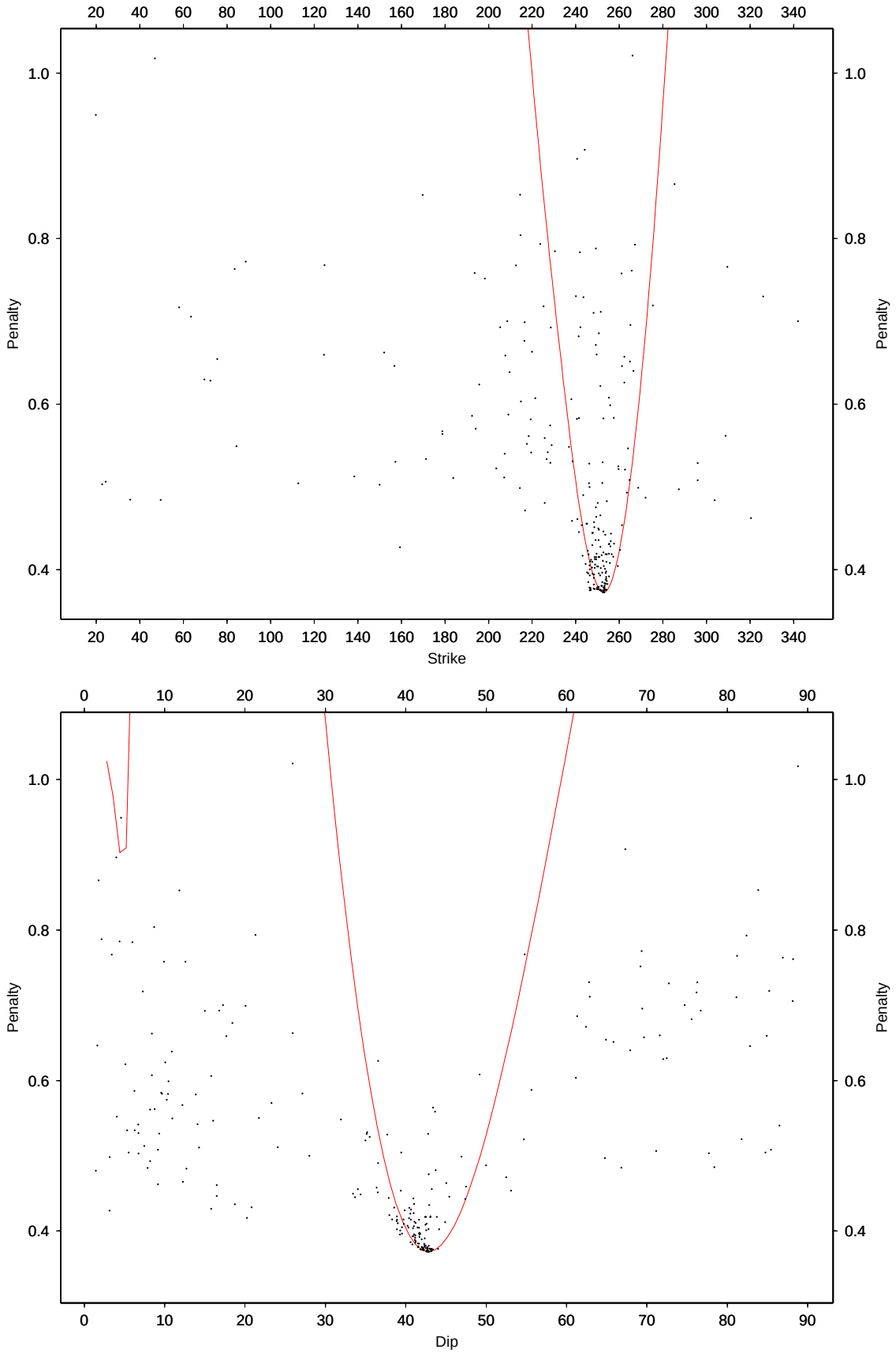

Figure 6.15: Scatter-plots of χ^2 against strike (top) and dip (bottom). Strike and dip are in degrees.

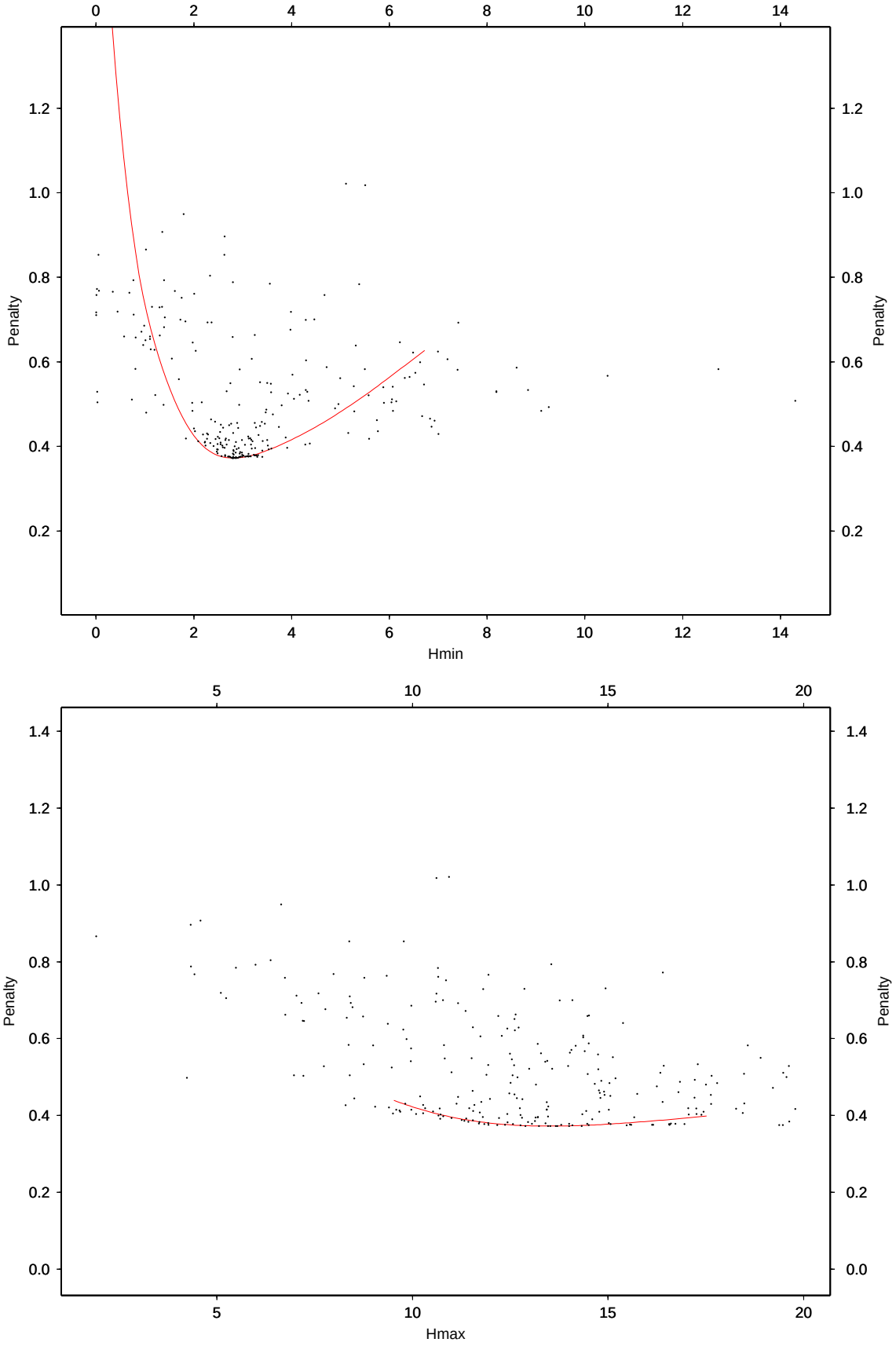

Figure 6.16: Scatter-plots of χ^2 against minimum fault depth (top) and maximum fault depth (bottom). All lengths are in km.

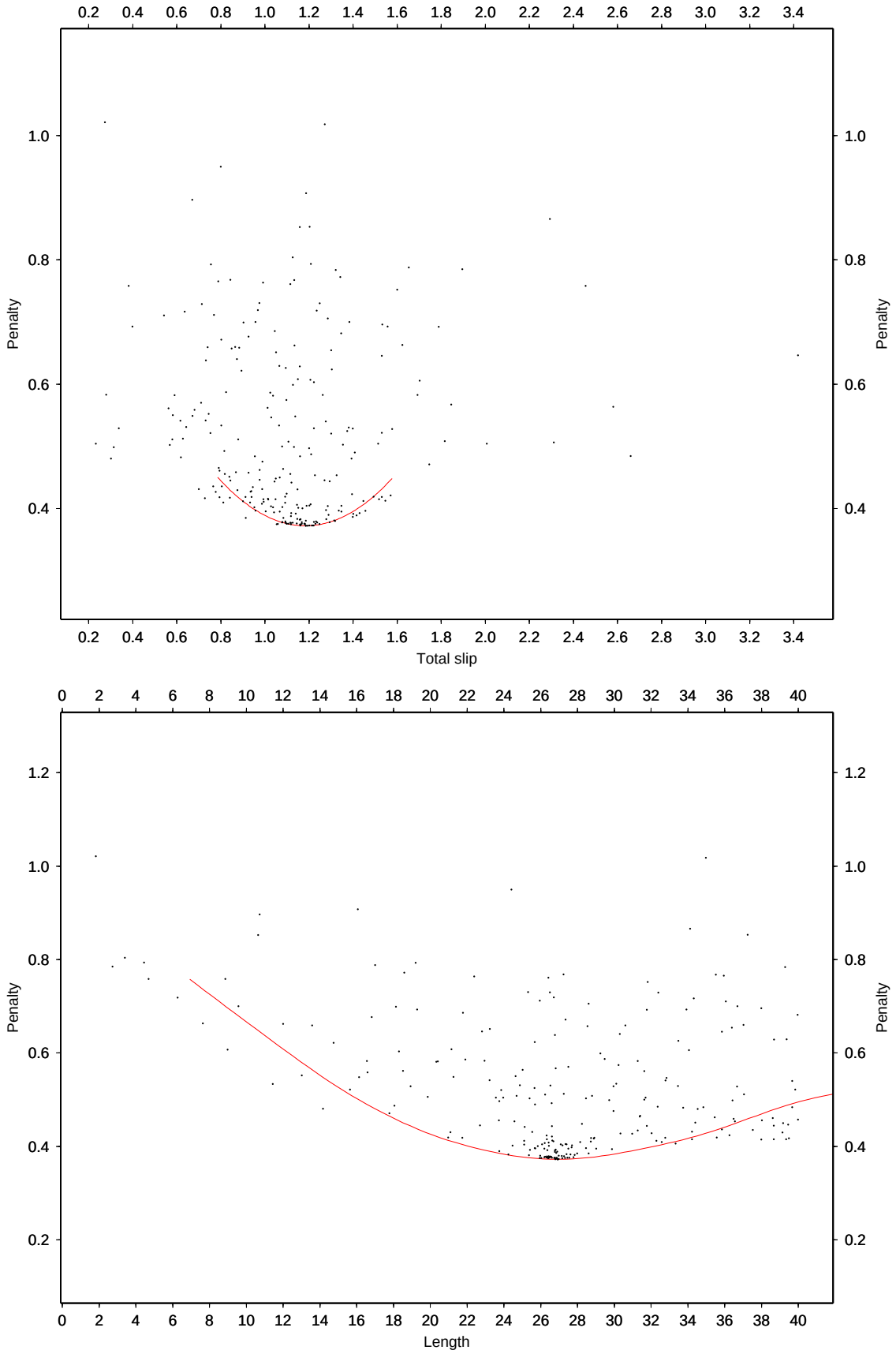

Figure 6.17: Scatter-plots of χ^2 against total slip magnitude (top, in m) and fault length (bottom, in km).

unlikely as the algorithm has been extensively tested for robustness. In either case, the bottom surface of the envelope represents the projection of the penalty function surface from many dimensions onto the single dimension of the parameter concerned, and so provides a crude estimate of the sensitivity of the inversion in the region of the lowest minimum found, when all parameters are included in the inversion. The red line shows the variation in χ^2 when only the displayed parameter is varied. Several of the minima lie beneath the single-parameter variation curve, demonstrating that trade-off between different parameters enlarges the region of 'good' solutions when compared with the situation in which only one parameter is varied. For minima close to the global minimum this observation is particularly important as it illustrates that for several parameters or combinations of parameters a range of parameter values can be compatible with both the dataset and other *a priori* information about the source mechanism. In contrast, some of the remarkably low minima of the penalty function that exist some distance from the global minimum could be eliminated if other *a priori* constraints (from seismology and field observations) were used to filter the solutions more tightly.

The minimum and maximum depth plots (Figure 6.16) show somewhat different characteristics. Minimum depth is well defined, but the maximum-depth χ^2 curve has a very broad minimum, and is poorly constrained in the direction of greater maximum depths. This lack of constraint is partly due to regional secular strain which tends to 'smear out' the co-seismic signal by masking it (increasing maximum depth has the same effect), and partly due to the lack of vertical displacement data (which provides particularly good information about fault depth and dip). Fault length (Figure 6.17, bottom) is relatively well defined, although again less so on the upper bound. Total slip magnitude (Figure 6.17, top) is well defined, but because fault length and maximum depth are poorly defined on the upper bound, the estimate for M_0 inferred from these three quantities (and also minimum depth and fault dip) must lie toward the lower end of the values compatible with the geodetic dataset.

6.2.5 Solution stability

It is not immediately clear whether the scatter of minima found by `okinv` is due to noise in the observed displacements, sub-optimal network configuration, inherent instability of the inverse problem, or some combination of the three. In order to investigate these possible causes I conducted a series of inversion experiments with real data with the observed network geometry, synthetic (noise-free) data generated from the best-fitting geodetic solution with the observed network geometry, and synthetic (noise-free) data generated from the above solution with coarse and fine gridded network configuration (Figure 6.18). Only horizontal displacement data were used. For a discussion of inversions from synthetic data (with or without vertical displacements) with noise, see Subsection 3.5.3.

In each case, the *a priori* bounds of the parameters were fixed to the values given in Table 6.4, and the

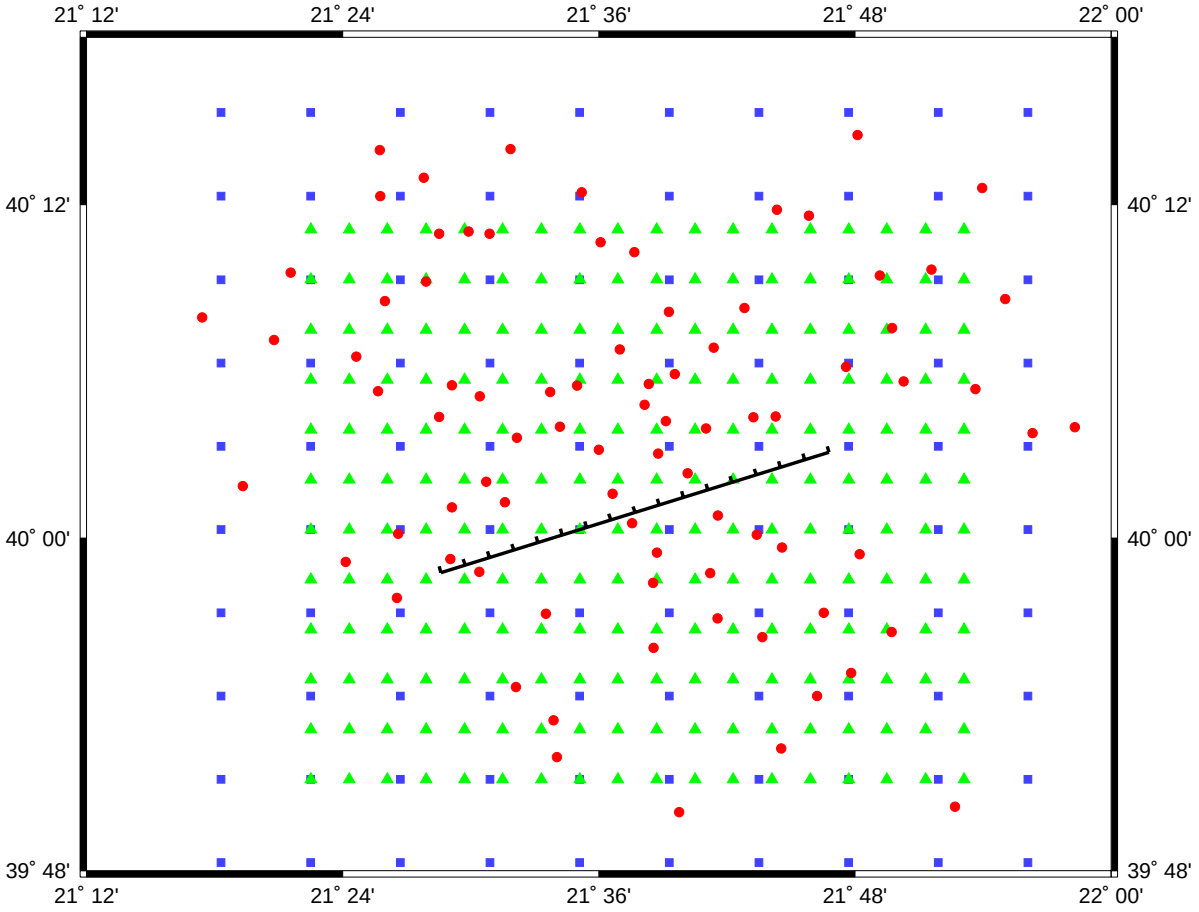

Figure 6.18: Synthetic networks used for the stability experiments. The coarse grid is shown by blue squares, the fine grid by green triangles, and the observed network by red circles. The model fault scarp is shown as a heavy black line with ticks on the downthrown side.

search region for the Monte-Carlo seed points was fixed at different multiples of these values. A series of inversions (normally 100 or 1000) was then run, and the proportion of minima lying exactly on the global minimum (which is known for the synthetic data) or close to it (in terms of *a priori* errors) indicates the stability of the solution within the specified perturbations, for that particular network.

Synthetic gridded data

Experiments were carried out first on a 3×2.5 km rectilinear grid of 216 points centred on the fault scarp (Figure 6.18). Restarts within $\pm$ 0.5σ of the true parameters obtained the ideal solution in all completed attempts, whereas restarts within $\pm$ 1σ obtained 'good' solutions (within $\pm$ 0.5σ of the true values for all parameters) in 83% of completed attempts (including the ideal solution in 50% of completed attempts).

Scarp		Total slip	Strike	Dip	Rake	Length	Dmin	Dmax	Scale &	Offsets
Lat.	Lon.	/m				/km	/km	/km	rotation	/m
± 0.1°	± 0.1°	± 0.1	± 20°	± 20°	± 20°	± 10	± 2	± 5	± 1 p.p.m.	± 0.1

Table 6.4: *A priori* bounds (1σ) for Monte-Carlo restart experiments.

For restarts within ± 2σ, 'good' solutions were only obtained in 9% of completed attempts, although the ideal solution was recovered in several cases.

A second set of experiments was performed on a larger but coarser grid of 100 points spaced 5×5 km. Restarts within ± 0.5σ recovered the ideal solution in 99% of completions, and a very close solution in the other 1%. For restarts within ± 1σ, the ideal solution was obtained in 64% of completions, and 'good' solutions in a further 22% of cases. However, restarts within ± 2σ only obtained 'good' solutions in 20% of completed inversions, although this is a better result than that for the finer grid.

The advantage of the second grid is that it covers a wider area and this appears to outweigh the effects of coarser observation spacing and halving the number of observations, particularly when inverting from more widely distributed starting parameters (corresponding to the situation of poor *a priori* knowledge of these parameters). A series of inversions was performed using the coarsely-gridded data, but with the scarp parameters (latitude, longitude, strike and length) fixed and the other parameters restarted from within ± 2σ of the true values. This method obtained the ideal solution in 85% of completed inversions, indicating that very good *a priori* knowledge of some parameters can make up for very poor knowledge of others.

Synthetic data with observed station distribution

Restarts from within ± 2σ find 'good' solutions in 5% of completed inversions, whereas restarts from within ± 1σ obtain the ideal solution in 32% of completions and 'good' solutions in a further 33% of cases. For restarts within ± 0.5σ, 'good' solutions were obtained in all completed runs, and the ideal solution in 95% of these cases. This performance is worse than that for the coarse grid, which has a similar number of data points, but still shows that the theoretical solution is reasonably stable even to perturbations of ± 1σ about the true parameters, given the network geometry that we actually observed.

Observed data

The presence of significant noise in the real data would be expected to smear the co-seismic signal present in the observed displacements and this would adversely affect the stability of the solution. This is indeed

the case: for restarts within $\pm$ 0.5σ of the best-fitting solution, the best-fitting solution is only recovered in 17% of completed inversions, although close approximations (within the limits specified above) are found in a total of 82% of completed runs. For restarts within $\pm$ 0.2σ, close approximations to the best-fitting solution are recovered in 99% of cases, including the precise best-fitting solution 50% of the time, so the solution is still stable with real data although within somewhat narrower limits than for the synthetic (noise-free) case.

Implications

This series of experiments shows that the `okinv` algorithm is able to recover global minima to the inverse problem even given widely-scattered starting parameters. The presence of parameters with good *a priori* constraints further increases the effectiveness of the `okinv` algorithm, but is not required. The stability of the solution is affected more by the distribution of data points than their number, with a widely-spaced grid better able to recover the solution than either a denser but smaller grid or a network of the observed geometry, unless good *a priori* constraints can be applied.

However, inversions from the observed network are able to recover the synthetic parameters well, and the presence of closely-spaced observations acts as insurance against noise or outliers in the real data. These have an adverse effect on the stability of the solution, but the best-fitting solution from the real data is still reasonably stable.

6.3 Summary

A pre-existing triangulation network surrounding the 13 May 1995 Kozani – Grevena earthquake has been occupied since the earthquake using GPS and the co-seismic horizontal displacements have been used to determine the event parameters. The observations are capable of resolving the event parameters without the need for *a priori* information except to eliminate physically unreasonable solutions, but the addition of reliable vertical displacement data would strengthen the results. Experiments with real and synthetic data have shown that the solution is stable, and the inversion algorithm can robustly recover this solution without the need for accurate *a priori* information.

The majority of geodetically-determined parameters of the event are compatible with those determined from global seismic studies, which define the strike, dip and rake or the event, or the local aftershock study, which defines the depth extent of faulting, fault length, dip, strike and position. However, the scalar moment implied by the geodetic model is roughly twice that determined by global CMT inversions, and the lower bound of the geodetic estimate is well defined, so this discrepancy may be a significant one.

Some of the excess of geodetic over seismic (mainshock) deformation may have occurred during aftershocks, but at least 0.3 m of discrepant slip remains. This is unlikely to have occurred at a uniform rate between the two survey epochs, but may have occurred shortly before or after the mainshock. If such a proportion of 'near-seismic' slip were to occur in other earthquakes, as has been demonstrated by a literature review, it might resolve the discrepancy between geodetic and seismic strain in some regions. However the discrepancy in the western Gulf of Korinthos discussed in Chapter 5 is too large to be explained in this way, and represents real seismic hazard.

A network of GPS sites has been established by virtue of multiple occupations in September 1995 and May 1995 (for the majority of stations) which will enable the future study of post-seismic strain in the region of the event. Both post-seismic and co-seismic networks were designed with the aid of forward models of the expected deformation.

Chapter 7

Conclusions

This thesis has been concerned with the measurement and interpretation of short- to medium-term crustal deformation in an extensional tectonic setting. Two different, but related, processes have been observed using geodetic surveying techniques. Firstly, the inter-seismic accumulation of strain in central Greece has been measured using GPS surveying over a period of up to seven years, and has been related to the longer-term deformation revealed by both surveying and seismology, and to the potential for earthquake hazard. Secondly, co-seismic release of strain in northwestern Greece has been measured using a combination of GPS and conventional terrestrial surveying. A novel variation on previous methods of inversion for the source parameters from geodetic displacements has been described and used, and the results have been compared with those from seismological data. Here I summarise the sets of conclusions that have been made individually in the preceding chapters, and relate them and their implications to each other.

7.1 The Central Greece Network

The Central Greece Network (Figure 4.1) has been surveyed fully on three occasions, but on the first occasion (1989), only half the sites were occupied with dual-frequency GPS receivers, and even these dual-frequency measurements are too inaccurate to be used for short-term deformation monitoring. Processing of the two subsequent epochs of measurement using a fiducial GPS method has yielded relative coordinates with a high degree of precision which are complemented by relative coordinates from three further epochs at which the part of the network surrounding the Gulf of Korinthos has been occupied, which were processed using CODE precise orbits.

The 15 June 1995 Egion earthquake displaced sites bordering the western Gulf of Korinthos by a significant amount (Figure 5.7), but these co-seismic effects have been removed using a forward model based on the

combined SAR, GPS and seismological study of Bernard *et al.* (1996), leaving the inter-seismic strain. The smooth changes with time of inter-site baseline length (Figure 4.6) demonstrate that the scale of the GPS reference frame is reproducible from epoch to epoch, despite the mixing of coordinate sets from fiducial and precise orbit processing, but the cumulative site displacements are subject to systematic offsets which may be result of errors in the epoch positioning over the base station. By constraining site velocities to be constant within error, these systematic offsets can be estimated and eliminated, leaving a set of site displacements which vary smoothly with time (Figures 4.7 and 4.8). The offsets have r.m.s. values of 9.4 mm (east) and 6.1 mm (north), which compares with r.m.s. residuals of 7.2 mm per coordinate to the fit of all sites to their smoothed velocities. Solving for a rotational error in reference frame realisation at each epoch does not improve the fit, confirming that errors are predominantly translational, caused by setup problems at the base station. Expression of the velocities in a Europe-fixed reference frame (Figure 4.8) reveals that the northwestern part of the Central Greece Network is moving with respect to Europe, and so deformation must occur between the two.

The time series of coordinates of Central Greece Network sites cannot be fitted within error a polynomial velocity field of 4^{th} order or less (Figures 4.9 and 4.10), implying that crustal deformation in Greece is localised at a scale much smaller than that of the network. Analysis of uniform strain rates within small independent polygonal regions (Figures 4.11 and 4.12) reveals significant high strain rates in the western Gulf of Korinthos which are not matched elsewhere, confirming this hypothesis. Elsewhere in central Greece, strain rates are lower, even across other extensional structures such as the Gulf of Evvia. These short-term geodetic findings are confirmed by hundred-year measurements (Billiris *et al.*, 1991; Davies, 1996; Davies *et al.*, 1996).

7.2 The Gulf of Korinthos

The Gulf of Korinthos is surrounded by the most extensively surveyed part of the Central Greece Network, and as one of the most densely populated parts of Greece has a good record of historical seismicity and a high susceptibility to earthquake hazard. A reliable catalogue of seismicity over the last hundred years has been compiled by Ambraseys & Jackson (1990), and the rate of seismic moment release can be related to the geodetic extension rate using the method of Kostrov (1974).

The Central Greece Network data show very little internal deformation along the north of the Peleponnessos, and so can be rotated into a reference frame in which the Peleponnessos is fixed, whereby inter-seismic extension across the Gulf perpendicular to strike can be investigated, once co-seismic effects of the 15 June 1995 Egion earthquake have been eliminated. They and the hundred-year data of Davies *et al.* (1996) both show a smooth and significant increase in extension rate from east to west along

the Gulf (Figures 5.10 and 5.11), whereas the seismicity rates in the western and eastern Gulf are similar. Agreement between five-year and hundred-year inter-seismic extension rates (Figures 5.11 and 5.12) demonstrates that inter-seismic deformation is steady with time, which implies that it is not controlled by ephemeral co-seismic processes in the upper crust.

Seismic strain release rates this century in the eastern Gulf match present-day extension rates, if a 10 km thick seismogenic layer is assumed, and the two are not seriously in disagreement for a 15 km thick seismogenic layer (Table 5.4). However, in the western Gulf, there is a large deficit of seismic strain release, corresponding to several earthquakes the size of the 1995 Egion event, which would not be eliminated even if the magnitude of all events this century had been underestimated by a factor of two (see below). The record of higher seismicity over the preceding three centuries, while less reliable, also indicates that seismic activity in the western Gulf of Korinthos this century has been unusually sparse, and that medium-term seismic hazard in this region is likely to be high.

7.3 The 1995 Grevena earthquake

The 13 May 1995 Grevena earthquake (Figures 6.1 and 6.2), the largest in mainland Greece since 1981, struck a region of low historical seismicity, demonstrating that an absence of earthquakes in the past is no guarantee of their continued absence. The geodetic strain in this region has not previously been measured, although the fact that sites in the northwest of the Central Greece Network are moving with respect to 'stable' Europe (Figure 4.8)indicates that deformation must occur in this area. However, a geodetic network that was recently occupied with conventional terrestrial surveying techniques exists in the area, and part of this network was reoccupied with GPS immediately after the 1995 earthquake. In addition to observing co-seismic displacements of all the sites in the GPS network, a subset of the sites was occupied with higher precision in May and September 1995 so that future post-seismic displacements of these sites can be studied.

Inversion for the source parameters of the earthquake from a (necessarily imperfect) set of geodetic surface displacements is a highly non-linear problem, even for uniform displacement across a single fault plane. Use of a downhill simplex inversion method combined with 'Monte-Carlo' – style random selection of the seeding parameter values enables such problems to be solved without the need for *a priori* constraints, other than those necessary to restrict the solution to physically reasonable parameter values. Scatter of the solutions found by the combined algorithm yields more complete information about the confidence limits of the solution than can be found from a one-shot inversion scheme. Experiments with synthetic datasets show that the method is robust and yields stable solutions.

Parameters of the 1995 Grevena earthquake obtained by inversion from the geodetic data are in good

agreement with the evidence from teleseismic and aftershock studies, with the exception of the scalar moment which is approximately twice that expected from seismological analysis. Such an excess of geodetic displacement has been noted for other earthquakes, and bears no relation to the post-seismic or total interval between surveys, so may be the result of very short term aseismic creep before or soon after the earthquake. However, even if this does represent a general trend, the doubling of seismic (or 'near-seismic') strain release would not be sufficient to remove the deficit of seismic strain release compared with geodetic strain which is observed in the western Gulf of Korinthos.

The model fault scarp location is in agreement with the aftershock distribution (Figures 6.13 and 6.14), but does not correspond to any of the clear-cut surface features in the region, but instead to a poorly-defined scarp within the zone of soft surficial sediments, close to the location of the most prominent co-seismic ground cracking. Without geodetic observation and modelling it is likely that such a feature would have been overlooked in the search for faults associated with possible seismic hazard, and this illustrates the value of geodetic studies in seismic hazard assessment.

7.4 General conclusions

In this thesis I have demonstrated that GPS is a viable tool for the measurement of crustal strain over the time-scale of a few years, and that the observed strain rates in central Greece are broadly similar to the rates observed over a hundred-year time-scale. The implication of this result for continental deformation studies is, that provided sufficient strain has accumulated to be detectable, a five-year period can provide an estimate of the inter-seismic strain as representative as that obtained over a hundred-year period. Set against this is the fact that over longer time-scales, the effects of individual earthquakes will average out over sufficiently large regions, but for short time-scales and small regions where a few large events may bias the measurement, the co-seismic effects of each earthquake must be removed by modelling to leave the inter-seismic strain.

The geodetic (actual) deformation caused by the release of strain in an earthquake does not always match that expected from teleseismic evidence. This study has shown that while the majority of conclusions from the geodetic study of the 1995 Grevena earthquake are in agreement with the global CMT and local aftershock studies, the scalar moment of the event estimated from the geodetic evidence is much higher than that of the CMT solution. It is possible that aseismic strain release shortly before or after an earthquake ('near-seismic' strain) may account for such a discrepancy. For the forward modelling of the 1995 Egion earthquake that was necessary to estimate inter-seismic strain in the Gulf of Korinthos, a model of the event based on a combined GPS, SAR and seismological study was used.

Although the inter-seismic deformation in central Greece seems temporally uniform, the spatial distribution

of strain is not. Across the whole region, a low-order polynomial velocity field is unable to fit the time series of observed coordinates, and within the Gulf of Korinthos, hundred-year and five-year observations confirm that extension this century is higher at the western end. Strain is therefore localised at some scale within the continental crust of Greece, and so the stress will more quickly build up to the level of failure in these localities. Based on the historical seismic record and on comparison with the eastern Gulf of Korinthos, much of the geodetic strain in the western Gulf of Korinthos is likely to be released during future earthquakes, even if 'near-seismic' strain in past earthquakes has been underestimated by a factor similar to the seismic underestimate of deformation in the 1995 Grevena earthquake.

The agreement between hundred-year seismic and five- to hundred-year geodetic strain rates in the eastern Gulf of Korinthos is compatible with the assumption of a 10 – 15 km seismogenic layer under the Gulf of Korinthos. This compares with the 13.5 km maximum depth of faulting estimated by inversion from the geodetic horizontal displacements in the 1995 Grevena earthquake in northern Greece, although in this case, faulting only comes to within 2.8 km of the surface because of the presence of a surficial layer of unconsolidated sediments. Such agreement between different regions with different histories of extension bodes well for seismic hazard studies based on the comparison of geodetic and seismic strain which must necessarily assume a seismogenic layer thickness, often by extrapolation or comparison with nearby regions.

7.5 Future work

A scheme of reoccupations of the Grevena and Egion post-seismic GPS networks is currently being executed. As these networks are denser than that surrounding the location of the 1981 Alkyonides earthquakes and have been surveyed with GPS from the start, they offer a far better opportunity to quantify post-seismic deformation after normal-faulting earthquakes. In particular, the Grevena network seems likely to yield good results because of the low level of background crustal strain expected in the light of historical seismicity in the region. However, the region lies outside the extent of the Central Greece Network and the 1890s geodetic network and so the background strain must be quantified by future geodetic measurements throughout northern Greece. Such measurements will also improve understanding of the seismic hazard in northern Greece and the northwestward extent of Aegean deformation as 'stable' Europe is approached. Further modelling of the expected post-seismic deformation is required in the light of existing models of the co-seismic perturbing signal which are continually being updated.

Future work on the Grevena and Egion earthquakes will also include an integration of GPS and SAR observations as the latter become available. It is possible that more detailed models will be required in order to reconcile the two sets of observations, because GPS surveying reveals more about the horizontal deformation whereas SAR interferometry reveals more about the vertical deformation. One possible refinement

to the simple homogeneous elastic model that has been used in this thesis is a layered-Earth elastic model. Geologically, this model is far more realistic, especially in the case of the Grevena earthquake where the bedrock is overlain by a thick layer of very soft sediments which may have deformed differently during the earthquake. Such layered-Earth modelling may also be necessary for the post-seismic deformation because the implementation of the current model is only for the case of a single elastic layer overlying a visco-elastic half-space, and the effects of a weaker elastic or visco-elastic surficial layer and / or a strong elastic upper mantle beneath the visco-elastic lower crust may be significant.

Surveying of the Central Greece Network with GPS has proved capable of detecting crustal deformation over short time-scales, but much of the network has only been occupied reliably on two occasions (1991 and 1993). To obtain good estimates of crustal strain in these parts of the network, which are straining at much lower rates than the Gulf of Korinthos, it will be necessary to reoccupy more sites. The successful detection of low strain rates across the eastern Gulf of Korinthos demonstrates that a sufficient interval has already elapsed to make this proposition worthwhile. It is expected that such measurements will confirm the low strain rates in the northwest of the network and the surprisingly low strain detected across the extensional Gulf of Evvia, while confirming that significant movement (not yet revealed by SLR) occurs between the combined GPS / SLR sites at DION and XRIS.

7.6 Summary

With reference to the aims of this thesis (Section 1.5), the following conclusions have been demonstrated:

- Inter-seismic deformation rates observed in central Greece on the five-year time-scale matches hundred-year deformation rates very closely and can be quantified precisely using three or more epochs of GPS data.
- The velocity field of central Greece 1991 – 1996 cannot be fitted by a low (up to 4^{th}) order polynomial. Strain is present everywhere, but high strains are localised within the Gulf of Korinthos.
- An algorithm has been presented which robustly inverts for earthquake source mechanisms given local geodetic displacements. The mechanism determined geodetically for the 1995 Grevena earthquake agrees well with the aftershock measurements, but implies a higher scalar moment than the CMT solution.

In addition, the following have been achieved:

- A method has been presented which corrects possible errors in relative GPS base station positioning, and allows smooth velocities to be estimated for all sites in the network.

- Medium-term seismic hazard has been identified in the western Gulf of Korinthos using five-year and hundred-year geodetic data, and historical seismicity data.
- Forward modelling of post-seismic visco-elastic relaxation has been used to design networks that enable future measurement of post-seismic strain surrounding the 1995 Grevena and Egion earthquakes.

References

Abercrombie, R.E., Main, I.G., Douglas, A., & Burton, P.W. 1995. The nucleation and rupture process of the 1981 Gulf of Corinth earthquakes from deconvolved broad-band data. *Geophys. J. Int.*, **120**, 393–405.

Agatza-Balodimou, A.M., Briole, P., Lyon-Caen, H., Mitsikaki, C., Rigo, A., Ruegg, J.-C., Papazissi, K., & Veis, G. 1995. Recent developments in deformation studies from geodetic data in the Corinthian Gulf (Greece). *Pages 759–769 of: Proc. 1st International Symposium on Deformations in Turkey.* Ankara: TMMOB–HKMO.

Ambraseys, N.N., & Jackson, J.A. 1990. Seismicity and associated strain of central Greece between 1890 and 1988. *Geophys. J. Int.*, **101**, 663–708.

Anderson, H., & Jackson, J.A. 1987. Active tectonics of the Adriatic region. *Geophys. J. R. Astron. Soc.*, **91**, 937–983.

Ando, M. 1975. Source mechanisms and tectonic significance of historical earthquakes along the Nankai trough, Japan. *Tectonophys.*, **27**, 119–140.

Angelier, J., Lybéris, N., Pichon, X. Le, Barrier, E., & Huchon, P. 1982. The tectonic development of the Hellenic arc and the Sea of Crete: a synthesis. *Tectonophys.*, **86**, 159–196.

Argus, D.F., & Gordon, R.G. 1990. Pacific–North American plate motion from Very Long Baseline Interferometry compared with motion inferred from magnetic anomalies, transform faults, and earthquake slip vectors. *J. Geophys. Res.*, **95**(B11), 17,315–17,324.

Argus, D.F., & Gordon, R.G. 1991. No-net-rotation model of current plate velocities incorporating plate motion model NUVEL-1. *Geophys. Res. Lett.*, **18**(11), 2,039–2,042.

Armijo, R., Meyer, B., King, G.C.P., Rigo, A., & Papanastassiou, D. 1996. Quaternary evolution of the Corinth Rift and its implications for the Late Cenozoic evolution of the Aegean. *Geophys. J. Int.*, **126**, 11–53.

Bernard, P., Briole, P., Meyer, B., Lyon-Caen, H., Gomez, J.-M., Tiberi, C., Berge, C., Hatzfeld, D., Lachet, C., Lebrun, B., Deschamps, A., Courboulex, F., Laroque, C., Rigo, A., Massonnet, D., Papadimitriou, P., Kassaras, J., Diagourtas, D., Makropoulos, K., Veis, G., Papazisi, E., Mitsikaki, C., Karakostas, V., Papadimitriou, E., & Papanastassiou, D. 1996. The $M_s = 6.2$, June 15, 1995 Aigion earthquake (Greece): Results of a multidisciplinary study. *J. seismol.* Under review.

Beutler, G., Gurtner, W., Rothacher, M., Wild, U., & Frei, E. 1989. Relative static positioning with the Global Positioning System: Basic technical considerations. *In:* Bock, Y., & Leppard, N. (eds), *Global Positioning System: an Overview*. Springer-Verlag. IAG Symposium 102.

Bibby, H.M. 1976. Crustal strain across the Marlborough faults, New Zealand. *N.Z. J. Geol. Geophys.*, **19**(4), 407–425.

Bibby, H.M. 1982. Unbiased estimate of strain from triangulation data using the method of simultaneous reduction. *Tectonophys.*, **82**, 161–174.

Biju-Duval, B., & Montadert, L. (eds). 1977. *Structural history of the Mediterranean basins*. Paris: Technip.

Billiris, H., Paradissis, D., Veis, G., England, P., Featherstone, W., Parsons, B., Cross, P., Rands, P., Rayson, M., Sellers, P., Ashkenazi, V., Davison, M., Jackson, J., & Ambraseys, N. 1991. Geodetic determination of tectonic deformation in central Greece from 1900 to 1988. *Nature*, **350**, 124–129.

Bird, P. 1991. Lateral extrusion of lower crust from under high topography, in the isostatic limit. *J. Geophys. Res.*, **96**(B6), 10,275–10,286.

Blewitt, G., Heflin, M.B., Webb, F.H., Lindqwister, U.J., & Malla, R.P. 1992. Global coordinates with centimetre accuracy in the International Terrestrial Reference Frame using GPS. *Geophys. Res. Lett.*, **19**, 853–856.

Blewitt, G., Heflin, M.B., Hurst, K.J., Jefferson, D.C., Webb, F.H., & Zumberge, J.F. 1993. Absolute far-field displacements from the 28 June 1992 Landers earthquake sequence. *Nature*, **361**, 340–342.

Bock, Y., Agnew, D.C., Fang, P., Genrich, J.F., Hager, B.H., Herring, T.A., Hudnut, K.W., King, R.W., Larson, S., Minster, J.-B., Stark, K., Wdowinski, S., & Wyatt, F.K. 1993. Detection of crustal deformation from the Landers earthquake sequence using continuous geodetic measurements. *Nature*, **361**, 337–340.

Bomford, G. 1980. *Geodesy*. Fourth edn. Oxford: Oxford University Press.

Bott, M.H.P., & Dean, D.S. 1973. Stress diffusion from plate boundaries. *Nature*, **243**, 339–341.

Boucher, C., & Altamimi, Z. 1991. *ITRF 89 and other realizations of the IERS terrestrial reference system for 1989*. IERS Technical Note, no. 6. Paris: Observatoire de Paris.

Boucher, C., Altamimi, Z., & Duhem, L. 1993. *ITRF 92 and its associated velocity field.* IERS Technical Note, no. 15. Paris: Observatoire de Paris.

Boucher, C., Altamimi, Z., & Duhem, L. 1994. *Results and analysis of the ITRF93.* IERS Technical Note, no. 18. Paris: Observatoire de Paris.

Bourne, S., Clarke, P., England, P., Parsons, B., Rowe, G., Beavan, J., Blick, G., Darby, D., Wood, P., Arnadottir, T., & Walcott, R. 1995. Geodetic determination of crustal deformation in the north of the South Island, New Zealand. *Terra Abstracts*, 37–37. Abstract supplement no. 1 to Terra Nova, vol. 7.

Bourne, S.J. 1996. *Distributed deformation of the South Island of New Zealand.* D.Phil. thesis, University of Oxford, Oxford, UK. (In preparation).

Brace, W.F., & Kohlstedt, D.L. 1980. Limits on lithostatic stress imposed by laboratory experiments. *J. Geophys. Res.*, **85**(B2), 6,248–6,252.

Braunmiller, J., & Nábělek, J. 1996. Geometry of normal faults: Seismological constraints. *J. Geophys. Res.*, **101**(B2), 3,045–3,052.

Briole, P., de Natale, G., Gaulon, R., Pingue, F., & Scarpa, R. 1986. Inversion of geodetic data and seismicity associated with the Friuli earthquake sequence (1976–1977). *Annales geophysicae*, **4**(B4), 481–492.

Brooks, M., & Ferentinos, G. 1984. Tectonics and sedimentation in the Gulf of Corinth and the Zakynthos and Kefallinia channels, western Greece. *Tectonophys.*, **101**, 25–54.

Byerlee, J. 1978. Friction of rocks. *Pure. Appl. Geophys.*, **116**, 615–626.

Chen, W.-P., & Molnar, P. 1983. Focal depths of intracontinental and intraplate earthquakes and their implications for the thermal and mechanical properties of the lithosphere. *J. Geophys. Res.*, **88**(B5), 4,183–4,214.

Cheng, A., Jackson, D.D., & Matsu'ura, M. 1987. Aseismic crustal deformation in the Transverse Ranges of southern California. *Tectonophys.*, **144**, 159–180.

Clarke, P., Curtis, A., England, P., Parsons, B., Paradissis, D., Veis, G., & Billiris, H. 1995a. Geodetic studies of aspects of strain accumulation and release during the seismic cycle: Results from the Gulfs of Korinthos and Argos. *Pages 746–747 of: Proc. of the 1st International Symposium on Deformations in Turkey.* Ankara: TMMOB–HKMO. (abstract only).

Clarke, P., Paradissis, D., England, P., Parsons, B., Billiris, H., Veis, G., Briole, P., & Ruegg, J.-C. 1996. Geodetic investigation of the 13 May 1995 Kozani – Grevena (Greece) earthquake. *Submitted to Geophys. Res. Lett.*.

Clarke, P.J., Curtis, A., Galanis, I., Billiris, H., England, P., Paradissis, D., Parsons, B., & Veis, G. 1995b. Geodetic determination of strain during the seismic cycle: Results from the Alkyonides region, central Greece. *Terra Abstracts*, 35–35. Abstract supplement no. 1 to Terra Nova, vol. 7.

Cohen, S.C. 1980. Postseismic viscoelastic surface deformation and stress 1: Theoretical considerations, displacement and strain calculations. *J. Geophys. Res.*, **85**, 3,131–3,150.

Cohen, S.C., & Kramer, M.J. 1984. Crustal deformation, the earthquake cycle, and models of viscoelastic flow in the asthenosphere. *Geophys. J. R. Astron. Soc.*, **78**, 735–750.

Collier, R.E.L., Leeder, M.R., Rowe, P.J., & Atkinson, T.C. 1992. Rates of tectonic uplift in the Corinth and Megara basins, Central Greece. *Tectonics*, **11**, 1,159–1,167.

Cost, T.L. 1964. Approximate Laplace transform inversions in visco-elastic stress analysis. *AIAA J.*, **2**, 2,157–2,166.

Cross, P.A. 1983. *Advanced least squares applied to position fixing.* First edn. Department of Land Surveying Working Papers, no. 6. London: University of East London. (Reprinted with corrections, 1990).

Cross, P.A. 1984. *The Geodetic Suite.* Second edn. Department of Land Surveying Working Papers, no. 4. London: Polytechnic of East London. (MS-DOS version).

Cross, P.A. 1989. The detection of rotations by surveying techniques. *In:* Kissel, C., & Laj, C. (eds), *Paleomagnetic rotations and continental deformation.* Dordrecht: Kluwer.

Cross, P.A. 1990. Practical integration of GPS and classical control networks. *Pages 336–348 of: Proc. of the XIX FIG congress.*

Cross, P.A., Hollwey, J.R., & Small, L.G. 1980. *Geodetic appreciation.* First edn. Department of Land Surveying Working Papers, no. 2. London: Polytechnic of East London. (Reprinted with corrections, 1989).

Curtis, A. 1994. *Surface-wave studies and elastic models of extensional zones: the Tibetan Plateau and Aegean region.* D.Phil. thesis, University of Oxford, Oxford, UK.

Curtis, A., England, P., Clarke, P., & Parsons, B. 1995. An analysis of geodetic observations in Greece and the Cyclades using a static elastic relaxation model. *Terra Abstracts*, 171–171. Abstract supplement no. 1 to Terra Nova, vol. 7.

Das, S., & Scholz, C.H. 1981. Off-faul aftershock clusters caused by shear stress increase? *Bull. Seismol. Soc. Am.*, **71(5)**, 1,669–1,675.

Davies, R.R. 1996. *Large geodetic and geomorphological studies of crustal strain and active faulting in southern and central Greece.* D.Phil. thesis, University of Oxford, Oxford, UK.

Davies, R.R., England, P.C., Parsons, B.E., Billiris, H., Paradissis, D., & Veis, G. 1996. Geodetic strain of Greece in the interval 1892–1992. *Submitted to J. Geophys. Res.*.

DeMets, C., Gordon, R.G., Argus, D.F., & Stein, S. 1990. Current plate motions. *Geophys. J. Int.*, **101**, 425–478.

DeMets, C., Gordon, R.G., Argus, D.F., & Stein, S. 1994. The effect of recent revisions to the geomagnetic reversal time scale on estimates of current plate motions. *Geophys. Res. Lett.*, **21**(20), 2,191–2,194.

Denys, P.H., Cross, P.A., Veis, G., Billiris, H., Paradissis, D., Ashkenazi, V., Bingley, R., Clarke, P., England, P., Parsons, B.E., Kahle, H.-G., & Mueller, M.V. 1995. GPS networks for determining the accumulation of current crustal strain in Central Greece. *Pages 748–758 of: Proc. of the 1st International Symposium on Deformations in Turkey*. Ankara: TMMOB–HKMO.

Dewey, J., & Şengör, A.M.C. 1979. Aegean and surrounding regions: Complex multiplate and continuum tectonics in a convergent zone. *Geol. Soc. Am. Bull.*, **90**, 84–92.

Dixon, T.H. 1991. An introduction to the Global Positioning System and some geological applications. *Rev. Geophys.*, **29**(2), 249–276.

Dixon, T.H. 1993. GPS measurement of relative motion of the Cocos and Caribbean plates and strain accumulation across the Middle America Trench. *Geophys. Res. Lett.*, **20**(20), 2,167–2,170.

Dong, D.-N., & Bock, Y. 1989. Global Positioning System network analysis with phase ambiguity resolution applied to crustal deformation studies in California. *J. Geophys. Res.*, **94**(B4), 3,949–3,966.

Doutsos, T., & Poulimenos, G. 1992. Geometry and kinematics of active faults and their seismotectonic significance in the western Corinth–Patras rift (Greece). *J. Struct. Geol.*, **14**(6), 689–699.

Dziewonski, A.M., & Woodhouse, J.H. 1983. An experiment in systematic study of global seismicity: Centroid-moment tensor solutions for 201 moderate and large earthquakes of 1981. *J. Geophys. Res.*, **88**(B4), 3,247–3,271.

Dziewonski, A.M., Ekström, G., Franzen, J.E., & Woodhouse, J.H. 1988. Global seismicity of 1981: Centroid-moment tensor solutions for 542 earthquakes. *Phys. Earth Planet. Inter.*, **50**(2), 155–182.

Ekström, G., & Dziewonski, A.M. 1988. Evidence of bias in the estimation of earthquake size. *Nature*, **332**, 319–323.

Ekström, G., & England, P. 1989. Seismic strain rates in regions of distributed deformation. *J. Geophys. Res.*, **94**(B8), 10,231–10,257.

England, P., & Jackson, J. 1989. Active deformation of the continents. *Ann. Rev. Earth. Planet. Sci.*, **17**, 197–226.

England, P., & McKenzie, D.P. 1982. A thin viscous sheet model for continental deformation. *Geophys. J. R. Astron. Soc.*, **70**, 295–321.

England, P., & McKenzie, D.P. 1983. Correction to: A thin viscous sheet model for continental deformation. *Geophys. J. R. Astron. Soc.*, **73**, 523–532.

Fernandez, J., Yu, T.-T., & Rundle, J.B. 1996. Horizontal viscoelastic - gravitational displacement due to a rectangular dipping thrust fault in a layered Earth model. *J. Geophys. Res.*, **101**(B6), 13,581–13,594.

Foulger, G., Hahn, C.-H., Seeber, G., Einarsson, P., Julian, B.R., & Heki, K. 1992. Post-rifting stress relaxation at the divergent plate boundary in Northeast Iceland. *Nature*, **358**, 488–490.

Frank, F.C. 1966. Deduction of earth strains from survey data. *Bull. Seismol. Soc. Am.*, **56**(1), 35–42.

Fung, Y.-C. 1965. *Foundations of solid mechanics*. First edn. New Jersey: Prentice-Hall.

Galanis, I. 1993. *Geodetic determination of displacements in the region of the Alkyonides Islands (1969 – 1981 – 1991).* Diploma thesis, Department of Surveying, National Technical University of Athens. (In Greek).

Gans, P.B. 1987. An open-system, two-layer crustal stretching model for the eastern Great Basin. *Tectonics*, **6**(1), 1–12.

Goetze, C. 1978. The mechanisms of creep in olivine. *Phil. Trans. R. Soc. Lond. A*, **288**, 99–119.

Govers, R., & Wortel, M.J.R. 1995. Extension of stable continental lithosphere and the initiation of lithospheric scale faults. *Tectonics*, **14**(4), 1,041–1,055.

Gurtner, W., & Mader, G. 1989. Receiver independent exchange format version 2. *GPS Bulletin of the CSTG*, **3**(3).

Gurtner, W., Mader, G., & Arthur, D. 1989. A common exchange format for GPS data. *GPS Bulletin of the CSTG*, **2**(3).

Hager, B.H., King, R.W., & Murray, M.H. 1991. Measurement of crustal deformation using the Global Positioning System. *Ann. Rev. Earth. Planet. Sci.*, **19**, 351–382.

Haines, A.J., & Holt, W.E. 1993. A procedure for obtaining the complete horizontal motions within zones of distributed deformation from the inversion of strain rate data. *J. Geophys. Res.*, **98**(B7), 12,057–12,082.

Hatzfeld, D., Nord, J., Paul, A., Guiguet, R., Briole, P., Ruegg, J.-C., Cattin, R., Meyer, B., Hubert, A., Bernard, P., Makropoulos, K., Karakostas, V., Papaioannou, C., Papanastassiou, D., & Veis, G. 1995. The Kozani–Grevena (Greece) earthquake of May 13, 1995, $M_s = 6.6$. Preliminary results of a field multidisciplinary survey. *Seis. res. lett.*, **66**, 61–70.

Hatzfeld, D., Karakostas, V., Ziazia, M., Selvaggi, G., Diagourtas, D., Kassaras, I., Makropoulos, K., Guiguet, R., Paul, A., Papaioannou, C., Azzara, R., Bona, M. Di, Bacceschi, S., & Bernard, P. 1996. The Kozani–Grevena earthquake of May 13, 1995. A seismological study. *Bull. Seismol. Soc. Am.*. (In press; preprint).

Hayford, J.F., & Baldwin, A.L. 1907. *The earth movements in the California earthquake of 1906.* U.S. Coast Geod. Surv. Rept., Appendix 3, 69–104. U.S. Coast & Geodetic Survey.

Heflin, M.B., Bertiger, W.I., Blewitt, G., Freedman, A.P., Hurst, K.J., Lichten, S.M., Lindqwister, U.J., Vigue, Y., Webb, F.H., Yunck, T.P., & Zumberge, J.F. 1992. Global geodesy using GPS without fiducial sites. *Geophys. Res. Lett.*, **19**, 131–134.

Hofmann-Wellenhof, B., Lichtenegger, H., & Collins, J. 1992. *Global Positioning System theory and practice.* Third edn. Wien: Springer-Verlag.

Holt, W.E., Ni, J.F., Wallace, T.C., & Haines, A.J. 1991. The active tectonics of the Eastern Himalayan Syntaxis and surrounding regions. *J. Geophys. Res.*, **96**(B9), 14,595–14,632.

Hopfield, H.S. 1969. Two-quartic tropospheric refractivity profile for correcting satellite data. *J. Geophys. Res.*, **74**, 4,487–4,499.

Hsű, K. 1982. *Mountain building processes.* London: Academic Press.

Jackson, J.A. 1994. Active tectonics of the Aegean region. *Ann. Rev. Earth. Planet. Sci.*, **22**, 239–271.

Jackson, J.A., & McKenzie, D.P. 1988a. Rates of active deformation in the Aegean Sea and surrounding regions. *Basin Research*, **1**, 121–128.

Jackson, J.A., & McKenzie, D.P. 1988b. The relationship between plate motions and seismic moment tensors and the rate of active deformation in the Mediterranean and Middle East. *Geophys. J. R. Astron. Soc.*, **93**, 45–73.

Jackson, J.A., & White, N.J. 1989. Normal faulting in the upper continental crust: Observations from regions of active extension. *J. Struct. Geol.*, **11**(1/2), 15–36.

Jackson, J.A., King, G., & Vita-Finzi, C. 1982a. The neotectonics of the Aegean: an alternative view. *Earth Planet. Sci. Lett.*, **61**, 303–318.

Jackson, J.A., Gagnepain, J., Houseman, G., King, G.C.P., Papadimitriou, P., Soufleris, C., & Virieux, J. 1982b. Seismicity, normal faulting and the geomorphological development of the Gulf of Corinth (Greece): the Corinth earthquakes of February and March 1981. *Earth Planet. Sci. Lett.*, **57**, 377–397.

Jackson, J.A., Haines, J., & Holt, W. 1992. The horizontal velocity field in the deforming Aegean Sea region determined from the moment tensors of earthquakes. *J. Geophys. Res.*, **97**(B12), 17,657–17,684.

Jackson, J.A., Haines, J., & Holt, W. 1994. A comparison of satellite laser ranging and seismicity data in the Aegean region. *Geophys. Res. Lett.*, **21**(25), 2,849–2,852.

Jaeger, J.C., & Cook, N.G.W. 1976. *Fundamentals of rock mechanics*. Second edn. London: Chapman and Hall.

Jeffreys, H. 1970. *The Earth*. Fifth edn. London: Cambridge University Press.

Johnson, H.O., & Wyatt, F.K. 1994. Geodetic network design for fault-mechanics studies. *Man. geod.*, **19**, 309–323.

Kagiadakis, V. 1995. *Geodetic determination of tectonic displacements in the region of the Alkyonides Islands and the Argolikos Gulf*. Diploma thesis, Department of Surveying, National Technical University of Athens. (In Greek).

Kahle, H.-G., Muller, M.V., Geiger, A., Danuser, G., Mueller, S., Veis, G., Billiris, H., & Paradissis, D. 1995. The strain field in northwestern Greece and the Ionian Islands: Results inferred from GPS measurements. *Tectonophys.*, **249**, 41–52.

Kahle, H.-G., Muller, M.V., & Veis, G. 1996. Trajectories of crustal deformation of Western Greece from GPS observations 1989–1994. *Geophys. Res. Lett.*, **23**(6), 677–680.

Kaplan, G.H. 1981. *The IAU resolutions of astronomical constants, time scales and the fundamental reference frame*. US Naval Observatory Circular, no. 163. Washington: US Naval Observatory.

Kastens, K.A., Balis, V., Billiris, H., Chayes, D., Elsner, C., Friedhoff, H., Habrich, H., Hurst, K., Koczynski, T., Milas, P., Mueller, A., Mueller, W., Papafitsorou, A., Paradissis, D., Raymond, C., Riecken, J., Sorge, B., Stephens, B., Stowell, J., Tsolakis, D., Tziotzis, J., Veis, G., & Vlachos, C. 1989. The Aegean GPS Project: 1988 results and 1989 plans. *Eos Trans. AGU*, **70**(15), 306.

King, G.C.P., Ouyang, Z.X., Papadimitriou, P., Deschamps, A., Gagnepain, J., Houseman, G., Jackson, J.A., Soufleris, C., & Virieux, J. 1985a. The evolution of the Gulf of Corinth, Greece: an aftershock study of the 1981 earthquakes. *Geophys. J. R. Astron. Soc.*, **80**, 677–693.

King, G.C.P., Stein, R.S., & Rundle, J.B. 1988. The growth of geological structures by repeated earthquakes 1: Conceptual framework. *J. Geophys. Res.*, **93**(B11), 13,307–13,318.

King, G.C.P., Stein, R.S., & Lin, J. 1994. Static stress changes and the triggering of earthquakes. *Bull. Seismol. Soc. Am.*, **84**(3), 935–953.

King, N.E., Segall, P., & Prescott, W. 1987. Geodetic measurements near Parkfield, California, 1959–1984. *J. Geophys. Res.*, **92**(B3), 2,747–2,766.

King, R.W., Masters, E.G., Rizos, C., Stolz, A., & Collins, J. 1985b. *Surveying with GPS.* Monograph, no. 9. School of Surveying, University of New South Wales, Australia.

Kissel, C., & Laj, C. 1988. The Tertiary geodynamical evolution of the Aegean arc: a palaeomagnetic reconstruction. *Tectonophys.*, **146**, 183–201.

Kissel, C., & Laj, C. 1989. *Palaeomagnetic rotations and continental deformation.* Dordrecht: Kluwer.

Kissel, C., Laj, C., & Müller, C. 1985. Tertiary geodynamical evolution of northwestern Greece: Palaeomagnetic results. *Earth Planet. Sci. Lett.*, **72**, 190–204.

Kostrov, B. 1974. Seismic moment and energy of earthquakes, and seismic flow of rock. *Izv. Acad. Sci. USSR Phys. Solid Earth*, **97**, 23–44.

Kusznir, N.J., & Matthews, D.H. 1988. Deep seismic reflections and the deformational mechanics of continental lithosphere. *J. Petrol. Special Lithosphere Issue*, 63–87.

Lamb, S.H. 1994. Behaviour of the brittle crust in wide plate boundary zones. *J. Geophys. Res.*, **99**(B3), 4,457–4,483.

Lambeck, K. 1988. *Geophysical geodesy.* Oxford: Oxford University Press.

Larsen, S., & Reilinger, R. 1992. Global Positioning System measurements of strain accumulation across the Imperial Valley, California: 1986–1989. *J. Geophys. Res.*, **97**(B6), 8,865–9,876.

Le Pichon, X. 1982. Land-locked oceanic basins and continental collision; the eastern Mediterranean as a case example. *Pages 129–146 of:* Hsü, K. (ed), *Mountain building processes.* London: Academic Press.

Le Pichon, X., & Angelier, J. 1979. The Hellenic arc and trench system: a key to the neotectonic evolution of the eastern Mediterranean area. *Tectonophys.*, **60**, 1–42.

Le Pichon, X., Chamot-Rooke, N., Lallemant, S., Noomen, R., & Veis, G. 1995. Geodetic determination of the kinematics of central Greece with respect to Europe: Implications for eastern Mediterranean tectonics. *J. Geophys. Res.*, **100**(B7), 12,765–12,690.

Leeder, M.R., & Jackson, J.A. 1993. The interaction between normal faulting and drainage in active extensional basins, with examples from the western United Sates and central Greece. *Basin. Res.*, **5**, 79–102.

Leick, A. 1990. *GPS satellite surveying.* John Wiley.

Lenk, O. 1995. Space based geodetic activities for geodynamics in Turkey. *Pages 12–29 of: Proc. of the 1st International Symposium on Deformations in Turkey.* Ankara: TMMOB–HKMO.

Lichten, S.M., & Border, J.S. 1987. Strategies for high-precision Global Positioning System orbit determination. *J. Geophys. Res.*, **92**(B12), 12,751–12,762.

Lisowski, M., Prescott, W.H., Savage, J.C., & Johnston, M.J. 1990. Geodetic estimate of coseismic slip during the 1989 Loma Prieta , California, earthquake. *Geophys. Res. Lett.*, **17**(9), 1,437–1,440.

Love, A.E.H. 1911. *Some problems of geodynamics.* Cambridge.

Lundgren, P.R., Kornreich Wolf, S., Protti, M., & Hurst, K.J. 1993. GPS measurements of crustal deformation associated with the 22 April 1991 Valle del la Estrella, Costa Rica earthquake. *Geophys. Res. Lett.*, **20**(5), 407–410.

Ma, X.Q., & Kusznir, N.J. 1995. Coseismic and postseismic subsurface displacements and strains for a dip-slip normal fault in a three-layer elastic-gravitational medium. *J. Geophys. Res.*, **100**(B7), 12,813–12,828.

Magistrale, H., & Zhou, H.-W. 1996. Lithologic control of the depth of earthquakes in southern California. *Science*, **273**, 639–642.

Makris, J. 1985. Geophysics and geodynamic implications from the evolution of the Hellenides. *Pages 231–248 of:* Stanley, D.J., & Wessel, F.-C. (eds), *Geological evolution of the Mediterranean Basin.* Springer-Verlag.

Makropoulos, K.C., & Burton, P.W. 1984. Greek tectonics and seismicity. *Tectonophys.*, **106**, 275–304.

Malvern, L.E. 1969. *Introduction to the mechanics of a continuous medium.* Prentice-Hall.

Marshall, G.A., Stein, R.S., & Thatcher, W. 1991. Faulting geometry and slip from co-seismic elevation changes: the 18 October 1989, Loma Prieta, California, earthquake. *Bull. Seismol. Soc. Am.*, **81**(5), 1,660–1,693.

Massonet, D., Rossi, M., Carmona, C., Adragna, F., Peltzer, G., Feigl, K., & Rabaute, T. 1993. The displacement field of the Landers earthquake mapped by radar interferometry. *Nature*, **364**, 138–142.

Massonnet, D., Thatcher, W., & Vadon, H. 1996. Detection of postseismic fault-zone collapse following the Landers earthquake. *Nature*, **382**, 612–616.

Matsu'ura, M., Tanimoto, T., & Iwasaki, T. 1981. Quasi-static displacements due to faulting in a layered half-space with an intervenient viscoelastic layer. *J. Phys. Earth*, **29**, 23–54.

Matsu'ura, M., Jackson, D.D., & Cheng, A. 1986. Dislocation model for aseismic crustal deformation at Hollister, California. *J. Geophys. Res.*, **91**(B12), 12,661–12,674.

McKenzie, D.P. 1972. Active tectonics of the Mediterranean region. *Geophys. J. R. Astron. Soc.*, **30**, 109–185.

McKenzie, D.P. 1978. Active tectonics of the Alpine-Himalayan belt: the Aegean Sea and surrounding regions. *Geophys. J. R. Astron. Soc.*, **55**, 217–254.

McKenzie, D.P., & Jackson, J.A. 1983. The relationship between strain rates, crustal thickening, palaeomagnetism, finite strain and fault movements within a deforming zone. *Earth Planet. Sci. Lett.*, **65**, 182–202.

McKenzie, D.P., & Jackson, J.A. 1989. The kinematics and dynamics of distributed deformation. *In:* Kissel, C., & Laj, C. (eds), *Paleomagnetic rotations and continental deformation.* Dordrecht: Kluwer.

McTigue, D.F., & Stein, R.S. 1984. Topographic amplification of tectonic displacement: Implications for geodetic measurement of strain changes. *J. Geophys. Res.*, **89**(B2), 1,123–1,131.

Menke, W. 1984. *Geophysical data analysis: Discrete inverse theory.* London: Academic Press.

Molnar, P. 1983. Average regional strain due to slip on numerous faults of different orientations. *J. Geophys. Res.*, **88**(B8), 6,430–6,432.

Molnar, P., & Chen, W.-P. 1982. Seismicity and mountain building. *Pages 41–57 of:* Hsű, K. (ed), *Mountain building processes.* London: Academic Press.

Mouyaris, N., Papastamatiou, D., & Vita-Finzi, C. 1992. The Helice fault? *Terra Nova*, **4**, 124–129.

Nelder, J.A., & Mead, R. 1965. A simplex method for function minimisation. *Computer Journal*, **7**, 308–313.

Noomen, R., Ambrosius, B.A.C., & Wakker, K.F. 1993. Crustal motions in the Mediterranean region determined from laser ranging to LAGEOS. *Pages 331–346 of:* Smith, D.E., & Turcotte, D.L. (eds), *Contributions of space geodesy to geodynamics: Crustal dynamics (AGU Geodynamics Series vol. 23).* Washington: AGU.

Nur, A., & Mavko, G. 1973. Postseismic viscoelastic rebound. *Science*, **183**, 204–206.

Okada, Y. 1985. Surface deformation due to shear and tensile faults in a half-space. *Bull. Seismol. Soc. Am.*, **75**(4), 1,135–1,154.

Oral, M.B., Reilinger, R.E., Toksöz, M.N., King, R.W., Barka, A.A., Kinik, I., & Lenk, O. 1994. Global Positioning System offers evidence of plate motions in Eastern Mediterranean. *Eos Trans. AGU*, **76**(2), 9–11.

Papadopoulos, G.A., Kondopoulou, D.P., Leventakis, G.-A., & Pavlides, S.B. 1986. Seismotectonics of the Aegean region. *Tectonophys.*, **124**, 67–84.

Papazachos, B.C. 1990. Seismicity of the Aegean and surrounding area. *Tectonophys.*, **178**, 287–308.

Papazachos, B.C., & Papazachos, C.B. 1989. *The earthquakes of Greece.* Thessaloniki: Ziti Publications. (In Greek).

Papazachos, B.C., Panagiotopoulos, D.G., Scordilis, E.M., Karakaisis, G.F., Papaioannou, Ch.A., Karacostas, B.G., Papdimitriou, E.E., Kiratzi, A.A., Hatzidimitriou, P.M., Leventakis, G.N., Voidomatis, Ph.S., Peftitselis, K.I., & Tsapanos, T.M. 1996. *Focal properties of the 13 May 1995 large ($M_s = 6.6$) earthquake in the Kozani area (north Greece).* Geophysical Laboratory Publication no. 4. Aristotle University of Thessaloniki.

Pavlides, S.B., Zouros, N.C., Chatzipetros, A.A., Kostopoulos, D.S., & Mountrakis, D.M. 1995. The 13 May 1995 western Macedonia, Greece (Kozani Grevena) earthquake; preliminary results. *Terra Nova*, **7**(5), 544–549.

Peltzer, G., Rosen, P., Rogez, F., & Hudnut, K. 1996. Postseismic rebound in fault step-overs caused by pore fluid flow. *Science*, **273**, 1,202–1,204.

Pollitz, F.F. 1992. Postseismic relaxation theory on the spherical Earth. *Bull. Seismol. Soc. Am.*, **82**, 422–453.

Prescott, W.H. 1976. An extension of Frank's method for obtaining crustal shear strains from survey data. *Bull. Seismol. Soc. Am.*, **66**(6), 1,847–1,853.

Prescott, W.H., Savage, J.C., & Kinoshita, W.T. 1979. Strain accumulation rates in the Western United States between 1970 and 1978. *J. Geophys. Res.*, **84**(B10), 5,423–5,435.

Press, F. 1965. Displacements, strains and tilts at teleseismic distances. *J. Geophys. Res.*, **70**(10), 2,395–2,411.

Press, W.H., Teukolsky, S.A., Vetterling, W.T., & Flannery, B.P. 1992. *Numerical Recipes in C: The art of scientific computing.* Second edn. Cambridge: Cambridge University Press.

Proffett, J.M. 1977. Cenozoic geology of the Yerington district, Nevada, and implications for the nature and origin of Basin and Range faulting. *Geol. Soc. Am. Bull.*, **88**, 247–266.

Reilinger, R. 1986. Evidence for postseismic viscoelastic relaxation following the 1959 $M = 7.5$ Hebgen Lake, Montana, earthquake. *J. Geophys. Res.*, **91**(B9), 9,488–9,494.

Rietbrock, A., Tiberi, C., Scherbaum, F., & Lyon-Caen, H. 1996. Seismic slip on a low angle normal fault in the Gulf of Corinth: Evidence from high-resolution cluster analysis of microearthquakes. *Geophys. Res. Lett.*, **23**(14), 1,817–1,820.

Rigo, A., Lyon-Caen, H., Armijo, R., Deschamps, A., Hatzfeld, D., Makropoulos, K., Papadimitriou, P., & Kassaras, I. 1996. A microseismic study in the Gulf of Corinth (Greece): Implications for large-scale normal faulting mechanisms. *Geophys. J. Int.*, **126**, 663–688.

Robbins, J.W., Smith, D.E., & Ma, C. 1993. Horizontal crustal deformation and large-scale plate motions inferred from space geodetic techniques. *Pages 21–36 of:* Smith, D.E., & Turcotte, D.L. (eds), *Contributions of space geodesy to geodynamics: Crustal dynamics (AGU Geodynamics Series vol. 23).* Washington: AGU.

Roberts, S., & Jackson, J.A. 1991. Active normal faulting in central Greece; an overview. *Pages 125–142 of:* Roberts, A.M., Yielding, G., & Freeman, B. (eds), *The geometry of normal faults, Geol. Soc. Spec. Publ. 56.* London: Geological Society of London.

Rothacher, M., Beutler, G., Gurtner, W., Brockmann, E., & Mervart, L. 1993. *Bernese GPS Software version 3.4 documentation.*

Rundle, J.B. 1978. Viscoelastic crustal deformation by finite quasi-static sources. *J. Geophys. Res.*, **83**(B12), 5,937–5,945.

Rundle, J.B. 1980. Static elastic-gravitational deformation of a layered half space by point couple sources. *J. Geophys. Res.*, **85**(B10), 5,355–5,363.

Rundle, J.B. 1981. Vertical displacements from a rectangular fault in layered elastic-gravitational media. *J. Phys. Earth*, **29**, 173–186.

Rundle, J.B. 1982. Viscoelastic-gravitational deformation by a rectangular thrust fault in a layered Earth. *J. Geophys. Res.*, **87**(B9), 7,787–7,796.

Saastamoinen, J. 1972. Atmospheric correction for the troposphere and stratosphere in radio ranging of satellites. *Pages 247–251 of:* Henriksen, S.W., Mancini, A., & Chovitz, B.H. (eds), *The use of artificial satellites for geodesy (AGU Geophysical Monographs Series, vol. 15).* Washington: AGU.

Savage, J.C. 1983. A dislocation model of strain accumulation and release at a subduction zone. *J. Geophys. Res.*, **88**(B6), 4,984–4,996.

Savage, J.C., & Gu, G. 1985. A plate flexure approximation to postseismic and interseismic deformation. *J. Geophys. Res.*, **90**(B10), 8,570–8,580.

Savage, J.C., & Hastie, L.M. 1966. Surface deformation associated with dip-slip faulting. *J. Geophys. Res.*, **71**, 4,897–4,904.

Savage, J.C., & Hastie, L.M. 1969. A dislocation model for the Fairview Peak, Nevada, earthquake. *Bull. Seismol. Soc. Am.*, **59**, 1,937–1,948.

Savage, J.C., Lisowski, M., & Svarc, J.L. 1994. Postseismic deformation following the 1989 ($M = 7.1$) Loma Prieta, California, earthquake. *J. Geophys. Res.*, **99**(B7), 13,757–13,765.

Scholz, C.H. 1988. The brittle-plastic transition and the depth of seismic faulting. *Geol. Rundschau*, **77**(1), 319–328.

Segall, P., & Harris, R. 1987. Earthquake deformation cycle on the San Andreas Fault near Parkfield, California. *J. Geophys. Res.*, **92**(B10), 10,511–10,525.

Segall, P., & Matthews, M.V. 1988. Displacement calculations from geodetic data and the testing of geophysical deformation models. *J. Geophys. Res.*, **93**(B12), 14,954–14,966.

Singh, S.J. 1970. Static deformation of a multilayered half-space by internal sources. *J. Geophys. Res.*, **75**(17), 3,257–3,263.

Smith, D.E., Kolenkiewicz, R., Dunn, P.J., Robbins, J.W., Torrence, M.H., Klosko, S.M., Williamson, R.G., Pavlis, E.C., Douglas, N.B., & Fricke, S.K. 1990. Tectonic motion and deformation from Satellite Laser Ranging to LAGEOS. *J. Geophys. Res.*, **95**(B13), 22,013–22,041.

Smith, D.E., Kolenkiewicz, R., Robbins, J.W., Dunn, P.J., & Torrence, M.H. 1994. Horizontal crustal motion in the central and eastern Mediterranean inferred from Satellite Laser Ranging measurements. *Geophys. Res. Lett.*, **21**(18), 1,979–1,982.

Snay, R.A., Neugebauer, H.C., & Prescott, W.H. 1991. Horizontal deformation associated with the Loma Prieta earthquake. *Bull. Seismol. Soc. Am.*, **81**(5), 1,647–1,659.

Sokoutis, D., Brun, J.P., Dreissche, J. Van Den, & Pavlides, S. 1993. A major Oligo-Miocene detachment in southern Rhodope controlling north Aegean extension. *J. Geol. Soc. London*, **150**, 243–246.

Spakman, W., Wortel, M.J.R., & Vlaar, N.J. 1988. The Hellenic subduction zone: a tomographic image and its geodynamic implications. *Geophys. Res. Lett.*, **15**, 60–63.

Stein, R.S., & Barrientos, S.E. 1985. High angle normal faulting in the intermountain seismic belt: Geodetic investigation of the 1983 Borah Peak, Idaho, earthquake. *J. Geophys. Res.*, **90**(B13), 11,355–11,366.

Stein, R.S., & Lisowski, M. 1983. The 1979 Homestead Valley earthquake sequence, California: Control of aftershocks and postseismic deformation. *J. Geophys. Res.*, **88**(B8), 6,477–6,490.

Steketee, J.A. 1958. On Volterra's dislocations in a semi-infinite elastic medium. *Can. J. Phys.*, **36**, 192–205.

Stewart, I., & Vita-Finzi, C. 1996. Coastal uplift on active normal faults: The Eliki Fault, Greece. *Geophys. Res. Lett.*, **23**(14), 1,853–1,856.

Stiros, S.C. 1993. Kinematics and deformation of central and southwestern Greece from historical triangulation data and implications for the active tectonics of the Aegean. *Tectonophys.*, **220**, 283–300.

Tarantola, A., & Valette, B. 1982. Generalised nonlinear inverse problems solved using the least squares criterion. *Rev. geophys. and space physics*, **20**(2), 219–232.

Taymaz, T., Jackson, J., & McKenzie, D. 1991. Active tectonics of the north and central Aegean Sea. *Geophys. J. Int.*, **106**, 433–490.

Thatcher, W. 1995. Microplate versus continuum descriptions of active tectonic deformation. *J. Geophys. Res.*, **100**(B3), 3,885–3,894.

Thatcher, W., & Rundle, J.B. 1979. A model for the earthquake cycle in underthrust zones. *J. Geophys. Res.*, **84**, 5,540–5,556.

Thatcher, W., & Rundle, J.B. 1984. A viscoelastic coupling model for the cyclic deformation due to periodically repeated earthquakes at subduction zones. *J. Geophys. Res.*, **89**(B9), 7,631–7,640.

Vita-Finzi, C., & King, G.C.P. 1985. The seismicity, geomorphology and structural evolution of the Corinth area of Greece. *Phil. Trans. R. Soc. Lond. A*, **314**, 379–407.

Volterra, Par M. Vito. 1907. L'equilibre des corps elastiques. *Ann. Ec. Norm.*, **3**(XXXIV), 401–517.

Walcott, R.I. 1984. The kinematics of the plate boundary zone through New Zealand: a comparison of short- and long-term deformations. *Geophys. J. R. Astron. Soc.*, **79**, 613–633.

Walsh, J.J., & Watterson, J. 1988. Analysis of the relationship between displacements and dimensions of faults. *J. Struct. Geol.*, **10**(3), 239–247.

Ward, S.N., & Barrientos, S.E. 1986. An inversion for slip distribution and fault shape from geodetic observations of the 1983, Borah Peak, Idaho, earthquake. *J. Geophys. Res.*, **91**(B5), 4,909–4,919.

Wells, D. 1986. *Guide to GPS positioning.* Fredericton: Canadian GPS Associates.

Welsch, W.M. 1983. Finite element analysis of strain patterns from geodetic observations across a plate margin. *Tectonophys.*, **97**, 57–71.

Wernicke, B. 1981. Low-angle normal faults in the Basin and Range Province: Nappe tectonics in an extending orogen. *Nature*, **291**, 645–648.

Wessel, P., & Smith, W.H.F. 1991. Free software helps map and display data. *Eos Trans. AGU*, **72**, 441–446.

Wessel, P., & Smith, W.H.F. 1995. New version of the Generic Mapping Tools released. *Eos Trans. AGU*, **76**, 329.

Westaway, R. 1991. Continental extension on sets of parallel faults: Observational evidence and theoretical models. *Pages 143–169 of:* Roberts, A.M., G. Yielding, & Freeman, B. (eds), *The geometry of normal faults, Geol. Soc. Spec. Publ. 56.* London: Geological Society of London.

Williams, C.R., Arnadottir, T., & Segall, P. 1993. Coseismic deformation and dislocation models of the 1989 Loma Prieta earthquake derived from Global Positioning System measurements. *J. Geophys. Res.*, **98**(B3), 753–770.

Yu, T.-T., Rundle, J.B., & Fernandez, J. 1996. Surface deformation due to a strike-slip fault in an elastic gravitational layer overlying a viscoelastic gravitational half-space. *J. Geophys. Res.*, **101**(B2), 3,199–3,214.

Appendix A

Central Greece Network occupations

The purpose of this Appendix is to tabulate unwieldy data appertaining to the three full Central Greece Network occupations (June 1989, October 1991 and May 1993), and the three part-occupations of Central Greece Network sites around the Gulf of Korinthos (June 1995, October 1995 and May 1996).

A.1 Epoch 1989.44 (June 1989)

Table A.1 gives the observation schedule and antenna heights, and Table A.2 gives offsets from the ITRF fiducial site coordinates to those sites occupied in 1989, where necessary.

Campaign coordinates are given in Table A.3.

A.2 Epoch 1991.78 (October 1991)

Table A.4 gives the observation schedule and antenna heights, and Table A.5 gives offsets from the ITRF fiducial site coordinates to those sites occupied in 1991, where necessary.

Campaign coordinates are given in Table A.6.

A.3 Epoch 1993.39 (May 1993)

Table A.7 gives the observation schedule and antenna heights, and Table A.8 gives offsets from the ITRF fiducial site coordinates to those sites occupied in 1993, where necessary.

Date (6/89)	5	6	7	8	9	10	11	12	13	14	15	16	17
GPS Week			491							492			
DOY	156	157	158	159	160	161	162	163	164	165	166	167	168
1	16	16	45	57	-	40	39	14	13	13	28	28	05
T4000SLD	1.134	1.171	1.295	1.544		1.197	1.045	0.912	0.129	0.128	1.302	1.266	1.447
2	59	59	58	52	40	39	-	27	09	21	14	43	-
WM102	0.164	0.164	1.379	1.354	1.145	1.239		1.037	1.364	0.129	0.825	1.120	
3	44	44	66	66	54	54	54	54	54	54	54	54	54
WM102	0.129	0.129	1.417	1.357	1.280	1.280	1.280	1.280	1.281	1.318	1.318	1.318	1.318
4	62	62	48	48	38	38	04	04	18	18	30	31	31
WM102	1.085	1.144	0.126	0.127	1.083	1.150	1.238	1.240	1.330	1.226	1.239	1.096	0.941
5	30	65	65	56	56	-	12	12	rcvr	06	06	26	26
WM102	0.712	1.033	1.001	1.058	1.280		1.180	0.924	fail	1.101	1.119	1.129	1.184
6	54	54	54	54	51	-	11	11	02	02	29	29	34
T4000SLD	1.357	1.357	1.357	1.357	1.180		0.123	0.123	1.371	1.259	1.272	1.248	1.456
7	rcvr	45	57	58	52	37	37	19	19	23	23	41	-
T4000SL	fail	1.137	1.139	1.170	1.122	1.285	1.153	1.234	1.116	1.076	1.110	1.266	
8	32	32	49	49	35	35	rcvr	rcvr	22	22	34	34	-
WM101	0.952	0.740	0.132	0.132	1.034	0.883	fail	fail	0.125	0.126	1.310	1.405	
9	42	46	46	55	55	27	09	01	01	20	20	60	61
T4000SX	1.337	1.112	1.223	1.363	1.473	1.285	1.126	1.287	1.131	1.193	1.354	0.983	0.890
10	43	47	47	50	41	17	17	03	03	24	24	36	-
T4000SL	1.422	1.485	1.586	1.354	1.292	1.311	1.278	1.506	1.476	1.345	1.340	1.139	
11	60	63	64	51	50	10	10	07	08	25	25	42	-
WM101	1.279	1.212	1.055	1.063	1.096	1.199	1.114	1.273	1.265	1.181	1.114	1.051	
12	61	64	63	53	53	36	15	08	07	15	33	33	-
T4000SL	0.866	1.138	1.331	1.177	1.340	1.293	1.390	1.257	1.257	1.288	1.073	1.150	
Hers							Hers						
T4000SLD							0.904						
Onsa							Onsa						
TI4100							1.434						
Wett							Wett						
TI4100							0.000						

T4000SL = Trimble 4000SL (L1 only); T4000SLD = Trimble 4000SLD (L1, L2[2]); T4000SX = Trimble 4000SX (L1 only);

WM101 = Leica WM101 (L1 only); WM102 = Leica WM102 (L1, L2).

Table A.1: Receiver types, site occupations and antenna heights in metres (to bottom of pre-amp for Trimble and TI receivers, bottom of antenna for Leica receivers) for June 1989 survey.

Site	ΔX/m	ΔY/m	ΔZ/m	Note
Dion (54)	+1.893	+2.152	–3.314	Point C
Hers	–11.158	–46.352	+1.997	Solar pillar
Wett	–32.644	–26.926	+29.304	

Table A.2: Fiducial site eccentricities, June 1989. Vectors are from the ITRF marker (Table 2.3) to the campaign GPS marker.

Site	X/m	Y/m	Z/m	Note
CG02	4554541.2273	1862482.3669	4044666.4537	
CG04	4545704.6360	1920346.0555	4027956.2927	
CG05	4539618.8943	1944401.4699	4023759.3833	
CG06	4588151.4187	1849820.4498	4012934.8947	
CG09	4565606.6067	1920347.3156	4005106.2465	
CG11	4536964.9513	1988571.7574	4005402.0005	
CG12	4524262.7742	2006260.5306	4010418.7128	
CG13	4615299.9739	1845836.1314	3984776.8735	
CG14	4592868.1368	1897227.6328	3985190.0586	
CG16	4745949.4754	1905705.9542	3799169.0388	
CG18	4596042.5597	1733476.6806	4055720.8053	
CG21	4591931.4077	1930877.7121	3971480.5523	
CG26	4571271.3915	1972601.6892	3974038.4709	
CG27	4563037.4527	1980600.6299	3978750.1266	
CG28	4589879.4913	1967399.7172	3955033.8464	
CG29	4606396.4889	1935360.4765	3951352.0455	
CG30	4636745.4403	1886703.0457	3940596.4718	
CG31	4634559.0295	1899331.7911	3935867.8918	
CG34	4597929.4538	1972797.3480	3942758.8112	
CG38	4552114.8351	2037147.3915	3963188.4086	
CG39	4522199.8799	2065010.9171	3982702.2417	
CG40	4525889.1335	2070725.8600	3975606.9170	
CG43	4652055.7875	1897724.6903	3917372.1055	
CG44	4644472.9117	1937384.8131	3906731.2130	
CG45	4620984.3857	1948528.3695	3928588.6013	
CG48	4627578.6043	1976842.4875	3908073.2972	
CG51	4599894.4677	2004286.6086	3925361.1217	
CG52	4574393.6020	2033129.0050	3939302.4416	
54-C	4595218.3172	2039437.4819	3912626.1896	Fixed
CG56	4578277.9918	2075992.2237	3913287.2309	
CG57	4613032.1205	2000514.9558	3911362.0015	
CG58	4610054.5444	2015493.5263	3907135.6518	
CG59	4700881.0201	1817837.1155	3896017.6066	
CG62	4661660.7160	1924282.2900	3892694.0539	
CG65	4641801.2614	1979193.1216	3888527.4808	
CG66	4610660.6983	2047374.8541	3889970.5498	

Table A.3: Site coordinates, June 1989 (L3 solution).

Date (10/91)	5	6	7	8	9	10	11	13	14	15	16	17	18	19
GPS Week	612	613						614						
DOY	278	279	280	281	282	283	284	286	287	288	289	290	291	292
1	59	59	32	32	50	50	41	41	15	15	02	02	13	13
LM-XII	0.120	0.120	1.116	1.088	1.254	1.200	1.273	1.499	1.416	1.518	1.331	1.295	0.117	0.116
2	61	62	62	48	48	28	28	11	11	14	14	23	23	61
WM102	0.868	1.171	1.095	0.127	0.125	1.126	1.116	0.123	0.123	0.645	0.640	0.712	0.709	0.893
3	60	60	63	63	66	66	33	33	10	10	01	01	19	19
WM102	1.324	1.338	1.270	1.310	1.009	1.251	1.063	1.188	1.072	1.038	1.195	1.220	1.203	1.177
4	42	45	45	35	35	56	56	39	39	17	17	36	36	30
WM102	1.037	1.158	1.098	1.173	1.206	1.196	1.212	1.028	1.217	1.264	1.215	1.266	1.379	1.235
5	43	43	64	64	57	57	29	29	05	05	06	06	27	27
WM102	1.074	1.055	0.874	0.912	1.226	1.368	1.079	0.992	1.339	1.300	1.284	1.298	1.254	1.315
6	44	65	65	58	58	34	34	12	12	09	09	22	22	44
LM-XII	0.123	1.080	0.953	1.066	1.067	1.341	1.380	0.990	0.875	1.049	1.076	0.131	0.130	0.125
7	24	24	47	47	53	53	21	21	04	04	03	03	08	08
LM-XII	1.255	1.185	1.339	1.135	1.052	1.108	0.119	0.121	1.044	0.991	0.960	0.994	1.030	1.033
8	30	46	46	51	51	52	52	40	40	26	26	20	20	42
LM-XII	1.167	0.957	1.020	0.966	0.960	1.130	0.960	1.204	1.215	1.112	1.064	1.049	1.131	0.922
9	31	31	49	49	55	55	38	38	37	37	07	07	25	25
LM-XII	1.171	0.997	0.120	0.121	1.199	1.271	1.001	0.997	1.352	1.320	1.256	1.038	1.093	0.882
10	54	54	54	54	54	54	54	54	54	54	54	54	54	54
T4000SST	1.413	1.415	1.415	1.415	1.415	1.415	1.415	1.414	1.414	1.404	1.404	1.404	1.404	1.398
11	16	16	16	16	16	16	16	18	18	18	18	18	18	18
T4000SST	1.562	1.519	1.520	1.549	1.535	1.543	1.535	1.560	1.519	1.512	1.535	1.460	1.572	1.574
HERS	HERS													
SNR-8A	0.200													
MADR	MADR													
SNR-8	0.000													
ONSA	ONSA													
SNR-800	0.995													
WETT	WETT													
SNR-800	0.000													

LM-XII = Ashtech LM-XII (L1, L2[2]); SNR-8, SNR-8A, SNR-800 = Rogue series (L1, L2);
T4000SST = Trimble 4000SST (L1, L2); WM102 = Leica WM102 (L1, L2).

Table A.4: Receiver types, site occupations and antenna heights in metres (to bottom of pre-amp for Trimble receivers, bottom of choke-ring for Rogue receivers, bottom of antenna for Leica and Ashtech receivers) for October 1991 survey.

Site	ΔX/m	ΔY/m	ΔZ/m	Note
DION (54)	+1.893	+2.152	−3.314	Point C

Table A.5: Fiducial site eccentricities, October 1991. Vectors are from the ITRF marker (Table 2.3) to the campaign GPS marker.

Site	X/m	Y/m	Z/m	Note
CG01	4549888.5012	1895762.7066	4034573.8712	
CG02	4554541.1106	1862482.4531	4044666.4483	
CG03	4563198.0294	1874447.0126	4029837.4070	
CG04	4545704.5982	1920346.1170	4027956.3134	
CG05	4539618.8019	1944401.5754	4023759.3121	
CG06	4588151.4055	1849820.5387	4012934.9234	
CG07	4564567.1069	1903044.7975	4014921.5250	
CG08	4588623.8907	1876481.3722	4000362.0754	
CG09	4565606.4304	1920347.3478	4005106.1956	
CG10	4550930.0601	1955644.8936	4005111.0514	
CG11	4536964.9187	1988571.7974	4005401.9614	
CG12	4524262.7672	2006260.5535	4010418.6770	
CG13	4615299.8651	1845836.1587	3984776.8137	
CG14	4592868.1432	1897227.7085	3985190.1022	
CG15	4577903.7810	1938011.9052	3982736.8882	
CG16	4745949.5648	1905705.9402	3799169.0196	
CG17	4556996.8072	1969382.9407	3991140.1682	
CG18	4596042.5821	1733476.7840	4055720.9244	
CG19	4601771.4362	1901220.0913	3973719.5007	
CG20	4604106.4843	1918649.6197	3962296.9770	
CG21	4591931.3194	1930877.7270	3971480.4710	
CG22	4612635.2577	1900869.1213	3961679.9455	
CG23	4610905.3588	1909825.5576	3959468.2446	
CG24	4597863.5975	1938508.9826	3959991.1911	
CG25	4595359.0230	1950490.3807	3958135.9602	
CG26	4571271.3234	1972601.7398	3974038.4039	
CG27	4563037.4050	1980600.6764	3978750.1107	
CG28	4589879.4455	1967399.7395	3955033.7657	
CG29	4606396.3465	1935360.5051	3951351.9585	
CG30	4636745.4496	1886703.0775	3940596.4938	
CG31	4634559.0387	1899331.7579	3935867.8842	
CG32	4621884.2661	1921702.0957	3940613.9346	
CG33	4610013.6830	1944405.5905	3943029.0322	
CG34	4597929.3656	1972797.3821	3942758.7297	
CG35	4585749.3712	1997779.2546	3944379.2734	
CG36	4571812.4658	1996184.8089	3961739.3394	
CG37	4567938.0698	2006907.3593	3960889.9072	
CG38	4552114.8345	2037147.4102	3963188.3609	
CG39	4522199.9422	2065010.9842	3982702.2410	
CG40	4525889.1452	2070725.8785	3975606.8418	
CG41	4579631.9858	2014445.3208	3943140.9458	
CG42	4652969.9520	1877352.3581	3925788.0691	
CG43	4652055.6439	1897724.6995	3917371.9767	
CG44	4644472.9330	1937384.8159	3906731.1886	
CG45	4620984.3928	1948528.4087	3928588.5750	
CG46	4615566.4915	1961954.8564	3927646.1806	
CG47	4633203.1483	1961421.8430	3907452.5994	
CG48	4627578.6022	1976842.5266	3908073.3049	
CG49	4619101.9757	1980972.4855	3914066.1487	
CG50	4607401.1635	1989498.6827	3924138.8882	
CG51	4599894.4191	2004286.6106	3925361.0335	
CG52	4574393.7863	2033128.9969	3939302.3949	
CG53	4588486.6623	2028895.7576	3925877.6555	
54-C	4595218.3256	2039437.4981	3912626.1629	Fixed
CG55	4569149.7702	2052266.5186	3935686.8163	
CG56	4578278.0600	2075992.2504	3913287.2276	
CG57	4613032.0451	2000514.9483	3911361.9193	
CG58	4610054.5222	2015493.5364	3907135.6270	
CG59	4700881.0487	1817837.1289	3896017.5686	
CG60	4663885.9926	1879924.8754	3911789.9962	
CG61	4679033.3863	1850628.0834	3906821.6881	
CG62	4661660.7376	1924282.3005	3892694.0704	
CG63	4654493.0912	1963758.4171	3881625.0681	
CG64	4647892.5724	1967179.4624	3887940.7126	
CG65	4641801.2655	1979193.1439	3888527.4612	
CG66	4610660.7366	2047374.8754	3889970.5530	

Table A.6: Site coordinates, October 1991.

Campaign coordinates are given in Table A.9.

A.4 Epoch 1995.44 (June 1995)

Occupation times and antenna heights (to the L1 phase centre) for the campaign are shown in Figures A.1 – A.4. Receivers OX1, OX2 and OX3 are Ashtech LM-XII; 0105, 0111, 0256 and 0380 are Ashtech Z-XII; DION is a Trimble 4000SSE.

Table A.10 gives offsets from the ITRF fiducial site coordinates to the Dionysos permanent site occupied from 1995.

Campaign coordinates are given in Table A.14.

A.5 Epoch 1995.76 (October 1995)

Occupation times and antenna heights (to L1 phase centre) for the June 1995 Central Greece campaign are depicted in Figures A.5 – A.10. All receivers are Ashtech LM-XII, except AST_1, AST_2, OMP_16, INSU_13, INSU_14, INSU_16, 0340, 0616, 0673, CT00, T000 (Ashtech Z-XII); CT01, TRIM_1, TRIM_2 (Trimble 4000SST); DION (Trimble 4000SSE).

Table A.10 gives offsets from the ITRF fiducial site coordinates to the Dionysos permanent site occupied from 1995.

Campaign coordinates are given in Table A.12.

Date (5/93)	17	18	19	20	21	24	25	26	27	28
GPS Week			697					698		
DOY	137	138	139	140	141	144	145	146	147	148
1	11	11	-	12	12	22	48	-	57	57
LM-XII2	0.119	0.119		0.690	0.829	0.126	0.120		1.046	1.036
2	40	40	40	39	39	19	51	51	56	56
LM-XII2	0.990	1.124	0.838	0.924	0.889	1.205	0.824	0.945	1.288	1.287
3	10	10	02	02	30	30	34	34	52	52
LM-XII2	1.214	1.232	1.374	1.362	1.450	1.569	1.300	1.264	1.168	1.181
4	17	17	15	15	27	27	26	26	37	37
LM-XII2	1.317	1.281	1.375	1.415	1.317	1.237	1.047	1.033	1.269	1.099
5	36	36	03	03	25	25	41	41	38	38
SR299	1.089	1.147	1.159	1.147	1.031	1.073	1.163	1.306	0.905	0.910
6	55	55	08	08	32	32	45	45	58	58
LM-XII2	1.041	1.042	1.016	1.223	0.871	0.881	0.884	0.884	1.125	1.119
7	66	14	14	06	06	60	60	47	47	66
T4000SST	1.158	0.734	0.704	1.074	0.932	1.202	1.328	1.291	1.196	1.169
8	50	07	07	29	29	62	62	46	46	50
T4000SST	1.118	1.160	1.105	0.986	0.972	0.831	0.831	0.886	0.824	0.913
9	35	09	09	31	31	-	62	46	46	50
SR299	1.478	1.328	1.275	1.325	1.274	0.678	0.693	0.702	0.711	1.592
10	33	01	01	20	20	43	43	53	53	53
SR299	1.091	1.228	1.239	1.388	1.470	1.292	1.302	1.355	1.251	1.299
11	28	04	04	23	23	61	rcvr	rcvr	65	28
LM-XII2	0.094	0.932	1.060	0.960	0.942	0.871	fail	fail	1.028	1.137
12	22	21	21	13	13	59	59	50	64	64
LM-XII2	0.133	0.133	0.132	0.144	0.144	0.139	0.140	1.242	1.325	1.045
13	19	05	05	24	24	42	42	63	63	rcvr
LM-XII2	1.230	1.387	1.363	1.312	1.334	1.312	1.373	1.316	1.308	fail
14	16	16	16	16	16	16	16	16	16	16
SR299	1.463	1.478	1.454	1.380	1.308	1.388	1.368	1.386	1.403	1.371
15	18	18	18	18	18	18	18	18	18	18
LM-XII2	1.601	1.543	1.589	1.540	1.592	1.590	1.567	1.639	1.559	1.602
16a	DION A									
LM-XII2	1.057	1.349	1.349	1.349	1.349	1.348	1.348	-	-	-
16c	DION C									
T4000SST	1.275	1.275	1.275	1.275	1.273	1.273	1.273	1.273	1.273	1.273
16d	DION D									
SR299	1.436									
HERS	HERS									
SNR-8C	0.200									
MADR	MADR									
SNR-8	0.000									
ONSA	ONSA									
SNR-800	0.995									
WETT	WETT									
SNR-800	0.000									

LM-XII2 = Ashtech LM-XII2 (L1, L2); SNR-8, SNR-8C, SNR-800 = Rogue series (L1, L2);

SR299 = Leica 200 (L1, L2); T4000SST = Trimble 4000SST (L1, L2).

Table A.7: Receiver types, site occupations and antenna heights in metres (to bottom of antenna for Ashtech, Leica, Trimble receivers, bottom of choke-ring for Rogue receivers) for May 1993 survey.

Site	ΔX/m	ΔY/m	ΔZ/m	Note
DION (54)	0.000	0.000	0.000	Point A (LM-XII2)
DION (54)	+1.893	+2.152	−3.314	Point C (T4000SST)
DION (54)	+2.821	−0.344	−3.111	Point D (SR299)

Table A.8: Fiducial site eccentricities, May 1993. Vectors are from the ITRF marker (Table 2.3) to the campaign GPS marker.

Site	X/m	Y/m	Z/m	Note
CG01	4549888.4898	1895762.6950	4034573.8450	
CG02	4554541.0744	1862482.4317	4044666.4176	
CG03	4563198.0211	1874447.0221	4029837.4042	
CG04	4545704.5771	1920346.1101	4027956.2943	
CG05	4539618.8176	1944401.5823	4023759.3004	
CG06	4588151.3997	1849820.5166	4012934.8976	
CG07	4564567.1201	1903044.7986	4014921.5317	
CG08	4588623.8704	1876481.3746	4000362.0545	
CG09	4565606.4172	1920347.3535	4005106.1894	
CG10	4550930.0226	1955644.8709	4005110.9941	
CG11	4536964.9040	1988571.7915	4005401.9390	
CG12	4524262.7746	2006260.5563	4010418.6643	
CG13	4615299.8428	1845836.1499	3984776.8111	
CG14	4592868.1394	1897227.6975	3985190.0913	
CG15	4577903.7940	1938011.9208	3982736.8737	
CG16	4745949.5604	1905705.9397	3799168.9583	
CG17	4556996.7673	1969382.9611	3991140.0909	
CG18	4596042.5285	1733476.7777	4055720.8902	
CG19	4601771.3983	1901220.0638	3973719.4394	
CG20	4604106.4910	1918649.6160	3962296.9852	
CG21	4591931.3232	1930877.7204	3971480.4613	
CG22	4612635.2419	1900869.1106	3961679.9307	
CG23	4610905.3371	1909825.5399	3959468.2207	
CG24	4597863.5703	1938508.9517	3959991.1641	
CG25	4595358.9785	1950490.4487	3958135.9130	
CG26	4571271.3266	1972601.7425	3974038.4014	
CG27	4563037.4300	1980600.6694	3978750.0988	
CG28	4589879.4265	1967399.7337	3955033.7416	
CG29	4606396.3635	1935360.5060	3951351.9771	
CG30	4636745.4014	1886703.0553	3940596.4689	
CG31	4634559.0285	1899331.7530	3935867.8972	
CG32	4621884.2644	1921702.1116	3940613.9004	
CG33	4610013.6606	1944405.6062	3943029.0037	
CG34	4597929.3794	1972797.3859	3942758.7222	
CG35	4585749.3101	1997779.2340	3944379.2232	
CG36	4571812.4041	1996184.7850	3961739.2776	
CG37	4567938.0683	2006907.3484	3960889.8927	
CG38	4552114.8422	2037147.4303	3963188.3630	
CG39	4522199.8898	2065010.9574	3982702.1769	
CG40	4525889.1461	2070725.8832	3975606.8202	
CG41	4579632.0022	2014445.3368	3943140.9482	
CG42	4652969.9791	1877352.3586	3925788.0681	
CG43	4652055.6765	1897724.6958	3917371.9923	
CG44	4644472.9435	1937384.8309	3906731.1910	
CG45	4620984.3248	1948528.3586	3928588.5150	
CG46	4615566.5128	1961954.8534	3927646.1840	
CG47	4633203.1763	1961421.8427	3907452.6115	
CG48	4627578.6544	1976842.5391	3908073.2927	
CG49	4619101.9805	1980972.4771	3914066.1389	
CG50	4607401.2192	1989498.7022	3924138.9212	
CG51	4599894.5434	2004287.3738	3925360.4658	
CG52	4574393.7780	2033128.9927	3939302.3731	
CG53	4588486.6814	2028895.7682	3925877.6586	
CG54	4595218.3314	2039437.5092	3912626.1446	Fixed
54-D	4595219.2594	2039435.0132	3912626.3476	Fixed
CG55	4569149.7715	2052266.5216	3935686.7908	
CG56	4578277.9888	2075992.2189	3913287.1625	
CG57	4613032.0700	2000514.9558	3911361.9172	
CG58	4610054.5234	2015493.5270	3907135.6031	
CG59	4700881.0507	1817837.1068	3896017.5884	
CG60	4663885.9834	1879924.8421	3911789.9847	
CG61	4679033.3657	1850628.0586	3906821.6630	
CG62	4661660.8049	1924282.3177	3892694.0702	
CG63	4654493.0551	1963758.3925	3881625.0202	
CG64	4647892.5925	1967179.4613	3887940.7111	
CG65	4641801.2481	1979193.1300	3888527.4426	
CG66	4610660.7545	2047374.8817	3889970.5504	

Table A.9: Site coordinates, May 1993.

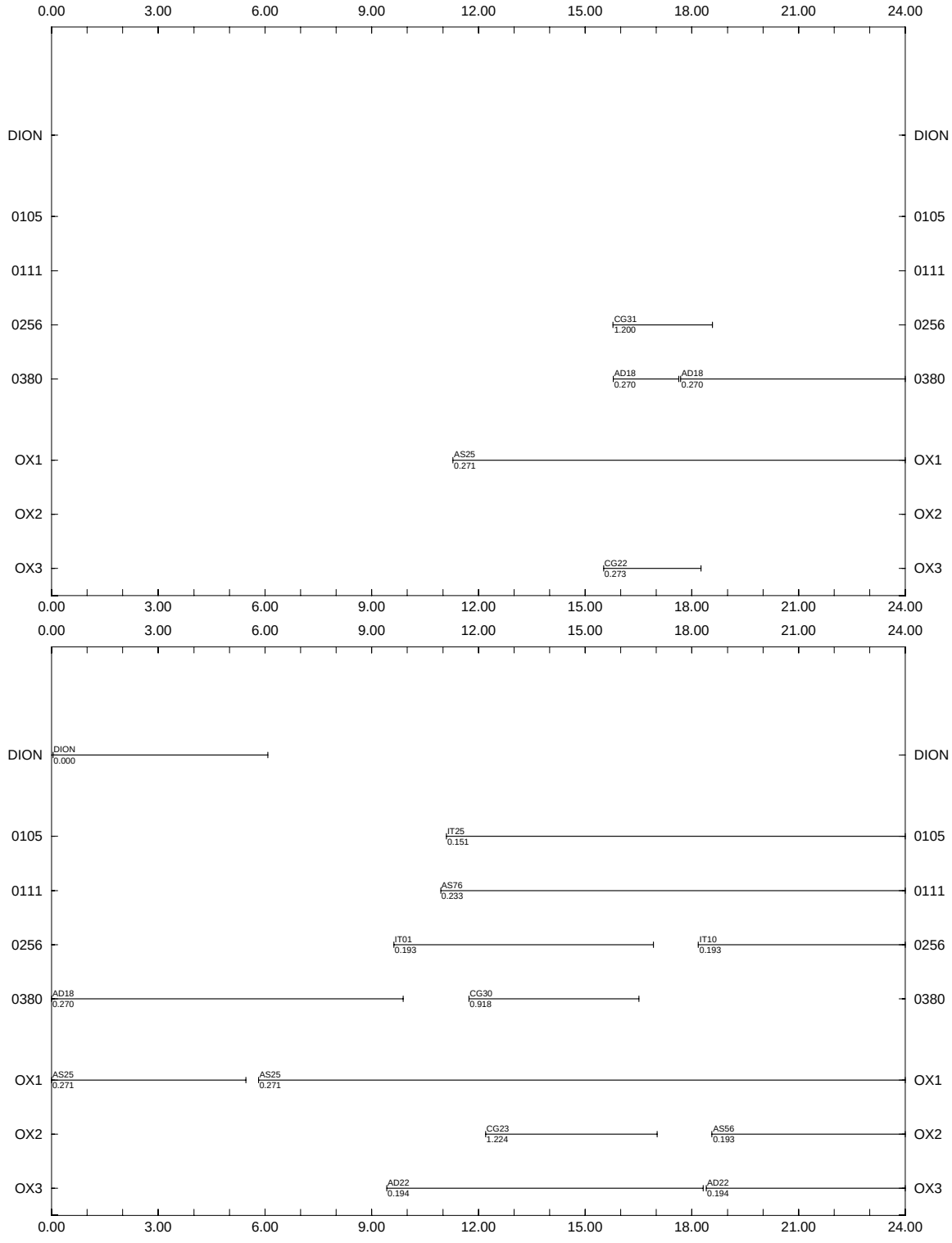

Figure A.1: June 1995 Central Greece site occupations, days 171 (above) and 172 (below).

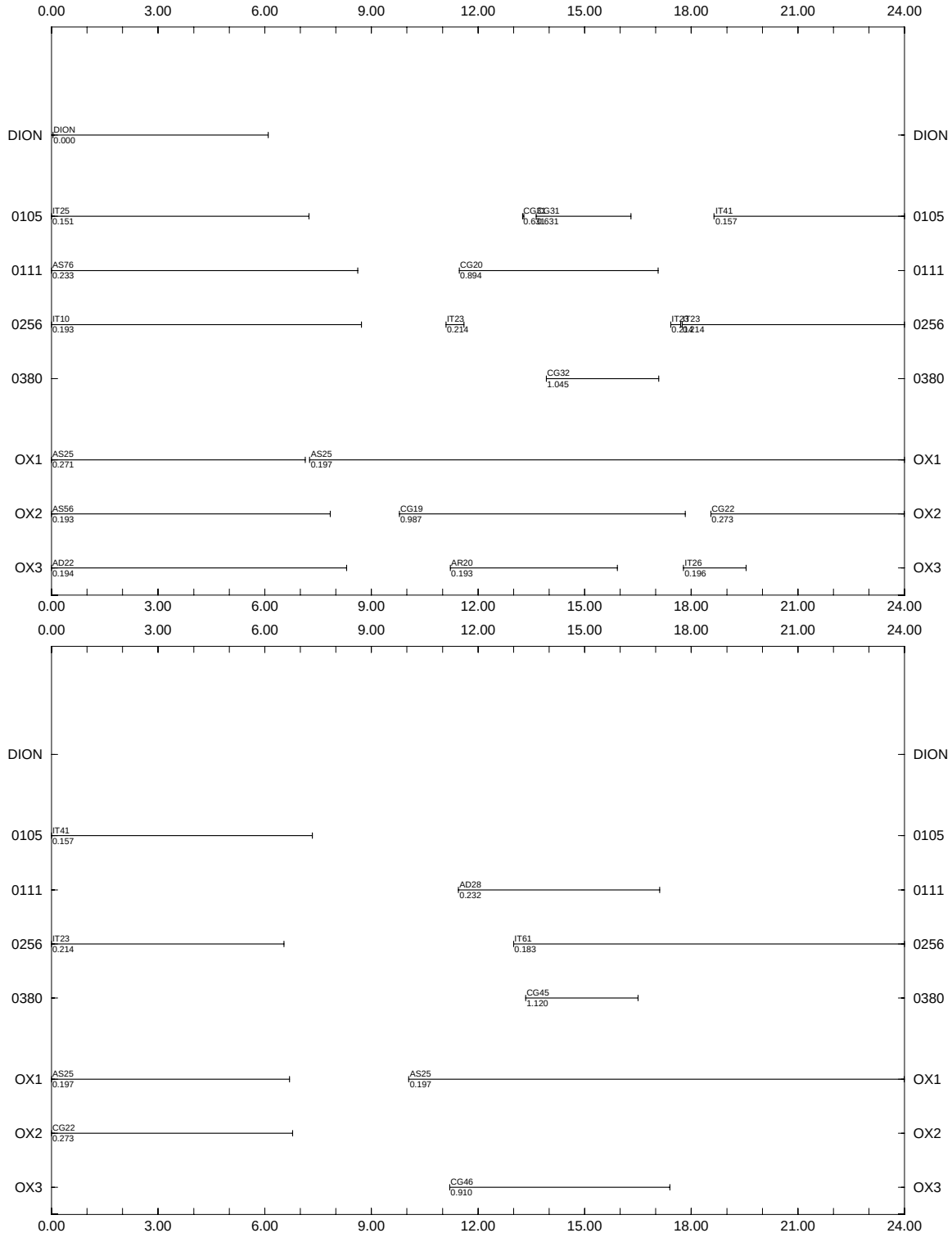

Figure A.2: June 1995 Central Greece site occupations, days 173 (above) and 174 (below).

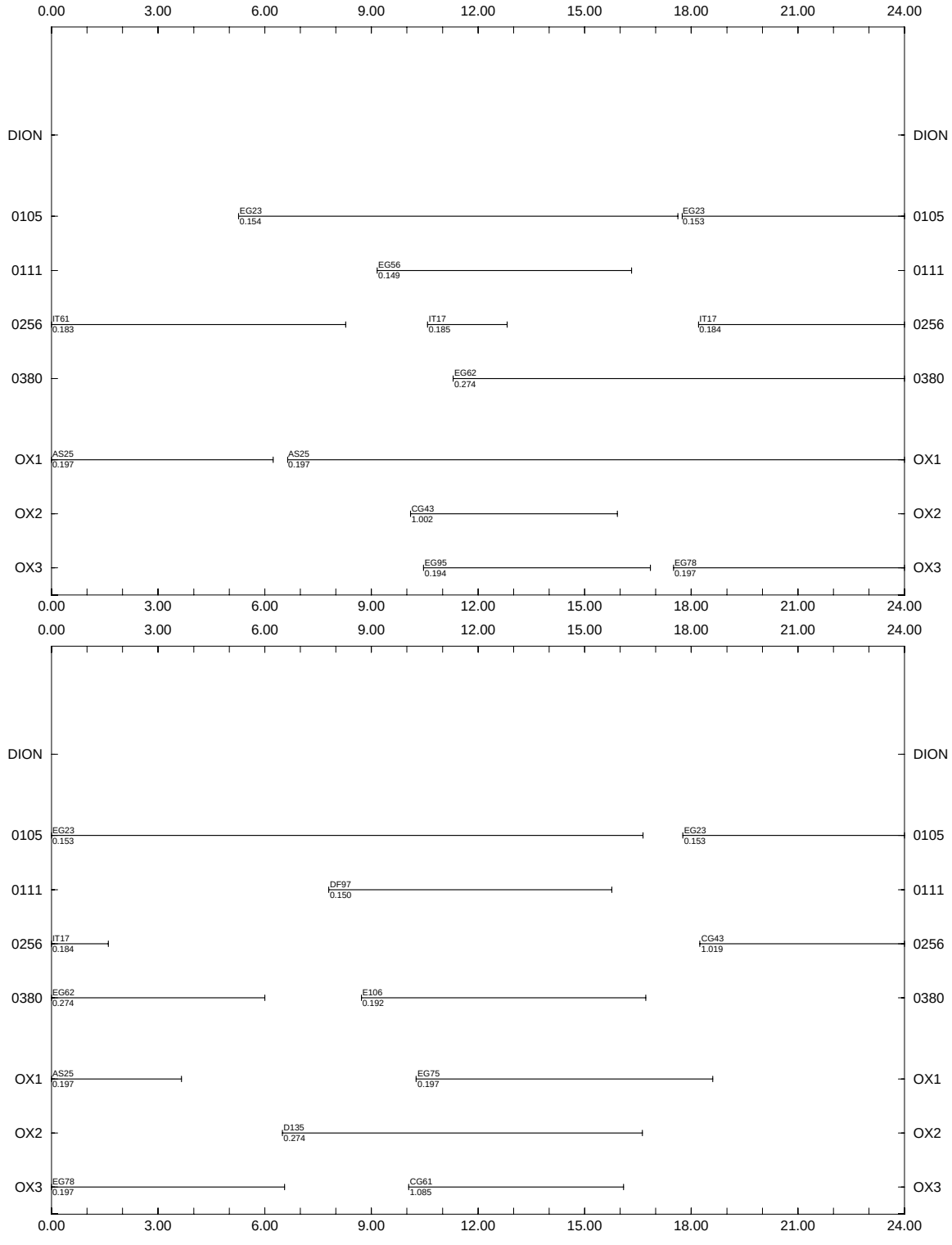

Figure A.3: June 1995 Central Greece site occupations, days 175 (above) and 176 (below).

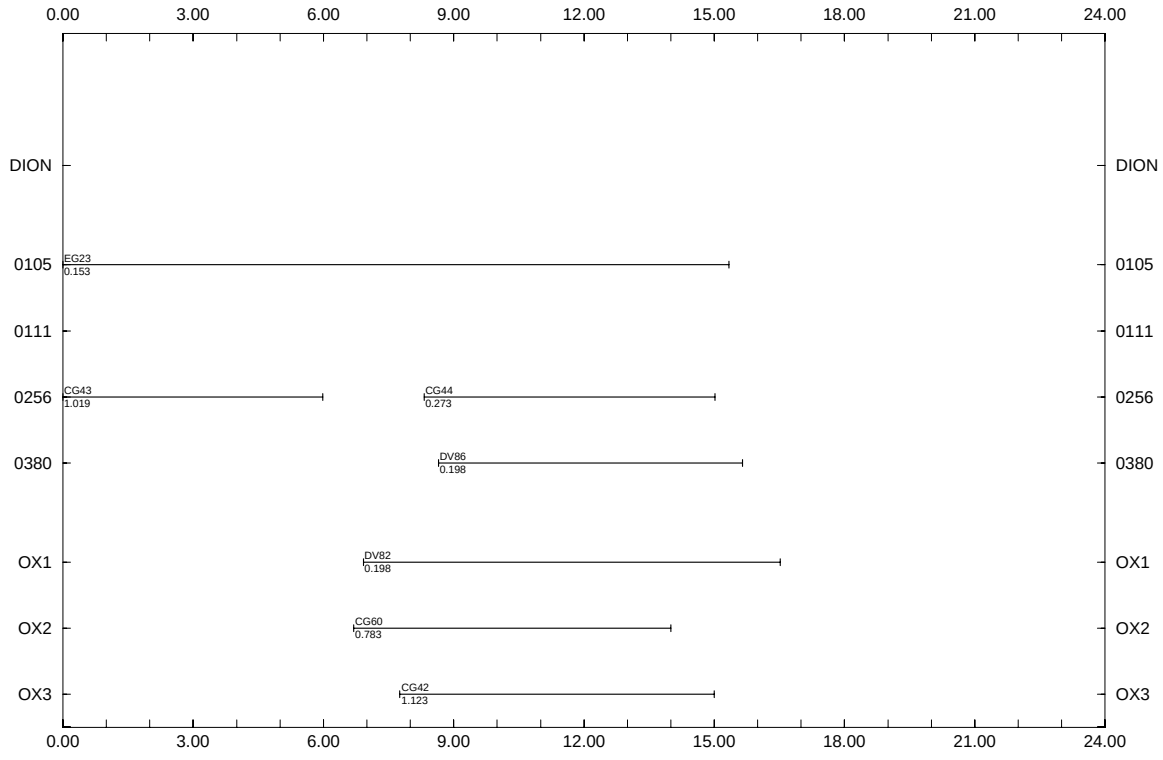

Figure A.4: June 1995 Central Greece site occupations, day 177.

Site	ΔX/m	ΔY/m	ΔZ/m	Note
Dion (54)	−0.005	+17.530	−2.632	GPS pillar

Table A.10: Site eccentricity for Dionysos since early 1995. Vectors are from the ITRF marker (Table 2.3) to the $\frac{5}{8}$" thread on the permanent GPS pillar.

Site	X/m	Y/m	Z/m	Note
CG19	4601771.3904	1901210.0865	3973719.4754	
CG20	4604106.4620	1918639.6246	3962296.9477	
CG22	4612635.2357	1900859.1356	3961679.9415	
CG23	4610905.3224	1909815.5538	3959468.2165	
CG30	4636745.4095	1886693.0398	3940596.5648	
CG31	4634558.9325	1899321.6854	3935867.8737	
CG32	4621884.2266	1921692.1118	3940613.8546	
CG42	4652969.9843	1877342.3529	3925788.0269	
CG43	4652055.6914	1897714.6893	3917371.9292	
CG44	4644472.9415	1937374.8322	3906731.1491	
CG45	4620984.2787	1948518.3271	3928588.4954	
CG46	4615566.4251	1961944.8312	3927646.1082	
DION	4595216.4408	2039442.9013	3912626.8032	Fixed
CG60	4663885.9810	1879914.8337	3911789.9395	
CG61	4679033.4190	1850618.0826	3906821.6815	

Table A.11: Site coordinates, June 1995 (central Greece stations only).

Site	X/m	Y/m	Z/m	Note
CG19	4601771.4200	1901220.1109	3973719.4566	
CG20	4604106.5015	1918649.6266	3962296.9866	
CG22	4612635.2568	1900869.1604	3961679.9479	
CG30	4636745.4106	1886703.0585	3940596.5502	
CG31	4634559.0407	1899331.7262	3935867.9328	
CG32	4621884.2707	1921702.1523	3940613.9046	
CG33	4610013.7101	1944405.6255	3943029.0158	
CG42	4652970.0188	1877352.4097	3925788.0273	
CG43	4652055.7709	1897724.7400	3917371.9441	
DION	4595216.4406	2039452.9010	3912626.8038	Fixed
CG60	4664034.6147	1879678.7083	3911668.4321	
CG61	4679033.4526	1850628.1137	3906821.6587	
CG62	4661660.8411	1924282.3444	3892694.0400	

Table A.12: Site coordinates, October 1995 (Central Greece stations only).

Site	ΔX/m	ΔY/m	ΔZ/m	Note
Dion (54)	+1.893	+2.152	–3.314	Point C

Table A.13: Fiducial site eccentricities, May 1996. Vectors are from the ITRF marker (Table 2.3) to the campaign GPS marker.

A.6 Epoch 1996.39 (May 1996)

Occupation times and antenna heights (to L1 phase centre) for the May 1996 Central Greece campaign are depicted in Figures A.11 and A.12. All receivers are Ashtech Z-XII, except OXF_1 and OXF_2 (Ashtech LM-XII).

Table A.13 gives offsets from the ITRF fiducial site coordinates to the Dionysos point C occupied in May 1996.

Campaign coordinates are given in Table A.14.

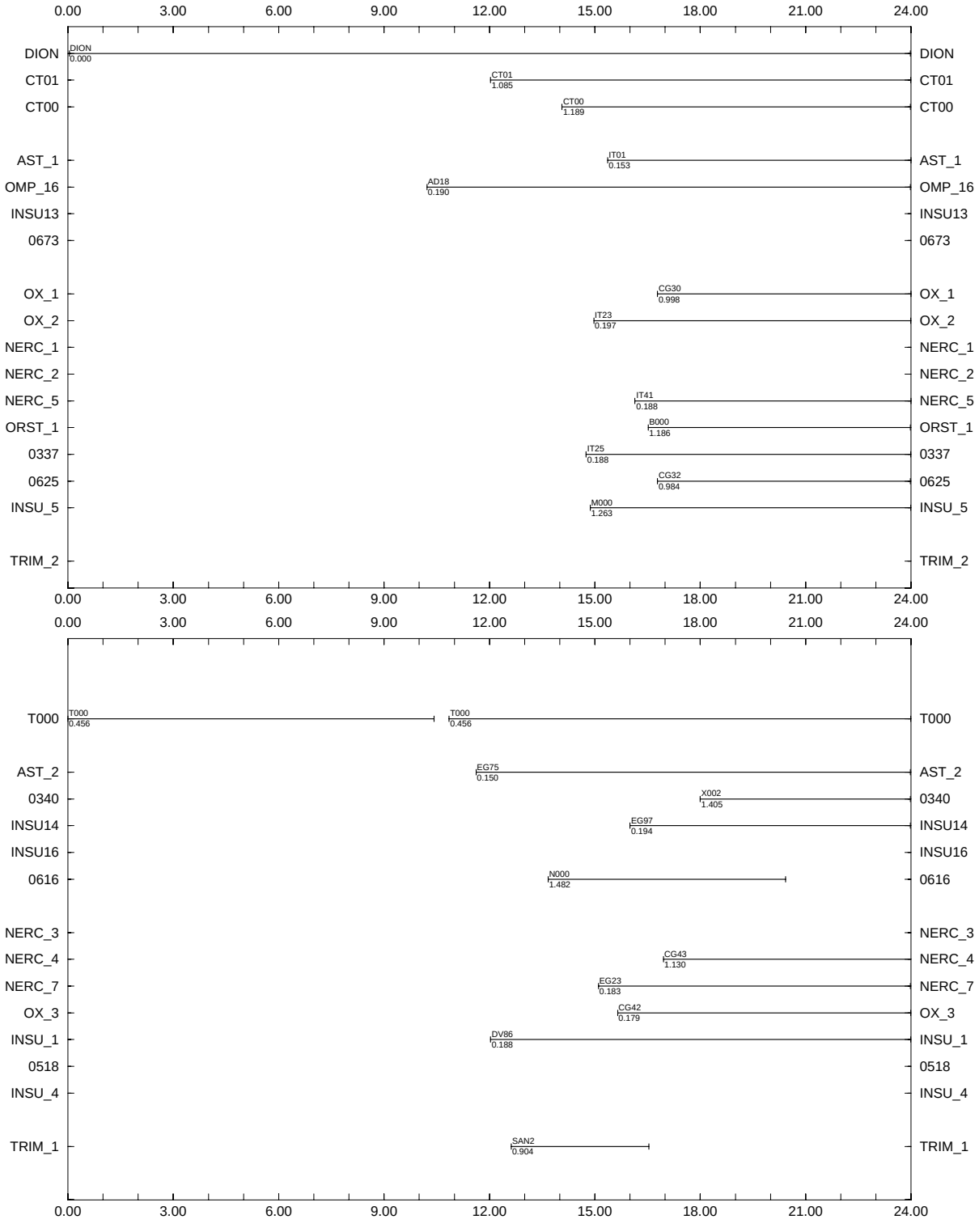

Figure A.5: October 1995 Central Greece site occupations, day 276 (north, above; south, below).

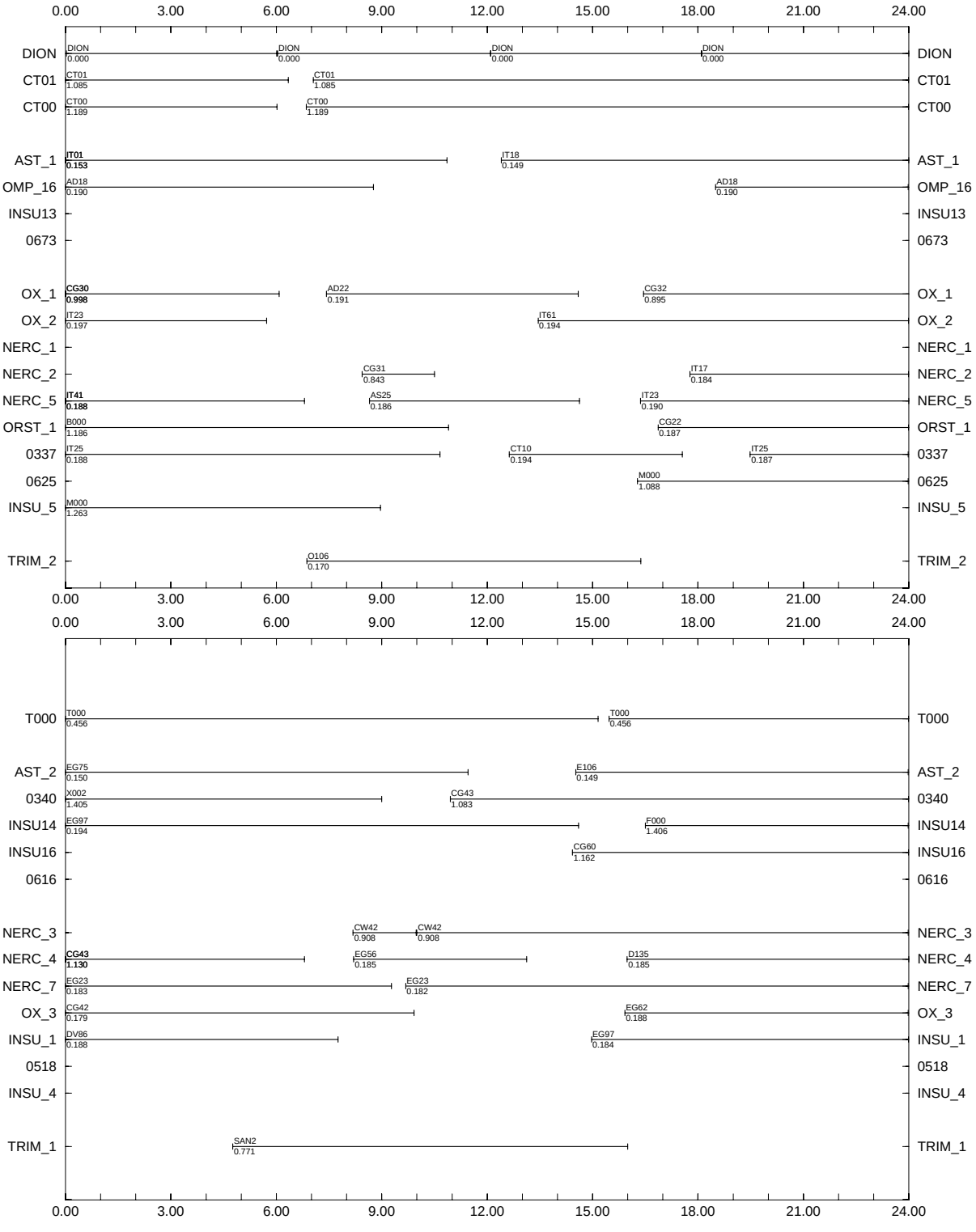

Figure A.6: October 1995 Central Greece site occupations, day 277 (north, above; south, below).

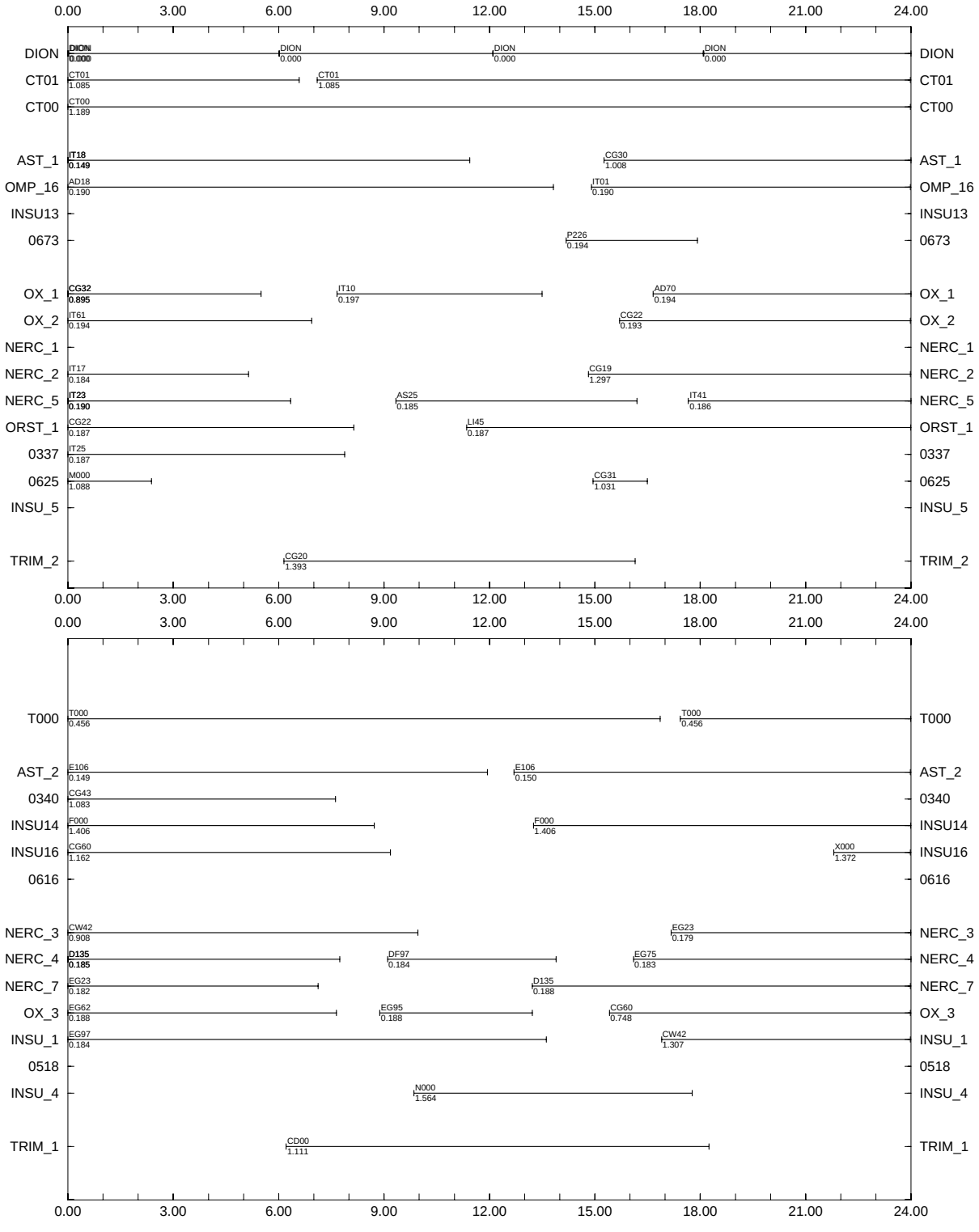

Figure A.7: October 1995 Central Greece site occupations, day 278 (north, above; south, below).

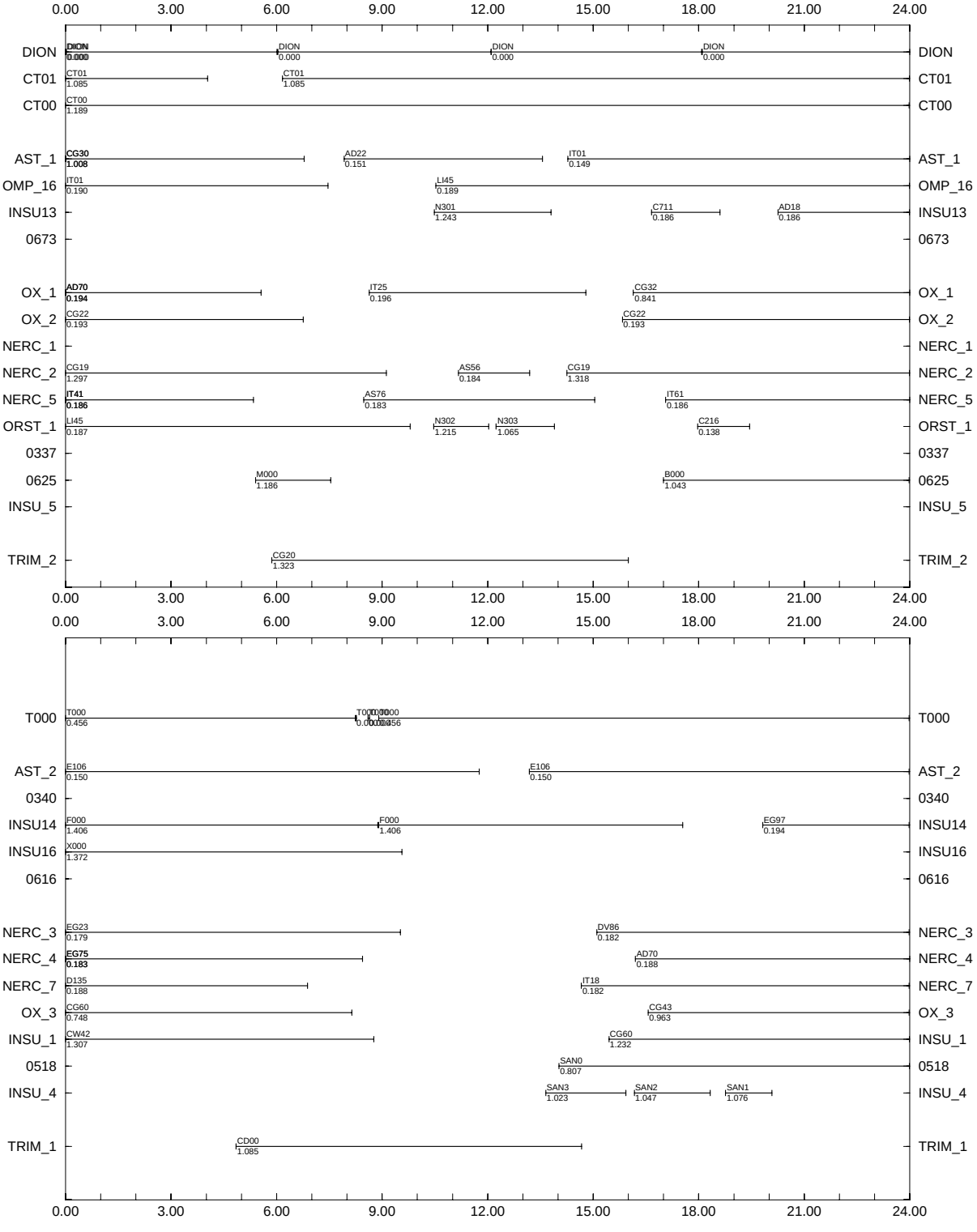

Figure A.8: October 1995 Central Greece site occupations, day 279 (north, above; south, below).

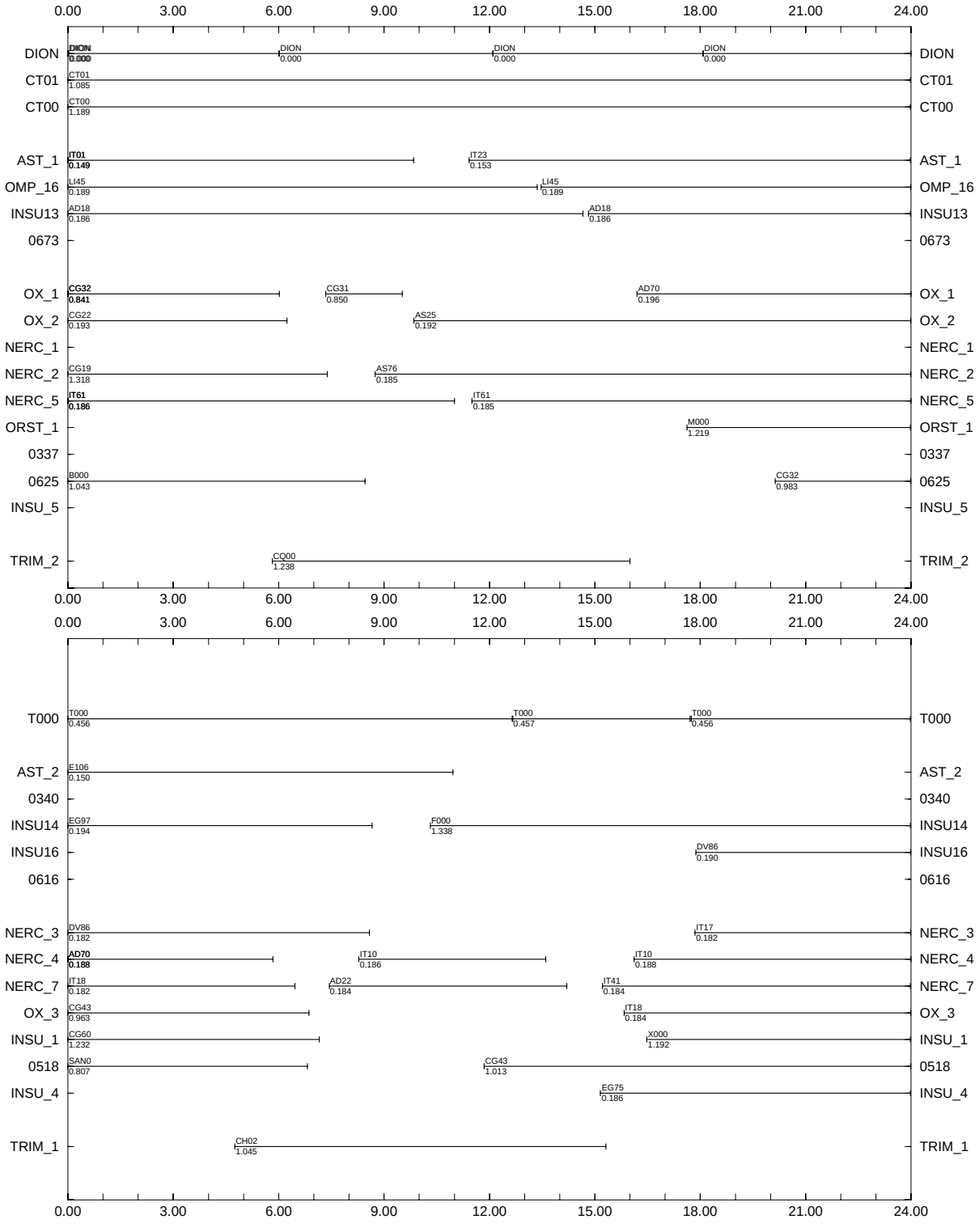

Figure A.9: October 1995 Central Greece site occupations, day 280 (north, above; south, below).

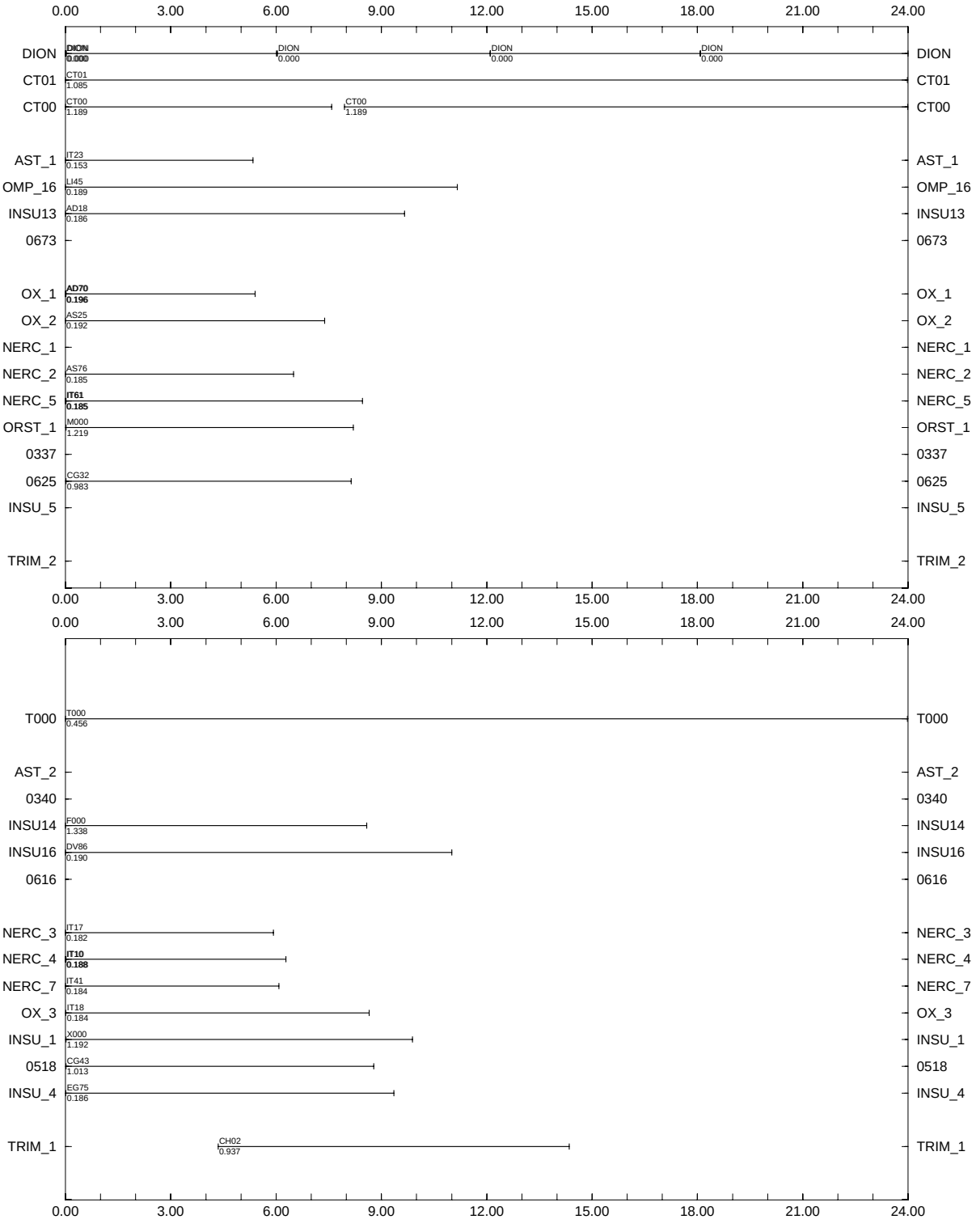

Figure A.10: October 1995 Central Greece site occupations, day 281 (north, above; south, below).

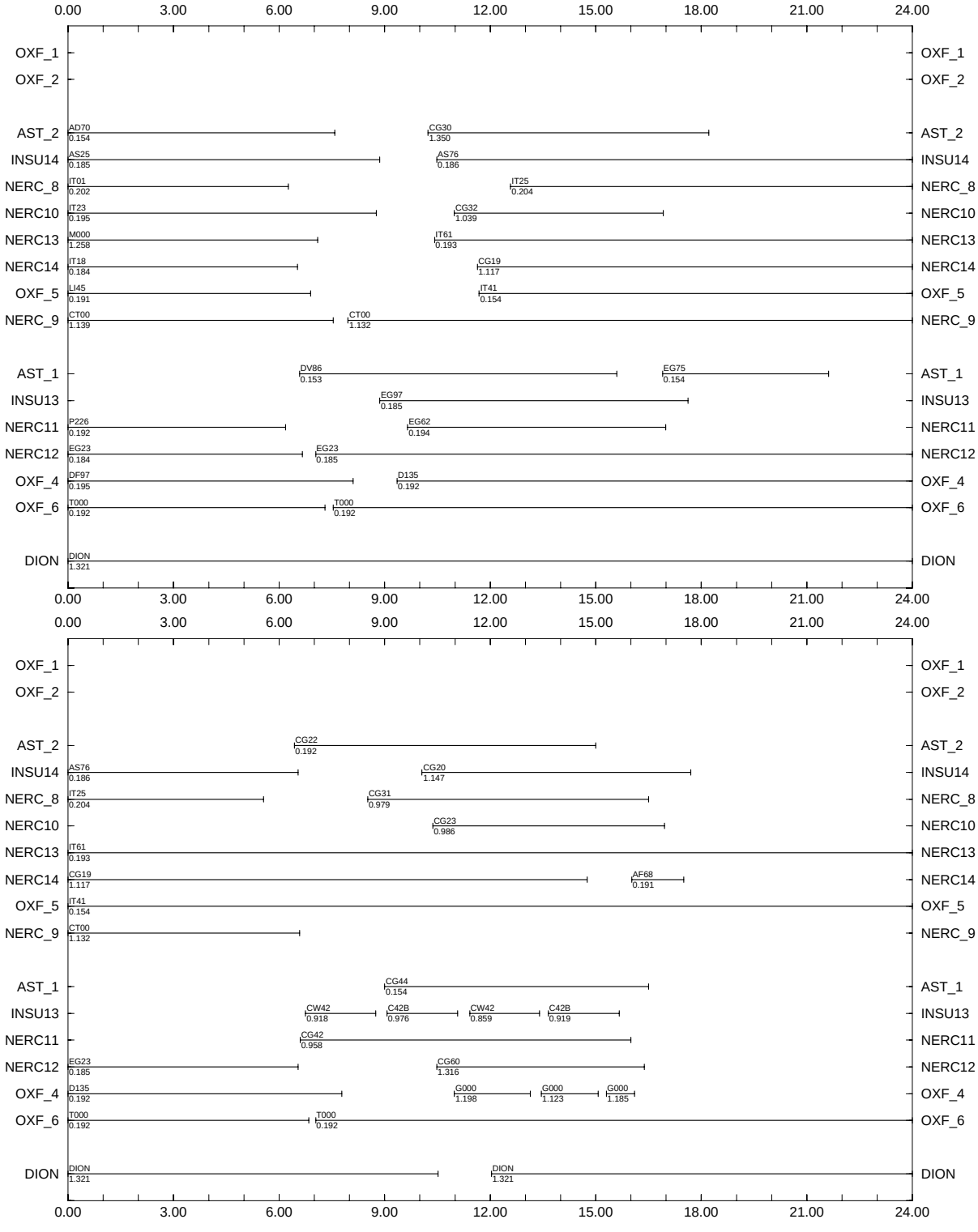

Figure A.11: May 1996 Central Greece site occupations, days 149 (above) and 150 (below).

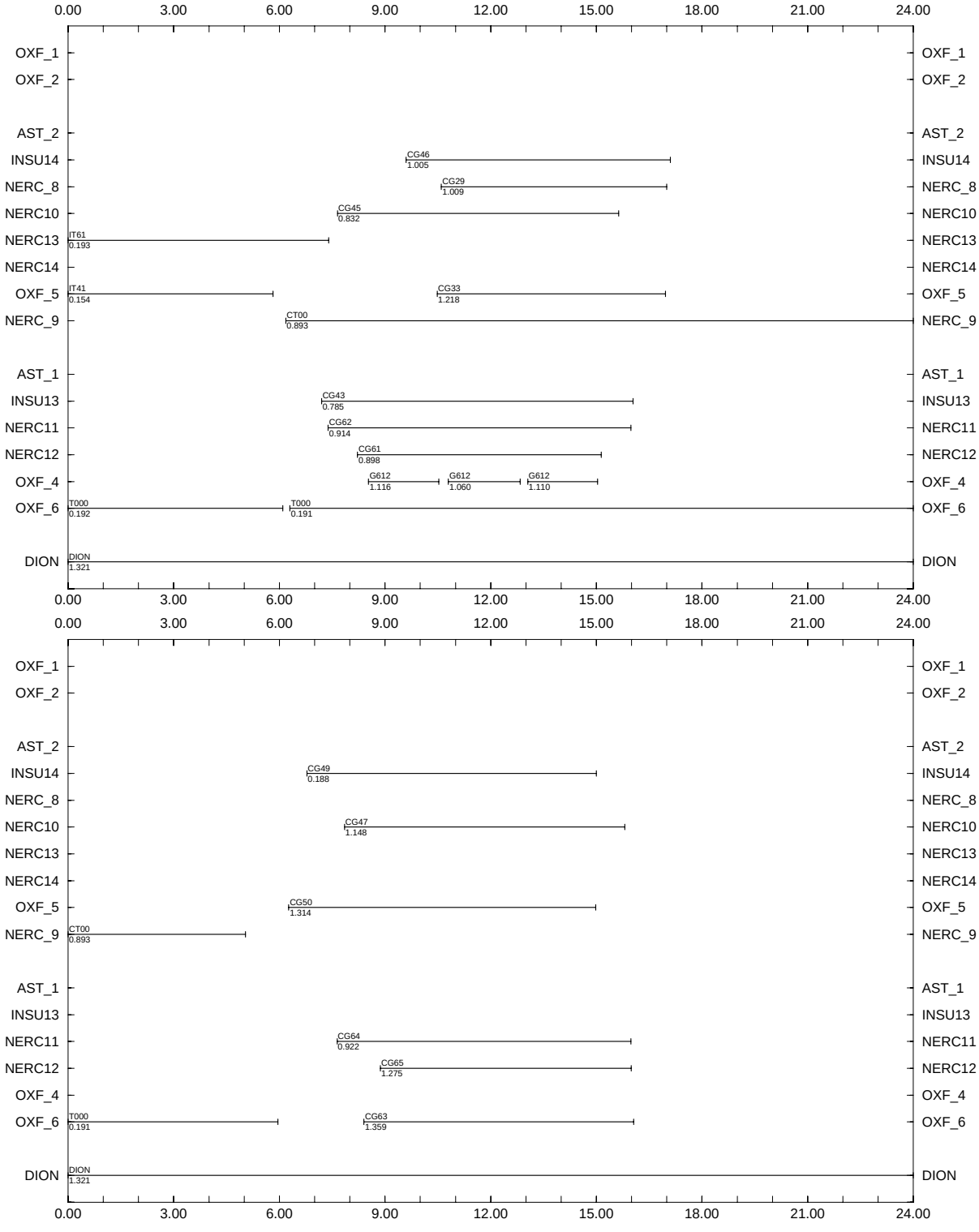

Figure A.12: May 1996 Central Greece site occupations, days 151 (above) and 152 (below).

Site	X/m	Y/m	Z/m	Note
CG01	4549888.4898	1895762.6950	4034573.8450	
CG19	4601771.4861	1901220.1580	3973719.5325	
CG20	4604106.5884	1918649.7014	3962297.0736	
CG22	4612635.3547	1900869.2185	3961680.0485	
CG23	4610905.4273	1909825.6347	3959468.3215	
CG29	4606396.4374	1935360.5777	3951352.0291	
CG30	4636745.4604	1886703.1074	3940596.6242	
CG31	4634559.0938	1899331.7764	3935868.0146	
CG32	4621884.3433	1921702.1905	3940613.9708	
CG33	4610013.7467	1944405.6746	3943029.0553	
CG42	4652970.1046	1877352.4473	3925788.1123	
CG43	4652055.8185	1897724.7639	3917371.9953	
CG44	4644472.9953	1937384.8748	3906731.1628	
CG45	4620984.4077	1948528.4336	3928588.5680	
CG46	4615566.5657	1961954.9064	3927646.2042	
CG47	4633203.0491	1961421.7914	3907452.6183	
CG49	4619102.1143	1980972.5583	3914066.1911	
CG50	4607401.2846	1989498.7672	3924138.9143	
CG54	4595218.3422	2039437.5299	3912626.1104	Fixed
CG60	4663886.0832	1879924.9178	3911790.0135	
CG61	4679033.5111	1850628.1534	3906821.7310	
CG62	4661660.8957	1924282.3845	3892694.0858	
CG63	4654493.1756	1963758.4791	3881625.0507	
CG64	4647892.7370	1967179.5530	3887940.7676	
CG65	4641801.3858	1979193.2170	3888527.4753	

Table A.14: Site coordinates, May 1996 (Central Greece sub-network).

Appendix B

Grevena Network occupations

The purpose of this Appendix is to tabulate unwieldy data appertaining to the two Grevena Network occupations that are referred to in this thesis (May 1995 and September 1995).

B.1 Epoch 1995.39 (May 1995)

Figures B.1 – B.4 show the times of observations and antenna heights (to L1 phase centre) pertaining to the May 1995 campaign. All receivers are Ashtech LM-XII, except P12 (Ashtech P-XII); IN_z12, Z12a, Z12b (Ashtech Z-XII); 03_T, 08_T (Trimble 4000SST); DION (Trimble 4000SSE)

Campaign coordinates are given in Tables B.1 (co-seismic sites) and B.2 (post-seismic sites).

B.2 Epoch 1995.74 (September 1995)

Occupation times and antenna heights (to L1 phase centre) for the September 1995 Grevena campaign are depicted in Figures B.5 – B.8. All receivers are Ashtech LM-XII, except INSU_10, INSU_11, AST_1, AST_2 (Ashtech Z-XII); TRIMBLE (Trimble 4000SST); DION (Trimble 4000SSE)

Campaign coordinates are given in Table B.3.

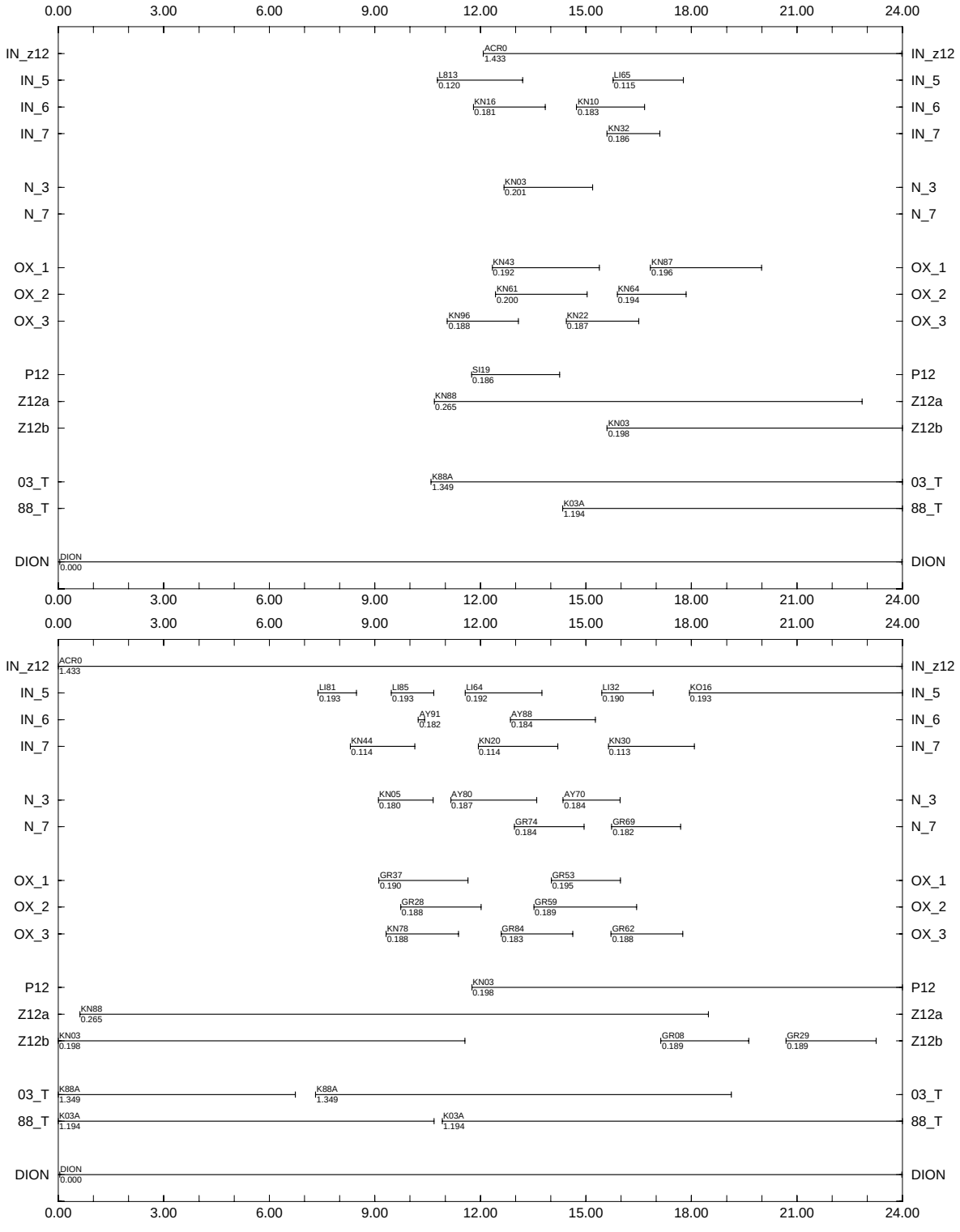

Figure B.1: May 1995 Grevena site occupations, days 138 (above) and 139 (below).

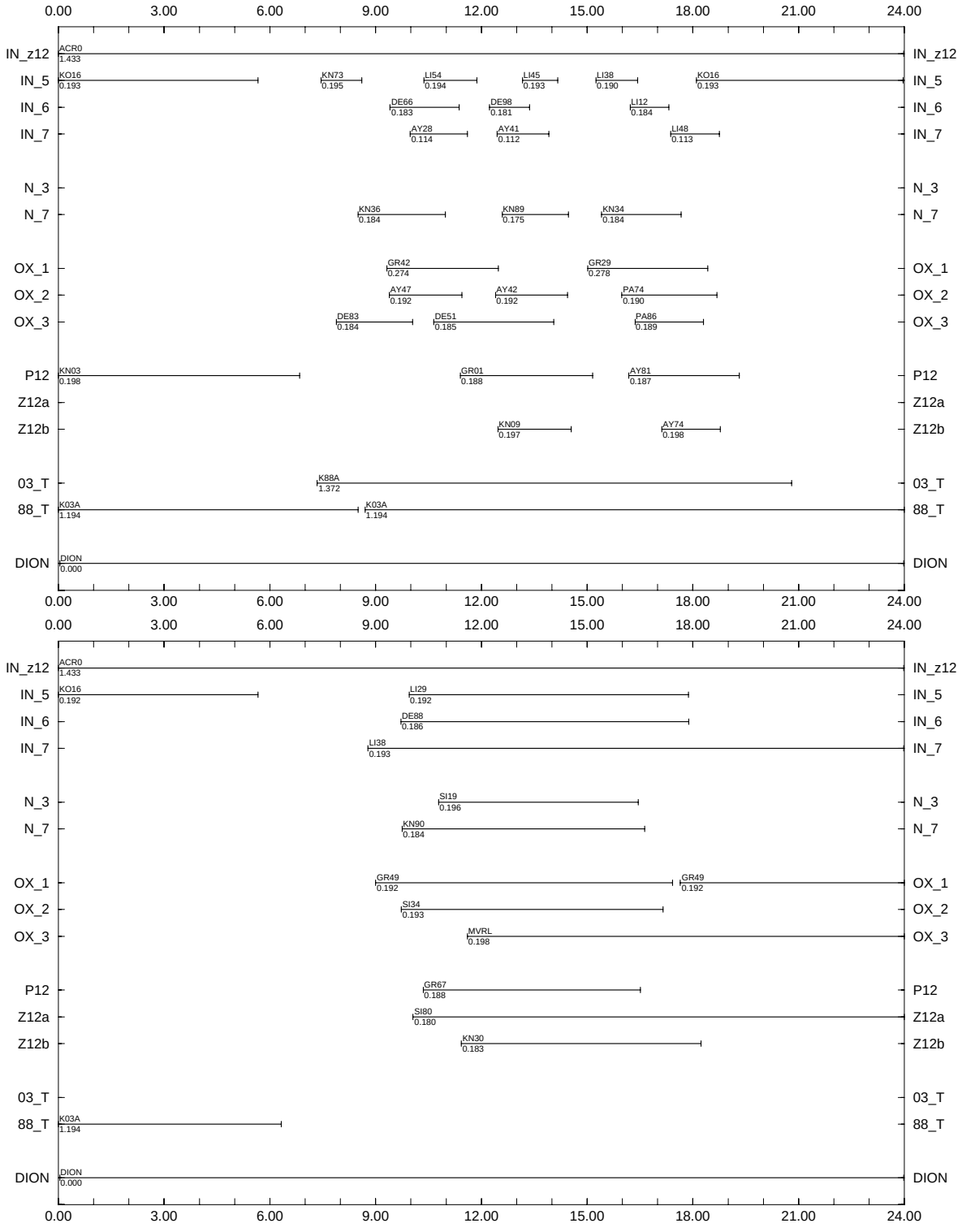

Figure B.2: May 1995 Grevena site occupations, days 140 (above) and 141 (below).

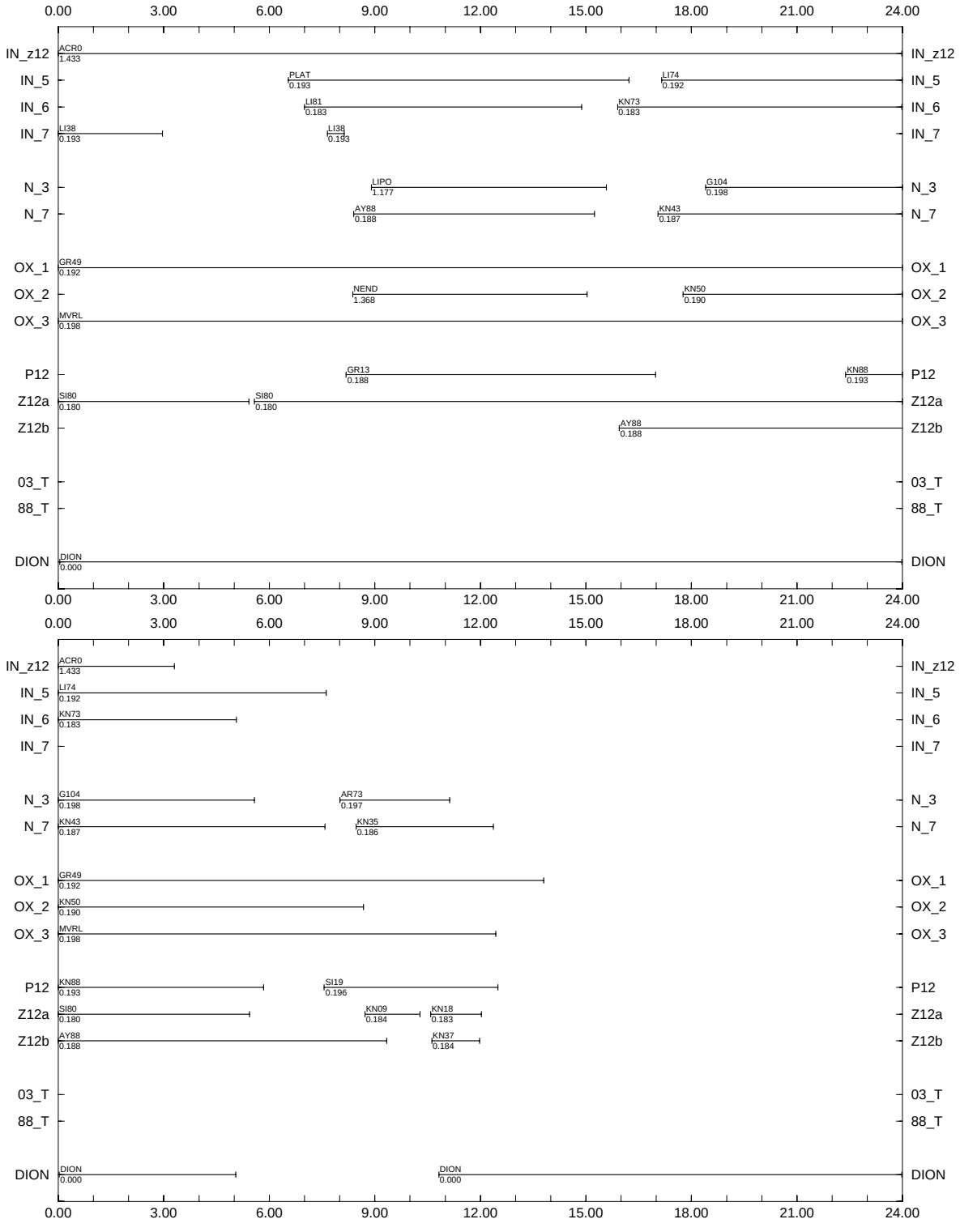

Figure B.3: May 1995 Grevena site occupations, days 142 (above) and 143 (below).

Site	X/m	Y/m	Z/m	Note
ACR0	4530309.6943	1822729.7013	4090841.9268	Local base
AR73	4527520.8146	1768331.6516	4117240.1567	
AY28	4558489.6033	1810603.5600	4064342.1417	
AY41	4553609.4058	1816001.4696	4067744.4995	
AY42	4559342.9861	1802154.6837	4067204.8775	
AY47	4557936.9020	1801372.7319	4069071.1886	
AY51	4555223.2290	1807372.2660	4069291.0478	
AY53	4557663.8523	1798544.4003	4070859.0426	
AY58	4553594.0086	1808424.5212	4070599.2748	
AY66	4552542.1670	1806396.8412	4072712.2586	
AY69	4549443.4138	1812984.3015	4073299.9149	
AY70	4549744.5574	1809895.5787	4074199.4001	
AY74	4554004.2019	1799256.6877	4074625.9387	
AY80	4550010.7820	1805352.5322	4076072.2246	
AY81	4554005.3414	1794480.8052	4076821.8849	
AY84	4548184.6974	1808757.8840	4076583.4462	
AY91	4548785.7626	1805169.0246	4077698.6893	
DE51	4550598.2405	1817385.6473	4070426.0768	
DE66	4546620.0821	1821172.6489	4074090.9493	
DE83	4543759.1770	1817714.8688	4077521.1877	
DE98	4547736.0527	1816715.3787	4075221.8953	
DION	4595216.4408	2039452.9013	3912626.8032	Fixed
GR01	4554453.0391	1788858.9383	4078725.5075	
GR08	4552076.8667	1791761.3953	4080082.0185	
GR28	4549119.4800	1785327.0226	4085901.1310	
GR29	4548665.4145	1789500.0879	4084587.9362	
GR36	4547086.7150	1789794.5457	4086205.9820	
GR37	4548292.3911	1783428.7576	4087665.6253	
GR42	4549706.8825	1778106.4909	4088613.3962	
GR53	4545394.1138	1784334.3876	4090523.3072	
GR59	4543595.2600	1786576.3202	4091499.9354	
GR69	4540625.3703	1788450.0927	4094079.6456	
GR74	4541400.5160	1782424.9487	4095935.4211	
GR78	4539489.7745	1781668.1193	4098196.1523	
GR84	4539517.0988	1784823.3230	4096783.3986	
KN05	4545825.6313	1808347.5364	4079632.5454	
KN09	4550493.0105	1794933.1279	4080279.3891	
KN10	4547598.5969	1801504.3399	4080759.2257	
KN16	4544988.8590	1805846.8992	4081865.9541	
KN18	4550107.2845	1793443.4264	4081285.7648	
KN20	4544910.2491	1803714.4681	4082863.7683	
KN21	4546149.9148	1799951.6527	4082986.0437	
KN22	4546073.3128	1797146.9379	4084063.3907	
KN28	4543346.0633	1803643.6386	4084430.5797	
KN31	4543183.5256	1802060.3395	4085224.5202	
KN32	4544972.2797	1796007.5554	4085872.3407	
KN34	4546854.4304	1791689.5435	4085652.1357	
KN35	4544150.7614	1797585.9934	4086254.8044	
KN36	4541329.6778	1803477.1716	4086899.0726	
KN37	4542321.6234	1802021.3445	4086343.9095	
KN44	4539368.4976	1805500.0699	4088285.2679	
KN61	4537528.7340	1799062.0251	4093173.3461	
KN64	4537809.1932	1796768.1581	4093548.8348	
KN65	4540217.3592	1789772.2513	4093978.0561	
KN87	4536266.1000	1789733.2371	4098222.4237	
KN89	4547579.7209	1794652.3879	4083501.9671	
KN96	4542826.4914	1806309.6145	4084221.5088	
L813	4530299.5854	1822758.4465	4090847.8557	
LI12	4540638.7597	1819534.7013	4080678.2637	
LI32	4533670.2687	1829105.3907	4084298.3217	
LI45	4535870.5548	1817694.3110	4086438.5810	
LI48	4536654.0727	1813875.5273	4087110.6057	
LI54	4533876.8604	1816085.0940	4089027.4157	
LI64	4532159.3850	1814521.4995	4091743.7260	
LI65	4530522.3967	1817556.4559	4091938.3952	
LI85	4531685.5476	1809254.2338	4094918.8943	
PA74	4557066.2413	1789778.8185	4075501.6191	
PA85	4556855.7091	1786033.3924	4077300.4997	
PA86	4554177.9096	1792464.9793	4077446.9807	

Table B.1: Site coordinates, May 1995, co-seismic sites.

Site	X/m	Y/m	Z/m	Note
ACR0	4530309.6943	1822729.7013	4090841.9268	Local base
AY88	4545470.7128	1812821.9045	4077944.3830	
DE30	4551946.8339	1827848.7052	4065085.0193	
DE88	4548931.6640	1819154.4888	4071675.7915	
DION	4595216.4408	2039452.9013	3912626.8032	Fixed
G104	4546777.3527	1778140.6533	4092180.4387	
GR13	4556430.2678	1778481.3217	4081256.0010	
GR49	4550740.4525	1773360.6616	4089932.0380	
GR67	4541398.9582	1786648.2148	4093966.7537	
K03A	4548249.9573	1803143.0856	4079254.5940	
K88A	4536452.2484	1794900.0870	4096255.0345	
KN03	4548252.6347	1803181.9796	4079240.0404	
KN29	4541329.0720	1809120.7791	4084888.1166	
KN30	4540730.8296	1810473.1116	4084910.3601	
KN43	4541609.3290	1799656.3069	4088076.3949	
KN49	4539073.9069	1802155.0486	4090128.2568	
KN50	4537069.0739	1806779.2780	4090350.9918	
KN73	4532384.0308	1807223.4852	4095348.5242	
KN88	4536466.9567	1794897.7505	4096243.2058	
KO16	4521381.5117	1815841.3927	4103354.2666	
LI29	4535003.9105	1826603.7479	4084046.4042	
LI38	4534543.6053	1822308.7805	4086166.3667	
LI74	4526008.2808	1819394.6323	4096085.9839	
LI81	4527161.7726	1810928.0259	4098917.6118	
LIPO	4563305.8229	1829480.3482	4050871.8208	
LO79	4523239.8531	1831453.5575	4094532.7226	
NEND	4517426.1870	1776006.0164	4125321.4415	
SI19	4526163.8753	1801344.6495	4104338.7574	
SI34	4528211.8282	1789327.4961	4107820.4438	
SI80	4521577.2007	1781972.1061	4119114.8122	

Table B.2: Site coordinates, May 1995, post-seismic sites.

Site	X/m	Y/m	Z/m	Note
ACR0	4530309.4880	1822729.7065	4090841.8771	
AR44	4529837.4639	1780090.7616	4110079.4234	
AR69	4524558.0247	1777327.8679	4117075.4678	
AR73	4527520.6378	1768331.6618	4117240.1624	
AY68	4556200.7392	1797531.3403	4073141.2732	
AY88	4545470.5521	1812821.9370	4077944.3580	
DE30	4551946.7148	1827848.7199	4065085.0297	
DE88	4548931.4881	1819154.5188	4071675.7838	
DION	4595216.4419	2039452.9035	3912626.7995	Fixed
G104	4546777.2260	1778140.7108	4092180.4849	
GR04	4553395.2620	1790409.5661	4079255.1900	
GR13	4556430.1286	1778481.3603	4081256.0139	
GR49	4550740.2602	1773360.6615	4089932.0277	
GR69	4540625.2197	1788450.1098	4094079.6782	
KN03	4548252.5200	1803182.0149	4079240.0103	
KN29	4541328.9051	1809120.7620	4084888.1004	
KN43	4541609.1508	1799656.3120	4088076.3805	
KN49	4539073.7568	1802155.0371	4090128.2460	
KN50	4537068.9243	1806779.2831	4090350.9977	
KN73	4532383.8724	1807223.4950	4095348.5362	
KN88	4536466.7669	1794897.7592	4096243.1962	
KO16	4521381.3640	1815841.3953	4103354.2550	
LI29	4535003.7610	1826603.7461	4084046.4023	
LI38	4534543.5680	1822308.8675	4086166.4619	
LI74	4526008.1795	1819394.7078	4096086.0356	
LI81	4527161.6567	1810928.0604	4098917.6411	
LIPO	4563305.6333	1829480.3039	4050871.6722	
LO79	4523239.7223	1831453.5787	4094532.7212	
NEND	4517425.9995	1776006.0218	4125321.4119	
SI19	4526163.7082	1801344.6535	4104338.7543	
SI34	4528211.6428	1789327.5116	4107820.4369	
SI80	4521576.9993	1781972.1622	4119114.8020	
SIF0	4534874.0826	1792619.6154	4098679.4065	
SIF1	4534884.5913	1792631.8911	4098667.6991	Local base
SIF2	4534880.1908	1792623.0759	4098673.7565	
SIFA	4534874.3007	1792625.1521	4098677.0554	
SIFB	4534870.4028	1792620.1017	4098681.9281	
SIFC	4534880.2242	1792609.3916	4098677.4365	

Table B.3: Site coordinates, September 1995.

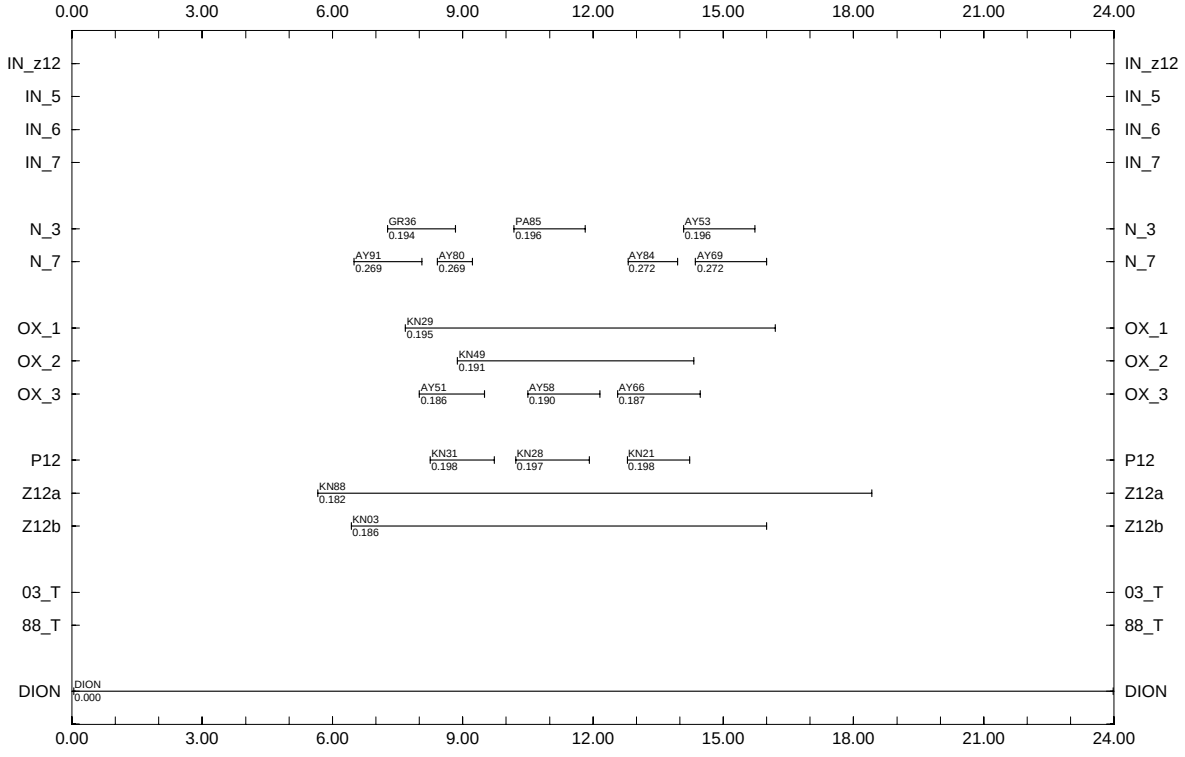

Figure B.4: May 1995 Grevena site occupations, day 144.

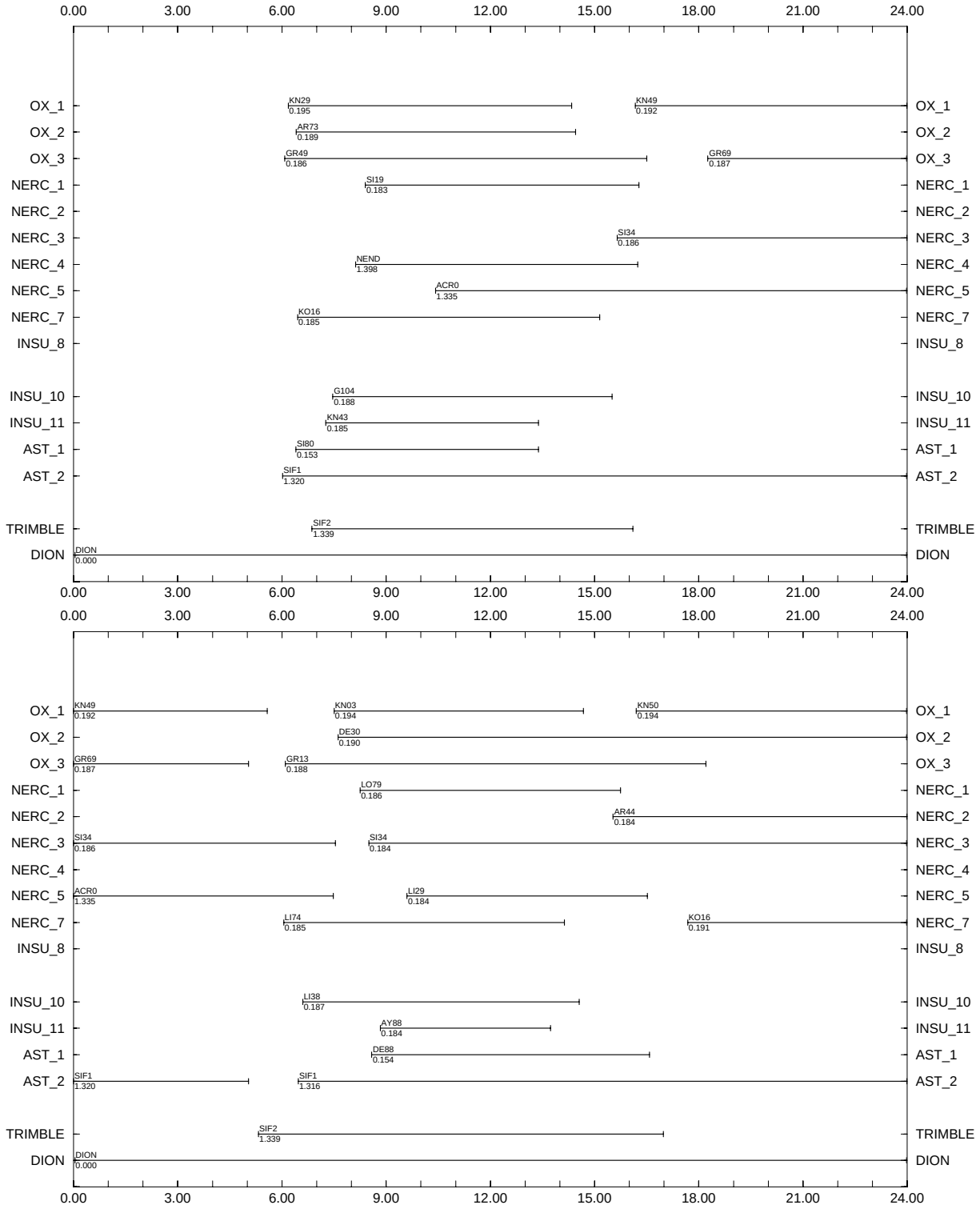

Figure B.5: September 1995 Grevena site occupations, days 267 (above) and 268 (below).

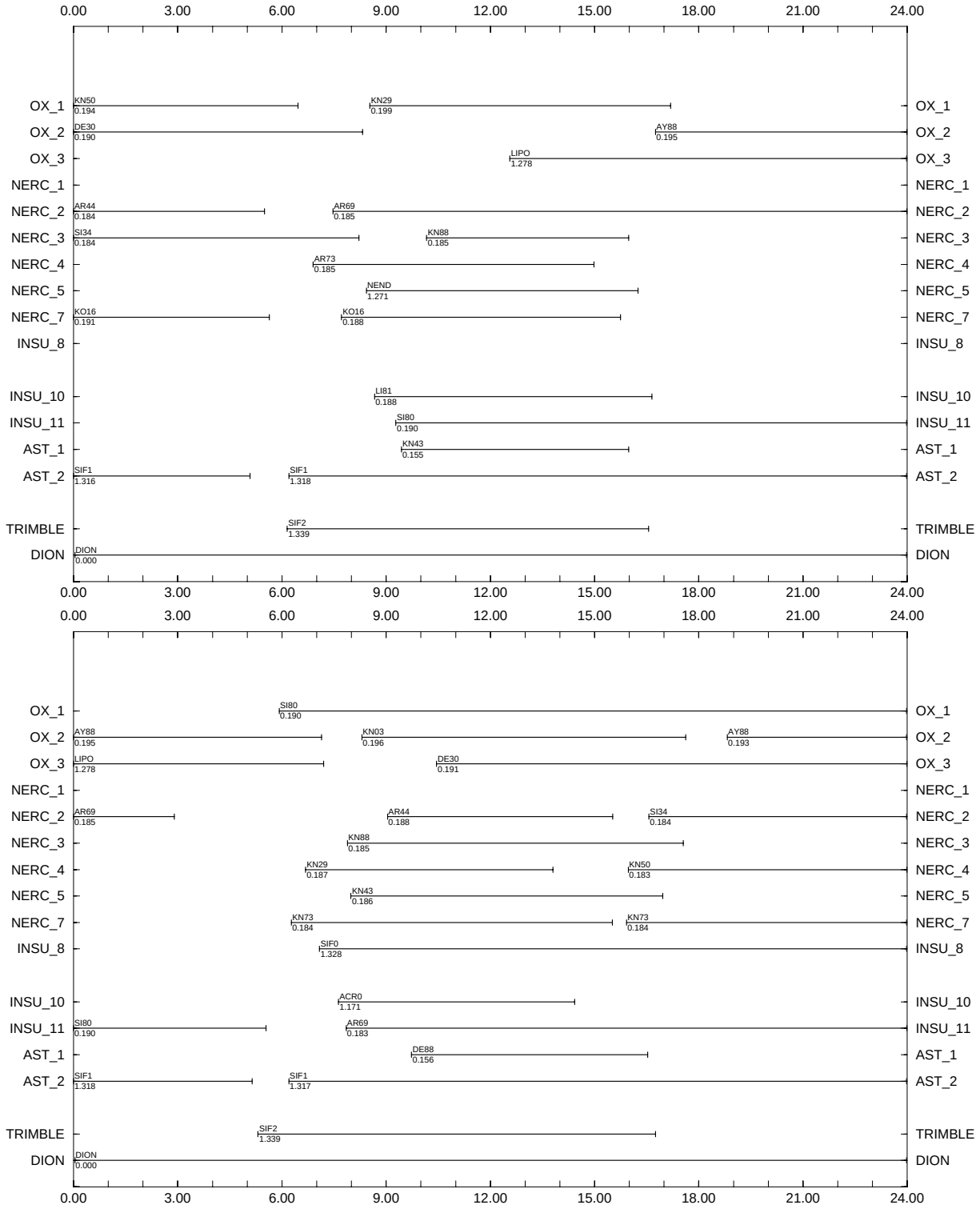

Figure B.6: September 1995 Grevena site occupations, days 269 (above) and 270 (below).

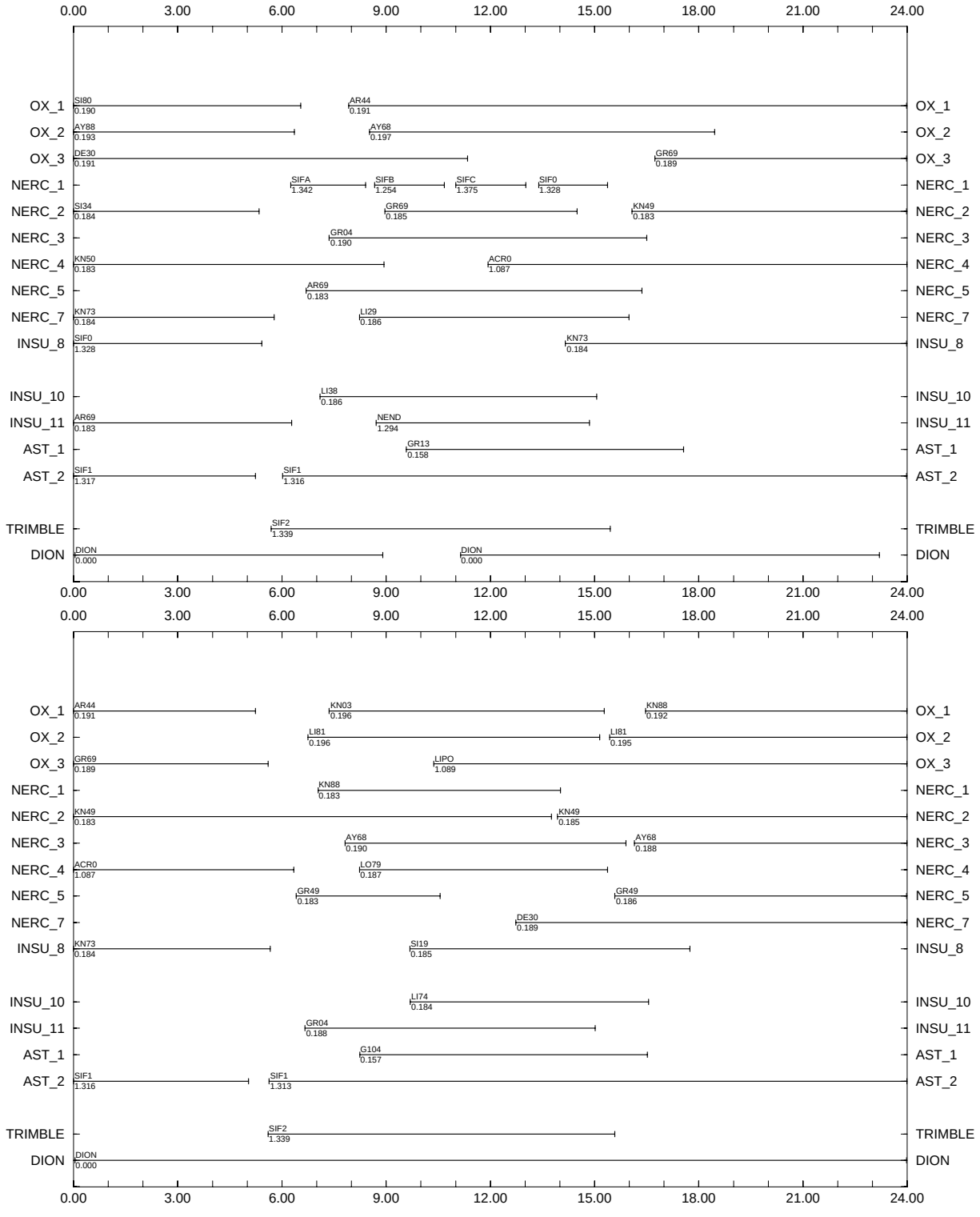

Figure B.7: September 1995 Grevena site occupations, days 271 (above) and 272 (below).

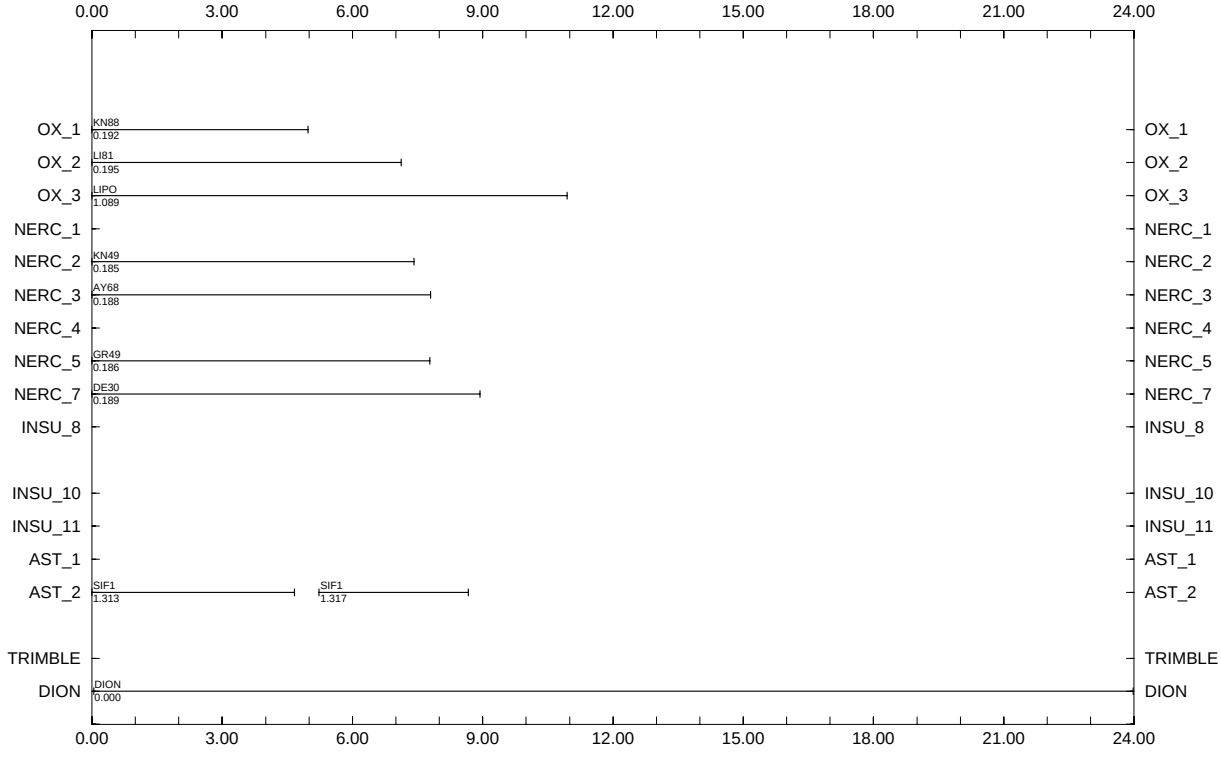

Figure B.8: September 1995 Grevena site occupations, day 273.

Appendix C

C source code

This Appendix describes the various software packages and programs that are referred to in this thesis. They can be considered in two categories:

Firstly, Table C.1 shows the name, authorship and functionality of all programs. Not all programs in this table have been named in this thesis; some others are included for completeness. Where a program forms part of the main Bernese 3.4 GPS Processing Software package (Rothacher *et al.*, 1993), authorship is simply listed as 'Bernese'.

Secondly, this Appendix contains the source code for programs that I have written in C to analyse geodetic measurements. All programs have been compiled with Sun C or GNU C and use standard library routines from the Sun C or GNU C packages. Additional subroutines used by the programs come from Numerical Recipes in C (Press *et al.*, 1992, second edition).

Programs are listed with the main routine first, then the subroutines in alphabetical order, then the input file. Each unit of code or input file is delimited by a line consisting entirely of asterisks (which are not part of the code or input file). The source code of the NR routines is not reproduced here except in cases where the routines have been modified (beyond conversion to double precision arithmetic throughout) for use with my programs.

C.1 `velsmooth`

`velsmooth` is a C program that smooths the horizontal velocities of survey points by estimating a weighted linear fit to the time series of coordinate observations at each site. Because the reference frame of the relative coordinates derived from GPS observations can have errors in its realisation at each epoch, it is

Program	Author	Function
`ashtorin`	Ashtech Ltd.	Convert raw GPS observations to RINEX format
`rxobv3`	Bernese	Transfer RINEX observations to Bernese format
`rxnbv3`	Bernese	Transfer RINEX satellite orbit data to Bernese format
`codchk`	Bernese	Identify outliers in code pseudoranges
`codspp`	Bernese	Calculate coordinates from code pseudoranges and model receiver clock
`satclk`	Bernese	Transfer satellite clock data to Bernese format
`pretab`	Bernese	Transfer satellite precise orbit data to Bernese format
`defstd`	Bernese	Generate orbit model from orbit data
`updstd`	Bernese	Update orbit model from fiducial improvements
`stddif`	Bernese	Display differences in orbit models
`sngdif`	Bernese	Form station-to-station single difference phase observables
`mauprp`	Bernese	Clean cycle slips and outliers from station-to-station single difference phase observations
`ionest`	Bernese	Generate ionospheric TEC model from code or phase zero-difference observable
`gpsest`	Bernese	Form double-difference phase observables; estimate site coordinates, troposphere delays, ambiguity parameters and orbit parameters
`gt`	UNAVCO	Display double-difference phase residuals; permit manual cycle slip and outlier removal
`compar`	Bernese	Combine daily network coordinate solutions
`L3D`	Newcastle Univ.	Combine daily network coordinate solutions, display statistics
`velsmooth`	PJC	Estimate smooth velocities of sites and epoch network offsets / rotations from geodetic coordinates
`svdvel2`	PJC	Estimate polynomial velocity fields from geodetic coordinates
`polystr2`	PJC	Estimate uniform strain in polygons from geodetic coordinates
`okinv`	PJC	Estimate earthquake mechanisms from geodetic displacements

Table C.1: Program authorship and functionality, for programs referred to in this thesis. 'Bernese' refers to Rothacher *et al.* (1993)

possible to estimate global network translations and rotations at any epoch under the assumption that the site velocities are actually smooth with time (Section 3.1). The best-fitting set of translations and rotations are those that minimise the penalty function (usually weighted L_2 norm) calculated from observed minus computed site coordinates at each epoch, for sites with three or more epochs of observation. At least one epoch must be fixed, or the problem is underdetermined. Likewise, a second epoch must be fixed, or the velocities of one site (for translations) or two sites (if rotations are also estimated) must be fixed.

Estimation of a network rotation introduces non-linearity to the problem, so the downhill simplex algorithm (Nelder & Mead, 1965) is used to invert for the network translations and rotations, with 'Monte-Carlo' randomly-selected starting parameters (see Section C.4 for further discussion). The full variance-covariance matrices of the coordinates can be used to weight this part of the inversion. The site velocities and zero-epoch coordinates (*i.e.* the slope and intercept of the best weighted linear fit to the time series of coordinates) are estimated at each inversion step using standard linear regression formulæ, with variance weighting only. Even if a site velocity is fixed, the zero-epoch coordinate is still estimated as a nuisance parameter.

`velsmooth` has been compiled and run on Sun workstations using both Sun C and GNU C. For 5 epochs of data, with 64 sites (23 sites with 3 or more occupations), the time per individual simplex inversion is 2.5 s on a SPARC 10. Execution time is subject to a large overhead: the time taken to read in the variance-covariance matrix files (50 s in the above example).

```
*******************************************************************************
/* velsmooth.c
      program to take a series of coordinate files (with or without covariances)
   and solve for network translations and infinitesimal rotations at each epoch
   such that the sites have the smoothest possible velocities (as defined by
   the L1, L2 or Linf norm of the residuals).  At least two epochs must be fixed
   otherwise the problem is underdetermined, and more than one epoch may be
   fixed.  Also, station velocities may be constrained to a priori values.  A
   downhill simplex algorithm (from NR) is used to perform the non-linear
   inversion.

   Version 2 only uses the simplex to invert for the epoch rotation/translation
   while the site velocities and zero-offsets are solved direct site-by-site as
   the penalty function for each simplex vertex is calculated.
   This implicitly assumes that the site velocities are uncorrelated.

   Pete Clarke, June 1996
*/

#include <math.h>
#include <stdio.h>
#include <string.h>
#include "nr.h"
#include "nrutil.h"

#include "globdef.c"
#include "global.c"


main(argc,argv)

int argc;
char **argv;

{
        char inpname[NAMELEN];                  /* input file name */
        char outname[NAMELEN];                  /* output file name */
        char crdnames[MAXEPO+1][NAMELEN];       /* coordinate file names */
        char covarnames[MAXEPO+1][NAMELEN];     /* vcm file names */
        char residname[NAMELEN];                /* residuals file name */
        char velname[NAMELEN];                  /* smoothed velocity file name */
        char corname[NAMELEN];                  /* translated epoch-to-epoch velocity
file name */
        char sitename[MAXSITE+1][NAMELEN];      /* site names */
        double *excoord,*eycoord;               /* lon, lat of sites */
        double **params;                        /* array for simplex */
        double ***cvdisp;                       /* coordinate vcm */
        double det;                             /* temp for regression calcs */
        double sigx,sigy,correl;                /* temps for errors in residual file
*/
        double finalpen,bestpen;                /* final chisq, best so far */
        double *penalty;                        /* chisq for each vertex of simplex
*/
        double *best_dU,*best_dV,*best_rot;     /* best translation params so far */
        double *best_x0,*best_y0;               /* best site zero-epoch positions so
far */
        double *best_Vx,*best_Vy;               /* best site velocities so far */
        double *sig_Vx, *sig_Vy;                /* a posteriori errors on velocities
*/
        register int i,j,k;                     /* counters */
        int ii,jj;                              /* more counters! */
        int nobs;                               /* number of constraining obs */
        int nfunks,nfunks_tot;                  /* no. of penalty evaluations so far
and in total */
        int niter,niter_tot;                    /* no. of simplex moves so far and in
total */
        int imonte;                             /* counter for Monte-Carlo restarts
*/
        int nmonte;                             /* no. of M-C restarts to be done */
        int usevcm;                             /* 1=use full vcm; 0=use diagonal */
        int wgterr;                             /* flag of problem in inversion of
vcm */
        long seedval;                           /* seed for random numbers */
        FILE *outfile,*resfile;                 /* file handles for output, residuals
files */
        FILE *velfile,*corfile;                 /* file handles for smoothed and
epoch-to-epoch velocity files */

/* reserve memory for global and local arrays */
        globres();
        excoord = dvector(1,MAXSITE);
        eycoord = dvector(1,MAXSITE);
        best_dU = dvector(1,MAXEPO);
        best_dV = dvector(1,MAXEPO);
        best_rot = dvector(1,MAXEPO);
        best_x0 = dvector(1,MAXSITE);
        best_y0 = dvector(1,MAXSITE);
        best_Vx = dvector(1,MAXSITE);
        best_Vy = dvector(1,MAXSITE);
        sig_Vx = dvector(1,MAXSITE);
        sig_Vy = dvector(1,MAXSITE);
        penalty = dvector(1,MAXSITE*4+MAXEPO*3);
        params = dmatrix(1,MAXEPO*3+MAXSITE*4+1,1,MAXEPO*3+MAXSITE*4);
        cvdisp = d3tensor(1,MAXEPO,1,MAXSITE*NDIM,1,MAXSITE*NDIM);
        occup = imatrix(0,MAXEPO,0,MAXSITE);


/* print title */
        printf("\nEstimation of uniform velocities for more than two epochs\n");
        printf("-------------------------------------------------------------------
-------\n\n");


/* start clock, so we can use it as seed for random number generator later */
        seedval = clock();


/* read input file, and coord files specified therein */
        if (argc == 2) strcpy(inpname,argv[1]);
        else strcpy(inpname,"velsmooth.inp");
        readinps(inpname, crdnames,covarnames, outname, velname, residname, corname,
excoord,eycoord, xdisp,ydisp, sitename,&nobs, &usevcm,&nmonte);


/* load covariance matrices or set to identity matrix if "UNIT" entered. */
        getcovs(covarnames,sitename,cvdisp,pwgt);
        if ((usevcm == 0) && (covtype == 3)) {
                printf("\n### using only diagonal elements of weight matrices\n");
                covtype = 2;
        }


/* set up sums of times of observations (weighted/unweighted) for speed later */
        for(j=1; j<=nsites; j++) {
                sum_x[j] = sumt_x[j] = sumtt_x[j] = 0.0;
                sum_y[j] = sumt_y[j] = sumtt_y[j] = 0.0;
                for(k=1; k<=nepochs; k++) if (occup[k][j] == 1) {
                        sum_x[j] += pwgt[k][j][j];
                        sum_y[j] += pwgt[k][j+nsites][j+nsites];
                        sumt_x[j] += (epoch[k] - ZEPOCH) * pwgt[k][j][j];
                        sumt_y[j] += (epoch[k] - ZEPOCH) *
```

```
pwgt[k][j+nsites][j+nsites];
                        sumtt_x[j] += (epoch[k] - ZEPOCH) * (epoch[k] - ZEPOCH) *
pwgt[k][j][j];
                        sumtt_y[j] += (epoch[k] - ZEPOCH) * (epoch[k] - ZEPOCH) *
pwgt[k][j+nsites][j+nsites];
                }
        }


/* determine centroid of network and convert site coords to xy (km) */
        ell2xy(excoord,eycoord,xcoord,ycoord,nsites,1);
        printf("\nCentroid of network at %9.4f , %9.4f\n", orig_x,orig_y);


/* get seed for random number generator */
        seedval = clock();
        srand48(seedval);


/* loop over Monte-Carlo restarts */
        if (nfloat > 0) printf("\nInverting for %d of %d parameter(s) ...\n",
nfloat,nparams);
        else {
                printf("\nNo inversion performed.\n");
                nmonte = 0;
        }
        niter_tot = 0; nfunks_tot = 0; bestpen = BIGNUM;
        imonte = 1;
        do {


        /* load up parameters into arrays for simplex routine */
                makesimp(params,nmonte);


        /* evaluate penalty function at initial vertices of simplex */
                for(j=1; j<=nfloat+1; j++)
                        penalty[j] = penfunc(params[j]);


        /* do the inversion! */
                if (nfloat > 0) {
                        amoeba(params,penalty,nfloat,FTOL,penfunc, &nfunks, &niter);
                        niter_tot += niter;
                        nfunks_tot += nfunks;
                }


        /* convert simplex params back to ordinary stuff,
           compute final penalty */
                finalpen = penfunc(params[nfloat+1]);


        /* if parameters are best so far (and we collapsed the simplex
           successfully), store them for later */
                if ((finalpen < bestpen) && (nfunks < NMAX)) {
                        bestpen = finalpen;
                        for(j=1,jj=1; j<=nepochs; j++) {
                                if (inv_dU[j] == 1)
                                        best_dU[j] = params[nfloat+1][(jj++)];
                                else best_dU[j] = dU[j];
                                if (inv_dV[j] == 1)
                                        best_dV[j] = params[nfloat+1][(jj++)];
                                else best_dV[j] = dV[j];
                                if (inv_rot[j] == 1)
                                        best_rot[j] = params[nfloat+1][(jj++)];
                                else best_rot[j] = rot[j];
                        }
                        for(i=1,ii=jj; i<=nsites; i++) {
                                best_x0[i] = xz[i];
                                best_y0[i] = yz[i];
                                best_Vx[i] = Vx[i];
                                best_Vy[i] = Vy[i];
                        }
                }


        /* end of loop over Monte-Carlo restarts */
                imonte++;
        } while (imonte <= nmonte);


/* recalculate penalty and residuals for best solution */
        for(j=1,jj=1; j<=nepochs; j++) {
                if (inv_dU[j] == 1) params[1][(jj++)] = dU[j] = best_dU[j];
                else dU[j] = best_dU[j];
                if (inv_dV[j] == 1) params[1][(jj++)] = dV[j] = best_dV[j];
                else dV[j] = best_dV[j];
                if (inv_rot[j] == 1) params[1][(jj++)] = rot[j] = best_rot[j];
                else rot[j] = best_rot[j];
        }
        for(i=1; i<=nsites; i++) {
                xz[i] = best_x0[i];
                yz[i] = best_y0[i];
                Vx[i] = best_Vx[i];
                Vy[i] = best_Vy[i];
        }
        finalpen = penfunc(params[1]);


/* compute errors in smoothed velocities */
        for(i=1; i<=nsites; i++) {
                if (occup[0][i] < 2) continue;
                det = sum_x[i]*sumtt_x[i] - sumt_x[i]*sumt_x[i];
                if (inv_Vx[i] == 1) sig_Vx[i] = sum_x[i]/det;
                else sig_Vx[i] = 0.0;
                det = sum_y[i]*sumtt_y[i] - sumt_y[i]*sumt_y[i];
                if (inv_Vy[i] == 1) sig_Vy[i] = sum_y[i]/det;
                else sig_Vy[i] = 0.0;
        }


/* output helmert parameters and statistics */
        printf("\nEpoch           dU      inv     dV      inv     rot     inv\n");
        printf("----------      ****.** *       ****.** *       ****.** *\n");
        for(j=1; j<=nepochs; j++) {
                printf("%7.21f         %7.21f %1d       %7.21f %1d       %7.21f
%1d\n", epoch[j], dU[j],inv_dU[j], dV[j],inv_dV[j], rot[j],inv_rot[j]);
        }
        printf("\n");

        printf("%9d                                          no. of constraining obs\n",
nobs);
        printf("  %11.4le                                      total penalty\n",
finalpen);
        printf("  %11.4le                                      mean penalty\n",
finalpen/((double)(nobs)));
        if (norm == 2) {
                printf("  %11.4le                                          root mean
penalty\n", sqrt(finalpen/((double)(nobs))));
                if (covtype != 1) {
                        covtype = 1;
```

```
                        finalpen = penfunc(params[1]);
                        printf("  %11.4le                                    root mean
residual\n", sqrt(finalpen/((double)(nobs))));
                }
        }
        printf("\n");
        printf("%9d                                        no. of Monte-Carlo
restarts\n", nmonte);
        printf("%9d                                        no. of simplex moves\n",
niter_tot);
        printf("%9d                                        no. of function calls\n",
nfunks_tot);
        printf("\n\n");


/* create output file for parameters, if required */
        outfile = NULL;
        if (strcmp(outname,"NONE") != 0) {
                outfile = fopen(outname,"w");
                if (outfile == NULL) {
                        printf("\n*** error opening output file \"%s\"\n", outname);
                        exit();
                }
                fprintf(outfile,"\nEpoch           dU      inv     dV      inv
rot     inv\n");
                fprintf(outfile,"----------      ****.** *       ****.** *
****.** *\n");
                for(j=1; j<=nepochs; j++) {
                        fprintf(outfile,"%7.2lf         %7.2lf %1d       %7.2lf
%1d       %7.2lf %1d\n", epoch[j], dU[j],inv_dU[j], dV[j],inv_dV[j],
rot[j],inv_rot[j]);
                }
                fprintf(outfile,"\n");

                fprintf(outfile,"%9d                                       no. of
constraining obs\n", nobs);
                fprintf(outfile,"  %11.4le                                   total
penalty\n", finalpen);
                fprintf(outfile,"  %11.4le                                   mean
penalty\n", finalpen/((double)(nobs)));
                if (norm == 2) {
                        fprintf(outfile,"  %11.4le
root mean penalty\n", sqrt(finalpen/((double)(nobs))));
                        if (covtype != 1) {
                                covtype = 1;
                                finalpen = penfunc(params[1]);
                                fprintf(outfile,"  %11.4le
root mean residual\n", sqrt(finalpen/((double)(nobs))));
                        }
                }
                fprintf(outfile,"\n");
                fprintf(outfile,"%9d                                       no. of
Monte-Carlo restarts\n", nmonte);
                fprintf(outfile,"%9d                                       no. of
simplex moves\n", niter_tot);
                fprintf(outfile,"%9d                                       no. of
function calls\n", nfunks_tot);
                fclose(outfile);
        }


/* create smoothed velocity file if required */
        velfile = NULL;
        if (strcmp(velname,"NONE") != 0) {
                velfile = fopen(velname,"w");
                if (velfile == NULL) {
                        printf("\n*** error opening velocity output file \"%s\"\n",
velname);
                        exit();
                }
                for(i=1; i<=nsites; i++) {
                        if (occup[0][i] > 1) fprintf(velfile,"%10.6lf %10.6lf  %8.4lf
%8.4lf  %7.4lf %7.4lf 0 %s\n", excoord[i],eycoord[i], best_Vx[i]/1000.0,
best_Vy[i]/1000.0, sig_Vx[i]/1000.0,sig_Vy[i]/1000.0, sitename[i]);
                }
                fclose(velfile);
        }


/* create residuals file if required */
        resfile = NULL;
        if (strcmp(residname,"NONE") != 0) {
                resfile = fopen(residname,"w");
                if (resfile == NULL) {
                        printf("\n*** error opening residual output file \"%s\"\n",
residname);
                        exit();
                }
                for(k=1; k<=nepochs; k++) {
                        fprintf(resfile,"# Residuals at epoch %7.2lf\n", epoch[k]);
                        for(i=1; i<=nsites; i++) {
                                if ((occup[k][i] == 0) || (occup[0][i] <= 2))
continue;
                                if (strcmp(covarnames[k],"UNIT") == 0)
                                        sigx = sigy = 0.0;
                                else {
                                        sigx = sqrt(cvdisp[k][i][i]);
                                        sigy = sqrt(cvdisp[k][nsites+i][nsites+i]);
                                }
                                correl = 0.0;
                                if ((sigx != 0.0) && (sigy != 0.0))
                                        correl = cvdisp[k][i][nsites+i] /
(sigx*sigy);
                                fprintf(resfile,"%10.6lf %10.6lf  %8.4lf %8.4lf
%6.4lf %6.4lf %6.4lf  %s\n", excoord[i], eycoord[i],
(xdisp[k][i]-comp_xdisp[k][i])/1000.0, (ydisp[k][i]-comp_ydisp[k][i])/1000.0,
sigx/1000.0,sigy/1000.0,correl, sitename[i]);
                        }
                }
                fclose(resfile);
        }


/* create corrected epoch-to-epoch velocities file if required */
        corfile = NULL;
        if (strcmp(corname,"NONE") != 0) {
                corfile = fopen(corname,"w");
                if (corfile == NULL) {
                        printf("\n*** error opening corrected velocity output file
\"%s\"\n", corname);
                        exit();
                }
                for(j=1; j<nepochs; j++) for(k=j+1; k<=nepochs; k++) {
                        fprintf(corfile,"# Corrected displacements from epoch %7.2lf
to epoch %7.2lf\n", epoch[j],epoch[k]);
                        for(i=1; i<=nsites; i++) {
                                if ((occup[j][i] == 1) && (occup[k][i] == 1))
fprintf(corfile,"%10.6lf %10.6lf  %8.4lf %8.4lf  0 0 0 %s\n", excoord[i],eycoord[i],
((xdisp[k][i]-(dU[k]-(ycoord[i]*rot[k])/1.0e3)) -
(xdisp[j][i]-(dU[j]-(ycoord[i]*rot[j])/1.0e3)))/1000.0,
((ydisp[k][i]-(dV[k]+(xcoord[i]*rot[k])/1.0e3)) -
```

```
(ydisp[j][i]-(dV[j]+(xcoord[i]*rot[j])/1.0e3)))/1000.0, sitename[i]);
                        }
                }
                fclose(corfile);
        }


/* free local and global arrays */
        globfree();
        free_dvector(excoord,1,MAXSITE);
        free_dvector(eycoord,1,MAXSITE);
        free_dvector(best_dU,1,MAXEPO);
        free_dvector(best_dV,1,MAXEPO);
        free_dvector(best_rot,1,MAXEPO);
        free_dvector(best_x0,1,MAXSITE);
        free_dvector(best_y0,1,MAXSITE);
        free_dvector(best_Vx,1,MAXSITE);
        free_dvector(best_Vy,1,MAXSITE);
        free_dvector(sig_Vx,1,MAXSITE);
        free_dvector(sig_Vy,1,MAXSITE);
        free_dvector(penalty,1,MAXSITE*4+MAXEPO*3);
        free_dmatrix(params,1,MAXEPO*3+MAXSITE*4+1,1,MAXEPO*3+MAXSITE*4);
        free_d3tensor(cvdisp,1,MAXEPO,1,MAXSITE*NDIM,1,MAXSITE*NDIM);
        free_imatrix(occup,0,MAXEPO,0,MAXSITE);
}

#include "globend.c"
*******************************************************************************


*******************************************************************************
/* amoeba.c
  Taken directly from NR in C, section 10.4
/*
*******************************************************************************


*******************************************************************************
/* function to compute epoch site displacements, for velsmooth program.
   makes use of the global variables sum_x,sum_y, sumt_x,sumt_y, sumtt_x,sumtt_y
   which have the weighted sums of observations computed beforehand to save time
   (see NR, section 15.2 for details).

   pjc jun '96
*/

#include <stdio.h>
#include <math.h>
#include "nrutil.h"

#include "global.c"


void comp_disps(param_vals)

double *param_vals;

{
        int i,j;
        int jj;
        double xj,yj;
        double sumx,sumy,sumtx,sumty;
        double det;

/* unwrap the helmert parameters from the vertex of the simplex. */
        for(j=1,jj=1; j<=nepochs; j++) {
                if (inv_dU[j] == 1) dU[j] = param_vals[jj++];
                if (inv_dV[j] == 1) dV[j] = param_vals[jj++];
                if (inv_rot[j] == 1) rot[j] = param_vals[jj++];
        }

/* loop over all sites with more than 2 ooccupations */
        for(i=1; i<=nsites; i++) {
                if (occup[0][i] < 2) continue;

        /* compute sums of obs given network translations */
                sumx = sumy = sumtx = sumty = 0.0;
                for(j=1; j<=nepochs; j++) {
                        if (occup[j][i] ==0) continue;
                        xj = xdisp[j][i] - (dU[j] - (ycoord[i]*rot[j])/1.0e3);
                        yj = ydisp[j][i] - (dV[j] + (xcoord[i]*rot[j])/1.0e3);
                        sumx += xj * pwgt[j][i][i];
                        sumy += yj * pwgt[j][i+nsites][i+nsites];
                        sumtx += xj * (epoch[j]-ZEPOCH) * pwgt[j][i][i];
                        sumty += yj * (epoch[j]-ZEPOCH) *
pwgt[j][i+nsites][i+nsites];
                }

        /* solve for zero-epoch offset (and velocity unless it is constrained) */
                det = sum_x[i]*sumtt_x[i] - sumt_x[i]*sumt_x[i];
                if (inv_Vx[i] == 1) {
                        Vx[i] = (sum_x[i]*sumtx - sumt_x[i]*sumx) / det;
                        xz[i] = (sumtt_x[i]*sumx - sumt_x[i]*sumtx) / det;
                }
                else xz[i] = (sumx - Vx[i]*sumt_x[i]) / sum_x[i];

                det = sum_y[i]*sumtt_y[i] - sumt_y[i]*sumt_y[i];
                if (inv_Vy[i] == 1) {
                        Vy[i] = (sum_y[i]*sumty - sumt_y[i]*sumy) / det;
                        yz[i] = (sumtt_y[i]*sumy - sumt_y[i]*sumty) / det;
                }
                else yz[i] = (sumy - Vy[i]*sumt_y[i]) / sum_y[i];
        }

/* compute model displacements */
        for(i=1; i<=nsites; i++) {
                for(j=1; j<=nepochs; j++) {
                        comp_xdisp[j][i] = dU[j] - (ycoord[i]*rot[j])/1.0e3 + xz[i]
+ Vx[i]*(epoch[j]-ZEPOCH);
                        comp_ydisp[j][i] = dV[j] + (xcoord[i]*rot[j])/1.0e3 + yz[i]
+ Vy[i]*(epoch[j]-ZEPOCH);
                }
        }
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to convert site coords in the arrays xx[],yy[] into
  x-y coords in km in the arrays x[],y[].
  Conversion from lat/lon to km must be done. The centroid of the network is
  returned in the global variables orig_x,orig_y and all coords are relative
  to this.

   pjc jun '95
*/

#include <stdio.h>
#include <math.h>
#include "nrutil.h"

#include "global.c"
```

```
void ell2xy(xx,yy,x,y,ncoords,change_orig)

double *x,*y,*xx,*yy;
int ncoords,change_orig;

{
        int i;
        double rprimev,rmerid,tmp;

/* on first call to subroutine, change_orig==1, so move origin */
        if (change_orig == 1) {
                orig_x = 0.0; orig_y = 0.0;
                for(i=1; i<=ncoords; i++) {
                        orig_x += xx[i];
                        orig_y += yy[i];
                }
                orig_x /= (double)ncoords; orig_y /= (double)ncoords;
        }

/* shift the coords to the new origin and convert to km by multiplying by
   appropriate radius of curvature */
        tmp = 1 - pow((ECCEN*sin(orig_y*DEG2RAD)),2.0);
        rprimev = RADIUS / sqrt(tmp);
        rmerid = RADIUS*(1 - ECCEN*ECCEN) / (tmp*sqrt(tmp));

        for(i=1; i<=ncoords; i++) {
                x[i] = (xx[i]-orig_x) * (rprimev/1000.0)*cos(orig_y*DEG2RAD) *
DEG2RAD;
                y[i] = (yy[i]-orig_y) * (rmerid/1000.0) * DEG2RAD;
        }
}


#include "globend.c"
*******************************************************************************

*******************************************************************************
/* subroutine to find the station number of a named station (ie its position in
   the array namelist[1..nsites]), or return -999 if station not found.

   pjc jul '94
*/

#include <string.h>

#define MAXSITE 100     /* max number of sites */


int findstnum(name,namelist,nsites)

char name[];
char namelist[][32];
int nsites;

{
        int i;
        int found=0;

        for(i=1;found<1;i++) {
                if (i > nsites) {
                        found=2;
                }
                if (strcmp(name,namelist[i]) == 0) {
                        found=1;
                }
        }
        if (found == 1) {
                return(i-1);
        } else {
                return(-999);
        }
}


#undef MAXSITE
*******************************************************************************

*******************************************************************************
/* subroutine to read in a covariance matrix from a file in .CEN format (which
   is equivalent to the Bernese 3.4 format, but having had a similarity
   transform applied so that the components are in the coordinate system of
   east, north, up appropriate at the centroid of the network).
   Covariances are stored in units of mm.
   Modified dec '95 for multiple covar files for polystr2.
   Modified further jun '96 to invert vcm's for velsmooth, using
   LU-decomposition routines from NR (sections 2.3, 2.5).

   pjc aug '95
*/

#include <string.h>
#include <stdio.h>
#include <math.h>
#include "nrutil.h"
#include "nr.h"

#define SIGSQMIN 4.0            /* default std error of a fixed coord is 2mm */
#define SIGSQTOL 1.0e-2         /* 'zero' variance of a fixed coordinate */

#include "global.c"


void getcovs(covarnames,sitename,cvmatrix,pwgt)

char covarnames[][NAMELEN],sitename[][NAMELEN];
double ***cvmatrix,***pwgt;

{
        FILE *covarfile;
        int status;
        int h,i,j,k,ii,jj,n;
        int stnum1,stnum2;
        int nepsites;
        int wgterr = 0;
        char st1[NAMELEN],st2[NAMELEN];
        char neu1[3],neu2[3];
        char dummystring[255];
        double elem,unitwgt;
        double **evcm,**evcmcopy,**wgttmp;
        double det,*indx,*tmpvec,*tmpcopy;
        double rtmp;

        indx = dvector(1,MAXSITE*NDIM);
        tmpvec = dvector(1,MAXSITE*NDIM);
        tmpcopy = dvector(1,MAXSITE*NDIM);
        evcm = dmatrix(1,MAXSITE*NDIM,1,MAXSITE*NDIM);
        evcmcopy = dmatrix(1,MAXSITE*NDIM,1,MAXSITE*NDIM);
        wgttmp = dmatrix(1,MAXSITE*NDIM,1,MAXSITE*NDIM);


/* loop over all epochs */
        printf("\n");
        for(k=1; k<=nepochs; k++) {
                printf("Loading covariance submatrix %s ... ", covarnames[k]);

        /* set whole matrix to zero in case of accidents */
```

```
                for(i=1;i<=nsites*NDIM;i++)
                        for(j=1;j<=nsites*NDIM;j++)
                                cvmatrix[k][i][j] = 0.0;
                covtype = 0;

        /* for unit matrix, do not bother inverting (!) */
                if (strcmp(covarnames[k],"UNIT") == 0) {
                        for(i=1; i<=nsites*NDIM; i++) {
                                cvmatrix[k][i][i] = 1.0;
                                pwgt[k][i][i] = 1.0;
                        }
                        covtype = 1;
                }

        /* for real vcm, load it in then invert as necessary */
                else {
                        covtype = 3;
                        covarfile=fopen(covarnames[k],"r");
                        if (covarfile == NULL) {
                                printf("\n*** error opening file:
'%s'\n",covarnames[k]);
                                exit();
                        }
                        status=0;
                        for(i=1;i<=6;i++) fgets(dummystring,255,covarfile);
                        fscanf(covarfile, "RMS OF UNIT WEIGHT: %lf", &unitwgt);
                        unitwgt *= 1.0e3;       /* convert to mm */
                        for(i=1;i<=5;i++) fgets(dummystring,255,covarfile);
                        while (status != EOF) {
                                status = fscanf(covarfile,"%s %s %s %s %lf\n", st1,
neu1, st2, neu2, &elem);
                                if ((status != 0) && (status != EOF)) {
                                        stnum1=findstnum(st1,sitename,nsites);
                                        stnum2=findstnum(st2,sitename,nsites);
                                        i=j=-999;
                                        if (strcmp(neu1,"E") == 0) i=0;
                                        if (strcmp(neu1,"N") == 0) i=nsites;
                                        if (strcmp(neu2,"E") == 0) j=0;
                                        if (strcmp(neu2,"N") == 0) j=nsites;
                                        i+=stnum1; j+=stnum2;
                                        if ((i>0) && (j>0)) {
                                                cvmatrix[k][i][j] = elem *
unitwgt*unitwgt;

                                                cvmatrix[k][j][i] = elem *
unitwgt*unitwgt;
                                        }
                                }
                        }
                        fclose(covarfile);

                /* check covariance matrix for zero variances of fixed sites,
adjust them to SIGSQMIN */
                        for(i=1;i<=nsites;i++) {
                                if (occup[k][i] == 0) continue;
                                if (cvmatrix[k][i][i] < SIGSQTOL) {
                                        printf("\n### Variance of  %s:E increased
from %10.3e to %10.3le",sitename[i],cvmatrix[k][i][i],SIGSQMIN);
                                        cvmatrix[k][i][i] = SIGSQMIN;
                                }
                                if (cvmatrix[k][i+nsites][i+nsites] < SIGSQTOL) {
                                        printf("\n### Variance of  %s:N increased
from %10.3e to %10.3le", sitename[i], cvmatrix[k][i+nsites][i+nsites], SIGSQMIN);
                                        cvmatrix[k][i+nsites][i+nsites] = SIGSQMIN;
                                }
                        }

                /* first copy epoch covariance matrix */
                        nepsites = occup[k][0];
                        for(i=1,ii=1; i<=nsites; i++) {
                                if (occup[k][i] == 1) {
                                        for(j=1,jj=1; j<=nsites; j++) {
                                                if (occup[k][j] == 1) {
                                                        evcmcopy[ii][jj] =
evcm[ii][jj] = cvmatrix[k][i][j];
                                                        evcmcopy[ii+nepsites][jj] =
evcm[ii+nepsites][jj] = cvmatrix[k][i+nsites][j];
                                                        evcmcopy[ii][jj+nepsites] =
evcm[ii][jj+nepsites] = cvmatrix[k][i][j+nsites];
                                                        evcmcopy[ii+nepsites]
[jj+nepsites] = evcm[ii+nepsites][jj+nepsites] = cvmatrix[k][i+nsites][j+nsites];
                                                        jj++;
                                                }
                                        }
                                        ii++;
                                }
                        }
                /* now invert by LU-decomposition and back-substitution, with
iterative improvement */
                        ludcmp(evcm,NDIM*nepsites,indx,&det);
                        for(j=1; j<=NDIM*nepsites; j++) {
                                for(i=1; i<=NDIM*nepsites; i++)
                                        tmpcopy[i] = tmpvec[i] = 0.0;
                                tmpcopy[j] = tmpvec[j] = 1.0;
                                lubksb(evcm,NDIM*nepsites,indx,tmpvec);
                                mprove(evcmcopy,evcm, NDIM*nepsites, indx, tmpcopy,
tmpvec);
                                for(i=1; i<=NDIM*nepsites; i++) wgttmp[i][j] =
tmpvec[i];
                        }
                /* check success by multiplying inverted matrix with original */
                        for(i=1; i<=NDIM*nepsites; i++) {
                                for(j=1; j<=NDIM*nepsites; j++) {
                                        rtmp=0.0;
                                        for(h=1; h<=NDIM*nepsites; h++)
                                                rtmp += evcmcopy[i][h] *
wgttmp[h][j];
                                        if ((i==j) && (fabs(rtmp-1.0) > INVTOL))
                                                wgterr = 1;
                                        if ((i!=j) && (fabs(rtmp) > INVTOL))
                                                wgterr = 1;
                                }
                        }
                /* copy into array pwgt[][][] for all epochs */
                        for(i=1,ii=1; i<=nsites; i++) {
                                if (occup[k][i] == 1) {
                                        for(j=1,jj=1; j<=nsites; j++) {
                                                if (occup[k][j] == 1) {
                                                        pwgt[k][i][j] =
wgttmp[ii][jj];
                                                        pwgt[k][i+nsites][j] =
wgttmp[ii+nepsites][jj];
                                                        pwgt[k][i][j+nsites] =
wgttmp[ii][jj+nepsites];
                                                        pwgt[k][i+nsites][j+nsites]
= wgttmp[ii+nepsites][jj+nepsites];
                                                        jj++;
                                                }
                                                else {
                                                        pwgt[k][i][j] = 0.0;
                                                        pwgt[k][i+nsites][j] = 0.0;
                                                        pwgt[k][i][j+nsites] = 0.0;
```

```
                                                pwgt[k][i+nsites][j+nsites]
= 0.0;
                                        }
                                }
                                ii++;
                        }
                        else for(j=1; j<=NDIM*nsites; j++) {
                                pwgt[k][i][j] = 0.0;
                                pwgt[k][i+nsites][j] = 0.0;
                        }
                }
        }
        printf(" ... done\n");
    }

    free_dvector(indx,1,MAXSITE*NDIM);
    free_dvector(tmpvec,1,MAXSITE*NDIM);
    free_dvector(tmpcopy,1,MAXSITE*NDIM);
    free_dmatrix(evcm,1,MAXSITE*NDIM,1,MAXSITE*NDIM);
    free_dmatrix(evcmcopy,1,MAXSITE*NDIM,1,MAXSITE*NDIM);
    free_dmatrix(wgttmp,1,MAXSITE*NDIM,1,MAXSITE*NDIM);
}


#undef SIGSQMIN
#undef SIGSQTOL


#include "globend.c"
*******************************************************************************

*******************************************************************************
/* subroutine to get coordinates and sitenames from B3.4 .CRD files, then make
   displacements (in mm) for each epoch in local reference frame relative to
   coordinates at the first epoch encountered.

   pjc dec '95
*/

#include<math.h>
#include<stdio.h>
#include<string.h>

#include "nr.h"
#include "nrutil.h"

#include "global.c"


void getcrds(crdnames, excoord,eycoord, xdisp,ydisp, sitename, nobs)

char crdnames[][NAMELEN],sitename[][NAMELEN];
double *excoord,*eycoord;
double **xdisp,**ydisp;
int *nobs;

{
    FILE *crdfile;
    char dummystring[255],dummy[NAMELEN],*flag,crdflag[2];
    int i,j,n,site,status;
    int n1sites,n2sites,n3sites;
    double cx,cy,cz;
    double *cx0,*cy0,*cz0;
    double lat,lon,hgt;
    double dx,dxy,dy,dz;


    cx0 = dvector(1,MAXSITE);
    cy0 = dvector(1,MAXSITE);
    cz0 = dvector(1,MAXSITE);


/* set occupation flags to zero at start */
    nsites = 0;
    for(i=0; i<=MAXEPO; i++) for(j=0; j<=MAXSITE; j++) occup[i][j] = 0;

/* loop over all epochs, each of which should have a .CRD file */
    for(i=1; i<=nepochs; i++) {
        crdfile=fopen(crdnames[i],"r");
        if (crdfile == NULL) {
            printf("\n*** error opening coordinate file: '%s'\n",
crdnames[i]);
            exit();
        }
        printf("Reading coordinate file \"%s\" ... ", crdnames[i]);
        for(j=1; j<=6; j++) fgets(dummystring,255,crdfile);

    /* read coordinates from file */
        status=0; n=0;
        while (status != EOF) {
            flag = fgets(dummystring,255,crdfile);
            if (flag == NULL) break;
            strcpy(crdflag,"");
            status = sscanf(dummystring,"%*d %s %lf %lf %lf %s", dummy,
&cx,&cy,&cz, crdflag);
            if (status == EOF) break;
            /* only use coordinates with flag set */
            if (strcmp(crdflag,"") != 0) {
                n++;
                site = findstnum(dummy,sitename,nsites);
                if (site < 0) {
                    nsites++;
                    if (nsites > MAXSITE) {
                        printf("\n*** maximum number of
sites exceeded\n");
                        printf("    number of sites >= %d
\n", nsites);
                        exit();
                    }
                    site = nsites;
                    strcpy(sitename[site],dummy);
                /* origin for each site is first epoch coordinate */
                    cx0[site] = cx;
                    cy0[site] = cy;
                    cz0[site] = cz;
                }
                occup[i][site] = 1;
                occup[0][site] += 1;

            /* convert to ellipsoidal coordinates */
                xyz2llh(cx,cy,cz, &lat,&lon,&hgt);
                excoord[site] = lon / DEG2RAD;
                eycoord[site] = lat / DEG2RAD;

            /* compute displacement relative to site origin */
                dx = cx - cx0[site];
                dy = cy - cy0[site];
                dz = cz - cz0[site];

            /* rotate into local e,n,u coordinate frame */
                dxy = dx*cos(lon) + dy*sin(lon);
                xdisp[i][site] = (dy*cos(lon) - dx*sin(lon)) *
1000.0;
                ydisp[i][site] = (dz*cos(lat) - dxy*sin(lat)) *
1000.0;
```

```
                }
        }

        occup[i][0] = n;
        printf("%d sites found\n", n);
        fclose(crdfile);
}

/* check totals of site occupations */
        printf("Total of %d sites found\n", nsites);
        n1sites = n2sites = n3sites = *nobs = 0;
        for(i=1; i<=nsites; i++) {
                if (occup[0][i] == 1) {
                        n1sites++;
                        for(j=1; j<=nepochs; j++) {
                                occup[j][0] -=occup[j][i];
                                occup[j][i] = 0;
                        }
                }
                if (occup[0][i] == 2) n2sites++;
                if (occup[0][i] >= 3) {
                        n3sites++;
                        *nobs += occup[0][i] * NDIM;
                }
        }
        printf("%d sites with 1 occupation (not used)\n", n1sites);
        printf("%d sites with 2 occupations\n", n2sites);
        printf("%d sites with >=3 occupations\n", n3sites);
        if (n3sites < 3) {
                printf("\n### problem may be underdetermined - fewer than 3 sites
with >=3 epochs\n");
        }


        free_dvector(cx0,1,MAXSITE);
        free_dvector(cy0,1,MAXSITE);
        free_dvector(cz0,1,MAXSITE);
}


#include "globend.c"
*******************************************************************************

*******************************************************************************
/* global.c

  global variables for velsmooth.c and subroutines */


/* get macro defs (array sizes etc) */
#include "globdef.c"


/* function prototypes */
void amoeba();
void comp_disps();
void ell2xy();
int findstnum();
void getcovs();
void getcrds();
void globres();
void globfree();
void lubksb();
void ludcmp();
void makesimp();
double monte();
void mprove();
double monte();
void nrutil();
double penfunc();
double pen_L1dis();
double pen_L2dis();
double pen_Linfdis();
void readinps();
void xy2ell();
void xyz2llh();


/* Global variables */

double orig_x;          /* coordinates of centroid of network, used as origin */
double orig_y;          /* always in lon,lat (degrees) */

double *xcoord,*ycoord; /* coordinates of site, relative to origin, in km */

double **comp_xdisp;    /* computed site displacements at each epoch */
double **comp_ydisp;

double **xdisp,**ydisp; /* actual site displacements at each epoch */

double *epoch;          /* dates of epochs */

double ***pwgt;         /* weight matrix for coordinates, for each epoch */

double *ap_dU, *ap_dV;  /* a priori helmert parameters for each epoch */
double *ap_rot;

double *ap_Vx, *ap_Vy;  /* a priori site velocities for specified sites */

double *dU, *dV, *rot;  /* estimates helmert parameters for each epoch */

double *xz, *yz;        /* estimated zero-epoch offset for each site */

double *Vx, *Vy;        /* estimated velocities for each site */

double *sum_x,*sumt_x;  /* sum of weights, sum of epoch*weight for each site */
double *sumtt_x;        /* sum of epoch*epoch*weight for each site */
double *sum_y,*sumt_y;  /* similarly for y-displacement weights */
double *sumtt_y;

int norm;               /* norm of penalty function (1,2,0=infinity) */

int covtype;            /* vcm type: 1=unit,2=diagonal,3=full */

int *inv_dU,*inv_dV;    /* flags to estimate/fix each helmert parameter */
int *inv_rot;

int *inv_Vx, *inv_Vy;   /* flags to estimate/fix each velocity component */

int **occup;            /* matrix of occupation flags for sites */

int nsites;             /* total no. of sites */

int nepochs;            /* total no. of epochs */

int nparams;            /* no. of parameters to estimate in total (inc velocities */

int nfloat;             /* no. of parameters in simplex (helmert params only) */
*******************************************************************************

*******************************************************************************
/* globdef.c

  macro and structure definitions for velsmooth.c and subroutines */
```

```
/* Constants and array sizes */
#define BIGNUM  1.0e30          /* (almost) largest representable number */
#define DEG2RAD 0.017453293     /* degrees to radians factor */
#define ECCEN   0.081819191     /* eccentricity of GRS-80 ellipsoid */
#define FTOL    1.0e-5          /* tolerance for fractional change in penalty */
#define INVTOL  1.0e-12         /* tolerance for goodness of vcm inversion */
#define MAXEPO  5               /* max no. of epochs in survey */
#define MAXITER 5000            /* max no. of iterations */
#define MAXSITE 100             /* max no. of sites */
#define NAMELEN 32              /* max length of site names */
#define NDIM    2               /* no. of dimensions in problem */
#define NMAX    5000            /* max no. of fn calls permitted in amoeba.c */
#define NSIG    2.0             /* search width (sigmas) for Monte-Carlo */
#define PERTFAC 0.05            /* perturbation factor for simplex */
#define RADIUS  6378137.0       /* radius of GRS-80 ellipsoid */
#define SIGPOS  25.0            /* a priori sigma of a position /mm */
#define SIGROT  1.0             /* a priori sigma of a rotation /microrad */
#define SIGVEL  10.0            /* a priori sigma of a velocity /mm/yr */
#define ZEPOCH  1988.00         /* zero epoch for coordinate diffs */
*******************************************************************************

*******************************************************************************
/* globend.c

   undefine macros used in velsmooth.c and subroutines.
*/

#undef BIGNUM
#undef DEG2RAD
#undef ECCEN
#undef INVTOL
#undef MAXEPO
#undef MAXITER
#undef MAXSITE
#undef MAXVERT
#undef NAMELEN
#undef NDIM
#undef RADIUS
#undef ZEPOCH
*******************************************************************************

*******************************************************************************
/* globfree.c
   subroutine to free memory from global arrays for velsmooth.c.

   pjc jun '96
*/

#include "nrutil.h"

#include "global.c"


void globfree()


{
        free_dvector(xcoord,1,MAXSITE);
        free_dvector(ycoord,1,MAXSITE);
        free_dvector(ap_dU,1,MAXEPO);
        free_dvector(ap_dV,1,MAXEPO);
        free_dvector(ap_rot,1,MAXEPO);
        free_dvector(ap_Vx,1,MAXSITE);
        free_dvector(ap_Vy,1,MAXSITE);
        free_dvector(dU,1,MAXEPO);
        free_dvector(dV,1,MAXEPO);
        free_dvector(rot,1,MAXEPO);
        free_dvector(xz,1,MAXSITE);
        free_dvector(yz,1,MAXSITE);
        free_dvector(Vx,1,MAXSITE);
        free_dvector(Vy,1,MAXSITE);
        free_dvector(sum_x,1,MAXSITE);
        free_dvector(sumt_x,1,MAXSITE);
        free_dvector(sumtt_x,1,MAXSITE);
        free_dvector(sum_y,1,MAXSITE);
        free_dvector(sumt_y,1,MAXSITE);
        free_dvector(sumtt_y,1,MAXSITE);
        free_dmatrix(xdisp,1,MAXEPO,1,MAXSITE);
        free_dmatrix(ydisp,1,MAXEPO,1,MAXSITE);
        free_dmatrix(comp_xdisp,1,MAXEPO,1,MAXSITE);
        free_dmatrix(comp_ydisp,1,MAXEPO,1,MAXSITE);
        free_dvector(epoch,1,MAXEPO);
        free_d3tensor(pwgt,1,MAXEPO,1,MAXSITE*NDIM,1,MAXSITE*NDIM);

        free_ivector(inv_dU,1,MAXEPO);
        free_ivector(inv_dV,1,MAXEPO);
        free_ivector(inv_rot,1,MAXEPO);
        free_ivector(inv_Vx,1,MAXSITE);
        free_ivector(inv_Vy,1,MAXSITE);
        free_imatrix(occup,0,MAXEPO,0,MAXSITE);
}


#include "globend.c"
*******************************************************************************

*******************************************************************************
/* subroutine to reserve memory for global arrays for velsmooth.c.

   pjc jun '96
*/

#include "nrutil.h"
#include "nr.h"

#include "global.c"


void globres()


{
        xcoord = dvector(1,MAXSITE);
        ycoord = dvector(1,MAXSITE);
        ap_dU = dvector(1,MAXEPO);
        ap_dV = dvector(1,MAXEPO);
        ap_rot = dvector(1,MAXEPO);
        ap_Vx = dvector(1,MAXSITE);
        ap_Vy = dvector(1,MAXSITE);
        dU = dvector(1,MAXEPO);
        dV = dvector(1,MAXEPO);
        rot = dvector(1,MAXEPO);
        xz = dvector(1,MAXSITE);
        yz = dvector(1,MAXSITE);
        Vx = dvector(1,MAXSITE);
        Vy = dvector(1,MAXSITE);
        sum_x = dvector(1,MAXSITE);
        sumt_x = dvector(1,MAXSITE);
        sumtt_x = dvector(1,MAXSITE);
        sum_y = dvector(1,MAXSITE);
        sumt_y = dvector(1,MAXSITE);
        sumtt_y = dvector(1,MAXSITE);
        xdisp = dmatrix(1,MAXEPO,1,MAXSITE);
        ydisp = dmatrix(1,MAXEPO,1,MAXSITE);
```

```
        comp_xdisp = dmatrix(1,MAXEPO,1,MAXSITE);
        comp_ydisp = dmatrix(1,MAXEPO,1,MAXSITE);
        epoch = dvector(1,MAXEPO);
        pwgt = d3tensor(1,MAXEPO,1,MAXSITE*NDIM,1,MAXSITE*NDIM);

        inv_dU = ivector(1,MAXEPO);
        inv_dV = ivector(1,MAXEPO);
        inv_rot = ivector(1,MAXEPO);
        inv_Vx = ivector(1,MAXSITE);
        inv_Vy = ivector(1,MAXSITE);
        occup = imatrix(0,MAXEPO,0,MAXSITE);
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* lubksb.c
  Taken directly from NR in C, section 2.3
/*
*******************************************************************************


*******************************************************************************
/* ludcmp.c
  Taken directly from NR in C, section 2.3
/*
*******************************************************************************


*******************************************************************************
/* subroutine to get set up simplex in array params[1..nfloat+1][1..nfloat].

   pjc jun '95
   Adapted for velsmooth.c   pjc jun '96
*/

#include <stdio.h>
#include <math.h>
#include "nrutil.h"

#include "global.c"


void makesimp(params,nmonte)

int nmonte;
double **params;


{
        int i,j;
        int ii,jj;

/* first set up (nfloat+1)th vertex of simplex with initial parameters */
        for(j=1,jj=1; j<=nepochs; j++) {
                if (inv_dU[j] == 1) params[nfloat+1][(jj++)] = monte(ap_dU[j],
SIGPOS, NSIG, nmonte);
                else dU[j] = ap_dU[j];
                if (inv_dV[j] == 1) params[nfloat+1][(jj++)] = monte(ap_dV[j],
SIGPOS, NSIG, nmonte);
                else dV[j] = ap_dV[j];
                if (inv_rot[j] == 1) params[nfloat+1][(jj++)] = monte(ap_rot[j],
SIGROT, NSIG, nmonte);
                else rot[j] = ap_rot[j];
        }


/* copy parameters over to other vertices of simplex */
        for(i=1; i<=nfloat; i++) {
                for(j=1; j<=nfloat; j++) {
                        params[i][j] = params[nfloat+1][j];
                }
        }


/* now perturb parameters by fraction PERTFAC of a priori sigma */
        for(j=1,jj=1; j<=nepochs; j++) {
                if (inv_dU[j] == 1) params[jj][jj++] += SIGPOS*PERTFAC;
                if (inv_dV[j] == 1) params[jj][jj++] += SIGPOS*PERTFAC;
                if (inv_rot[j] == 1) params[jj][jj++] += SIGROT*PERTFAC;
        }
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* function to compute amount of dither when setting up Monte-Carlo start
  points ie return a value randomly selected from the interval (x-2*sig,x+2*sig)
  with uniform probability.

   pjc jun '95
*/

#include <stdio.h>
#include <math.h>
#include "nrutil.h"
double drand48();

#include "global.c"


double monte(x,sig,searchnsig,nmonte)

double x,sig,searchnsig;
int nmonte;


{
        double value;

        if (nmonte > 0) value = x-searchnsig*sig + 2.0*searchnsig*sig*drand48();
        else value = x;

        return value;
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* mprove.c
  Taken directly from NR in C, section 2.5
/*
*******************************************************************************


*******************************************************************************
/* nr.h
  Taken directly from NR in C, appendix A
/*
```

```
*******************************************************************************


*******************************************************************************
/* nrutil.c
  Taken directly from NR in C, appendix B
/*
*******************************************************************************


*******************************************************************************
/* nrutil.h
  Taken directly from NR in C, appendix B
/*
*******************************************************************************


*******************************************************************************
/* function to compute penalty function (weighted L1 norm of displacement
  residuals), for velsmooth.

   pjc jun '96
*/

#include <stdio.h>
#include <math.h>
#include "nrutil.h"

#include "global.c"


double pen_L1dis()


{
        register int i,j,k;
        double pen_val=0.0;
        double xtmp,ytmp,ztmp;

/* for unit variance, just sum moduluf of (observed-computed) */
        if (covtype == 1) {
                for(k=1; k<=nepochs; k++) for(i=1; i<=nsites; i++) {
                        if ((occup[k][i] == 1) && (occup[0][i] > 2)) {
                                pen_val += fabs(xdisp[k][i]-comp_xdisp[k][i]);
                                pen_val += fabs(ydisp[k][i]-comp_ydisp[k][i]);
                        }
                }
        }
/* otherwise, weight with variances */
        else {
                for(k=1; k<=nepochs; k++) for(i=1; i<=nsites; i++) {
                        if ((occup[k][i] == 1) && (occup[0][i] > 2)) {
                                pen_val += fabs(xdisp[k][i]-comp_xdisp[k][i]) *
sqrt(pwgt[k][i][i]);
                                pen_val += fabs(ydisp[k][i]-comp_ydisp[k][i]) *
sqrt(pwgt[k][nsites+i][nsites+i]);
                        }
                }
        }

        return pen_val;
}


#include "globend.c"
*******************************************************************************

*******************************************************************************
/* function to compute penalty function (weighted L2 norm of displacement
  residuals) for velsmooth.

   pjc jun '96
*/

#include <stdio.h>
#include <math.h>
#include "nrutil.h"

#include "global.c"


double pen_L2dis()


{
        register int i,j,k;
        double pen_val=0.0;
        double xtmp,ytmp;
        double xres,yres;

/* for unit variance, just calculate sum of squares of (o-c) */
        if (covtype == 1) {
                for(k=1; k<=nepochs; k++) for(i=1; i<=nsites; i++) {
                        if ((occup[k][i] == 1) && (occup[0][i] > 2)) {
                                pen_val += (xdisp[k][i]-comp_xdisp[k][i]) *
(xdisp[k][i]-comp_xdisp[k][i]);
                                pen_val += (ydisp[k][i]-comp_ydisp[k][i]) *
(ydisp[k][i]-comp_ydisp[k][i]);
                        }
                }
        }
/* for diagonal weight matrix, calculate weighted sum of (o-c) */
        else if (covtype == 2) {
                for(k=1; k<=nepochs; k++) for(i=1; i<=nsites; i++) {
                        if ((occup[k][i] == 1) && (occup[0][i] > 2)) {
                                pen_val += (xdisp[k][i]-comp_xdisp[k][i]) *
(xdisp[k][i]-comp_xdisp[k][i]) * pwgt[k][i][i];
                                pen_val += (ydisp[k][i]-comp_ydisp[k][i]) *
(ydisp[k][i]-comp_ydisp[k][i]) * pwgt[k][i+nsites][i+nsites];
                        }
                }
        }
/* for full weight matrix, calculate the inner product (o-c)*W*(o-c) */
        else {
                for(k=1; k<=nepochs; k++) for(i=1; i<=nsites; i++) {
                        if ((occup[k][i] == 1) && (occup[0][i] > 2)) {
                                xtmp = 0.0; ytmp = 0.0;
                                for(j=1; j<=nsites; j++) {
                                        if ((occup[k][j] == 1) && (occup[0][j] > 2))
{
                                                xres = xdisp[k][j] -
comp_xdisp[k][j];
                                                yres = ydisp[k][j] -
comp_ydisp[k][j];
                                                xtmp += pwgt[k][i][j] * xres +
pwgt[k][i][nsites+j] * yres;
                                                ytmp += pwgt[k][nsites+i][j] * xres
+ pwgt[k][nsites+i][nsites+j] * yres;
                                        }
                                }
                                pen_val += (xdisp[k][i]-comp_xdisp[k][i])*xtmp;
                                pen_val += (ydisp[k][i]-comp_ydisp[k][i])*ytmp;
                        }
                }
```

```
        }
        return pen_val;
}


#include "globend.c"
*******************************************************************************

*******************************************************************************
/* function to compute penalty function (weighted L1 norm of displacement
  residuals).

   pjc jun '95
*/

#include <stdio.h>
#include <math.h>
#include "nrutil.h"

#include "global.c"


double pen_Linfdis()


{
        register int i,j,k;
        double pen_val=0.0;
        double tmp;

/* for unit variance, just find max residual */
        if (covtype == 1) {
                for(k=1; k<=nepochs; k++) for(i=1; i<=nsites; i++) {
                        if ((occup[k][i] == 1) && (occup[0][i] > 2)) {
                                tmp = fabs(xdisp[k][i]-comp_xdisp[k][i]);
                                if (tmp > pen_val) pen_val = tmp;
                                tmp = fabs(ydisp[k][i]-comp_ydisp[k][i]);
                                if (tmp > pen_val) pen_val = tmp;
                        }
                }
        }
/* otherwise, find max weighted residual */
        else {
                for(k=1; k<=nepochs; k++) for(i=1; i<=nsites; i++) {
                        if ((occup[k][i] == 1) && (occup[0][i] > 2)) {
                                tmp = fabs(xdisp[i]-comp_xdisp[i]) *
sqrt(pwgt[k][i][i]);
                                if (tmp > pen_val) pen_val = tmp;
                                tmp = fabs(ydisp[i]-comp_ydisp[i]) *
sqrt(pwgt[k][nsites+i][nsites+i]);
                                if (tmp > pen_val) pen_val = tmp;
                        }
                }
        }

        return pen_val;
}


#include "globend.c"
*******************************************************************************

*******************************************************************************
/* function to compute general penalty function, for velsmooth program.

   pjc jun '96
*/


#include <stdio.h>
#include <math.h>
#include "nrutil.h"

#include "global.c"


double penfunc(param_vals)

double *param_vals;

{
        double pen_val;

/* first compute model displacements */
        comp_disps(param_vals);

/* now use the appropriate penalty function */
        if (norm == 0) pen_val = pen_Linfdis();
        else if (norm == 1) pen_val = pen_L1dis();
        else if (norm == 2) pen_val = pen_L2dis();
        else return BIGNUM;

        return(pen_val);
}


#include "globend.c"
*******************************************************************************

*******************************************************************************
/* subroutine to read the input options for velsmooth from the given file.
   This file must adhere to the format given in the template file
   "velsmooth.inx".  Filenames and various flags are returned.
   pjc jun '96
*/

#include <string.h>
#include <stdio.h>

#include "global.c"


void readinps(inpname, crdnames,covarnames, outname, velname,residname, corname,
excoord,eycoord, xdisp,ydisp, sitename,nobs, usevcm,nmonte)

char *inpname,*outname;
char crdnames[][NAMELEN],covarnames[][NAMELEN];
char *velname,*residname,*corname;
char sitename[][NAMELEN];
double *excoord,*eycoord;
double **xdisp,**ydisp;
int *nobs;
int *nmonte;
int *usevcm;

{
        FILE *inpfile;
        char dummystring[255];
        char dummy1[NAMELEN],dummy2[NAMELEN];
        double depoch,ddU,ddV,drot;
        double dVx,dVy;
        int idU,idV,idrot;
        int idVx,idVy;
        int i;
        int status,nchars,nfixed;
        int site;
```

```
        void getpsvm();
        int findstnum();


        inpfile=fopen(inpname,"r");
        if (inpfile == NULL) {
                printf("\n*** error opening input file '%s'\n",inpname);
                exit();
        }

/* skip header */
        for(i=1;i<=5;i++) fgets(dummystring,255,inpfile);

/* get filenames and general options */
        fscanf(inpfile,"%s",outname);
        fgets(dummystring,255,inpfile);

        fscanf(inpfile,"%s",residname);
        fgets(dummystring,255,inpfile);

        fscanf(inpfile,"%s",velname);
        fgets(dummystring,255,inpfile);

        fscanf(inpfile,"%s",corname);
        fgets(dummystring,255,inpfile);

        fscanf(inpfile,"%d",&nepochs);
        fgets(dummystring,255,inpfile);

        fscanf(inpfile,"%d",&norm);
        fgets(dummystring,255,inpfile);

        fscanf(inpfile,"%d",usevcm);
        fgets(dummystring,255,inpfile);

        fscanf(inpfile,"%d",nmonte);
        fgets(dummystring,255,inpfile);

        for(i=1;i<=4;i++) fgets(dummystring,255,inpfile);

/* get coordinate and vcm filenames, then get actual coordinates */
        for(i=1; i<=nepochs; i++) {
                fscanf(inpfile,"%lf %s %s",&depoch,dummy1,dummy2);
                epoch[i] = depoch;
                strcpy(crdnames[i],dummy1);
                strcpy(covarnames[i],dummy2);
                fgets(dummystring,255,inpfile);
        }
        getcrds(crdnames, excoord,eycoord, xdisp,ydisp, sitename, nobs);

/* get a priori helmert params, and which params are to be estimated */
        for(i=1;i<=4;i++) fgets(dummystring,255,inpfile);
        status = 0;
        nparams = nsites*4 + nepochs*3;
        nfloat = 0;
        for(i=1; i<=nepochs; i++) {
                fscanf(inpfile,"%*s %lf %d %lf %d %lf %d", &ddU,&idU, &ddV,&idV,
&drot,&idrot);
                ap_dU[i] = ddU;
                inv_dU[i] = idU;
                ap_dV[i] = ddV;
                inv_dV[i] = idV;
                ap_rot[i] = drot;
                inv_rot[i] = idrot;
                if ((idU == 0) && (idV == 0) && (idrot == 0)) status++;
                nfloat += idU + idV + idrot;
                fgets(dummystring,255,inpfile);
        }
        if (status == 0) {
                printf("\n### problem underdetermined - no epochs fixed\n");
        }

/* set a priori sites velocities to zero, estimate all velocities unless
otherwise stated */
        for(i=1; i<=nsites; i++) {
                ap_Vx[i] = 0.0;
                inv_Vx[i] = 1;
                ap_Vy[i] = 0.0;
                inv_Vy[i] = 1;
        }

/* get any site velocity constraints that are specified */
        for(i=1;i<=7;i++) fgets(dummystring,255,inpfile);
        nfixed = 0;
        sscanf(dummystring,"%s %n",dummy1,&nchars);
        while (strcmp(dummy1,"XXXX") != 0) {
                site = findstnum(dummy1,sitename,nsites);
                if (site > 0) {
                        sscanf(dummystring,"%*s %lf %d %lf %d", &dVx,&idVx,
&dVy,&idVy);
                        Vx[site] = ap_Vx[site] = dVx;
                        inv_Vx[site] = idVx;
                        Vy[site] = ap_Vy[site] = dVy;
                        inv_Vy[site] = idVy;
                        nfixed += 2 - (idVx + idVy);
                }
                else printf("*** Site \"%s\" constraint ignored\n", dummy1);

                fgets(dummystring,255,inpfile);
                sscanf(dummystring,"%s %n",dummy1,&nchars);
        }
/*      nfloat += nsites*4 - nfixed; */
        if ((status < 2) && (nfixed < 4)) {
                printf("\n### problem may be underdetermined - <2 epochs and <4
velocities fixed\n");
        }

        fclose(inpfile);

}

#include "globend.c"
*******************************************************************************

*******************************************************************************
/* subroutine to convert coordinates from cartesian (in m) to lat,lon (in
  radians) and ellipsoidal height (in m) by iterative method (see Heiskanen &
  Moritz for theory).

   pjc dec '95
*/

#include <math.h>
#include <stdio.h>

#include "globdef.c"


void xyz2llh(x,y,z, lat,lon,hgt)

double x,y,z;
double *lat,*lon,*hgt;

{
        double newlat,dlat,newhgt,dhgt,N,Lxy;
```

```
        *lon = atan2(y,x);

        Lxy = sqrt(x*x + y*y);
        *lat = atan2(z,(1.0-ECCEN*ECCEN)*Lxy);
        *hgt = 0.0;
        dlat = 0.5; dhgt = 0.5;

        while ((dlat > 1.0e-10) || (dhgt > 1.0e-4)) {
                N = RADIUS / (sqrt(1.0 - pow(ECCEN*sin(*lat),2.0)));
                newlat = atan2(z , (Lxy - ECCEN*ECCEN*N*cos(*lat)));
                newhgt = (Lxy/cos(*lat)) - N;
                dlat = fabs(*lat - newlat);
                dhgt = fabs(*hgt - newhgt);
                *lat = newlat; *hgt = newhgt;
        }
}


#include "globend.c"
*******************************************************************************

*******************************************************************************
Input option file for velsmooth program
-------------------------------------------------------------------------------

Option                                  Description
******************************          ----------------------------------------
cg91-93.out                             output filename / "NONE"
cg91-93.res                             residual filename / "NONE"
cg91-93.vel                             velocity output filename /"NONE"
cg91-93.cor                             corrected velocity output file /"NONE"
3                                       no. of epochs
2                                       norm of penalty (1/2/0=Linf)
0                                       use vcm (1=full,0=diag)
100                                     no. of Monte-Carlo restarts


Epoch           Filename.CRD                    Filename.CEN / "UNIT"
**********      ****************************    ****************************
1991.78         CG910U.CRD                      ../rawcrds/CG910U.CEN
1993.39         CG93.CRD                        ../rawcrds/CG93.CEN
1995.44         95JUNR3C.CRD                    ../rawcrds/95JUNR3.CEN


Epoch no.       dU      inv     dV      inv     rot     inv
----------      ****.** *       ****.** *       ****.** *
1                  0.00 1          0.00 1          0.00 0
2                  0.00 0          0.00 0          0.00 0
3                  0.00 1          0.00 1          0.00 0


Site velocity constraints (1 line per site,XXXX to finish):

Site                    Vx      inv     Vy      inv
********************    ***.*** *       ***.*** *
CG54                    -19.40  0       -24.60  0
XXXX
*******************************************************************************
```

C.2 `svdvel2`

`svdvel2` is a C program that estimates coefficients of a polynomial describing the time-invariant velocity field in a region (Section 3.2). The polynomial can have terms in latitude, longitude, or mixed terms, with north and east components of the velocity field treated separately. The observational constraints used are time-separated coordinate observations, which can be weighted with their variances. To improve numerical stability, the geographic coordinates (latitude, longitude) of sites are stored and regarded separately from the geodetic coordinates which are stored as local offsets relative to the first epoch of observation.

The estimation of higher-order terms in the polynomial can result in instability of the inversion of the design matrix. For this reason, the robust technique of Singular Value Decomposition (SVD) is used to perform the inversion. SVD identifies poorly-determined parameters and enables the user to eliminate them from the inversion. In each run, the user is shown a list of parameters and their singular values, and asked to specify the minimum acceptable singular value. For further discussion of the SVD technique, see Press *et al.* (1992), Sections 2.6 and 15.4.

`svdvel2` has been compiled and run on Sun workstations using Sun C. Execution time is dependent on the number of coefficients estimated, but subject to the overhead of loading in the coordinate variances from the variance-covariance matrix files.

```
*******************************************************************************
/*
     Program to take site coordinates from *.CRD files and variances from *.CEN
   files, then compute the coefficients of the polynomials in lat and long that
   best fit the velocity field in north and east components separately.
   Polynomials are developed about the centroid of the network.
   The observations are weighted using the variances. Singular value
   decomposition is used to limit the number of poorly-determined coefficients.
   The user enters the tolerance after viewing the singular values.
   Velocities are stored internally in units of 1mm, hence must be divided
   by 1000 for residuals, etc.

     Routines from Numerical Recipes for singular value decomposition and LU
   matrix inversion are used, but are modified to use double precision
   arithmetic throughout.

     Based on the old program svdvel, which attempted to use full weighting
   (mathematically incorrect for svd), and only worked for one set of
   displacements (ie 2 epochs).

   Pete Clarke, Feb 1996
*/

#include<math.h>
#include<stdio.h>
#include<string.h>
#include "nr.h"
#include "nrutil.h"

#include "globdef.c"


main(argc,argv)

int argc;
char **argv;

{
        char inpname[NAMELEN];                  /* input file name */
        char outname[NAMELEN];                  /* output file name */
        char crdnames[MAXEPO+1][NAMELEN];       /* coordinate file names */
        char covarnames[MAXEPO+1][NAMELEN];     /* vcm file names */
        char sitename[MAXSITE+1][NAMELEN];      /* site names */
        char funcname[NAMELEN];                 /* fitted polynomial powers file */
        char clipname[NAMELEN];                 /* clipping polygon vertices file */
        char dispgrid[NAMELEN];                 /* model velocity grid output file */
        char gradgrid[NAMELEN];                 /* model velocity gradient tensor
grid output file */
        char straingrid[NAMELEN];               /* model principal strain grid output
file */
        char residname[NAMELEN];                /* residuals output file */
        char dispsta[NAMELEN];                  /* model site velocities output file
*/
        double *wx,*wy;                         /* weight vectors for x and y
displacements */
        double *xcoeff,*ycoeff;                 /* polynomial coeffs for x and y
velocity fields */
        double *epoch;                          /* times of survey epochs */
        double *xcoord,*ycoord;                 /* geographic site coords */
        double **xdisp,**ydisp;                 /* local site coords relative to
first survey epoch */
        double **u,**v;                         /* matrices for svd */
        double **cvxdisp,**cvydisp;             /* variances of x and y local coords
at each epoch */
        double **cvxcoeff,**cvycoeff;           /* a posteriori vcms of poly coeffs
*/
        double xchisq,ychisq;                   /* goodness of x,y fits */
        double xrms,yrms;                       /* rms weighted resids of x,y fits */
        double xcent,ycent;                     /* centroid of network (origin) */
        double minlon,minlat,maxlon,maxlat;     /* bounds of region/grid */
        double lonstep,latstep;                 /* grid spacing */
        double gridscale;                       /* 0=lat/lon, otherwise unit2km
factor */
        double *vertx,*verty;                   /* vertices of clipping polygon */
        int npar;                               /* no. of estimated parameters */
        int nsites;                             /* total no. of sites */
        int ngoodsites;                         /* no. of sites with 2 or more
occupations */
        int nverts;                             /* no. of vertices in clipping
polygon */
        int nepochs;                            /* no. of survey epochs */
        int *xpow,*ypow;                        /* powers of x,y in x and y velocity
field polynomials */
        int **occup;                            /* matrix of site occupation flags */

        FILE *outfile;                          /* generic output file handle */


/* reserve arrays */
        epoch = dvector(1,MAXEPO);
        wx = dvector(1,MAXPAR+MAXSITE);
        wy = dvector(1,MAXPAR+MAXSITE);
        xcoeff = dvector(1,MAXPAR+MAXSITE);
        ycoeff = dvector(1,MAXPAR+MAXSITE);
        xcoord = dvector(1,MAXSITE);
        ycoord = dvector(1,MAXSITE);
        xdisp = dmatrix(1,MAXEPO,1,MAXSITE);
        ydisp = dmatrix(1,MAXEPO,1,MAXSITE);
        vertx = dvector(1,MAXVERT);
        verty = dvector(1,MAXVERT);
        u = dmatrix(1,MAXSITE*MAXEPO,1,MAXPAR+MAXSITE);
        v = dmatrix(1,MAXPAR+MAXSITE,1,MAXPAR+MAXSITE);
        cvxdisp = dmatrix(1,MAXEPO,1,MAXSITE);
        cvydisp = dmatrix(1,MAXEPO,1,MAXSITE);
        cvxcoeff = dmatrix(1,MAXPAR,1,MAXPAR);
        cvycoeff = dmatrix(1,MAXPAR,1,MAXPAR);
        xpow = ivector(1,MAXPAR);
        ypow = ivector(1,MAXPAR);
        occup = imatrix(0,MAXEPO,0,MAXSITE);


/* print title */
        printf("\nEstimation of polynomial displacement fields\n");
        printf("-------------------------------------------\n");


/* read input file and coord files specified therein */
        if (argc == 2) strcpy(inpname,argv[1]);
        else strcpy(inpname,"svdvel2.inp");
        readinps(inpname,outname,&nepochs,epoch,crdnames,covarnames, xcoord, ycoord,
&xcent,&ycent, xdisp,ydisp,sitename,&nsites,&ngoodsites,occup, funcname, residname,
dispsta,dispgrid,gradgrid,straingrid, &gridscale,clipname, &lonstep,&latstep);


/* load covariance matrices or set to identity matrix if "UNIT" entered */
        getcovs(covarnames,nepochs,sitename,cvxdisp,cvydisp,nsites,occup);


/* get no. of terms to estimate, check bounds */
        printf("\nNumber of basis functions for each of Vx, Vy : ");
        scanf("%d",&npar);
        if (npar > MAXPAR) {
```

```
                printf("\n*** maximum number of  basis functions exceeded\n");
                printf("*** max number of basis functions in each of Vx,Vy = %d\n",
MAXPAR);
                exit(1);
        }

/* set up functions automatically or from file of powers of x and y */
        getfuncs(funcname,npar,xpow,ypow);


/* call the main svd fitting routine for x displacements */
        printf("\nFitting polynomial for x ... \n");
        svdfit3(xcoord,ycoord,xdisp,cvxdisp,nsites,ngoodsites,epoch,nepochs, occup,
xcoeff,npar, u,v,wx, &xchisq,&xrms, polynom, xpow,ypow);


/* compute covariances of polynomial coefficients for x */
        printf("Computing postfit covariance matrix for x ... \n");
        svdvar(v,npar,wx,cvxcoeff);


/* call the main svd fitting routine for y displacements */
        printf("\nFitting polynomial for y ... \n");
        svdfit3(xcoord,ycoord,ydisp,cvydisp,nsites,ngoodsites,epoch,nepochs, occup,
ycoeff,npar, u,v,wy, &ychisq,&yrms, polynom, xpow,ypow);


/* compute covariances of polynomial coefficients for y */
        printf("Computing postfit covariance matrix for y ... \n");
        svdvar(v,npar,wy,cvycoeff);


/* print results to screen */
        printf("\nResults of polynomial fit: \n");
        pcoeffs(stdout,npar,nsites,xpow,ypow,xcoeff,ycoeff,cvxcoeff,cvycoeff, wx,wy,
xchisq,ychisq,xrms,yrms,xcent,ycent);


/* open output file for parameters, if required, and print params */
        outfile = NULL;
        if (strcmp(outname,"NONE") != 0) {
                outfile = fopen(outname,"w");
                if (outfile == NULL) {
                        printf("\n*** error opening output file \"%s\"\n", outname);
                        exit(1);
                }
                pcoeffs(outfile,npar,nsites,xpow,ypow,xcoeff,ycoeff,
cvxcoeff,cvycoeff, wx,wy, xchisq,ychisq,xrms,yrms,xcent,ycent);
        }


/* output residuals to residual .psvm file (if output filename given) */
        if (strcmp(residname,"NONE") != 0) {
                printf("\nMaking residual file\n");
                presids(residname, nsites,sitename,xcoord,ycoord,xdisp,ydisp,
cvxdisp,cvydisp,xcent,ycent,nepochs,epoch,occup,npar,xpow,ypow,xcoeff,ycoeff);
        }


/* output velocities computed at stations, if output file specified */
        if (strcmp(dispsta,"NONE") != 0) {
                printf("\nMaking station velocity file\n");
                pdispsta(dispsta, nsites,sitename,occup, xcoord,ycoord, xcent,ycent,
npar, xpow,ypow, xcoeff,ycoeff,cvxcoeff,cvycoeff);
        }


/* find vertices of clipping polygon from file or select default rectangle */
        getclips(clipname,&nverts,vertx,verty,nsites,xcoord,ycoord,xcent,ycent,
&minlon,&minlat,&maxlon,&maxlat,lonstep,latstep);


/* output velocities computed at points on grid, if output file specified */
        if (strcmp(dispgrid,"NONE") != 0) {
                printf("\nMaking gridded displacement file\n");
                pvelgrid(dispgrid, npar,xcoeff,ycoeff,cvxcoeff,cvycoeff, xpow,ypow,
xcent,ycent, minlon,minlat, maxlon,maxlat, lonstep,latstep, nverts,vertx,verty);
        }


/* output grid of velocity gradient tensor, if output file specified */
        if (strcmp(gradgrid,"NONE") != 0) {
                printf("\nMaking gridded vel. grad. tensor file\n");
                pgradgrid(gradgrid, npar,xcoeff,ycoeff,xpow,ypow, xcent,ycent,
gridscale, minlon,minlat, maxlon,maxlat, lonstep,latstep, nverts,vertx,verty);
        }


/* output principal strain rates computed at points on grid, if output file is
   specified */
        if (strcmp(straingrid,"NONE") != 0) {
                printf("\nMaking gridded principal strain file\n");
                pepsgrid(straingrid, npar,xcoeff,ycoeff,xpow,ypow, xcent,ycent,
gridscale, minlon,minlat, maxlon,maxlat, lonstep,latstep, nverts,vertx,verty);
        }


        printf("\nFinished.\n\n");


/* free arrays */
        free_dvector(epoch,1,MAXEPO);
        free_dvector(wx,1,MAXPAR+MAXSITE);
        free_dvector(wy,1,MAXPAR+MAXSITE);
        free_dvector(xcoeff,1,MAXPAR+MAXSITE);
        free_dvector(ycoeff,1,MAXPAR+MAXSITE);
        free_dvector(xcoord,1,MAXSITE);
        free_dvector(ycoord,1,MAXSITE);
        free_dmatrix(xdisp,1,MAXEPO,1,MAXSITE);
        free_dmatrix(ydisp,1,MAXEPO,1,MAXSITE);
        free_dvector(vertx,1,MAXVERT);
        free_dvector(verty,1,MAXVERT);
        free_dmatrix(u,1,MAXSITE*MAXEPO,1,MAXPAR+MAXSITE);
        free_dmatrix(v,1,MAXPAR+MAXSITE,1,MAXPAR+MAXSITE);
        free_dmatrix(cvxdisp,1,MAXEPO,1,MAXSITE);
        free_dmatrix(cvydisp,1,MAXEPO,1,MAXSITE);
        free_dmatrix(cvxcoeff,1,MAXPAR,1,MAXPAR);
        free_dmatrix(cvycoeff,1,MAXPAR,1,MAXPAR);
        free_ivector(xpow,1,MAXPAR);
        free_ivector(ypow,1,MAXPAR);
        free_imatrix(occup,0,MAXEPO,0,MAXSITE);
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to return the partial differential wrt x(lon) of the nfuncs basis
   functions at a given point (x,y) The arrays xpow[] and ypow[] contain the
   exponents x^xpow[n] and y^ypow[n] of the nth basis function.
```

```
   pjc sep '94
*/

#include<math.h>

#include "globdef.c"


void dpolbydx(x,y,xcent,ycent,gridscale,afunc,nfuncs,xpow,ypow)

int nfuncs;
int xpow[],ypow[];
double x,y,xcent,ycent,gridscale,afunc[];

{
        int n;
        double rprimev;


        if (gridscale == 0.0) {
                rprimev = RADIUS / sqrt(1 - power((ECCEN*sin(ycent*PI/180)),2));
                gridscale = rprimev * cos(ycent*PI/180.0) * PI/180.0;
        }

        for(n=1;n<=nfuncs;n++) {
                if (xpow[n] > 0) afunc[n] = xpow[n] * power(x,xpow[n]-1) *
power(y,ypow[n]) / gridscale;
                else afunc[n] = 0.0;
        }

}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to return the partial differential wrt y(lat) of the nfuncs basis
   functions at a given point (x,y) The arrays xpow[] and ypow[] contain the
   exponents x^xpow[n] and y^ypow[n] of the nth basis function.

   pjc sep '94
*/

#include<math.h>

#include "globdef.c"


void dpolbydy(x,y,xcent,ycent,gridscale,afunc,nfuncs,xpow,ypow)

int nfuncs;
int xpow[],ypow[];
double x,y,xcent,ycent,gridscale,afunc[];

{
        int n;
        double rmerid,tmp;


        if (gridscale == 0.0) {
                tmp = 1 - power((ECCEN*sin(ycent * PI/180)),2);
                rmerid = RADIUS * (1 - power(ECCEN,2)) / (tmp * sqrt(tmp));
                gridscale = rmerid * PI/180.0;
        }

        for(n=1;n<=nfuncs;n++) {
                if (ypow[n] > 0) afunc[n] = ypow[n] * power(x,xpow[n]) *
power(y,ypow[n]-1) / gridscale;
                else afunc[n] = 0.0;
        }

}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to find the station number of a named station ie its position in
   the array namelist[1..nsites], or return -999 if station not found

   pjc jul '94
*/

#include<string.h>

#include "globdef.c"


int findstnum(name,namelist,nsites)

char name[];
char namelist[][NAMELEN];
int nsites;

{
        int i;
        int found=0;

        for(i=1;found<1;i++) {
                if (i > nsites) {
                        found=2;
                }
                if (strcmp(name,namelist[i]) == 0) {
                        found=1;
                }
        }
        if (found == 1) {
                return(i-1);
        } else {
                return(-999);
        }
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to get vertices of clipping polygon.  If the file is "NONE", then
   a rectangular polygon will be chosen such that all the sites for which
   cordinates are given lie within it, and the vertices are at integer
   multiples of (lonstep,latstep).

   pjc sep '94
*/

#include<math.h>
#include<stdio.h>
#include<string.h>

#include "globdef.c"
```

```
void getclips(filename,nverts,vertx,verty,nsites,xcoord,ycoord,xcent,ycent,
minlon,minlat,maxlon,maxlat,lonstep,latstep)

char filename[NAMELEN];
double *vertx,*verty,*xcoord,*ycoord;
double *minlon,*minlat,*maxlon,*maxlat;
double xcent,ycent,lonstep,latstep;
int *nverts,nsites;

{
        FILE *clipfile;
        double vx,vy;
        int status,i;

        *minlon=*minlat=999;
        *maxlon=*maxlat=-999;
        for(i=1;i<=nsites;i++) {
                if (xcoord[i] < *minlon) *minlon=xcoord[i];
                if (xcoord[i] > *maxlon) *maxlon=xcoord[i];
                if (ycoord[i] < *minlat) *minlat=ycoord[i];
                if (ycoord[i] > *maxlat) *maxlat=ycoord[i];
        }
        *minlon = lonstep*floor((*minlon+xcent)/lonstep) - xcent;
        *maxlon = lonstep*ceil((*maxlon+xcent)/lonstep) - xcent;
        *minlat = latstep*floor((*minlat+ycent)/latstep) - ycent;
        *maxlat = latstep*ceil((*maxlat+ycent)/latstep) - ycent;

        if (strcmp(filename,"NONE") == 0) {
                *nverts = 4;
                vertx[1] = *minlon; verty[1] = *minlat;
                vertx[2] = *minlon; verty[2] = *maxlat;
                vertx[3] = *maxlon; verty[3] = *maxlat;
                vertx[4] = *maxlon; verty[4] = *minlat;
        }
        else {
                clipfile=fopen(filename,"r");
                if (clipfile == NULL) {
                        printf("\n*** error opening clipping file: %s\n", filename);
                        exit(1);
                }
                i=0;
                status=0;
                while (status != EOF) {
                        i++;
                        status = fscanf(clipfile,"%lf %lf\n",&vx,&vy);
                        if (i > MAXVERT) {
                                printf("\n*** maximum number of clipping polygon
vertices exceeded\n");
                                printf("    number of vertices >= %d", i);
                                exit(1);
                        }
                        vx -= xcent; vy -= ycent;
                        if (vx < *minlon) *minlon = floor(vx/lonstep) * lonstep;
                        if (vx > *maxlon) *maxlon = ceil(vx/lonstep) * lonstep;
                        if (vy < *minlat) *minlat = floor(vy/latstep) * latstep;
                        if (vy > *maxlat) *maxlat = ceil(vy/latstep) * latstep;
                        vertx[i] = vx;
                        verty[i] = vy;
                }
                fclose(clipfile);
                *nverts = i-1;
        }

}


#include "globend.c"
```

```
*******************************************************************************


*******************************************************************************
/* subroutine to read in a covariance matrix from a file in .CEN format (which
   is equivalent to the Bernese 3.4 format, but having had a similarity
   transform applied so that the components are in the coordinate system of
   east, north, up appropriate at the centroid of the network).
   Covariances are stored in units of mm.
   Modified dec '95 for multiple covar files for polystr2.
   Modified feb '96 for diagonal elements only for svdvel2.

   pjc aug '95
*/

#include <string.h>
#include <stdio.h>

#define SIGSQMIN 4.0            /* default std error of a fixed coord is 2mm */
#define SIGSQTOL 1.0e-2         /* 'zero' variance of a fixed coordinate */

#include "globdef.c"


void getcovs(covarnames,nepochs,sitename,cvx,cvy,nsites,occup)

char covarnames[][NAMELEN],sitename[][NAMELEN];
double **cvx,**cvy;
int nsites,nepochs,**occup;

{
        FILE *covarfile;
        int status;
        int i,k;
        int stnum1,stnum2;
        char st1[NAMELEN],st2[NAMELEN];
        char neu1[3],neu2[3];
        char dummystring[255];
        double elem,unitwgt;

/* loop over all epochs */
        for(k=1; k<=nepochs; k++) {
                printf("\nLoading covariance submatrix %s ...\n", covarnames[k]);


        /* for unit vcm, just synthesise it */
                if (strcmp(covarnames[k],"UNIT") == 0) {
                        for(i=1; i<=nsites; i++) {
                                cvx[k][i] = 1.0;
                                cvy[k][i] = 1.0;
                        }
                }

        /* load up B3.4-style vcm, check for singularities (zero variances) */
                else {
                        covarfile=fopen(covarnames[k],"r");
                        if (covarfile == NULL) {
                                printf("\n*** error opening file:
'%s'\n",covarnames[k]);
                                exit(1);
                        }
                        status=0;
                        for(i=1;i<=6;i++) fgets(dummystring,255,covarfile);
                        fscanf(covarfile, "RMS OF UNIT WEIGHT: %lf", &unitwgt);
                        unitwgt *= 1.0e3;       /* convert to mm */
                        for(i=1;i<=5;i++) fgets(dummystring,255,covarfile);
                        while (status != EOF) {
                                status = fscanf(covarfile,"%s %s %s %s %lf\n", st1,
neu1, st2, neu2, &elem);
```

```
                                if ((status != 0) && (status != EOF)) {
                                        stnum1=findstnum(st1,sitename,nsites);
                                        stnum2=findstnum(st2,sitename,nsites);
                                        if ((stnum1==stnum2) && (strcmp(neu1,"E")==0)
&& (strcmp(neu2,"E")==0)) cvx[k][stnum1] = elem * unitwgt*unitwgt;
                                        else if ((stnum1==stnum2) &&
(strcmp(neu1,"N")==0) && (strcmp(neu2,"N")==0)) cvy[k][stnum1] = elem *
unitwgt*unitwgt;
                                }
                        }
                        fclose(covarfile);
                }

/* check covariance matrix for zero variances, adjust them to SIGSQMIN */
                for(i=1;i<=nsites;i++) {
                        if (occup[k][i] == 0) continue;
                        if (cvx[k][i] < SIGSQTOL) {
                                printf("### Variance of  %s:E increased from %10.3e
to %10.3le\n",sitename[i],cvx[k][i],SIGSQMIN);
                                cvx[k][i] = SIGSQMIN;
                        }
                        if (cvy[k][i] < SIGSQTOL) {
                                printf("### Variance of  %s:N increased from %10.3e
to %10.3le\n",sitename[i],cvy[k][i],SIGSQMIN);
                                cvy[k][i] = SIGSQMIN;
                        }
                }
        }
}


#undef SIGSQMIN
#undef SIGSQTOL


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to get coordinates and sitenames from B3.4 .CRD files, then make
   displacements (in mm) in local coords for each epoch.

   pjc dec '95
*/

#include<math.h>
#include<stdio.h>
#include<string.h>

#include "nr.h"
#include "nrutil.h"

#include "globdef.c"


void getcrds(nepochs,crdnames,occup,excoord,eycoord,xcent,ycent,xdisp,ydisp,
sitename,nsites,ngoodsites)

char crdnames[][NAMELEN],sitename[][NAMELEN];
double *excoord,*eycoord,*xcent,*ycent;
double **xdisp,**ydisp;
int *nsites,*ngoodsites,nepochs;
int **occup;

{
        FILE *crdfile;
        char dummystring[255],dummy[NAMELEN],*flag,crdflag[2];
        int i,j,n,site,status;
        double cx,cy,cz;
        double *cx0,*cy0,*cz0;
        double lat,lon,hgt;
        double dx,dxy,dy,dz;


        cx0 = dvector(1,MAXSITE);
        cy0 = dvector(1,MAXSITE);
        cz0 = dvector(1,MAXSITE);


/* zero all arrays at start to avoid accidents */
        *nsites = 0;
        for(i=0; i<=MAXEPO; i++) for(j=0; j<=MAXSITE; j++) occup[i][j] = 0;


/* loop over all epochs */
        for(i=1; i<=nepochs; i++) {
                crdfile=fopen(crdnames[i],"r");
                if (crdfile == NULL) {
                        printf("\n*** error opening coordinate file: '%s'\n",
crdnames[i]);
                        exit(1);
                }
                printf("Reading coordinate file \"%s\" ... ", crdnames[i]);
                for(j=1; j<=6; j++) fgets(dummystring,255,crdfile);

        /* read in all lines with the coord flag set */
                status=0; n=0;
                while (status != EOF) {
                        flag = fgets(dummystring,255,crdfile);
                        if (flag == NULL) break;
                        strcpy(crdflag,"");
                        status = sscanf(dummystring,"%*d %s %lf %lf %lf %s", dummy,
&cx,&cy,&cz, crdflag);
                        if (status == EOF) break;
                        if (strcmp(crdflag,"") != 0) {
                                n++;
                                site = findstnum(dummy,sitename,*nsites);
                                if (site < 0) {
                                        (*nsites)++;
                                        if (*nsites > MAXSITE) {
                                                printf("\n*** maximum number of sites
exceeded\n");
                                                printf("    number of sites >= %d\n",
*nsites);
                                                exit(1);
                                        }
                                        site = *nsites;
                                        strcpy(sitename[site],dummy);
                                        cx0[site] = cx;
                                        cy0[site] = cy;
                                        cz0[site] = cz;
                                }
                                occup[i][site] = 1;
                                occup[0][site] += 1;

                        /* get geodetic coords of site */
                                xyz2llh(cx,cy,cz, &lat,&lon,&hgt);
                                excoord[site] = lon / DEG2RAD;
                                eycoord[site] = lat / DEG2RAD;

                        /* get offset of site from its first position */
                                dx = cx - cx0[site];
                                dy = cy - cy0[site];
                                dz = cz - cz0[site];

                        /* rotate offset into local coord frame */
```

```
                                dxy = dx*cos(lon) + dy*sin(lon);
                                xdisp[i][site] = (dy*cos(lon) - dx*sin(lon)) *
1000.0;
                                ydisp[i][site] = (dz*cos(lat) - dxy*sin(lat)) *
1000.0;
                        }
                }

                occup[i][0] = n;
                printf("%d sites found\n", n);
                fclose(crdfile);
        }


/* check for sites with only 1 occupation - they are no good */
        for(i=1,*ngoodsites=0; i<=*nsites; i++) {
                if (occup[0][i] < 2) {
                        printf("Site %s has fewer than 2 occupations - will be
ignored\n", sitename[i]);
                        for(j=0; j<=nepochs; j++) occup[j][i] = 0;
                }
                else (*ngoodsites)++;
        }


/* find centroid of network, to be used as origin for polynomial development */
        *xcent = 0.0; *ycent = 0.0;
        for(i=1; i<=*nsites; i++) {
                *xcent += excoord[i]; *ycent += eycoord[i];
        }


/* convert all coords to be relative to origin */
        *xcent /= (double)(*nsites); *ycent /= (double)(*nsites);
        for(i=1; i<=*nsites; i++) {
                excoord[i] -= *xcent; eycoord[i] -= *ycent;
        }

        printf("Total of %d sites found (%d good)\n", *nsites,*ngoodsites);


        free_dvector(cx0,1,MAXSITE);
        free_dvector(cy0,1,MAXSITE);
        free_dvector(cz0,1,MAXSITE);
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to get powers of x and y to be used in basis functions 1..nfunc
   from the file filename.  Each line of the file should have two integers,
   being the powers of x and y respectively for that particular basis function.
   The special filenames X,Y,XY are recognised as being complete polynomials up
   to the required degree and order in X, Y and both X and Y respectively, and
   no file need be read to generate these.

   pjc sep '94
*/

#include<stdio.h>
#include<string.h>

#include "globdef.c"

void getfuncs(filename,nfunc,xpow,ypow)

char filename[NAMELEN];
int nfunc,xpow[],ypow[];

{
        FILE *funcfile;
        int status,i;

        if (strcmp(filename,"X") == 0) {
                for(i=1;i<=nfunc;i++) {
                        xpow[i] = i-1;
                        ypow[i] = 0;
                }
        }
        else if (strcmp(filename,"Y") == 0) {
                for(i=1;i<=nfunc;i++) {
                        xpow[i] = 0;
                        ypow[i] = i-1;
                }
        }
        else if (strcmp(filename,"XY") == 0) {
                for(i=1;i<=nfunc;i++) {
                        polypower(i,xpow+i,ypow+i);
                }
        }
        else {
                funcfile=fopen(filename,"r");
                if (funcfile == NULL) {
                        printf("\n*** error opening function file: %s\n", filename);
                        exit(1);
                }
                for(i=1;i<=nfunc;i++) {
                        status = fscanf(funcfile,"%d %d\n",xpow+i,ypow+i);
                        if (status == EOF) {
                                printf("\n*** error reading function file: %s\n",
filename);
                                printf("*** end of file reached after %d
functions\n", i-1);
                                exit(1);
                        }
                }
                fclose(funcfile);
        }

}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* globdef.c

  macro and structure definitions for svdvel2.c and subroutines */


/* Constants and array sizes */
#define DEG2RAD 0.017453293     /* degrees to radians factor */
#define ECCEN   0.081819191     /* eccentricity of GRS-80 ellipsoid */
#define MAXEPO  5               /* max no. of epochs in survey */
#define MAXPAR  100             /* max no. of params */
#define MAXSITE 100             /* max no. of sites */
#define MAXVERT 20              /* max no. of vertices of clipping polygon */
#define MINTOL 1.0e-10  /* minimum permitted non-zero value for w[j]/wmax */
#define NAMELEN 32              /* max length of site names etc */
#define PI      3.1415926536    /* pi */
```

```
#define RADIUS  6378137.0       /* radius of GRS-80 ellipsoid */
#define ZEPOCH  1988.00         /* zero epoch for coordinate diffs */


/* function prototypes */
void dpolbydx();
void dpolbydy();
void getclips();
void getcovs();
void getcrds();
void getfuncs();
void matmult();
void pcoeffs();
void pdispsta();
void pepsgrid();
void polynom();
void polypower();
void presids();
void pvelgrid();
void readinps();
void svdfit3();
void svedit();
void xyz2llh();

int findstnum();
int inpoly();

double power();
*****************************************************************************


*****************************************************************************
/* globend.c

   undefine macros used in svdvel2.c
*/

#undef DEG2RAD
#undef ECCEN
#undef MAXEPO
#undef MAXPAR
#undef MAXSITE
#undef MAXVERT
#undef MINTOL
#undef NAMELEN
#undef PI
#undef RADIUS
#undef ZEPOCH
*****************************************************************************


*****************************************************************************
/* subroutine to determine whether a point (px,py) lies within a polygon
   defined by vertices (vertx[1..nverts],verty[1..nverts]).  The angles
   subtended at the point by the sides of the polygon in turn are summed : if
   the sum    is 360 degrees the point lies inside the polygon.  A check is
   also made to see if (px,py) is on a vertex, as numerical instability lies
   here!

   pjc sep '94
*/

#include<math.h>

#include "globdef.c"

int inpoly(px,py,nverts,vertx,verty)

int nverts;
double px,py,*vertx,*verty;

{
        int i,j;
        double a,b,adotb;
        double theta;

        theta=0;
        for(i=1;i<=nverts;i++) {
                if ((px == vertx[i]) && (py == verty[i])) return(1);
                if (i < nverts) j=i+1;
                else j=1;
                a = sqrt((px-vertx[i])*(px-vertx[i]) + (py-verty[i])*(py-verty[i]));
                b = sqrt((px-vertx[j])*(px-vertx[j]) + (py-verty[j])*(py-verty[j]));
                adotb = (vertx[i]-px)*(vertx[j]-px) + (verty[i]-py)*(verty[j]-py);
                theta += acos(adotb/(a*b)) * 180.0 / PI;
        }
        if ((theta > 359.0) && (theta < 361.0)) return(1);
        else return(0);
}


#include "globend.c"
*****************************************************************************


*****************************************************************************
/* nr.h
  Taken directly from NR in C, appendix A
/*
*****************************************************************************


*****************************************************************************
/* nrutil.c
  Taken directly from NR in C, appendix B
/*
*****************************************************************************


*****************************************************************************
/* nrutil.h
  Taken directly from NR in C, appendix B
/*
*****************************************************************************


*****************************************************************************
/* subroutine to print out results of svd fitting, also chisq and
   the origin of development of the polynomials.
   If a parameter has not been estimated (the weight has been zeroed), this
   is indicated.

   pjc feb '96
*/

#include<stdio.h>
#include<math.h>

#include "globdef.c"


void pcoeffs(dest,nfunc,ndata,xpow,ypow,xcoeff,ycoeff,cvxcoeff,cvycoeff,wx,wy,
xchisq,ychisq,xrms,yrms,xcent,ycent)

FILE *dest;
int nfunc,ndata,*xpow,*ypow;
double xchisq,ychisq,xrms,yrms,xcent,ycent;
```

```
double *wx,*wy,*xcoeff,*ycoeff,**cvxcoeff,**cvycoeff;

{
        int i;

        fprintf(dest,"\Origin of development (lon,lat) = %10.6lf , %10.6lf\n",
xcent,ycent);

        fprintf(dest,"\n\nChi-squared  (x) = %10.4le \n",xchisq);
        fprintf(dest,"rms residual (x) = %10.4le \n", xrms);
        fprintf(dest,"\nCoefficients of Vx: \n");
        fprintf(dest,"    n   x^ y^      coeff         error\n");
        for(i=1;i<=nfunc;i++) {
                fprintf(dest,"%5d  %2d %2d  %13.4le %13.4le", i,xpow[i],ypow[i],
xcoeff[i]/1000.0,sqrt(cvxcoeff[i][i])/1000.0);
                if (wx[i] == 0.0) fprintf(dest,"   (w=0)");
                fprintf(dest,"\n");
        }

        fprintf(dest,"\n\nChi-squared  (y) = %10.4le \n",ychisq);
        fprintf(dest,"rms residual (y) = %10.4le \n", yrms);
        fprintf(dest,"\nCoefficients of Vy: \n");
        fprintf(dest,"    n   x^ y^      coeff         error\n");
        for(i=1;i<=nfunc;i++) {
                fprintf(dest,"%5d  %2d %2d  %13.4le %13.4le", i,xpow[i],ypow[i],
ycoeff[i]/1000.0, sqrt(cvycoeff[i][i])/1000.0);
                if (wy[i] == 0.0) fprintf(dest,"   (w=0)");
                fprintf(dest,"\n");
        }
        fprintf(dest,"\n");

}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to compute and print out the computed velocities at sites
   from svd2 fitting, to  a file.

   pjc feb '96
*/

#include<stdio.h>
#include<math.h>
#include "nrutil.h"

#include "globdef.c"


void pdispsta(dispsta,nsites,sitename,occup, xcoord,ycoord, xcent,ycent, npar,
xpow,ypow, xcoeff,ycoeff,cvxcoeff,cvycoeff)

char dispsta[],sitename[][NAMELEN];
int nsites,**occup,npar,*xpow,*ypow;
double xcent,ycent;
double *xcoord,*ycoord,*xcoeff,*ycoeff,**cvxcoeff,**cvycoeff;

{
        int i,j;
        double vx,vy;
        double sigxx,sigyy;
        double *rfunc;
        FILE *dispstafile;

        rfunc=dvector(1,npar);


/* open file and check */
        dispstafile = fopen(dispsta,"w");
        if (dispstafile == NULL) {
                printf("\n*** error opening station displacement file: %s\n",
dispsta);
                exit(1);
        }

/* loop over all sites, generate basis functions at each one and multiply by
coefficients, then output */
        for(i=1;i<=nsites;i++) {
                if (occup[0][i] < 2) continue;
                polynom(xcoord[i],ycoord[i],rfunc,npar,xpow,ypow);
                vx = 0.0; vy = 0.0;
                sigxx=0.0;sigyy=0.0;
                for(j=1;j<=npar;j++) {
                        vx += xcoeff[j]*rfunc[j];
                        vy += ycoeff[j]*rfunc[j];
                        sigxx += cvxcoeff[j][j] * rfunc[j]*rfunc[j];
                        sigyy += cvycoeff[j][j] * rfunc[j]*rfunc[j];
                }
                sigxx = sqrt(sigxx);
                sigyy = sqrt(sigyy);
                fprintf(dispstafile,"%10.6lf %10.6lf  %8.4lf %8.4lf  %8.4lf %8.4lf
%8.4lf  %s\n", xcoord[i]+xcent,ycoord[i]+ycent, vx/1000.0, vy/1000.0,
sigxx/1000.0,sigyy/1000.0, 0.0, sitename[i]);
        }

        fclose(dispstafile);

        free_dvector(rfunc,1,npar);
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to compute velocity function partial differentials at points
   on grid, turn them into principal strains, and print them to file.
   Polynomials are expanded about (xcent,ycent), at intervals (xstep,ystep)
   from (minlon,minlat) to (maxlon,maxlat). Points not lying within the polygon
   specified by corners (vertx[1..nverts], verty[1..nverts] are not computed
   or printed. For converting from (lon,lat) in degrees to (x,y) in metres,
   the radii of curvature in the meridian and prime vertical at the point
   concerned are used if the gridscale is zero (ie ellipsoidal coordinates), or
   the gridscale is used to convert from grid (position) units to metres.

   pjc sep '94
   modified feb '96 for svdvel2
*/

#include<stdio.h>
#include<math.h>
#include "nrutil.h"

#include "globdef.c"


void pepsgrid(epsname, nfunc,xcoeff,ycoeff,xpow,ypow, xcent,ycent, gridscale,
minlon,minlat, maxlon,maxlat, xstep,ystep, nverts,vertx,verty)

char epsname[];
int nfunc,nverts,*xpow,*ypow;
double xcent,ycent;
```

```
double gridscale;
double minlon,minlat,maxlon,maxlat;
double xstep,ystep;
double *vertx,*verty,*xcoeff,*ycoeff;

{
        int j;
        double b,c,e12,sq;
        double azim,eps1,eps2;
        double x,y;
        double dubydx,dubydy,dvbydx,dvbydy;
        double *rfunc;
        FILE *epsfile;

        rfunc=dvector(1,nfunc);

/* open file and check */
        epsfile = fopen(epsname,"w");
        if (epsfile == NULL) {
                printf("\n*** error opening principal strain file: %s\n", epsname);
                exit(1);
        }

/* loop over grid points */
        for(x=minlon;x<=maxlon;x+=xstep) {
                for(y=minlat;y<=maxlat;y+=ystep) {
                        if (inpoly(x,y,nverts,vertx,verty) == 0) continue;

                /* compute functions, add up terms in poly for vel grad tensor */
                        dpolbydx(x,y,xcent,ycent,gridscale,rfunc,nfunc, xpow,ypow);
                        dubydx=0.0; dvbydx=0.0;
                        for(j=1;j<=nfunc;j++) {
                                dubydx += xcoeff[j]*rfunc[j]/1000.0;
                                dvbydx += ycoeff[j]*rfunc[j]/1000.0;
                        }
                        dpolbydy(x,y,xcent,ycent,gridscale,rfunc,nfunc, xpow,ypow);
                        dubydy=0.0; dvbydy=0.0;
                        for(j=1;j<=nfunc;j++) {
                                dubydy += ycoeff[j]*rfunc[j]/1000.0;
                                dvbydy += ycoeff[j]*rfunc[j]/1000.0;
                        }

                /* compute eps1 and eps2, output */
                        e12 = 0.5 * (dubydy + dvbydx);
                        b = dubydx + dvbydy;
                        c = e12*e12 - dubydx*dvbydy;
                        sq = sqrt(b*b + 4.0*c);
                        eps1 = (b+sq)/2.0; eps2 = (b-sq)/2.0;
                        azim = 90.0 + (atan(e12/(eps1-dubydx)) * 180.0/PI);
                        fprintf(epsfile,"%10.6lf %10.6lf  %12.4le %12.4le  %8.2f \n",
x+xcent,y+ycent, eps1,eps2,azim);
                }
        }

        fclose(epsfile);

        free_dvector(rfunc,1,nfunc);
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to compute velocity function partial differentials (ie elements
   of velocity gradient tensor) at points on grid, and print them to file.

   Polynomials are expanded about (xcent,ycent), at intervals (xstep,ystep)
   from (minlon,minlat) to (maxlon,maxlat). Points not lying within the polygon
   specified by corners (vertx[1..nverts], verty[1..nverts] are not computed
   or printed. For converting from (lon,lat) in degrees to (x,y) in metres,
   the radii of curvature in the meridian and prime vertical at the point
   concerned are used if the gridscale is zero (ie ellipsoidal coordinates), or
   the gridscale is used to convert from grid (position) units to metres.

   pjc feb '96
*/

#include<stdio.h>
#include<math.h>
#include "nrutil.h"

#include "globdef.c"


void pgradgrid(gradname, nfunc,xcoeff,ycoeff,xpow,ypow, xcent,ycent, gridscale,
minlon,minlat, maxlon,maxlat, xstep,ystep, nverts,vertx,verty)

char gradname[];
int nfunc,nverts,*xpow,*ypow;
double xcent,ycent;
double gridscale;
double minlon,minlat,maxlon,maxlat;
double xstep,ystep;
double *vertx,*verty,*xcoeff,*ycoeff;

{
        int j;
        double b,c,e12,sq;
        double azim,eps1,eps2;
        double x,y;
        double dubydx,dubydy,dvbydx,dvbydy;
        double *rfunc;
        FILE *gradfile;

        rfunc=dvector(1,nfunc);

/* open file, check */
        gradfile = fopen(gradname,"w");
        if (gradfile == NULL) {
                printf("\n*** error opening principal strain file: %s\n", gradname);
                exit(1);
        }

/* loop over grid points */
        for(x=minlon;x<=maxlon;x+=xstep) {
                for(y=minlat;y<=maxlat;y+=ystep) {
                        if (inpoly(x,y,nverts,vertx,verty) == 0) continue;

                /* compute funcs at point, add terms in poly, print */
                        dpolbydx(x,y,xcent,ycent,gridscale,rfunc,nfunc, xpow,ypow);
                        dubydx=0.0; dvbydx=0.0;
                        for(j=1;j<=nfunc;j++) {
                                dubydx += xcoeff[j]*rfunc[j]/1000.0;
                                dvbydx += ycoeff[j]*rfunc[j]/1000.0;
                        }
                        dpolbydy(x,y,xcent,ycent,gridscale,rfunc,nfunc, xpow,ypow);
                        dubydy=0.0; dvbydy=0.0;
                        for(j=1;j<=nfunc;j++) {
                                dubydy += ycoeff[j]*rfunc[j]/1000.0;
                                dvbydy += ycoeff[j]*rfunc[j]/1000.0;
                        }
                        fprintf(gradfile,"%10.6lf %10.6lf  %12.4le %12.4le  %12.4le
%12.4le\n", x+xcent,y+ycent, dubydx,dubydy, dvbydx,dvbydy);
```

```
                }
        }

        fclose(gradfile);

        free_dvector(rfunc,1,nfunc);
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to return the values of the nfuncs basis functions at a given
   point (x,y) which is specified by giving a vector x[] of all coordinates and
   the index number i of the required point.  The number of points in x[] has
   to be specified so that the x and y coordinates can be separated.  The arrays
   xpow[] and ypow[] contain the exponents x^xpow[n] and y^ypow[n] of the nth
   basis function.

   pjc sep '94
*/

#include<math.h>

#include "globdef.c"


void polynom(x,y,afunc,nfuncs,xpow,ypow)

int nfuncs;
int xpow[],ypow[];
double x,y,afunc[];

{
        int n;

        for(n=1;n<=nfuncs;n++) {
                afunc[n] = power(x,xpow[n]) * power(y,ypow[n]);
        }
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to work out the indices p and q of x and y in the n-th basis
   function in ascending powers of x and y.  By definition (x^0)(y^0) is
   function n=1, not n=0, etc.

   pjc jul '94
*/

#define MAXORDER 100


void polypower(n,p,q)

int n;
int *p,*q;

{
        int order;
        int base=1,ok=0;

        for(order=0;ok<1;order++) {
                base+=order;
                if (n <= (base+order)) ok=1;
                if (order > MAXORDER) ok=2;
        }
        *p = order+base-n-1;
        *q = n-base;
}

#undef MAXORDER
*******************************************************************************


*******************************************************************************
/* subroutine to raise a double-precision number to a positive-indefinite
   integer power.

   pjc sep '94
*/


double power(a,b)

double a;
int b;

{
        double ans;
        int i;

        ans = 1.0;
        for(i=1;i<=b;i++) ans *= a;
        return(ans);
}
*******************************************************************************


*******************************************************************************
/* subroutine to print out compute model site displacements at an epoch ,
   compute residuals to svd2 fitting, then print to file.

   pjc feb '96
*/

#include<stdio.h>
#include<math.h>
#include "nrutil.h"

#include "globdef.c"


void presids(residname,nsites,sitename,xcoord,ycoord,xdisp,ydisp, cvxdisp,cvydisp,
xcent,ycent, nepochs,epoch,occup, npar,xpow,ypow,xcoeff,ycoeff)

char residname[],sitename[][NAMELEN];
int nsites,npar,*xpow,*ypow,nepochs,**occup;
double xcent,ycent,*epoch;
double *xcoord,*ycoord,**xdisp,**ydisp,**cvxdisp,**cvydisp,*xcoeff,*ycoeff;

{
        int i,j,k,ii;
        double vxresid,vyresid;
        double sigxx,sigyy;
        double *rfunc;
        FILE *residfile;

        rfunc=dvector(1,npar);

/* open file and check */
        residfile = fopen(residname,"w");
        if (residfile == NULL) {
                printf("\n*** error opening residual file: %s\n", residname);
```

```
                exit(1);
        }


/* loop over all epochs */
        for(k=1; k<=nepochs; k++) {
                fprintf(residfile,"# Residuals at epoch %7.2lf (o-c)\n", epoch[k]);

        /* loop over all sites at each epoch */
                for(i=1,ii=0;i<=nsites;i++) {
                        if (occup[0][i] > 1) ii++;
                        if (occup[k][i] == 0) continue;
                /* compute funcs, calsulate (o-c) by subtraction */
                        polynom(xcoord[i],ycoord[i], rfunc,npar,xpow,ypow);
                        vxresid = xdisp[k][i] - xcoeff[npar+ii];
                        vyresid = ydisp[k][i] - ycoeff[npar+ii];
                        for(j=1;j<=npar;j++) {
                                vxresid -= xcoeff[j]*rfunc[j] * (epoch[k]-ZEPOCH);
                                vyresid -= ycoeff[j]*rfunc[j] * (epoch[k]-ZEPOCH);
                        }
                /* get variances, print */
                        sigxx = sqrt(cvxdisp[k][i]);
                        sigyy = sqrt(cvydisp[k][i]);
                        fprintf(residfile,"%10.6lf %10.6lf  %8.4lf %8.4lf  %8.4lf
%8.4lf %8.4lf  %s\n", xcoord[i]+xcent,ycoord[i]+ycent, vxresid/1000.0,
vyresid/1000.0, sigxx/1000.0,sigyy/1000.0, 0.0, sitename[i]);
                }
                fprintf(residfile,"\n");
        }

        fclose(residfile);

        free_dvector(rfunc,1,npar);
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to print out function values computed at points on grid to file.
   Polynomials are expanded about (xcent,ycent), at intervals (xstep,ystep) from
   (minlon,minlat) to (maxlon,maxlat). Points not lying within the polygon
   specified by corners (vertx[1..nverts],verty[1..nverts]) are not computed
   or printed.
   Modified for svdvel2 by pjc feb '96.

   pjc sep '94
*/

#include<stdio.h>
#include<math.h>
#include "nrutil.h"

#include "globdef.c"


void pvelgrid(gridname, nfunc,xcoeff,ycoeff,cvxcoeff,cvycoeff,xpow,ypow, xcent,ycent,
minlon,minlat, maxlon,maxlat, xstep,ystep, nverts,vertx,verty)

char gridname[];
int nfunc,nverts,*xpow,*ypow;
double xcent,ycent;
double minlon,minlat,maxlon,maxlat;
double xstep,ystep;
double *vertx,*verty,*xcoeff,*ycoeff,**cvxcoeff,**cvycoeff;
{
        int j;
        double x,y;
        double fx,fy;
        double sigxx,sigyy;
        double *rfunc;
        FILE *gridfile;

        rfunc=dvector(1,nfunc);

/* open file, check */
        gridfile = fopen(gridname,"w");
        if (gridfile == NULL) {
                printf("\n*** error opening grid file: %s\n", gridname);
                exit(1);
        }

/* loop over all grid points */
        for(x=minlon;x<=maxlon;x+=xstep) {
                for(y=minlat;y<=maxlat;y+=ystep) {
                        if (inpoly(x,y,nverts,vertx,verty) == 0) continue;
                /* compute funcs, add terms in poly and error terms, print */
                        polynom(x,y,rfunc,nfunc,xpow,ypow);
                        fx=0.0; fy=0.0;
                        sigxx=0.0;sigyy=0.0;
                        for(j=1;j<=nfunc;j++) {
                                fx += xcoeff[j]*rfunc[j];
                                fy += ycoeff[j]*rfunc[j];
                                sigxx += cvxcoeff[j][j] * rfunc[j]*rfunc[j];
                                sigyy += cvycoeff[j][j] * rfunc[j]*rfunc[j];
                        }
                        sigxx = sqrt(sigxx);
                        sigyy = sqrt(sigyy);
                        fprintf(gridfile,"%10.6lf %10.6lf  %8.4lf %8.4lf  %8.4lf
%8.4lf %8.4lf \n", x+xcent,y+ycent, fx/1000,fy/1000, sigxx/1000,sigyy/1000, 0.0);
                }
        }

        fclose(gridfile);

        free_dvector(rfunc,1,nfunc);
}

#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to read the input options for svdvel2 from the file <inpname>.
   This file must adhere to the format given in the template file "svdvel.inx".
   Filenames, site coords/displacements, and various flags are returned.
   pjc feb '96
*/

#include<string.h>
#include<stdio.h>

#include "globdef.c"


void readinps(inpname,outname,nepochs,epoch,crdnames,covarnames, xcoord,ycoord,
xcent,ycent,xdisp,ydisp, sitename,nsites,ngoodsites,occup, funcname,residname,
dispsta, dispgrid,gradgrid,straingrid, gridscale,clipname,lonstep,latstep)

char *inpname,*outname,*funcname,*residname;
char *dispsta,*dispgrid,*gradgrid,*straingrid,*clipname;
```

```
char crdnames[][NAMELEN],covarnames[][NAMELEN];
char sitename[][NAMELEN];
double *xcoord,*ycoord,*xcent,*ycent;
double **xdisp,**ydisp;
double *epoch;
double *lonstep,*latstep,*gridscale;
int *nsites,*ngoodsites,*nepochs;
int **occup;

{
        FILE *inpfile;
        char dummystring[255];
        char gridtype[32];
        char dummy1[NAMELEN],dummy2[NAMELEN];
        double depoch;
        int i;


/* open file and check */
        inpfile=fopen(inpname,"r");
        if (inpfile == NULL) {
                printf("\n*** error opening input file '%s'\n",inpname);
                exit(1);
        }

/* skip header */
        for(i=1;i<=5;i++) fgets(dummystring,255,inpfile);

/* get filenames and general flags */
        fscanf(inpfile,"%s",outname);
        fgets(dummystring,255,inpfile);

        fscanf(inpfile,"%s",funcname);
        fgets(dummystring,255,inpfile);

        fscanf(inpfile,"%s",residname);
        fgets(dummystring,255,inpfile);

        fscanf(inpfile,"%s",dispsta);
        fgets(dummystring,255,inpfile);

        fscanf(inpfile,"%s",dispgrid);
        fgets(dummystring,255,inpfile);

        fscanf(inpfile,"%s",gradgrid);
        fgets(dummystring,255,inpfile);

        fscanf(inpfile,"%s",straingrid);
        fgets(dummystring,255,inpfile);

        fscanf(inpfile,"%s",gridtype);
        fgets(dummystring,255,inpfile);
        if (strcmp(gridtype,"GEODETIC") == 0) {
                *gridscale = 0.0;
        }
        else {
                sscanf(gridtype,"%lf",gridscale);
        }

        fscanf(inpfile,"%s",clipname);
        fgets(dummystring,255,inpfile);

        fscanf(inpfile,"%lf",lonstep);
        fgets(dummystring,255,inpfile);

        fscanf(inpfile,"%lf",latstep);
        fgets(dummystring,255,inpfile);

        fscanf(inpfile,"%d",nepochs);
        fgets(dummystring,255,inpfile);

        for(i=1;i<=3;i++) fgets(dummystring,255,inpfile);

/* get epochs, coord filenames, vcm filenames */
        for(i=1; i<=*nepochs; i++) {
                fscanf(inpfile,"%lf %s %s",&depoch,dummy1,dummy2);
                epoch[i] = depoch;
                strcpy(crdnames[i],dummy1);
                strcpy(covarnames[i],dummy2);
                fgets(dummystring,255,inpfile);
        }

/* get all coordinates */
        getcrds(*nepochs,crdnames,occup, xcoord,ycoord, xcent,ycent, xdisp, ydisp,
sitename, nsites,ngoodsites);

        fclose(inpfile);

}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* svbksb.c
  Taken directly from NR in C, section 2.6
/*
*******************************************************************************


*******************************************************************************
/* svdcmp.c
  Taken directly from NR in C, section 2.6
/*
*******************************************************************************


*******************************************************************************
/* adaptation of NR svdfit.c routine to fit velocity polynomials through coord
   data.  Site coordinates at the aribitrary zero epoch are estimated as
   nuisance parameters.  The new routine svedit is called to set the tolerance
   for singular values.  Also, the chisq computation has been modified from the
   original.
   double precision arithmetic used throughout
   modified by pjc feb '96
*/

#include<math.h>
#include<stdio.h>
#include "nr.h"
#include "nrutil.h"

#include "globdef.c"


void svdfit3(xcoord,ycoord,y,sig,nsites,ngoodsites,epoch,nepochs,occup,a,ma, u,v,w,
chisq,rms, funcs, xpow,ypow)
double xcoord[],ycoord[],**y,**sig,epoch[],a[],**u,**v,w[],*chisq,*rms;
int nsites,ngoodsites,nepochs,ma,xpow[],ypow[],**occup;
void (*funcs)();        /* ANSI: void (*funcs)(double,double *,int); */

{
        int i,j,k,ii;
```

```
        double tmp,sum,*b,*afunc;
        int ndata;

        b=dvector(1,nsites*nepochs);
        afunc=dvector(1,ma);

/* clear the design matrix */
        for(i=1; i<=ngoodsites*nepochs; i++)
                for(j=1; j<=ma+ngoodsites; j++)
                        u[i][j] = 0.0;

/* set up the design matrix and data vector */
        ndata = 0;
        for (k=1; k<=nepochs; k++) for (i=1,ii=0;i<=nsites;i++) {
                if (occup[0][i] > 1) ii++;
                if (occup[k][i] == 0) continue;
                ndata++;
                (*funcs)(xcoord[i],ycoord[i],afunc,ma,xpow,ypow);
                tmp=1.0/sqrt(sig[k][i]);
                for (j=1;j<=ma;j++) u[ndata][j]=afunc[j]*tmp*(epoch[k]-ZEPOCH);
                u[ndata][ma+ii] = tmp;
                b[ndata]=y[k][i]*tmp;
        }

/* decompose the system */
        printf("Calling singular value decomposition\n");
        svdcmp(u,ndata,ma+ngoodsites,w,v);


/* new bit to print and edit out the singular values */
/* for(i=1; i<=ma+ngoodsites; i++) printf("%3d %9.21e\n", i,w[i]); */
        svedit(w,ma,xpow,ypow);
/* end of new bit */

/* back-substitute to get coeffs */
        printf("\nBacksubstituting svd fit\n");
        svbksb(u,w,v,ndata,ma+ngoodsites,b,a);

/* bit to compute chisq is completely new */
        for (k=1; k<=nepochs; k++) for (i=1,ii=0;i<=nsites;i++) {
                if (occup[0][i] > 1) ii++;
                if (occup[k][i] == 0) continue;
                (*funcs)(xcoord[i],ycoord[i],afunc,ma,xpow,ypow);
                for (sum=0.0,j=1;j<=ma;j++) sum += a[j]*afunc[j];
                sum *= (epoch[k]-ZEPOCH);
                sum += a[ma+ii];
                *chisq += (tmp=(y[k][i]-sum),tmp*tmp) / sig[k][i];
        }
        *rms = sqrt(*chisq/ndata);
/* end of new bit */

        free_dvector(afunc,1,ma);
        free_dvector(b,1,nsites*nepochs);
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* svdvar.c
  Taken directly from NR in C, section 15.4
/*
*******************************************************************************


*******************************************************************************
/* subroutine to edit the singular values w[1..ma*2] and zero any which fall
   below the given tolerance (input from the keyboard).  The minimum permitted
   tolerance is given by MINTOL*wmax.
   pjc aug '94
*/

#include<math.h>
#include<stdio.h>
#include "nrutil.h"
#include "nr.h"

#include "globdef.c"


void svedit(w,ma,xpow,ypow)

int ma;
int xpow[],ypow[];
double *w;

{
        int j;
        double tol,wmax;


/* find largest singular value */
        for (j=1,wmax=0.0;j<=ma+1;j++)
                if (w[j] > wmax) wmax=w[j];


/* print singular values */
        printf("\nSingular values:\n");
        printf("    n   x^ y^      weight     rel.wt\n");
        for (j=1;j<=ma;j++) printf("%5d  %2d %2d  %13.4le  %8.6lf\n",
j,xpow[j],ypow[j],w[j],w[j]/wmax);


/* get minimum sv */
        printf("Enter minimum acceptable weight: ");
        scanf("%lf",&tol);
        if (tol < MINTOL*wmax) {
                printf("### Too small, increasing to MINTOL*wmax = %12.4le\n",
MINTOL*wmax);
                tol = MINTOL*wmax;
        }

/* zero smaller sv's than 'tol' */
        for (j=1;j<=ma;j++) if (w[j] < tol) w[j] = 0.0;
*/

}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to convert coordinates from cartesian (in m) to lat,lon (in
  radians) and ellipsoidal height (in m) by iterative method (see Heiskanen &
  Moritz for theory).

   pjc dec '95
*/

#include <math.h>
#include <stdio.h>
```

```
#include "globdef.c"


void xyz2llh(x,y,z, lat,lon,hgt)

double x,y,z;
double *lat,*lon,*hgt;

{
        double newlat,dlat,newhgt,dhgt,N,Lxy;

        *lon = atan2(y,x);

        Lxy = sqrt(x*x + y*y);
        *lat = atan2(z,(1.0-ECCEN*ECCEN)*Lxy);
        *hgt = 0.0;
        dlat = 0.5; dhgt = 0.5;

        while ((dlat > 1.0e-12) || (dhgt > 1.0e-6)) {
                N = RADIUS / (sqrt(1.0 - pow(ECCEN*sin(*lat),2.0)));
                newlat = atan2(z , (Lxy - ECCEN*ECCEN*N*cos(*lat)));
                newhgt = (Lxy/cos(*lat)) - N;
                dlat = fabs(*lat - newlat);
                dhgt = fabs(*hgt - newhgt);
                *lat = newlat; *hgt = newhgt;
        }
}


#include "globend.c"
*******************************************************************************
```

```
*******************************************************************************
Input option file for svdvel displacement polynomial fitting program
-------------------------------------------------------------------------------

Option                              Description
******************************      ---------------------------------------
test.out                            output filename / "NONE"
XY                                  function filename / "X" / "Y" / "XY"
test.res                            residual filename / "NONE"
testcomp.psvm                       computed velocity filename / "NONE"
test.vel                            velocity grid output filename / "NONE"
test.vgrd                           vel. grad. tensor output file / "NONE"
test.eps                            principal strain output file / "NONE"
GEODETIC                            grid scale (metres per unit) /"GEODETIC"
NONE                                clipping polygon filename / "NONE"
0.1                                 lon(X) grid step size (units)
0.1                                 lat(Y) grid step size (units)
2                                   no. of epochs

Epoch          Filename.CRD                 Filename.CEN / "UNIT"
*********      **************************   **************************
1991.78        91NNO_2A.CRD                 UNIT
1993.39        93NNO_2A.CRD                 UNIT
*******************************************************************************
```

C.3 polystr2

`polystr2` is a C program that estimates a velocity gradient tensor uniform in both time and space, from coordinate observations at a number of specified sites. From the velocity gradient tensor, further strain parameters are derived (Sections 3.3 and 3.4). For the solution to be determined, at least two epochs of coordinate measurement must exist at at least three sites. If more sites or epochs or measurement exist, the best-fitting solution is determined by a least-squares criterion, and the estimated goodness of fit will indicate whether or not a uniform velocity grtadient tensor describes the data well.

LU decomposition routines (from Press *et al.*, 1992) are used to invert the design matrix, because this procedure generates the inverse matrix which can be used to estimate the variance-covariance matrix of the parameters. The inversion is performed iteratively. The observations can be fully weighted with the coordinate variance-covariance matrices at each epoch, but epochs are regarded as independent of each other. To improve numerical stability, the geographic site coordinates (latitude, longitude) are regarded separately from the geodetic coordinates (stored as local north and east offsets from the coordinate at the first epoch of observation). For the purposes of strain calculations, geographic coordinates are expressed in kilometres north and east of the centroid of the specified sites.

The derived strain parameters are estimated directly from the velocity gradient tensor, but their variance-covariance matrices are estimated by Jacobian transformations of the velocity gradient tensor variance-covariance matrix, after Welsch (1983). For the majority of parameters, this linear transform is correct, but for errors in the azimuths of the principal and maximum strains the approximation $\tan\theta \simeq \theta$ is not valid for larger errors, and will tend to overestimate them.

`polystr2` has been compiled and run on Sun workstations using Sun C. Execution time is dominated by the overhead time while coordinate variance-covariance matrix files are read.

```
*******************************************************************************
/* polystr2.c
     Program to take site coordinates from .CRD files, and vcm from .CEN file or
  unit matrix, then compute the velocity gradient tensor and resulting strain
  parameters for polygonal regions specified by groups of 3 or more stations,
  assuming uniform strain within the polygon and with time.
    This program corrects the bug in PCE's program whereby the first station
  specified in each polygon is regarded as infallible, and also allows for input
  of covariances, and printing of residuals and chisq statistic.
    Uses normal equation inversion not SVD, so the parameter vcm is correct.
    Based on polystr.c which solves the simpler but related problem of two
  epochs of data only, with velocities in .psvm file.

  Pete Clarke, December 1995
*/

#include <math.h>
#include <stdio.h>
#include <string.h>
#include "nr.h"
#include "nrutil.h"

#include "globdef.c"


main(argc,argv)

int argc;
char **argv;

{
        char inpname[NAMELEN];                  /* input file name */
        char outname[NAMELEN];                  /* output file name */
        char crdnames[MAXEPO+1][NAMELEN];       /* coordinate file names */
        char covarnames[MAXEPO+1][NAMELEN];     /* vcm file names */
        char residname[NAMELEN];                /* output resid file name */
        char strname[NAMELEN];                  /* output principal strains filename
*/
        char polyname[NAMELEN];                 /* output polynomial vertex coords
file name */
        char sitename[MAXSITE+1][NAMELEN];      /* site names */
        double *epoch;                          /* epoch times */
        double *xcoord,*ycoord;                 /* site coords (geographic) */
        double *pexcoord,*peycoord;             /* polygon vertex coords (geographic,
relative to local origin) */
        double *pxcoord,*pycoord;               /* polygon vertex coords (km,
relative to local origin) */
        double pxorig,pyorig;                   /* geographic coords of polygon
centroid */
        double **xdisp,**ydisp;                 /* e,n coords at each epoch relative
to first epoch of occupation of each site */
        double *u0,*v0;                         /* zero-epoch coord of each vertex
(nuisance param) */
        double **comp_xdisp,**comp_ydisp;       /* computed site local coords at each
epoch */
        double ***cvdisp;                       /* vcm of coords at each epoch */
        double **pwgt;                          /* weight matrix for obs in polygon
*/
        double **xvcm,**gamvcm,**epsvcm;        /* vcms of parameter sets */
        double **jac;                           /* jacobian matrix for vcm transform
*/
        double **a,**u,**m,**mcopy;             /* arrays for LU decomposition and
vcm inversion */
        double *m_indx,*b,*x;                   /* vectors for above */
        double xres,yres;                       /* temp var for residual */
        double xtmp,ytmp;                       /* temp var for chisq calcs */
        double mdet;                            /* temp var for vcm inversion */
        double chisq;                           /* goodness of fit to strain */
        double sigx,sigy,correl;                /* temp vars for variances */
        register int i,j,k;                     /* counters */
        int ii,jj,k0;                           /* more counters! */
        int covtype;                            /* 1=unit, other=full vcm */
        int nsites;                             /* no. of sites */
        int npolys;                             /* no. of polygons */
        int nepochs;                            /* no. of survey epochs */
        int poly;                               /* counter for polygon */
        int nverts,nepverts;                    /* no. of vertices in polygon, no of
vertices occupied at epoch */
        int ndata,npar;                         /* no. of parameters in inversion,
no. of observations */
        int site;                               /* site counter */
        int wgterr;                             /* flag: 1=error in vcm inversion */
        int niters;                             /* count of iterations in inversion
*/
        int reiter;                             /* flag; 1= reiteration required */
        int **verts;                            /* matrix of polygon vertices */
        int **occup;                            /* matrix of site occupation flags */
        FILE *outfile,*resfile;                 /* output, residual file handles */
        FILE *strfile,*polyfile;                /* principal strain, polygon vertices
file handles */
        struct polyparams par;                  /* results of inversion for strain */


/* reserve memory for arrays */
        epoch = dvector(1,MAXEPO);
        xcoord = dvector(1,MAXSITE);
        ycoord = dvector(1,MAXSITE);
        pexcoord = dvector(1,MAXVERT);
        peycoord = dvector(1,MAXVERT);
        pxcoord = dvector(1,MAXVERT);
        pycoord = dvector(1,MAXVERT);
        u0 = dvector(1,MAXVERT);
        v0 = dvector(1,MAXVERT);
        xdisp = dmatrix(1,MAXEPO,1,MAXSITE);
        ydisp = dmatrix(1,MAXEPO,1,MAXSITE);
        comp_xdisp = dmatrix(1,MAXEPO,1,MAXVERT);
        comp_ydisp = dmatrix(1,MAXEPO,1,MAXVERT);
        cvdisp = d3tensor(1,MAXEPO,1,MAXSITE*NDIM,1,MAXSITE*NDIM);
        pwgt = dmatrix(1,MAXVERT*MAXEPO*NDIM,1,MAXVERT*MAXEPO*NDIM);
        xvcm = dmatrix(1,MAXPAR,1,MAXPAR);
        epsvcm = dmatrix(1,NDERIV,1,NDERIV);
        gamvcm = dmatrix(1,NDERIV,1,NDERIV);
        jac = dmatrix(1,NDERIV,1,MAXPAR);
        a = dmatrix(1,MAXVERT*MAXEPO*NDIM,1,MAXPAR);
        m = dmatrix(1,MAXPAR,1,MAXPAR);
        mcopy = dmatrix(1,MAXPAR,1,MAXPAR);
        m_indx = dvector(1,MAXPAR);
        u = dmatrix(1,MAXPAR,1,MAXVERT*MAXEPO*NDIM);
        x = dvector(1,MAXPAR);
        b = dvector(1,MAXPAR);
        verts = imatrix(1,MAXPOLY,0,MAXVERT);
        occup = imatrix(0,MAXEPO,0,MAXSITE);


/* print title */
        printf("\nEstimation of uniform strain in polygonal regions for more than two
epochs\n");
        printf("-------------------------------------------------------------------
------\n\n");


/* read input file, and coord files specified therein */
```

```
        if (argc == 2) strcpy(inpname,argv[1]);
        else strcpy(inpname,"polystr2.inp");
        readinps(inpname, &nepochs,epoch,crdnames,covarnames, outname, strname,
residname, polyname, xcoord,ycoord, xdisp,ydisp, sitename,&nsites,occup, &npolys,
verts);


/* load covariance matrices or set it to identity matrix if "UNIT" entered. */
        getcovs(covarnames,nepochs,sitename,occup,cvdisp,nsites,&covtype);


/* open output file for parameters, if required */
        outfile = NULL;
        if (strcmp(outname,"NONE") != 0) {
                outfile = fopen(outname,"w");
                if (outfile == NULL) {
                        printf("\n*** error opening output file \"%s\"\n", outname);
                        exit();
                }
                for(i=1; i<=80; i++) fprintf(outfile,"-");
        }


/* open residual output file, if required */
        resfile = NULL;
        if (strcmp(residname,"NONE") != 0) {
                resfile = fopen(residname,"w");
                if (resfile == NULL) {
                        printf("\n*** error opening residual output file \"%s\"\n",
residname);
                        exit();
                }
        }


/* open principal strain output file, if required */
        strfile = NULL;
        if (strcmp(strname,"NONE") != 0) {
                strfile = fopen(strname,"w");
                if (strfile == NULL) {
                        printf("\n*** error opening principal strain output file
\"%s\"\n", strname);
                        exit();
                }
        }


/* open polygon vertex coords output file, if required */
        polyfile = NULL;
        if (strcmp(polyname,"NONE") != 0) {
                polyfile = fopen(polyname,"w");
                if (polyfile == NULL) {
                        printf("\n*** error opening polygon vertex output file
\"%s\"\n", polyname);
                        exit();
                }
                fprintf(polyfile,">\n");
        }


/* print a pretty line of dashes */
        printf("\n");
        for(i=1; i<=80; i++) printf("-");
        printf("\n");
        if (outfile != NULL) fprintf(outfile,"\n");
```

```
/* loop over all polygons */
        for(poly=1; poly<=npolys; poly++) {
                printf("\nPolygon: %d\n",poly);
                if (outfile != NULL) fprintf(outfile,"\nPolygon: %d\n",poly);


        /* determine centroid of polygon, print out vertices, compute origin */
                printf("Vertices: ");
                if (outfile != NULL) fprintf(outfile,"Vertices: ");
                nverts = verts[poly][0];
                pxorig = pyorig = 0.0;
                for(i=1; i<=nverts; i++) {
                        site = verts[poly][i];
                        pxorig += xcoord[site];
                        pyorig += ycoord[site];
                        pexcoord[i] = xcoord[site];
                        peycoord[i] = ycoord[site];
                        printf("%s ",sitename[site]);
                        if (outfile != NULL) fprintf(outfile,"%s ", sitename[site]);
                }
                printf("\n\n");
                if (outfile != NULL) fprintf(outfile,"\n\n");
                pxorig /= (double)nverts;
                pyorig /= (double)nverts;
                printf("Centroid at %10.6lf , %10.6lf\n\n",pxorig,pyorig);
                if (outfile != NULL) fprintf(outfile,"Centroid at %10.6lf ,
%10.6lf\n\n",pxorig,pyorig);


        /* determine the no. of data points and parameters for this polygon */
                ndata = 0;
                for(i=1; i<=nverts; i++) {
                        site = verts[poly][i];
                        ndata += occup[0][site];
                }
                occup[0][0] = ndata;
                npar = VBAR + NDIM*nverts;


        /* form covariance matrix and invert it to get weight matrix */
                for(i=1; i<=NDIM*ndata;i++)
                        for(j=1; j<=NDIM*ndata;j++)
                                pwgt[i][j] = 0.0;

                if (covtype == 1) {
                        for(i=1; i<=NDIM*ndata; i++) pwgt[i][i] = 1.0;
                }
                else {
                        wgterr = makewgt(poly,verts,cvdisp,pwgt,nsites,nepochs,
occup);
                        if (wgterr) {
                                printf("*** problems exist with inversion of
displacement vcm\n\n");
                                if (outfile != NULL) fprintf(outfile,"*** problems
exist with inversion of displacement vcm\n\n");
                        }
                }


        /* convert to local coords (relative to centroid) in km */
                ell2xy(pexcoord,peycoord, pxcoord,pycoord, nverts,1, pxorig,pyorig);


        /* set up the design matrix a[1..NDIM*ndata][1..npar] */
                for(i=1; i<=NDIM*ndata; i++)
                        for(j=1; j<=npar; j++)
```

```
                        a[i][j] = 0.0;
                k0 = 0;
                for(k=1; k<=nepochs; k++) {
                        nepverts = 0;
                        for(i=1; i<=nverts; i++) {
                                site = verts[poly][i];
                                if (occup[k][site] == 1) nepverts++;
                        }
                        for(i=1,ii=0; i<=nverts; i++) {
                                if (occup[k][i] ==0) continue;
                                ii++;
                                a[k0+ii][UX] = pxcoord[i] * (epoch[k]-ZEPOCH);
                                a[k0+ii][UY] = pycoord[i] * (epoch[k]-ZEPOCH);
                                a[k0+ii][UBAR] = (epoch[k]-ZEPOCH);
                                a[k0+ii][VBAR+2*i-1] = 1.0;
                                a[nepverts+k0+ii][VX] = pxcoord[i] *
(epoch[k]-ZEPOCH);
                                a[nepverts+k0+ii][VY] = pycoord[i] *
(epoch[k]-ZEPOCH);
                                a[nepverts+k0+ii][VBAR] = (epoch[k]-ZEPOCH);
                                a[nepverts+k0+ii][VBAR+2*i] = 1.0;
                        }
                        k0 += NDIM*nepverts;
                }


        /* multiply the design matrix transpose [a-t] by the weight matrix
           [pwgt] to get [u] (temp matrix) */
                matmult(a,pwgt,u, npar,NDIM*ndata,NDIM*ndata, 1,0);


        /* multiply [u] by [a] to get the normal equation matrix [m],
           copy [m] back into [mcopy] as ludcmp() will overwrite it,
           then do the LU-decomposition */
                matmult(u,a,m, npar,NDIM*ndata,npar, 0,0);
                for(i=1; i<=npar; i++) for(j=1; j<=npar; j++) mcopy[i][j] = m[i][j];
                ludcmp(m,npar,m_indx,&mdet);


        /* a priori parameters are all zero */
                par.ux = par.uy = 0.0;
                par.vx = par.vy = 0.0;
                par.ubar = par.vbar = 0.0;
                for(i=1; i<=nverts; i++)
                        u0[i] = v0[i] = 0.0;


        /* start of iterative non-linear least squares process */
                niters = 0;
                do {
                        niters++;


                /* calculate computed displacements at sites */
                        for(k=1; k<=nepochs; k++) {
                                for(i=1; i<=nverts; i++) {
                                        comp_xdisp[k][i] = (pxcoord[i]*par.ux +
pycoord[i]*par.uy + par.ubar) * (epoch[k]-ZEPOCH) + u0[i];
                                        comp_ydisp[k][i] = (pxcoord[i]*par.vx +
pycoord[i]*par.vy + par.vbar) * (epoch[k]-ZEPOCH) + v0[i];
                                }
                        }


                /* premultiply the a priori values b'[] by [u] to
                   get b[] (=[a-t][pwgt]b'[]) and copy b[] into x[] */
                        for(i=1; i<=npar; i++) {
                                b[i] = 0.0;
                                for(k=1,k0=0; k<=nepochs; k++) {
                                        for(j=1,nepverts=0; j<=nverts; j++) {
                                                site = verts[poly][j];
                                                if (occup[k][site] == 1) nepverts++;
                                        }
                                        for(j=1,jj=0; j<=nverts; j++) {
                                                site = verts[poly][j];
                                                if (occup[k][site] == 0) continue;
                                                jj++;
                                                b[i] += u[i][k0+jj] * (xdisp[k][site]
- comp_xdisp[k][j]) + u[i][k0+nepverts+jj] * (ydisp[k][site] - comp_ydisp[k][j]);
                                        }
                                        k0 += NDIM*nepverts;
                                }
                                x[i] = b[i];
                        }

                /* back-substitute the LU-decomposition of [m] into b to get
                   the parameter vector x[] (after iterative improvement of the
                   LU back-substitution) */
                        lubksb(m,npar,m_indx,x);
                        mprove(mcopy,m, npar, m_indx, b,x);


                /* check for max change to parameter vector, set reiter flag
                   if re-iteration necessary */
                        reiter = 0;
                        for(i=1; i<=npar; i++) if (x[i] > PARTOL) reiter = 1;


                /* interpret parameters in x[].  Units for coords were km,
                   displacements were in mm, so strains are in ppm */
                        par.ux += x[UX];
                        par.uy += x[UY];
                        par.vx += x[VX];
                        par.vy += x[VY];
                        par.ubar += x[UBAR];
                        par.vbar += x[VBAR];
                        for(i=1; i<=nverts; i++) {
                                u0[i] += x[VBAR+2*i-1];
                                v0[i] += x[VBAR+2*i];
                        }


                /* end of iterative least-squares process */
                } while ((reiter == 1) && (niters < MAXITER));


        /* back-substitute [m] into unit vectors to get the parameter vcm [xvcm]
           with iterative improvement.  Vectors b[] and x[] are overwritten. */
                for(j=1; j<=npar; j++) {
                        for(i=1; i<=npar; i++) b[i] = x[i] = 0.0;
                        b[j] = x[j] = 1.0;
                        lubksb(m,npar,m_indx,x);
                        mprove(mcopy,m, npar,m_indx, b,x);
                        for(i=1; i<=npar; i++) xvcm[i][j] = x[i];
                }


        /* calculate computed displacements at sites */
                for(k=1; k<=nepochs; k++) {
                        for(i=1; i<=nverts; i++) {
                                comp_xdisp[k][i] = (pxcoord[i]*par.ux +
```

```
pycoord[i]*par.uy + par.ubar) * (epoch[k]-ZEPOCH) + u0[i];
                        comp_ydisp[k][i] = (pxcoord[i]*par.vx +
pycoord[i]*par.vy + par.vbar) * (epoch[k]-ZEPOCH) + v0[i];
                        }
                }


        /* calculate chisq statistic and print it */
                chisq = 0.0;
                k0 = 0;
                for(k=1; k<=nepochs; k++) {
                        ii = 0; nepverts = 0;
                        for(i=1; i<=nverts; i++) {
                                site = verts[poly][i];
                                if (occup[k][site] == 1) nepverts++;
                        }
                        for(i=1; i<=nverts; i++) {
                                xtmp = ytmp = 0.0;
                                site = verts[poly][i];
                                if (occup[k][site] == 0) continue;
                                ii++; jj = 0;
                                for(j=1; j<=nverts; j++) {
                                        site = verts[poly][j];
                                        if (occup[k][site] == 0) continue;
                                        jj++;
                                        xres = xdisp[k][site] - comp_xdisp[k][j];
                                        yres = ydisp[k][site] - comp_ydisp[k][j];
                                        xtmp += pwgt[k0+ii][k0+jj] * xres +
pwgt[k0+ii][k0+nepverts+jj] * yres;
                                        ytmp += pwgt[nepverts+ii][jj] * xres +
pwgt[nepverts+ii][nepverts+jj] * yres;
                                }
                                site = verts[poly][i];
                                chisq += (xdisp[k][site] - comp_xdisp[k][i]) * xtmp;
                                chisq += (ydisp[k][site] - comp_ydisp[k][i]) * ytmp;
                        }
                        k0 += nepverts;
                }
                if (reiter == 0) printf("No. of iterations: %d\n",niters);
                else printf("*** SOLUTION HAS NOT CONVERGED AFTER %d ITERATIONS
***\n", niters);
                printf("Badness of fit (chisq): %10.4le\n",chisq);
                printf("RMS weighted residual : %10.4le\n\n",
sqrt(chisq/(double)(NDIM*ndata)));
                if (outfile != NULL) {
                        if (reiter == 0) fprintf(outfile,"No. of iterations:
%d\n",niters);
                        else fprintf(outfile,"*** SOLUTION HAS NOT CONVERGED ***\n");
                        fprintf(outfile,"Badness of fit (chisq): %11.4le\n", chisq);
                        fprintf(outfile,"RMS weighted residual : %10.4le\n\n",
sqrt(chisq/(double)(NDIM*ndata)));
                }


        /* convert to strain tensor, gammas and principal strains */
                par.rot = 0.5 * (par.uy - par.vx);
                par.dil = par.ux + par.vy;

                par.exx = par.ux;
                par.eyy = par.vy;
                par.exy = 0.5 * (par.uy + par.vx);

                par.gam1 = par.ux - par.vy;
                par.gam2 = par.uy + par.vx;
                par.tgam = hypot(par.gam1,par.gam2);

                par.eps1 = 0.5 * (par.dil + par.tgam);
                par.eps2 = 0.5 * (par.dil - par.tgam);

                if (par.gam2 == 0.0) par.azim2 = 0.0;
                else par.azim2 = atan((par.gam1-par.tgam)/par.gam2) / DEG2RAD;
                par.gazim = par.azim2 - 45.0;
                if (par.gazim < -45.0) par.gazim += 180.0;


        /* obtain vcm of gammas, rotation and dilatation by Jacobian xform */
                jac[GAM1][UX] = 1.0;
                jac[GAM1][UY] = jac[GAM1][VX] = 0.0;
                jac[GAM1][VY] = -1.0;

                jac[GAM2][UX] = 0.0;
                jac[GAM2][UY] = jac[GAM2][VX] = 1.0;
                jac[GAM2][VY] = 0.0;

                jac[ROT][UX] = 0.0;
                jac[ROT][UY] = 0.5;
                jac[ROT][VX] = -0.5;
                jac[ROT][VY] = 0.0;

                jac[DIL][UX] = 1.0;
                jac[DIL][UY] = jac[DIL][VX] = 0.0;
                jac[DIL][VY] = 1.0;

                jacxform(xvcm,VY,jac,gamvcm,NDERIV);


        /* obtain vcm of total gamma, principal strains and Eps2 azimuth by
           Jacobian xform */
                jac[TGAM][GAM1] = par.gam1 / par.tgam;
                jac[TGAM][GAM2] = par.gam2 / par.tgam;
                jac[TGAM][ROT] = jac[TGAM][DIL] = 0.0;

                jac[EPS1][GAM1] = 0.5 * par.gam1 / par.tgam;
                jac[EPS1][GAM2] = 0.5 * par.gam2 / par.tgam;
                jac[EPS1][ROT] = 0.0;
                jac[EPS1][DIL] = 0.5;

                jac[EPS2][GAM1] = -0.5 * par.gam1 / par.tgam;
                jac[EPS2][GAM2] = -0.5 * par.gam2 / par.tgam;
                jac[EPS2][ROT] = 0.0;
                jac[EPS2][DIL] = 0.5;

        /* NB this is the error in tan(2theta) */
                jac[AZIM2][GAM1] = par.gam2 / (par.gam1 * par.gam1);
                jac[AZIM2][GAM2] = -1.0 / par.gam1;
                jac[AZIM2][ROT] = jac[AZIM2][DIL] = 0.0;

                jacxform(gamvcm,NDERIV,jac,epsvcm,NDERIV);
        /* convert error in tan(2theta) to error in theta */
                epsvcm[AZIM2][AZIM2] = 0.5 * atan(epsvcm[AZIM2][AZIM2]);


        /* output parameters etc to screen */
                invprint(stdout, &par, epsvcm,gamvcm,xvcm);
                for(i=1; i<=80; i++) printf("-");
                printf("\n");


        /* output parameters etc to file if reqd */
                if (outfile != NULL) {
                        invprint(outfile, &par, epsvcm,gamvcm,xvcm);
```

```
                        fprintf(outfile,"\n\n\n");
                        for(i=1; i<=80; i++) fprintf(outfile,"-");
                        fprintf(outfile,"\n");
                }


        /* output residuals to residual .psvm file if reqd */
                if (resfile != NULL) {
                        for(k=1; k<=nepochs; k++) {
                                fprintf(resfile,"# Residuals for polygon %d at epoch
%7.2lf\n",poly,epoch[k]);
                                for(i=1; i<=nverts; i++) {
                                        site = verts[poly][i];
                                        if (occup[k][site] == 0) continue;
                                        sigx = sqrt(cvdisp[k][site][site]);
                                        sigy =
sqrt(cvdisp[k][nsites+site][nsites+site]);
                                        correl = 0.0;
                                        if ((sigx != 0.0) && (sigy != 0.0))
                                                correl = cvdisp[k][site][nsites+site]
/ (sigx*sigy);
                                        fprintf(resfile,"%10.6lf %10.6lf  %8.4lf
%8.4lf  %6.4lf %6.4lf %6.4lf %s\n", xcoord[site], ycoord[site],
(xdisp[k][site]-comp_xdisp[k][i])/1000.0, (ydisp[k][site]-comp_ydisp[k][i])/1000.0,
sigx/1000.0,sigy/1000.0,correl, sitename[site]);
                                        }
                        }
                        fprintf(resfile,"\n");
                }


        /* output principal strains to file if reqd */
                if (strfile != NULL) {
                        fprintf(strfile,"%10.6lf %10.6lf  %8.4lf %8.4lf  %7.2f\n",
pxorig, pyorig, par.eps1,par.eps2,par.azim2);
                }


        /* output polygon vertices to file if reqd */
                if (polyfile != NULL) {
                        for(i=1; i<=nverts; i++) {
                                site = verts[poly][i];
                                fprintf(polyfile,"%10.6lf %10.6lf\n", xcoord[site],
ycoord[site]);
                        }
                        fprintf(polyfile,">\n");
                }

/* end of loop over all polygons */
        }

        if (outfile != NULL) fclose(outfile);
        if (resfile != NULL) fclose(resfile);

        printf("\nFinished.\n\n");

/* free memory for arrays */
        free_dvector(epoch,1,MAXEPO);
        free_dvector(xcoord,1,MAXSITE);
        free_dvector(ycoord,1,MAXSITE);
        free_dvector(pexcoord,1,MAXVERT);
        free_dvector(peycoord,1,MAXVERT);
        free_dvector(pxcoord,1,MAXVERT);
        free_dvector(pycoord,1,MAXVERT);
        free_dmatrix(xdisp,1,MAXEPO,1,MAXSITE);
        free_dmatrix(ydisp,1,MAXEPO,1,MAXSITE);
        free_dmatrix(comp_xdisp,1,MAXEPO,1,MAXVERT);
        free_dmatrix(comp_ydisp,1,MAXEPO,1,MAXVERT);
        free_d3tensor(cvdisp,1,MAXEPO,1,MAXSITE*NDIM,1,MAXSITE*NDIM);
        free_dmatrix(pwgt,1,MAXVERT*MAXEPO*NDIM,1,MAXVERT*MAXEPO*NDIM);
        free_dmatrix(xvcm,1,MAXPAR,1,MAXPAR);
        free_dmatrix(epsvcm,1,NDERIV,1,NDERIV);
        free_dmatrix(gamvcm,1,NDERIV,1,NDERIV);
        free_dmatrix(jac,1,NDERIV,1,MAXPAR);
        free_dmatrix(a,1,MAXVERT*MAXEPO*NDIM,1,MAXPAR);
        free_dmatrix(m,1,MAXPAR,1,MAXPAR);
        free_dmatrix(mcopy,1,MAXPAR,1,MAXPAR);
        free_dmatrix(u,1,MAXPAR,1,MAXVERT*MAXEPO*NDIM);
        free_dvector(m_indx,1,MAXPAR);
        free_dvector(x,1,MAXPAR);
        free_dvector(b,1,MAXPAR);
        free_dvector(u0,1,MAXVERT);
        free_dvector(y0,1,MAXVERT);
        free_imatrix(verts,1,MAXPOLY,0,MAXVERT);
        free_imatrix(occup,0,MAXEPO,0,MAXSITE);
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to convert site coords in the arrays xx[],yy[] into
  x-y coords in km in the arrays x[],y[].  If coortype==1,
  conversion from lat/lon to km must be done, otherwise the coords are just
  copied over (relative to the centroid).

  pjc aug '95
*/

#include <stdio.h>
#include <math.h>
#include "nrutil.h"

#include "globdef.c"


void ell2xy(xx,yy,x,y,ncoords,coortype,orig_x,orig_y)

double *x,*y,*xx,*yy;
int ncoords,coortype;
double orig_x,orig_y;

{
        int i;
        double rprimev,rmerid,tmp;


        if (coortype==1) {
                rprimev = RADIUS / sqrt(1 - pow((ECCEN*sin(orig_y*DEG2RAD)),2.0) );
                tmp = 1 - pow((ECCEN*sin(orig_y*DEG2RAD)),2.0);
                rmerid = RADIUS*(1 - pow(ECCEN,2.0)) / (tmp*sqrt(tmp));
        }

        for(i=1; i<=ncoords; i++) {
                x[i] = xx[i]-orig_x;
                y[i] = yy[i]-orig_y;
                if (coortype==1) {
                        x[i] *= (rprimev/1000.0)*cos(orig_y*DEG2RAD) * DEG2RAD;
                        y[i] *= (rmerid/1000.0) * DEG2RAD;
                }
```

```
        }
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to find the station number of a named station ie its position
   in the list namelist[1..nsites], or return -999 if station not found
   pjc aug '95
*/

#include <string.h>

#include "globdef.c"


int findstnum(name,namelist,nsites)

char name[];
char namelist[][NAMELEN];
int nsites;

{
        int i;
        int found=0;

        for(i=1;found<1;i++) {
                if (i > nsites) {
                        found=2;
                }
                if (strcmp(name,namelist[i]) == 0) {
                        found=1;
                }
        }
        if (found == 1) {
                return(i-1);
        } else {
                return(-999);
        }
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to read in a covariance matrix from a file in .CEN format (which
   is equivalent to the Bernese 3.4 format, but having had a similarity
   transform applied so that the components are in the coordinate system of
   east, north, up appropriate at the centroid of the network).
   Covariances are stored in units of mm.
   Modified dec '95 for multiple covar files for polystr2.

   pjc aug '95
*/

#include <string.h>
#include <stdio.h>

#define SIGSQMIN 4.0            /* default std error of a fixed coord is 2mm */
#define SIGSQTOL 1.0e-6         /* 'zero' variance of a fixed coordinate */

#include "globdef.c"


void getcovs(covarnames,nepochs,sitename,occup,cvmatrix,nsites,covtype)

char covarnames[][NAMELEN],sitename[][NAMELEN];
double ***cvmatrix;
int nsites,nepochs,**occup,*covtype;

{
        FILE *covarfile;
        int status;
        int i,j,k,n;
        int stnum1,stnum2;
        char st1[NAMELEN],st2[NAMELEN];
        char neu1[3],neu2[3];
        char dummystring[255];
        double elem,unitwgt;


/* loop over all epochs */
        for(k=1; k<=nepochs; k++) {
                printf("\nLoading covariance submatrix %s ...\n", covarnames[k]);

        /* zero vcm at start to avoid accidents */
                for(i=1;i<=nsites*2;i++)
                        for(j=1;j<=nsites*2;j++)
                                cvmatrix[k][i][j] = 0.0;
                *covtype = 0;

        /* if unit vcm, just make it */
                if (strcmp(covarnames[k],"UNIT") == 0) {
                        for(i=1; i<=nsites*2; i++) cvmatrix[k][i][i] = 1.0;
                        *covtype = 1;
                }

        /* otherwise load up and check */
                else {
                        covarfile=fopen(covarnames[k],"r");
                        if (covarfile == NULL) {
                                printf("\n*** error opening file:
'%s'\n",covarnames[k]);
                                exit();
                        }
                        status=0;
                        for(i=1;i<=6;i++) fgets(dummystring,255,covarfile);
                        fscanf(covarfile, "RMS OF UNIT WEIGHT: %lf", &unitwgt);
                        unitwgt *= 1.0e3;       /* convert to mm */
                        for(i=1;i<=5;i++) fgets(dummystring,255,covarfile);
                        while (status != EOF) {
                                status = fscanf(covarfile,"%s %s %s %s %lf\n", st1,
neu1, st2, neu2, &elem);
                                if ((status != 0) && (status != EOF)) {
                                        stnum1=findstnum(st1,sitename,nsites);
                                        stnum2=findstnum(st2,sitename,nsites);
                                        i=j=-999;
                                        if (strcmp(neu1,"E") == 0) i=0;
                                        if (strcmp(neu1,"N") == 0) i=nsites;
                                        if (strcmp(neu2,"E") == 0) j=0;
                                        if (strcmp(neu2,"N") == 0) j=nsites;
                                        i+=stnum1; j+=stnum2;
                                        if ((i>0) && (j>0)) {
                                                cvmatrix[k][i][j] = elem *
unitwgt*unitwgt;
                                                cvmatrix[k][j][i] = elem *
unitwgt*unitwgt;
                                        }
                                }
                        }
                        fclose(covarfile);
                }
```

```
        /* check covariance matrix for zero variances, adjust them to SIGSQMIN */
                for(i=1;i<=nsites;i++) {
                        if (occup[k][i] == 1) {
                                if (cvmatrix[k][i][i] < SIGSQTOL) {
                                        printf("### Variance of  %s:E increased from
%10.3e to %10.3le\n",sitename[i],cvmatrix[k][i][i],SIGSQMIN);
                                        cvmatrix[k][i][i] = SIGSQMIN;
                                }
                                if (cvmatrix[k][i+nsites][i+nsites] < SIGSQTOL) {
                                        printf("### Variance of  %s:N increased from
%10.3e to %10.3le\n",sitename[i],cvmatrix[k][i+nsites][i+nsites],SIGSQMIN);
                                        cvmatrix[k][i+nsites][i+nsites] = SIGSQMIN;
                                }
                        }
                }
        }
}


#undef SIGSQMIN
#undef SIGSQTOL


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to get coordinates and sitenames from B3.4 .CRD files, then make
   displacements (in mm) in local e,n coords for each epoch, relative to the
   first epoch of observation.

   pjc dec '95
*/

#include<math.h>
#include<stdio.h>
#include<string.h>

#include "nr.h"
#include "nrutil.h"

#include "globdef.c"


void getcrds(nepochs,crdnames,occup,excoord,eycoord,xdisp,ydisp,sitename,nsites)

char crdnames[][NAMELEN],sitename[][NAMELEN];
double *excoord,*eycoord;
double **xdisp,**ydisp;
int *nsites,nepochs;
int **occup;

{
        FILE *crdfile;
        char dummystring[255],dummy[NAMELEN],*flag,*crdflag[2];
        int i,j,n,site,status;
        double cx,cy,cz;
        double *cx0,*cy0,*cz0;
        double lat,lon,hgt;
        double dx,dxy,dy,dz;


        cx0 = dvector(1,MAXSITE);
        cy0 = dvector(1,MAXSITE);
        cz0 = dvector(1,MAXSITE);


/* zero everything at start for safety */
        *nsites = 0;
        for(i=0; i<=MAXEPO; i++) for(j=0; j<=MAXSITE; j++) occup[i][j] = 0;

/* loop over all epochs */
        for(i=1; i<=nepochs; i++) {
                crdfile=fopen(crdnames[i],"r");
                if (crdfile == NULL) {
                        printf("\n*** error opening coordinate file: '%s'\n",
crdnames[i]);
                        exit();
                }
                printf("Reading coordinate file \"%s\" ... ", crdnames[i]);
                for(j=1; j<=6; j++) fgets(dummystring,255,crdfile);

        /* get all coords with flag set */
                status=0; n=0;
                while (status != EOF) {
                        flag = fgets(dummystring,255,crdfile);
                        if (flag == NULL) break;
                        strcpy(crdflag,"");
                        status = sscanf(dummystring,"%*d %s %lf %lf %lf %s", dummy,
&cx,&cy,&cz, crdflag);
                        if (status == EOF) break;
                        if (strcmp(crdflag,"") != 0) {
                                n++;
                                site = findstnum(dummy,sitename,*nsites);
                                if (site < 0) {
                                        (*nsites)++;
                                        if (*nsites > MAXSITE) {
                                                printf("\n*** maximum number of sites
exceeded\n");
                                                printf("    number of sites >= %d\n",
*nsites);
                                                exit();
                                        }
                                        site = *nsites;
                                        strcpy(sitename[site],dummy);
                                        cx0[site] = cx;
                                        cy0[site] = cy;
                                        cz0[site] = cz;
                                }
                                occup[i][site] = 1;
                                occup[0][site] += 1;

                        /* get ellipsoidal coords */
                                xyz2llh(cx,cy,cz, &lat,&lon,&hgt);
                                excoord[site] = lon / DEG2RAD;
                                eycoord[site] = lat / DEG2RAD;

                        /* get cartesian coords relative to first occupation */
                                dx = cx - cx0[site];
                                dy = cy - cy0[site];
                                dz = cz - cz0[site];

                        /* transform local coords into local enu ref frame */
                                dxy = dx*cos(lon) + dy*sin(lon);
                                xdisp[i][site] = (dy*cos(lon) - dx*sin(lon)) *
1000.0;
                                ydisp[i][site] = (dz*cos(lat) - dxy*sin(lat)) *
1000.0;
                        }
                }

                occup[i][0] = n;
                printf("%d sites found\n", n);
                fclose(crdfile);
        }
```

```
        printf("Total of %d sites found\n", *nsites);


        free_dvector(cx0,1,MAXSITE);
        free_dvector(cy0,1,MAXSITE);
        free_dvector(cz0,1,MAXSITE);
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* globdef.c

  macro and structure definitions for polystr2.c and subroutines */


/* Constants and array sizes */
#define BIGNUM  1.0e30          /* (almost) largest representable number */
#define DEG2RAD 0.017453293     /* degrees to radians factor */
#define ECCEN   0.081819191     /* eccentricity of GRS-80 ellipsoid */
#define INVTOL  1.0e-10         /* tolerance for goodness of vcm inversion */
#define MAXEPO  5               /* max no. of epochs in survey */
#define MAXITER 10              /* max no. of least-squares iterations */
#define MAXPAR  44              /* max no. of params per region (polystr2) */
#define MAXPOLY 150             /* max no. of polygonal regions */
#define MAXSITE 100             /* max no. of sites */
#define MAXVERT 20              /* max no. of stations in polygonal region */
#define NAMELEN 32              /* max length of site names */
#define NDERIV  4               /* no. of derived parameters per vcm */
#define NDIM    2               /* no. of dimensions in problem */
#define NPAR    6               /* no. of parameters per region (polystr) */
#define PARTOL  1.0e-6          /* max change in param vector for convergence */
#define RADIUS  6378137.0       /* radius of GRS-80 ellipsoid */
#define SVTOL   1.0e-8          /* singularity tolerance for singular values */
#define ZEPOCH  1988.00         /* zero epoch for coordinate diffs */


/* parameter numbers for fitted parameters */
#define UX      1               /* du/dx */
#define UY      2               /* du/dy */
#define VX      3               /* dv/dx */
#define VY      4               /* dv/dy */
#define UBAR    5               /* u-offset */
#define VBAR    6               /* v-offset */


/* parameter numbers for derived parameters */
#define GAM1    1               /* gamma-1 */
#define GAM2    2               /* gamma-2 */
#define ROT     3               /* rigid-body rotation */
#define DIL     4               /* areal dilatation */
#define TGAM    1               /* total gamma */
#define EPS1    2               /* most extensional principal strain */
#define EPS2    3               /* most compressional principal strain */
#define AZIM2   4               /* azimuth of EPS2 */


/* structure for results in polystr */
struct polyparams {
        double ux,uy,vx,vy;
        double ubar,vbar;
        double exx,exy,eyy;
        double rot,dil;
        double eps1,eps2,azim2;
        double gam1,gam2,tgam,gazim;
};
*******************************************************************************


*******************************************************************************
/* globend.c

   undfeine macros from polystr2
*/

#undef BIGNUM
#undef DEG2RAD
#undef ECCEN
#undef INVTOL
#undef MAXEPO
#undef MAXITER
#undef MAXPAR
#undef MAXPOLY
#undef MAXSITE
#undef MAXVERT
#undef NAMELEN
#undef NDERIV
#undef NDIM
#undef NPAR
#undef PARTOL
#undef RADIUS
#undef SVTOL
#undef ZEPOCH

#undef UX
#undef UY
#undef VX
#undef VY
#undef UBAR
#undef VBAR

#undef GAM1
#undef GAM2
#undef ROT
#undef DIL
#undef TGAM
#undef EPS1
#undef EPS2
#undef AZIM2
*******************************************************************************


*******************************************************************************
/* subroutine to print out results of polygon strain to specified file (which
   could be stdout).

   pjc aug '95
*/

#include <stdio.h>
#include <math.h>
#include "nrutil.h"

#include "globdef.c"


void invprint(outfile, par, epsvcm,gamvcm,parvcm)

FILE *outfile;
struct polyparams *par;
double **epsvcm,**gamvcm,**parvcm;

{
```

```
        fprintf(outfile,"Velocity gradient tensor (ppm/yr):\n");
        fprintf(outfile,"du/dx = %9.4f  +- %9.4f\n",par->ux, sqrt(parvcm[UX][UX]));
        fprintf(outfile,"du/dy = %9.4f  +- %9.4f\n",par->uy, sqrt(parvcm[UY][UY]));
        fprintf(outfile,"dv/dx = %9.4f  +- %9.4f\n",par->vx, sqrt(parvcm[VX][VX]));
        fprintf(outfile,"dv/dy = %9.4f  +- %9.4f\n",par->vy, sqrt(parvcm[VY][VY]));
        fprintf(outfile,"\n");

        fprintf(outfile,"Symmetric strain rate tensor (ppm/yr):\n");
        fprintf(outfile,"Exx   = %9.4f  +- %9.4f\n",par->exx, sqrt(parvcm[UX][UX]));
        fprintf(outfile,"Exy   = %9.4f  +- %9.4f\n",par->exy,
sqrt(0.25*(parvcm[UY][UY]+2.0*parvcm[UY][VX]+parvcm[VX][VX])));
        fprintf(outfile,"Eyy   = %9.4f  +- %9.4f\n",par->eyy, sqrt(parvcm[VY][VY]));
        fprintf(outfile,"\n");

        fprintf(outfile,"Rigid-body rotation rate (microdeg/yr):\n");
        fprintf(outfile,"Rot   = %9.4f  +- %9.4f\n", (par->rot)/DEG2RAD,
sqrt(gamvcm[ROT][ROT])/DEG2RAD);
        fprintf(outfile,"\n");

        fprintf(outfile,"Dilatation rate (ppm/yr):\n");
        fprintf(outfile,"Dil   = %9.4f  +- %9.4f\n",par->dil,
sqrt(gamvcm[DIL][DIL]));
        fprintf(outfile,"\n");

        fprintf(outfile,"Principal strain rates, azimuth of Eps2 (ppm/yr,deg):\n");
        fprintf(outfile,"Eps1  = %9.4f  +- %9.4f\n",par->eps1,
sqrt(epsvcm[EPS1][EPS1]));
        fprintf(outfile,"Eps2  = %9.4f  +- %9.4f\n",par->eps2,
sqrt(epsvcm[EPS2][EPS2]));
        fprintf(outfile,"Azim2 = %9.4f  +- %9.4f\n",par->azim2,
sqrt(epsvcm[AZIM2][AZIM2])/DEG2RAD);
        fprintf(outfile,"\n");

        fprintf(outfile,"Gammas (ppm/yr):\n");
        fprintf(outfile,"Gam1  = %9.4f  +- %9.4f    }\n",par->gam1,
sqrt(gamvcm[GAM1][GAM1]));
        fprintf(outfile,"Gam2  = %9.4f  +- %9.4f    }  c12 = %9.4f\n",par->gam2,
sqrt(gamvcm[GAM2][GAM2]), gamvcm[GAM1][GAM2]);
        fprintf(outfile,"\n");

        fprintf(outfile,"Total gamma, azimuth of max shear (ppm/yr,deg):\n");
        fprintf(outfile,"Tgam  = %9.4f  +- %9.4f\n",par->tgam,
sqrt(epsvcm[TGAM][TGAM]));
        fprintf(outfile,"Gazim = %9.4f  +- %9.4f\n",par->gazim,
sqrt(epsvcm[AZIM2][AZIM2])/DEG2RAD);
        fprintf(outfile,"\n");
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to transform a covariance matrix from one parameter frame to
   another by means of the Jacobian transform [Cy] = [J][Cx][Jt] where the
   parameters [x] are related to the parameters [y] by [y] = [J][x].

   pjc aug '95
*/

#include <math.h>
#include "nrutil.h"

#include "globdef.c"


void jacxform(covx,xsize, jac, covy,ysize)

double **covx,**jac,**covy;
int xsize,ysize;

{
        double **tmp;
        register int i,j,k;

        tmp = dmatrix(1,xsize,1,ysize);


/* first perform [tmp] = [Cx][Jt] */
        matmult(covx,jac,tmp, xsize,xsize,ysize, 0,1);


/* now perform [Cy] = [J][tmp] */
        matmult(jac,tmp,covy, ysize,xsize,ysize, 0,0);


        free_dmatrix(tmp,1,xsize,1,ysize);
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* lubksb.c
  Taken directly from NR in C, section 2.3
/*
*******************************************************************************


*******************************************************************************
/* ludcmp.c
  Taken directly from NR in C, section 2.3
/*
*******************************************************************************


*******************************************************************************
/* subroutine to make a weight matrix for certain stations from the full vcm
  cvdisp, given the station nos.

  pjc aug '95
*/

#include <stdio.h>
#include <math.h>
#include "nrutil.h"
#include "nr.h"

#include "globdef.c"


int makewgt(poly,verts,cvdisp,pwgt,nsites,nepochs,occup)

int poly,nsites,nepochs;
int **verts,**occup;
double ***cvdisp,**pwgt;

{
        double **pvcm,**pvcmcopy,**pwgttmp;
        double det,*indx,*tmpvec,*tmpcopy;
        double rtmp;
        register int ii,jj;
```

```
        int h,i,j,k;
        int nverts,ndata,nepverts;
        int site1,site2;
        int wgterr = 0;

/* local arrays for vcm inversion */
        indx = dvector(1,MAXVERT*NDIM);
        tmpvec = dvector(1,MAXVERT*NDIM);
        tmpcopy = dvector(1,MAXVERT*NDIM);
        pvcm = dmatrix(1,MAXVERT*NDIM,1,MAXVERT*NDIM);
        pvcmcopy = dmatrix(1,MAXVERT*NDIM,1,MAXVERT*NDIM);
        pwgttmp = dmatrix(1,MAXVERT*NDIM,1,MAXVERT*NDIM);


/* zero all elements of vcm at start */
        for(i=1; i<=MAXVERT*MAXEPO*NDIM; i++)
                for(j=1; j<=MAXVERT*MAXEPO*NDIM; j++)
                        pwgt[i][j] = 0.0;


/* copy over relevant columns of vcm for each submatrix in turn */
        nverts = verts[poly][0];
        ndata = 0;
        for(k=1; k<=nepochs; k++) {
                nepverts = 0;
                for(i=1; i<=nverts; i++) {
                        site1 = verts[poly][i];
                        if (occup[k][site1] == 1) nepverts++;
                }
                for(i=1,ii=0; i<=nverts; i++) {
                        site1 = verts[poly][i];
                        if (occup[k][site1] == 0) continue;
                        ii++;
                        for(j=1,jj=0; j<=nverts; j++) {
                                site2 = verts[poly][j];
                                if (occup[k][site2] == 0) continue;
                                jj++;
                                pvcmcopy[ii][jj] = pvcm[ii][jj] =
cvdisp[k][site1][site2];
                                pvcmcopy[nepverts+ii][jj] = pvcm[nepverts+ii][jj] =
cvdisp[k][nsites+site1][site2];
                                pvcmcopy[ii][nepverts+jj] = pvcm[ii][nepverts+jj] =
cvdisp[k][site1][nsites+site2];
                                pvcmcopy[nepverts+ii][nepverts+jj] =
pvcm[nepverts+ii][nepverts+jj] = cvdisp[k][nsites+site1][nsites+site2];
                        }
                }

        /* invert vcm of relevant sites */
                ludcmp(pvcm,NDIM*nepverts,indx,&det);
                for(j=1; j<=NDIM*nepverts; j++) {
                        for(i=1; i<=NDIM*nepverts; i++)
                                tmpcopy[i] = tmpvec[i] = 0.0;
                        tmpcopy[j] = tmpvec[j] = 1.0;
                        lubksb(pvcm,NDIM*nepverts,indx,tmpvec);
                        mprove(pvcmcopy,pvcm, NDIM*nepverts, indx, tmpcopy, tmpvec);
                        for(i=1; i<=NDIM*nepverts; i++) pwgttmp[i][j] = tmpvec[i];
                }

        /* check inversion for stability */
                for(i=1; i<=NDIM*nepverts; i++) {
                        for(j=1; j<=NDIM*nepverts; j++) {
                                rtmp=0.0;
                                for(h=1; h<=NDIM*nepverts; h++)
                                        rtmp += pvcmcopy[i][h] * pwgttmp[h][j];
                                if ((i==j) && (fabs(rtmp-1.0) > INVTOL)) wgterr = 1;
                                if ((i!=j) && (fabs(rtmp) > INVTOL)) wgterr = 1;
                        }
                }

        /* copy submatrix into full weight matrix */
                for(i=1; i<=NDIM*nepverts; i++) {
                        for(j=1; j<=NDIM*nepverts; j++) {
                                pwgt[NDIM*ndata+i][NDIM*ndata+j] = pwgttmp[i][j];
                        }
                }
                ndata += nepverts;

        }


/* free local arrays */
        free_dvector(indx,1,MAXVERT*NDIM);
        free_dvector(tmpvec,1,MAXVERT*NDIM);
        free_dvector(tmpcopy,1,MAXVERT*NDIM);
        free_dmatrix(pvcm,1,MAXVERT*NDIM,1,MAXVERT*NDIM);
        free_dmatrix(pvcmcopy,1,MAXVERT*NDIM,1,MAXVERT*NDIM);
        free_dmatrix(pwgttmp,1,MAXVERT*NDIM,1,MAXVERT*NDIM);


        return wgterr;
}

#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to multiply an m by n matrix [A] by a n by p matrix [B],
   putting the result in the m by p matrix [C].
   If at==1, then [At] (m by n) is used instead of [A] (m by n).
   If bt==1, then [Bt] (n by p) is used instead of [B] (n by p).

   pjc dec '95
*/

#include <math.h>
#include "nrutil.h"


void matmult(A,B,C, m,n,p, at,bt)

double **A,**B,**C;
int m,n,p;
int at,bt;

{
        int i,j,k;

        if (at == 0)
                if (bt == 0)
                        for(i=1; i<=m; i++)
                                for(j=1; j<=p; j++)
                                        for(k=1,C[i][j]=0.0; k<=n; k++)
                                                C[i][j] += A[i][k] * B[k][j];
                else
                        for(i=1; i<=m; i++)
                                for(j=1; j<=p; j++)
                                        for(k=1,C[i][j]=0.0; k<=n; k++)
                                                C[i][j] += A[i][k] * B[j][k];
        else
                if (bt == 0)
```

```
                        for(i=1; i<=m; i++)
                                for(j=1; j<=p; j++)
                                        for(k=1,C[i][j]=0.0; k<=n; k++)
                                                C[i][j] += A[k][i] * B[k][j];
                else
                        for(i=1; i<=m; i++)
                                for(j=1; j<=p; j++)
                                        for(k=1,C[i][j]=0.0; k<=n; k++)
                                                C[i][j] += A[k][i] * B[j][k];
}
*******************************************************************************


*******************************************************************************
/* mprove.c
  Taken directly from NR in C, section 2.5
/*
*******************************************************************************


*******************************************************************************
/* nr.h
  Taken directly from NR in C, appendix A
/*
*******************************************************************************


*******************************************************************************
/* nrutil.c
  Taken directly from NR in C, appendix B
/*
*******************************************************************************


*******************************************************************************
/* nrutil.h
  Taken directly from NR in C, appendix B
/*
*******************************************************************************


*******************************************************************************
/* subroutine to read the input options for polystr2 from the given file.
   This file must adhere to the format given in the template file
   "polystr2.inx".
   Filenames and various flags are returned.
   Modified for polystr2 to read .CRD files by pjc dec '95
   pjc aug '95
*/

#include <string.h>
#include <stdio.h>

#include "globdef.c"


void readinps(inpname, nepochs,epoch,crdnames,covarnames, outname, strname,residname,
polyname, xcoord,ycoord, xdisp,ydisp, sitename,nsites,occup, npolys, verts)

char *inpname,*outname;
char crdnames[][NAMELEN],covarnames[][NAMELEN];
char *strname,*residname;
char *polyname;
char sitename[][NAMELEN];
double *xcoord,*ycoord;
double **xdisp,**ydisp;
double *epoch;
int *npolys,*nsites,*nepochs;
int **verts,**occup;
{
        FILE *inpfile;
        char dummystring[255];
        char dummy1[NAMELEN],dummy2[NAMELEN];
        double depoch;
        int i,nverts,np;
        int status,nchars,indx;
        int site,inc;

        void getpsvm();
        int findstnum();


/* open file and check */
        inpfile=fopen(inpname,"r");
        if (inpfile == NULL) {
                printf("\n*** error opening input file '%s'\n",inpname);
                exit();
        }


/* skip header, load file names and flags */
        for(i=1;i<=5;i++) fgets(dummystring,255,inpfile);

        fscanf(inpfile,"%s",outname);
        fgets(dummystring,255,inpfile);

        fscanf(inpfile,"%s",residname);
        fgets(dummystring,255,inpfile);

        fscanf(inpfile,"%s",strname);
        fgets(dummystring,255,inpfile);

        fscanf(inpfile,"%s",polyname);
        fgets(dummystring,255,inpfile);

        fscanf(inpfile,"%d",nepochs);
        fgets(dummystring,255,inpfile);


/* load epoch times and coord/vcm filenames */
        for(i=1;i<=3;i++) fgets(dummystring,255,inpfile);

        for(i=1; i<=*nepochs; i++) {
                fscanf(inpfile,"%lf %s %s",&depoch,dummy1,dummy2);
                epoch[i] = depoch;
                strcpy(crdnames[i],dummy1);
                strcpy(covarnames[i],dummy2);
                fgets(dummystring,255,inpfile);
        }


/* get coords at all epochs */
        getcrds(*nepochs,crdnames,occup, xcoord,ycoord, xdisp,ydisp, sitename,
nsites);


/* get list of poygon vertices, one line per polygon, terminated with 'XXXX' */
        for(i=1;i<=4;i++) fgets(dummystring,255,inpfile);

        np = 0;
        status = sscanf(dummystring,"%s %n",dummy1,&nchars);
        while (strcmp(dummy1,"XXXX") != 0) {
                np += 1;
                nverts = 0;
                indx = 0;
```

```
                while (status != EOF) {
                        indx += nchars;
                        site = findstnum(dummy1,sitename,*nsites);
                        if (site > 0) {
                                if (occup[0][site] >= 2) {
                                        inc = 0;
                                        for(i=1; i<=nverts; i++)
                                                if (verts[np][i] == site) inc = 1;
                                        if (inc == 0) {
                                                nverts += 1;
                                                verts[np][nverts] = site;
                                        }
                                        else printf("*** Site \"%s\" named more than
once in polygon \"%s\"\n", dummy1,dummystring);
                                }
                                else printf("*** Site \"%s\" in polygon \"%s\" has
fewer than two occupations\n", dummy1,dummystring);
                        }
                        else printf("*** Unknown site \"%s\" ignored in polygon
\"%s\"\n", dummy1,dummystring);
                        status = sscanf(dummystring+indx,"%s %n",dummy1, &nchars);
                }
                if (nverts < 3) {
                        printf("*** Polygon %s has fewer than 3 recognised sites -
ignored\n",dummystring);
                        np -= 1;
                }
                else verts[np][0] = nverts;
                fgets(dummystring,255,inpfile);
                status = sscanf(dummystring,"%s %n",dummy1,&nchars);
        }

        *npolys = np;
        fclose(inpfile);

}

#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to convert coordinates from cartesian (in m) to lat,lon (in
  radians) and ellipsoidal height (in m) by iterative method (see Heiskanen &
  Moritz for theory).

   pjc dec '95
*/

#include <math.h>
#include <stdio.h>

#include "globdef.c"


void xyz2llh(x,y,z, lat,lon,hgt)

double x,y,z;
double *lat,*lon,*hgt;

{
        double newlat,dlat,newhgt,dhgt,N,Lxy;

        *lon = atan2(y,x);

        Lxy = sqrt(x*x + y*y);
        *lat = atan2(z,(1.0-ECCEN*ECCEN)*Lxy);
        *hgt = 0.0;
        dlat = 0.5; dhgt = 0.5;

        while ((dlat > 1.0e-10) || (dhgt > 1.0e-4)) {
                N = RADIUS / (sqrt(1.0 - pow(ECCEN*sin(*lat),2.0)));
                newlat = atan2(z , (Lxy - ECCEN*ECCEN*N*cos(*lat)));
                newhgt = (Lxy/cos(*lat)) - N;
                dlat = fabs(*lat - newlat);
                dhgt = fabs(*hgt - newhgt);
                *lat = newlat; *hgt = newhgt;
        }
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
Input option file for polystr polygon strain program
-------------------------------------------------------------------------------

Option                                  Description
*******************************         ---------------------------------------
test.out                                output filename / "NONE"
test.res                                residual filename / "NONE"
test.str                                principal strain output filename /"NONE"
poly.xyz                                polygon output filename /"NONE"
2                                       no. of epochs

Epoch           Filename.CRD                    Filename.CEN / "UNIT"
*********       ****************************  ****************************
1991.78         91NNO_2A.CRD                    91NNO_2A.CEN
1993.39         93NNO_2A.CRD                    93NNO_2A.CEN

Vertices of polygons (1 line per polygon,XXXX to finish)
-------------------------------------------------------
CG02 CG01 CG03
XXXX
*******************************************************************************
```

C.4 `okinv`

`okinv` is a C program that inverts for earthquake source parameters by minimising a penalty function (usually the L_2 norm of observed minus model displacements) using as its basis the downhill simplex inversion algorithm (Nelder & Mead, 1965). Because the outcome of a single simplex inversion is dependent on the starting parameters, for a non-linear problem which has many local minima, `okinv` can perform many simplex inversions with randomly-chosen starting parameters ('Monte-Carlo restarts'), from which the best minimum is chosen. Because the displacement computations are performed by a stripped-down version of Okada's (1985) FORTRAN-77 code (converted to C by myself) which is only valid for surface displacements resulting from slip on a rectangular fault plane wholly within an elastic half-space, bounds checking must be performed continually and the simplex inversion abandoned if physically unreasonable parameters are encountered. For further discussion of the theory, see Section 3.5.

In the current implementation, up to 4 fault segments and 250 observation sites can be included. For each fault segment, the strike, dip, rake, slip, length, minimum and maximum depth of faulting, and coordinates of the fault scarp can be either fixed or estimated. The latter refer to the point on the surface which is the projection up-dip of the centre of the fault plane. Additional global parameters that can be fixed or estimated are the translations (in north, east and up components), scale change and rotation about a vertical axis, that are applied to the surface displacements.

A priori errors assigned to the parameters are used to determine the bounds of starting parameters and the size of the initial simplex for each individual simplex inversion. So that the distribution of minima can be studied, the inital and final parameters in each simplex inversion are output to files. These can later be searched for minima corresponding to particular ranges of parameters. When the Monte-Carlo inversion is over, the *a priori* errors are used to determine the range over which each parameter in turn is varied, to investigate the sensitivity of the solution to that parameter alone.

`okinv` has been successfully compiled and run on Sun workstations using both Sun C and GNU C. The typical execution time on a SPARC 10, inverting for all nine parameters of one fault segment, two translation parameters, scale and rotation, with horizontal displacement observations at 100 sites, is 86 s per Monte-Carlo restart.

```
*******************************************************************************
/*
     Program to take site displacements from *.psvm file and covariances (in
   e,n[,h]) from covrotate output file, then invert for a dislocation on a
   finite plane in an elastic half-space that best fits the displacement field.
   The observations are weighted using the covariance matrix. A simplex
   algorithm (from Numerical Recipes) is used to perform the inversion, which
   can be repeated with different starting points selected Monte-Carlo style.
   Bootstrapping of the data to estimated errors can be performed (not yet!).
   Input options should be in the file "okinv.inp" or another file named on the
   command line.


   Pete Clarke, June 1995
*/

#include <math.h>
#include <stdio.h>
#include <string.h>
#include "nr.h"
#include "nrutil.h"
void srand48();

#include "global.c"


main(argc,argv)

int argc;
char **argv;

{
        char inpname[32];               /* input file name */
        char outname[32];               /* output file name */
        char psvmname[32];              /* psvm file name */
        char covarname[32];             /* vcm file name (B34 format) */
        char residname[32];             /* residual output file name */
        char dispgrid[32];              /* gridded displacement output file */
        char scratchname[32];           /* results of each Monte-carlo restart */
        char scseedname[32];            /* seed points for each M-C restart */
        char scvarname[32];             /* single-parameter variation file */
        char dispsta[32];               /* model site displacements file name */
        char sitename[MAXSITE+1][32];   /* site names */
        double **params;                /* array of simplex vertices */
        double *penalty;                /* array of simplex vertex penalties */
        double *excoord,*eycoord;       /* site coords in lon,lat */
        double **cvdisp;                /* vcm of site displacements */
        double **cvcopy;                /* copy of cvdisp */
        double rrake,rstrike;           /* rake and strike in radians */
        double finalpen;                /* penalty at end of simplex inversion */
        double bestpen;                 /* best penalty to date during M-C
restarts */
        double u,w;                     /* fault slip (m), down-dip width (km) */
        double rmerid,rtmp,rprimev;     /* radius of curvature in meridian, prime
vertical */
        double minx,miny;               /* lower left corner for model gridded
displacements */
        double xstep,ystep;             /* step size for model grid displacements */
        double pertfac;                 /* perturbation factor for initial simplex
size (sigmas) */
        double searchnsig;              /* search width (sigmas) for M-C seed
points */
        double varnsig;                 /* search width (sigmas) for single-parameter
variation */
        double det,*indx;               /* determinant, temp vector for vcm
inversion */
        double *tmpvec,*tmpcopy;        /* temp vectors for vcm inversion */
        double *best_strike,*best_dip;  /* best strike and dip found so far */
        double *best_rake,*best_slp;    /* best rake and slip found so far */
        double *best_fltx,*best_flty;   /* best fault coords so far */
        double *best_fltlen;            /* best fault length so far */
        double *best_hmin,*best_hmax;   /* best min/max fault depth so far */
        double best_u0,best_v0,best_w0; /* best translation params so far */
        double best_rigrot,best_scale;  /* best rotation/scale so far */
        int i,j,k;                      /* index variables */
        int dummy,iret;                 /* dummy flags */
        int imonte;                     /* no. of M-C restarts counter */
        int varnstep;                   /* no. of steps in single-param variation */
        int ndims;                      /* no. of dimensions of displacement data */
        int deltaz_out;                 /* output vertical displacement flag */
        int coortype;                   /* 1=lat/lon, 0=km */
        int nmonte;                     /* no. of M-C restarts to be performed */
        int nparams;                    /* no. of parameters in inversion */
        int nfunks,nfunks_tot;          /* no. of penalty function calls this
inversion, all inversions */
        int niter,niter_tot;            /* no. of simplex moves this inversion, all
inversions */
        int nxstep,nystep;              /* no. of grid points for model gridded
displacements */
        int wgterr;                     /* flag indicating error in displacement vcm
inversion */
        long seedval;                   /* seed value for random number generator */
        FILE *outfile;                  /* generic output file handle */
        FILE *scrfile,*scsfile;         /* file handles for M-C seed point and
results files */


/* start clock, so we can use it as seed for random number generator later */
        seedval = clock();


/* reserve memory for global arrays */
        globres();


/* reserve memory for local arrays */
        excoord = dvector(1,MAXSITE);
        eycoord = dvector(1,MAXSITE);
        indx = dvector(1,MAXSITE*MAXDIM);
        tmpvec = dvector(1,MAXSITE*MAXDIM);
        tmpcopy = dvector(1,MAXSITE*MAXDIM);
        penalty = dvector(1,NOFFS+MAXFLT*NFLAGS);
        params = dmatrix(1,1+NOFFS+MAXFLT*NFLAGS,1,NOFFS+MAXFLT*NFLAGS);
        best_strike = dvector(1,MAXFLT);
        best_dip = dvector(1,MAXFLT);
        best_rake = dvector(1,MAXFLT);
        best_slp = dvector(1,MAXFLT);
        best_fltx = dvector(1,MAXFLT);
        best_flty = dvector(1,MAXFLT);
        best_fltlen = dvector(1,MAXFLT);
        best_hmin = dvector(1,MAXFLT);
        best_hmax = dvector(1,MAXFLT);


/* print title */
        printf("\nEstimation of elastic dislocation parameters\n");
        printf("-------------------------------------------\n\n");


/* read input file */
        if (argc == 2) strcpy(inpname,argv[1]);
```

```
        else strcpy(inpname,"okinv.inp");
        printf("Reading from input file \"%s\"\n", inpname);
        readinps(inpname, psvmname,covarname, outname, residname, dispsta, dispgrid,
scratchname,scseedname,scvarname, &varnstep,&varnsig, &coortype, &deltaz_out,
&norm, &pertfac, &searchnsig, &minx,&miny, &xstep,&ystep, &nxstep,&nystep,
&nmonte, &nparams);


/* load displacements and coords from psvm file, if specified. */
        if (strcmp(psvmname,"NONE") != 0) {
                getpsvm(psvmname, excoord,eycoord, sitename);
                ell2xy(excoord,eycoord,xcoord,ycoord,nsites,coortype,1);
                printf("Centroid of network at %9.4f , %9.4f\n", orig_x,orig_y);
        }
        else {
                orig_x = minx + (double)(nxstep-1) * xstep;
                orig_y = miny + (double)(nystep-1) * ystep;
        }


/* load covariance matrix or set it to identity matrix if "UNIT" entered. If
   covarname is "PSVM", the sigmas in the psvm file are used. */
        ndims = 2 + abs(use_dz);
        getcovs(covarname,psvmname,sitename,cvdisp,ndims);


/* invert covariance matrix to get weight matrix */
        if (covtype == 1) {
        /* this bit used to generate a unit matrix, but has been removed
           to save memory */
/*              wgt = dmatrix(1,nsites*ndims,1,nsites*ndims);
                for(i=1; i<=nsites*ndims; i++) wgt[i][i] = 1.0; */
        }
        else if (covtype == 2) {
        /* use diagonal elements only */
                wgt = dmatrix(1,nsites*ndims,1,nsites*ndims);
                for(i=1; i<=nsites*ndims; i++) wgt[i][i] = 1.0 / cvdisp[i][i];
        }
        else {
        /* invert full vcm using NR LU-decomposition routines */
                wgt = dmatrix(1,nsites*ndims,1,nsites*ndims);
                printf("Inverting covariance matrix ...\n");
                cvcopy = dmatrix(1,nsites*ndims,1,nsites*ndims);
                for(i=1; i<=nsites*ndims; i++) for(j=1; j<=nsites*ndims; j++)
                        cvcopy[i][j] = cvdisp[i][j];
                ludcmp(cvdisp,nsites*ndims,indx,&det);
                for(j=1; j<=nsites*ndims; j++) {
                        for(i=1; i<=nsites*ndims; i++)
                                tmpcopy[i] = tmpvec[i] = 0.0;
                        tmpcopy[j] = tmpvec[j] = 1.0;
                        lubksb(cvdisp,nsites*ndims,indx,tmpvec);
                        mprove(cvcopy,cvdisp,nsites*ndims,indx,tmpcopy,tmpvec);
                        for(i=1; i<=nsites*ndims; i++) wgt[i][j] = tmpvec[i];
                }
                printf("Checking weight matrix ... ");
                wgterr = 0;
                for(i=1; i<=nsites*ndims; i++) for(j=1; j<=nsites*ndims; j++) {
                        rtmp=0.0;
                        for(k=1; k<=nsites*ndims; k++)
                                rtmp += cvcopy[i][k] * wgt[k][j];
                        if ((i==j) && (fabs(rtmp-1.0) > INVTOL)) wgterr = 1;
                        if ((i!=j) && (fabs(rtmp) > INVTOL)) wgterr = 1;
                }
                if (wgterr) printf("problems exist with inversion\n");
                else printf("OK\n");
        }


/* open scratch files if required for Monte-Carlo iterations */
        scrfile = NULL;
        if (strcmp(scratchname,"NONE") != 0) {
                scrfile = fopen(scratchname,"w");
                if (scrfile == NULL) {
                        printf("\n*** error opening scratch output file: %s\n",
scratchname);
                        exit();
                }
        }

        scsfile = NULL;
        if (strcmp(scseedname,"NONE") != 0) {
                scsfile = fopen(scseedname,"w");
                if (scsfile == NULL) {
                        printf("\n*** error opening scratch seed point file: %s\n",
scseedname);
                        exit();
                }
        }


/* convert coords of faults to km if necessary, copy over to right array */
        ell2xy(ap_fltx,ap_flty,ap_fltx,ap_flty,nfaults,coortype,0);


/* convert a priori sigmas of fault coords to km if necessary */
        if (coortype == 1) {
                rprimev = RADIUS / sqrt(1 - pow((ECCEN*sin(orig_y*DEG2RAD)),2.0) );
                rtmp = 1 - pow((ECCEN*sin(orig_y*DEG2RAD)),2.0);
                rmerid = RADIUS*(1 - pow(ECCEN,2.0)) / (rtmp*sqrt(rtmp));
                for(i=1; i<=nfaults; i++) {
                        apsigkm_fltx[i] = apsig_fltx[i] * (rprimev/1000.0)*
cos(orig_y*DEG2RAD) * DEG2RAD;
                        apsigkm_flty[i] = apsig_flty[i] * (rmerid/1000.0) * DEG2RAD;
                }
        }
        else {
                for(i=1; i<=nfaults; i++) {
                        apsigkm_fltx[i] = apsig_fltx[i];
                        apsigkm_flty[i] = apsig_flty[i];
                }
        }


/* get seed for random number generator */
        seedval = clock();
        srand48(seedval);


/* loop over Monte-Carlo restarts */
        if (nparams > 0) printf("\nInverting for %d parameter(s) ...\n", nparams);
        else {
                printf("\nNo inversion performed.\n");
                nmonte = 0;
        }
        niter_tot = 0; nfunks_tot = 0; bestpen = BIGNUM;
        imonte = 1;
        do {


        /* load up parameters into arrays for simplex routine */
                makesimp(nparams,params,pertfac,searchnsig,nmonte);
```

```
        /* evaluate penalty function at initial vertices of simplex */
                for(j=1; j<=nparams+1; j++)
                        penalty[j] = penfunc(params[j]);


        /* wrap the angular parameters back to reasonable values, for output
           to scratch seed file */
                for(j=1; j<=nfaults; j++) {
                        while (rake[j] > 180.0) rake[j] -= 360.0;
                        while (rake[j] < -180.0) rake[j] += 360.0;
                        if (dip[j] > 90.0) {
                                strike[j] += 180.0;
                                dip[j] = 180.0 - dip[j];
                                if (rake[j] > 0.0) rake[j] = 180.0 - rake[j];
                                if (rake[j] < 0.0) rake[j] = -180.0 - rake[j];
                        }
                        while (strike[j] > 360.0) strike[j] -= 360.0;
                        while (strike[j] < 0.0) strike[j] += 360.0;
                }


        /* dump seed point of each Monte-Carlo restart to scratch seed file */
                if (scsfile != NULL) {
                        xy2ell(fltx,flty, fltx,flty, nfaults, coortype);
                        fprintf(scsfile,"%04d  %11.4le  %4d  %4d\n", imonte,
penalty[nparams+1], 0,0);
                        for(j=1; j<=nfaults; j++) {
                                fprintf(scsfile,"%9.4lf %9.4lf %9.4lf %9.4lf\n",
strike[j],dip[j],rake[j], slp[j]);
                                fprintf(scsfile,"%9.4lf %9.4lf %9.4lf %9.4lf
%9.4lf\n", fltx[j],flty[j], fltlen[j], hmin[j],hmax[j]);
                        }
                        fprintf(scsfile,"%9.4lf %9.4lf %9.4lf %9.4lf %9.4lf\n\n",
u0,v0,w0, rigrot,scale);
                        fflush(scsfile);
                        ell2xy(fltx,flty, fltx,flty, nfaults, coortype, 0);
                }


        /* do the inversion! */
                if (nparams > 0) {
                        iret = amoeba(params,penalty,nparams,FTOL,penfunc, &nfunks,
&niter);
                        niter_tot += niter;
                        nfunks_tot += nfunks;
                        if ((iret == 1) && (nfunks < NMAX)) nfunks += NMAX;
                }


        /* convert simplex params back to ordinary stuff,
           compute final penalty */
                finalpen = penfunc(params[nparams+1]);
                xy2ell(fltx,flty,fltx,flty,nfaults,coortype);


        /* wrap the angular parameters back to reasonable values */
                for(j=1; j<=nfaults; j++) {
                        while (rake[j] > 180.0) rake[j] -= 360.0;
                        while (rake[j] < -180.0) rake[j] += 360.0;
                        if (dip[j] > 90.0) {
                                strike[j] += 180.0;
                                dip[j] = 180.0 - dip[j];
                                if (rake[j] > 0.0) rake[j] = 180.0 - rake[j];
                                if (rake[j] < 0.0) rake[j] = -180.0 - rake[j];
                        }
                        while (strike[j] > 360.0) strike[j] -= 360.0;
                        while (strike[j] < 0.0) strike[j] += 360.0;
                }


        /* if parameters are best so far (and we collapsed the simplex
           successfully), store them for later */
                if ((finalpen < bestpen) && (nfunks < NMAX)) {
                        bestpen = finalpen;
                        for(j=1; j<=nfaults; j++) {
                                best_strike[j] = strike[j];
                                best_dip[j] = dip[j];
                                best_rake[j] = rake[j];
                                best_slp[j] = slp[j];
                                best_fltx[j] = fltx[j];
                                best_flty[j] = flty[j];
                                best_fltlen[j] = fltlen[j];
                                best_hmin[j] = hmin[j];
                                best_hmax[j] = hmax[j];
                        }
                        best_u0 = u0; best_v0 = v0; best_w0 = w0;
                        best_rigrot = rigrot; best_scale = scale;
                }


        /* dump results of each Monte-Carlo restart to scratch file */
                if (scrfile != NULL) {
                        fprintf(scrfile,"%04d  %11.4le  %4d  %4d\n", imonte,
finalpen, nfunks,niter);
                        for(j=1; j<=nfaults; j++) {
                                fprintf(scrfile,"%9.4lf %9.4lf %9.4lf %9.4lf\n",
strike[j],dip[j],rake[j], slp[j]);
                                fprintf(scrfile,"%9.4lf %9.4lf %9.4lf %9.4lf
%9.4lf\n", fltx[j],flty[j], fltlen[j], hmin[j],hmax[j]);
                        }
                        fprintf(scrfile,"%9.4lf %9.4lf %9.4lf %9.4lf %9.4lf\n\n",
u0,v0,w0, rigrot,scale);
                        fflush(scrfile);
                }


        /* end of loop over Monte-Carlo restarts */
                imonte++;
        } while (imonte <= nmonte);


/* close scratch files */
        if (scrfile != NULL) fclose(scrfile);
        if (scsfile != NULL) fclose(scsfile);


/* recover best fitting parameters */
        for(j=1; j<=nfaults; j++) {
                strike[j] = best_strike[j];
                dip[j] = best_dip[j];
                rake[j] = best_rake[j];
                slp[j] = best_slp[j];
                fltx[j] = best_fltx[j];
                flty[j] = best_flty[j];
                fltlen[j] = best_fltlen[j];
                hmin[j] = best_hmin[j];
                hmax[j] = best_hmax[j];
                rrake = (rake[j]+90.0) * DEG2RAD;
```

```
                w = (hmax[j]-hmin[j])/sin(dip[j]*DEG2RAD);
                ud[j] =  100.0*slp[j]*cos(rrake);
                us[j] = -100.0*slp[j]*sin(rrake);
                al1[j] = -fltlen[j]/2.0;
                al2[j] = fltlen[j]/2.0;
                aw1[j] = hmin[j]/sin(dip[j]*DEG2RAD);
                aw2[j] = hmax[j]/sin(dip[j]*DEG2RAD);
        }
        u0 = best_u0; v0 = best_v0; w0 = best_w0;
        rigrot = best_rigrot; scale = best_scale;


/* recompute displacements and scarp end coords for best-fitting parameters */
        ell2xy(fltx,flty, fltx,flty, nfaults, coortype, 0);
        dummy = comp_disps();
        for(j=1; j<=nfaults; j++) {
                rstrike = (270.0-strike[j]) * DEG2RAD;
                dumx[1] = fltx[j] + 0.5*fltlen[j]*cos(rstrike);
                dumy[1] = flty[j] + 0.5*fltlen[j]*sin(rstrike);
                dumx[2] = fltx[j] - 0.5*fltlen[j]*cos(rstrike);
                dumy[2] = flty[j] - 0.5*fltlen[j]*sin(rstrike);
                xy2ell(dumx,dumy,dumx,dumy,2,coortype);
                end1x[j] = dumx[1]; end1y[j] = dumy[1];
                end2x[j] = dumx[2]; end2y[j] = dumy[2];
        }
        xy2ell(fltx,flty,fltx,flty,nfaults,coortype);


/* output parameters etc to screen */
        invprint(stdout, ndims,nmonte, niter_tot,nfunks_tot, bestpen);

/* output parameters etc to file */
        if (strcmp(outname,"NONE") != 0) {
                printf("\nWriting output file ...\n");
                outfile = fopen(outname,"w");
                if (outfile == NULL) {
                        printf("\n*** error opening output file \"%s\"\n", outname);
                        exit();
                }
                invprint(outfile, ndims,nmonte, niter_tot,nfunks_tot, bestpen);
        }

/* output residuals to residual .psvm file (if output filename given) */
        if (strcmp(residname,"NONE") != 0) {
                printf("\nMaking residual file ...\n");
                outfile = fopen(residname,"w");
                if (outfile == NULL) {
                        printf("\n*** error opening residual output file \"%s\"\n",
residname);
                        exit();
                }
                for(i=1; i<=nsites; i++) {
                        fprintf(outfile,"%10.6lf %10.6lf  %8.4lf %8.4lf  0 0 0",
excoord[i], eycoord[i], xdisp[i]-comp_xdisp[i], ydisp[i]-comp_ydisp[i]);
                        if (deltaz_out == 1) fprintf(outfile," %8.4lf %8.4lf",
zdisp[i]-comp_zdisp[i], 0.0);
                        fprintf(outfile," %s\n",sitename[i]);
                }
                fclose(outfile);
        }


/* output computed displacements to .psvm file (if output filename given) */
        if (strcmp(dispsta,"NONE") != 0) {
                printf("\nMaking computed station displacement file ...\n");
                outfile = fopen(dispsta,"w");
                if (outfile == NULL) {
                        printf("\n*** error opening station output file '%s'\n",
dispsta);
                        exit();
                }
                for(i=1; i<=nsites; i++) {
                        fprintf(outfile,"%10.6lf %10.6lf  %8.4lf %8.4lf  %8.4lf
%8.4lf %8.4lf", excoord[i], eycoord[i], comp_xdisp[i], comp_ydisp[i],
0.0,0.0,0.0);
                        if (deltaz_out == 1) fprintf(outfile," %8.4lf %8.4lf",
comp_zdisp[i], 0.0);
                        fprintf(outfile," %s\n",sitename[i]);
                }
                fclose(outfile);
        }


/* output penalty function values, varying one parameter at a time, if output
   file specified. */
        if (strcmp(scvarname,"NONE") != 0) {
                printf("\nMaking single-parameter variation file ...\n");
                outfile = fopen(scvarname,"w");
                if (outfile == NULL) {
                        printf("\n*** error opening single-parameter variation file
'%s'\n", scvarname);
                        exit();
                }
                ell2xy(fltx,flty, fltx,flty, nfaults, coortype, 0);

                j = 1;
                if (inv_u0 == 1) params[nparams+1][j++] = u0;
                if (inv_v0 == 1) params[nparams+1][j++] = v0;
                if (inv_w0 == 1) params[nparams+1][j++] = w0;
                if (inv_rigrot == 1) params[nparams+1][j++] = rigrot;
                if (inv_scale == 1) params[nparams+1][j++] = scale;
                for(i=1; i<=nfaults; i++) {
                        if (inv_strike[i] == 1) params[nparams+1][j++] = strike[i];
                        if (inv_dip[i] == 1) params[nparams+1][j++] = dip[i];
                        if (inv_rake[i] == 1) params[nparams+1][j++] = rake[i];
                        if (inv_slp[i] == 1) params[nparams+1][j++] = best_slp[i];
                        if (inv_fltx[i] == 1) params[nparams+1][j++] = fltx[i];
                        if (inv_flty[i] == 1) params[nparams+1][j++] = flty[i];
                        if (inv_fltlen[i] == 1) params[nparams+1][j++] = fltlen[i];
                        if (inv_hmin[i] == 1) params[nparams+1][j++] = hmin[i];
                        if (inv_hmax[i] == 1) params[nparams+1][j++] = hmax[i];
                }

                j = inv_u0 + inv_v0 + inv_w0 + inv_rigrot + inv_scale + 1;
                for(i=1; i<=nfaults; i++) {
                        if (inv_strike[i] == 1) {
                                fprintf(outfile, "# strike of fault %d\n", i);
                                var1par(params[nparams+1],j++, apsig_strike[i],
varnsig, varnstep, outfile);
                        }
                        if (inv_dip[i] == 1) {
                                fprintf(outfile, "# dip of fault %d\n", i);
                                var1par(params[nparams+1],j++, apsig_dip[i],
varnsig, varnstep, outfile);
                        }
                        if (inv_rake[i] == 1) {
                                fprintf(outfile, "# rake of fault %d\n", i);
                                var1par(params[nparams+1],j++, apsig_rake[i],
varnsig, varnstep, outfile);
                        }
```

```
                        if (inv_slp[i] == 1) {
                                fprintf(outfile, "# slp of fault %d\n", i);
                                var1par(params[nparams+1],j++, apsig_slp[i],
varnsig, varnstep, outfile);
                        }
                        if (inv_fltx[i] == 1) {
                                fprintf(outfile, "# lon(x) of fault %d\n", i);
                                var1crdpar(params[nparams+1],j++, i,1,coortype,
apsigkm_fltx[i], varnsig, varnstep, outfile);
                        }
                        if (inv_flty[i] == 1) {
                                fprintf(outfile, "# lat(y) of fault %d\n", i);
                                var1crdpar(params[nparams+1],j++, i,2,coortype,
apsigkm_flty[i], varnsig, varnstep, outfile);
                        }
                        if (inv_fltlen[i] == 1) {
                                fprintf(outfile, "# length of fault %d\n", i);
                                var1par(params[nparams+1],j++, apsig_fltlen[i],
varnsig, varnstep, outfile);
                        }
                        if (inv_hmin[i] == 1) {
                                fprintf(outfile, "# hmin of fault %d\n", i);
                                var1par(params[nparams+1],j++, apsig_hmin[i],
varnsig, varnstep, outfile);
                        }
                        if (inv_hmax[i] == 1) {
                                fprintf(outfile, "# hmax of fault %d\n", i);
                                var1par(params[nparams+1],j++, apsig_hmax[i],
varnsig, varnstep, outfile);
                        }
                }
        }


/* output displacements computed at points on grid, if output file specified.
   This is done in batches of nystep stations, for nxstep columns */
        if (strcmp(dispgrid,"NONE") != 0) {
                printf("\nMaking computed grid displacement file ...\n");
                outfile = fopen(dispgrid,"w");
                if (outfile == NULL) {
                        printf("\n*** error opening grid file '%s'\n", dispgrid);
                        exit();
                }
                if (nystep > MAXSITE) {
                        printf("\n*** no. of grid sites too large\n");
                        exit();
                }
                nsites = nystep;
                ell2xy(fltx,flty, fltx,flty, nfaults, coortype, 0);
                for(i=0; i<nxstep; i++) {
                        for(j=0; j<nystep; j++) {
                                excoord[j+1] = minx + ((double)i*xstep);
                                eycoord[j+1] = miny + ((double)j*ystep);
                        }
                        ell2xy(excoord,eycoord, xcoord,ycoord, nsites, coortype, 0);
                        dummy = comp_disps();
                        for(j=1; j<=nystep; j++) {
                                fprintf(outfile,"%10.6lf %10.6lf  %8.4lf %8.4lf
%8.4lf %8.4lf %8.4lf", excoord[j], eycoord[j], comp_xdisp[j],comp_ydisp[j],
0.0,0.0,0.0);
                                if (deltaz_out == 1) fprintf(outfile,"  %8.4lf
%8.4lf", comp_zdisp[j], 0.0);
                                fprintf(outfile,"\n");
                        }
                }
                fclose(outfile);
        }


        printf("\nFinished.\n\n");


/* free memory for global arrays */
        globfree();


/* free memory for local arrays */
        free_dvector(excoord,1,MAXSITE);
        free_dvector(eycoord,1,MAXSITE);
        free_dvector(indx,1,MAXSITE*MAXDIM);
        free_dvector(tmpvec,1,MAXSITE*MAXDIM);
        free_dvector(tmpcopy,1,MAXSITE*MAXDIM);
        free_dvector(penalty,1,NOFFS+MAXFLT*NFLAGS);
        free_dmatrix(params,1,1+NOFFS+MAXFLT*NFLAGS,1,NOFFS+MAXFLT*NFLAGS);
        free_dvector(best_strike,1,MAXFLT);
        free_dvector(best_dip,1,MAXFLT);
        free_dvector(best_rake,1,MAXFLT);
        free_dvector(best_slp,1,MAXFLT);
        free_dvector(best_fltx,1,MAXFLT);
        free_dvector(best_flty,1,MAXFLT);
        free_dvector(best_fltlen,1,MAXFLT);
        free_dvector(best_hmin,1,MAXFLT);
        free_dvector(best_hmax,1,MAXFLT);


/* arrays wgt (global), cvdisp (local) and cvcopy (local) are dynamically
  allocated so must be dynamically de-allocated (unless we only have unit vcm.
  in which case they were not allocated) */
        if (covtype != 1) {
                free_dmatrix(wgt,1,nsites*ndims,1,nsites*ndims);
                free_dmatrix(cvdisp,1,nsites*ndims,1,nsites*ndims);
        }
        if (covtype == 2) free_dmatrix(cvcopy,1,nsites*ndims,1,nsites*ndims);
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* amoeba.c
   slight modification of NR subroutine (do not print error message if max
  number of iterations exceeded).
  For comments and explanation see Press et al. (1980), section 10.4.
*/

#include <math.h>
#include <stdio.h>
#include "nr.h"
#include "nrutil.h"

#define NMAX 5000

#define GET_PSUM \
        for (j=1;j<=ndim;j++) { \
                for (i=1,sum=0.0;i<=mpts;i++) sum += p[i][j]; \
                psum[j]=sum;}
static double swap;
#define SWAP(a,b) {swap=(a); (a)=(b); (b)=swap;}
```

```
#include "global.c"


int amoeba(p,y,ndim,ftol,funk,nfunk,niter)
double **p,y[],ftol,(*funk)();
int ndim,*nfunk,*niter;
{
        int i,j,ilo,ihi,inhi,mpts=ndim+1;
        int iter=0;
        double ytry,ysave,sum,rtol,amotry(),*psum;

        psum=dvector(1,ndim);
        *nfunk=0;
        GET_PSUM
        for (;;) {
                iter += 1;
                ilo=1;
                ihi = y[1]>y[2] ? (inhi=2,1) : (inhi=1,2);
                for (i=1;i<=mpts;i++) {
                        if (y[i] < y[ilo]) ilo=i;
                        if (y[i] > y[ihi]) {
                                inhi=ihi;
                                ihi=i;
                        } else if (y[i] > y[inhi] && i != ihi) inhi=i;
                }
                rtol=2.0*fabs(y[ihi]-y[ilo])/(fabs(y[ihi])+fabs(y[ilo]));
                if (rtol < ftol) {
                        SWAP(y[mpts],y[ilo])
                        for(i=1;i<=ndim;i++) SWAP(p[mpts][i],p[ilo][i])
                        break;
                }
/*              if (*nfunk >= NMAX) nrerror("Too many iterations in AMOEBA"); */
                if (*nfunk >= NMAX) return 1;
                *nfunk += 2;
                ytry=amotry(p,y,psum,ndim,funk,ihi,-1.0);
                if (ytry <= y[ilo])
                        ytry=amotry(p,y,psum,ndim,funk,ihi,2.0);
                else if (ytry >= y[inhi]) {
                        ysave=y[ihi];
                        ytry=amotry(p,y,psum,ndim,funk,ihi,0.5);
                        if (ytry >= ysave) {
                                for (i=1;i<=mpts;i++) {
                                        if (i != ilo) {
                                                for (j=1;j<=ndim;j++)
                                                        p[i][j]=psum[j]=0.5*(p[i][j]+
p[ilo][j]);

                                                y[i]=(*funk)(psum);
                                        }
                                }
                                *nfunk += ndim;
                                GET_PSUM
                        }
                } else --(*nfunk);
                if (ytry > 1.0e25) {
/* hack to abort if we hit a boundary */
                        return 1;
                }
        }
        *niter = iter;
        free_dvector(psum,1,ndim);
        return 0;
}

double amotry(p,y,psum,ndim,funk,ihi,fac)
double **p,*y,*psum,(*funk)(),fac;
int ndim,ihi;
{
        int j;
        double fac1,fac2,ytry,*ptry;

        ptry=dvector(1,ndim);
        fac1=(1.0-fac)/ndim;
        fac2=fac1-fac;
        for (j=1;j<=ndim;j++) ptry[j]=psum[j]*fac1-p[ihi][j]*fac2;
        ytry=(*funk)(ptry);
        if (ytry < y[ihi]) {
                y[ihi]=ytry;
                for (j=1;j<=ndim;j++) {
                        psum[j] += ptry[j]-p[ihi][j];
                        p[ihi][j]=ptry[j];
                }
        }
        free_dvector(ptry,1,ndim);
        return ytry;
}

#undef GET_PSUM
#undef NMAX
#undef SWAP

#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to compute displacements at stations in array xcoord[1..nsites],
  ycoord[1..nsites].  The earthquake parameters are in global arrays.  Routine
  dc3d3 (a stripped-down C version of Okada's own code) is used to do the actual
  computation; all that is done here is to apply the scale and rotation
  parameters and sum the contributions of the faults.

   pjc jun '95
*/

#include <stdio.h>
#include <math.h>
#include "nrutil.h"


#include "global.c"


int comp_disps()


{
        register int i,j;
        int iret;
        static int singflag = 0;
        double ct,st;
        double alpha = (LAMBDA + MU) / (LAMBDA + 2.0*MU);
        double x,y,xx,yy;
        double ux,uy,uz;
        FILE *stafile;
        int dc3d3();


/* loop over all sites, apply scale and rotation */
        for(i=1; i<=nsites; i++) {
                comp_xdisp[i] = u0;
                comp_ydisp[i] = v0;
                comp_zdisp[i] = w0;

                comp_xdisp[i] -= rigrot*ycoord[i] / 1.0e3;
```

```
                comp_ydisp[i] += rigrot*xcoord[i] / 1.0e3;

                comp_xdisp[i] += scale*xcoord[i] / 1.0e3;
                comp_ydisp[i] += scale*ycoord[i] / 1.0e3;
        }

/* loop over all faults, apply quake dispacement to each site for each fault */
        for(i=1; i<=nfaults; i++) {
                ct = cos((strike[i]+90.0)*DEG2RAD);
                st = sin((strike[i]+90.0)*DEG2RAD);

                for(j=1; j<=nsites; j++) {
                        xx = xcoord[j]-fltx[i];
                        yy = ycoord[j]-flty[i];
                        x = ct*xx-st*yy;
                        y = ct*yy+st*xx;
                        iret = dc3d3(alpha, x,y, -dip[i], al1[i],al2[i], aw1[i],
aw2[i], us[i],ud[i], &ux,&uy,&uz);
                        if ((iret==1) && (singflag == 0)) {
                                printf("\n*** Singularities encountered in
displacement computations.\n");
                                singflag = 1;
                        }
                        comp_xdisp[j] += (ct*ux+st*uy)/100.0;
                        comp_ydisp[j] += (ct*uy-st*ux)/100.0;
                        comp_zdisp[j] += uz/100.0;
                }
        }

        return iret;
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* translation of Okada's DC3D.f subroutine into 'good' C code.
   ... which has then been pruned to compute ux,uy,uz for z=0 only, no tensile
   components, depth=0.

   pjc july '95
*/


#include <math.h>
#include <stdio.h>
#include "nr.h"
#include "nrutil.h"


/* fortran common blocks are now global variables */
/* common block 0 */
double c0_alp3;
double c0_sd,c0_cd;
double c0_cdcd,c0_sdcd;
/* common block 2 */
double c2_r;
double c2_y,c2_d,c2_tt;
double c2_ale;
double c2_x11,c2_y11;


/* constants are implemented as macros */
#define F0 0.0
#define F1 1.0
#define F2 2.0
#define PI2 6.283185307179586
#define EPS 1.0e-6


int dc3d3(alpha, x,y, dip, al1,al2, aw1,aw2, disl1,disl2, ux,uy,uz)

double alpha;
double x,y;
double dip;
double al1,al2,aw1,aw2;
double disl1,disl2;
double *ux,*uy,*uz;

{
        int i,j,k;
        int jxi,jet;
        double et,xi;
        double p,q;
        double *u,*du;
        double *dub;
        void dccon0(),dccon2(),ub();


        u = dvector(1,3);
        du = dvector(1,3);
        dub = dvector(1,3);


        for(i=1; i<=3; i++) {
                u[i] = F0;
                dub[i] = F0;
        }

        dccon0(alpha,dip);

        p = y*c0_cd;
        q = y*c0_sd;
        jxi = 0;
        jet = 0;
        if (((x-al1)*(x-al2)) <= F0) jxi = 1;
        if (((p-aw1)*(p-aw2)) <= F0) jet = 1;

        for(k=1; k<=2; k++) {
                if (k == 1) et = p-aw1;
                else et = p-aw2;
                for(j=1; j<=2; j++) {
                        if (j == 1) xi = x-al1;
                        else xi = x-al2;
                        dccon2(&xi,&et,&q,c0_sd,c0_cd);
                        if      (((q == F0)     && (((jxi == 1) && (et == F0))
                                                || ((jet == 1) && (xi == F0))))
                                || (c2_r == F0)) {
                                /* singular problem: return error code */
                                *ux = *uy = *uz = F0;
                                free_dvector(u,1,3);
                                free_dvector(du,1,3);
                                free_dvector(dub,1,3);
                                return 1;
                        }
                        ub(xi,et,q,disl1,disl2,dub);
                        du[1] = dub[1];
                        du[2] = dub[2]*c0_cd - dub[3]*c0_sd;
                        du[3] = dub[2]*c0_sd + dub[3]*c0_cd;
                        for(i=1; i<=3; i++) {
                                if ((j+k) != 3) u[i] += du[i];
                                else u[i] -= du[i];
                        }
                }
        }
```

```
        *ux = u[1]; *uy = u[2]; *uz = u[3];

        free_dvector(u,1,3);
        free_dvector(du,1,3);
        free_dvector(dub,1,3);

        return 0;
}


/* subroutine dccon0 calculates medium and fault-dip constants */

void dccon0(alpha,dip)

double alpha,dip;

{
        double p18;

        c0_alp3 = (F1-alpha)/alpha;

        p18 = PI2/360.0;
        c0_sd = sin(dip*p18);
        c0_cd = cos(dip*p18);
        if (fabs(c0_cd) < EPS) {
                c0_cd = F0;
                if (c0_sd > F0) c0_sd = F1;
                if (c0_sd < F0) c0_sd = -F1;
        }
        c0_cdcd = c0_cd * c0_cd;
        c0_sdcd = c0_sd * c0_cd;
}


/* subroutine dccon2 calculates station geometry constants for finite source */

void dccon2(xi,et,q,sd,cd)

double *xi,*et,*q,sd,cd;

{
        double rxi,ret;
        double dc_xi,dc_et,dc_q;
        double dc_max;

        dc_max = DMAX(fabs(*xi),DMAX(fabs(*et),fabs(*q)));
        if ((fabs(*xi/dc_max) < EPS) || (fabs(*xi) < EPS)) *xi = F0;
        if ((fabs(*et/dc_max) < EPS) || (fabs(*et) < EPS)) *et = F0;
        if ((fabs(*q /dc_max) < EPS) || (fabs(*q ) < EPS)) *q  = F0;

        dc_xi = *xi; dc_et = *et; dc_q = *q;

        c2_r = hypot(dc_xi,hypot(dc_et,dc_q));
        if (c2_r == F0) return;
        c2_y = dc_et*cd + dc_q*sd;
        c2_d = dc_et*sd - dc_q*cd;

        if (dc_q == F0) c2_tt = F0;
        else c2_tt = atan(dc_xi*dc_et/(dc_q*c2_r));

        if ((dc_xi < F0) && (dc_q == F0) && (dc_et == F0)) {
                c2_x11 = F0;
        }
        else {
                rxi = c2_r + dc_xi;
                c2_x11 = F1/(c2_r*rxi);
        }

        if ((dc_et < F0) && (dc_q == F0) && (dc_xi == F0)) {
c2_r-dc_et, c2_r, dc_et, dc_q, dc_xi);
                c2_ale = -log(c2_r-dc_et);
                c2_y11 = F0;
        }
        else {
                ret = c2_r + dc_et;
ret, c2_r, dc_et, dc_q, dc_xi);
                c2_ale = log(ret);
                c2_y11 = F1/(c2_r*ret);
        }
}


/* subroutine ub is part B of displacement and strain at depth due to buried
   fault in semi-infinite medium */

void ub(xi,et,q,disl1,disl2,u)

double xi,et,q;
double disl1,disl2;
double *u;


{
        int i;
        double rd,rd2;
        double ai1,ai2,ai3,ai4;
        double x,qx,qy;
        double *du;

        du = dvector(1,3);

        rd = c2_r+c2_d;
        if (c0_cd != F0) {
                if (xi == F0) ai4 = F0;
                else {
                        x = hypot(xi,q);
                        ai4 = F1/c0_cdcd * (xi/rd*c0_sdcd + F2*atan((et*(x+q*c0_cd) +
x*(c2_r+x)*c0_sd) / (xi*(c2_r+x)*c0_cd)) );
                }
                ai3 = (c2_y*c0_cd/rd - c2_ale + c0_sd*log(rd)) / c0_cdcd;
        }
        else {
                rd2 = rd*rd;
                ai3 = (et/rd + c2_y*q/rd2 - c2_ale) / F2;
                ai4 = xi*c2_y/rd2/F2;
        }

        ai1 = -xi/rd*c0_cd - ai4*c0_sd;
        ai2 = log(rd) + ai3*c0_sd;

        for(i=1; i<=3; i++) u[i] = F0;
        qx = q*c2_x11;
        qy = q*c2_y11;

        /* strike-slip contribution */
        if (disl1 != F0) {
                du[ 1] = - xi*qy  - c2_tt  - c0_alp3*ai1*c0_sd;
                du[ 2] = - q/c2_r          + c0_alp3*c2_y/rd*c0_sd;
                du[ 3] =   q*qy            - c0_alp3*ai2*c0_sd;
                for(i=1; i<=3; i++) u[i] += disl1/PI2*du[i];
        }

        /* dip-slip contribution */
        if (disl2 != F0) {
                du[ 1] = - q/c2_r          + c0_alp3*ai3*c0_sdcd;
                du[ 2] = - et*qx - c2_tt   - c0_alp3*xi/rd*c0_sdcd;
                du[ 3] =   q*qx            + c0_alp3*ai4*c0_sdcd;
```

```
                for(i=1; i<=3; i++) u[i] += disl2/PI2*du[i];
        }

        free_dvector(du,1,3);
}


#undef F0
#undef F2
#undef F2
#undef PI2
#undef EPS
*******************************************************************************


*******************************************************************************
/* subroutine to convert site coords in the arrays xx[],yy[] into
  x-y coords in km in the arrays x[],y[].  If coortype==1,
  conversion from lat/lon to km must be done, otherwise the coords are just
  copied over (relative to the centroid). The centroid of the network is
  returned in the global variables orig_x,orig_y and all coords are relative
  to this.

   pjc jun '95
*/

#include <stdio.h>
#include <math.h>
#include "nrutil.h"

#include "global.c"


void ell2xy(xx,yy,x,y,ncoords,coortype,change_orig)

double *x,*y,*xx,*yy;
int ncoords,coortype,change_orig;

{
        int i;
        double rprimev,rmerid,tmp;

/* if necessary, compute the origin (centroid of the network).  This is
  done on the first call to ell2xy from okinv */
        if (change_orig == 1) {
                orig_x = 0.0; orig_y = 0.0;
                for(i=1; i<=ncoords; i++) {
                        orig_x += xx[i];
                        orig_y += yy[i];
                }
                orig_x /= (double)ncoords; orig_y /= (double)ncoords;
        }

/* if lat/lon conversion required, compute radii of curvature */
        if (coortype==1) {
                tmp = 1 - pow((ECCEN*sin(orig_y*DEG2RAD)),2.0);
                rprimev = RADIUS / sqrt(tmp);
                rmerid = RADIUS*(1 - ECCEN*ECCEN) / (tmp*sqrt(tmp));
        }

/* for each station, apply a translation to the origin, then apply lat/lon
  conversion if required */
        for(i=1; i<=ncoords; i++) {
                x[i] = xx[i]-orig_x;
                y[i] = yy[i]-orig_y;
                if (coortype==1) {
                        x[i] *= (rprimev/1000.0)*cos(orig_y*DEG2RAD) * DEG2RAD;
                        y[i] *= (rmerid/1000.0) * DEG2RAD;
                }
        }
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to find the station number of a named station, or -999 if
   station not found
   pjc jul '94
*/

#include <string.h>

#define MAXSITE 100     /* max number of sites */


int findstnum(name,namelist,nsites)

char name[];
char namelist[][32];
int nsites;

{
        int i;
        int found=0;

        for(i=1;found<1;i++) {
                if (i > nsites) {
                        found=2;
                }
                if (strcmp(name,namelist[i]) == 0) {
                        found=1;
                }
        }
        if (found == 1) {
                return(i-1);
        } else {
                return(-999);
        }
}


#undef MAXSITE
*******************************************************************************


*******************************************************************************
/* subroutine to read in a covariance matrix from a file in .CEN format (which
   is equivalent to the Bernese 3.4 format, but having had a similarity
   transform applied so that the components are in the coordinate system of
   east, north, up appropriate at the centroid of the network).

   pjc jun '95
*/

#include <string.h>
#include <stdio.h>
#include "nr.h"
#include "nrutil.h"

#define SIGSQMIN 4.0e-06        /* default std error of a fixed coord is 2mm */
#define SIGSQTOL 1.0e-10        /* 'zero' variance of a fixed coordinate */

#include "global.c"
```

```
void getcovs(filename,psvmname,sitename,cvdisp,ndims)

char filename[],psvmname[],sitename[][32];
double **cvdisp;
int ndims;

{
        FILE *covarfile,*psvmfile;
        int status;
        int i,j,n;
        int stnum1,stnum2;
        char st1[32],st2[32];
        char neu1[3],neu2[3];
        char dummystring[255];
        double elem,unitwgt;
        double sx,sy,sz;

        covtype = 0;

/* set up unit vcm (unweighted displacements) if required */
        if (strcmp(filename,"UNIT") == 0) {
                printf("Setting covariance matrix to unit matrix\n");
                covtype = 1;
                return;
        }

/* alternatively, load in variances only from .psvm file */
        else if (strcmp(filename,"PSVM") == 0) {
                printf("\nInitialising covariance matrix ...\n");
                cvdisp = dmatrix(1,nsites*ndims,1,nsites*ndims);
                for(i=1;i<=nsites*MAXDIM;i++) {
                        for(j=1;j<=nsites*MAXDIM;j++) cvdisp[i][j] = 0.0;
                }
                printf("Reading covariances from .psvm file\n");
                covtype = 2;
                psvmfile = fopen(psvmname,"r");
                if (psvmfile == NULL) {
                        printf("\n*** error opening displacement file: '%s'\n",
psvmname);
                        exit();
                }
                n = 0;
                status = 0;
                while (status != EOF) {
                        n++;
                        if (deltaz_in != 0) status = fscanf(psvmfile,"%*f %*f %*f %*f
%lf %lf %*f %*f %lf %s\n", &sx,&sy,&sz, st1);
                        else status = fscanf(psvmfile,"%*f %*f %*f %*f %lf %lf %*f
%s\n", &sx,&sy, st1);
                        if (n > MAXSITE) {
                                printf("\n*** maximum number of sites exceeded\n");
                                printf("    number of sites >= %d\n", n);
                                exit();
                        }
                        stnum1 = findstnum(st1,sitename,nsites);
                        cvdisp[stnum1][stnum1] = sx*sx;
                        stnum1 += nsites;
                        cvdisp[stnum1][stnum1] = sy*sy;
                        if (deltaz_in != 0) {
                                stnum1 += nsites;
                                cvdisp[stnum1][stnum1] = sz*sz;
                        }
                }
        }

/* or load in the full vcm from a .CEN file */
        else {
                printf("\nInitialising covariance matrix ...\n");
                cvdisp = dmatrix(1,nsites*ndims,1,nsites*ndims);
                for(i=1;i<=nsites*MAXDIM;i++) {
                        for(j=1;j<=nsites*MAXDIM;j++) cvdisp[i][j] = 0.0;
                }
                printf("Reading covariances ... ");
                covarfile=fopen(filename,"r");
                if (covarfile == NULL) {
                        printf("\n*** error opening file: '%s'\n",filename);
                        exit();
                }
                status=0;
                for(i=1;i<=6;i++) fgets(dummystring,255,covarfile);
                fscanf(covarfile, "RMS OF UNIT WEIGHT: %lf", &unitwgt);
                for(i=1;i<=5;i++) fgets(dummystring,255,covarfile);
                while (status != EOF) {
                        status = fscanf(covarfile,"%s %s %s %s %lf\n", st1, neu1,
st2, neu2, &elem);
                        if ((status != 0) && (status != EOF)) {
                                stnum1=findstnum(st1,sitename,nsites);
                                stnum2=findstnum(st2,sitename,nsites);
                                i=j=-999;
                                if (strcmp(neu1,"E") == 0) i=0;
                                if (strcmp(neu1,"N") == 0) i=nsites;
                                if ((strcmp(neu1,"U") == 0) && (deltaz_in != 0))
i=nsites*2;
                                if (strcmp(neu2,"E") == 0) j=0;
                                if (strcmp(neu2,"N") == 0) j=nsites;
                                if ((strcmp(neu2,"U") == 0) && (deltaz_in != 0))
j=nsites*2;
                                i+=stnum1; j+=stnum2;
                                if ((i>0) && (j>0)) {
                                        cvdisp[i][j] = elem*unitwgt*unitwgt;
                                        cvdisp[j][i] = elem*unitwgt*unitwgt;
                                }
                        }
                }
                printf("done\n");
                fclose(covarfile);
        }

/* check covariance matrix for zero variances, adjust them to SIGSQMIN ( this
  prevents nastiness if fixed sites are included */
        for(i=1;i<=nsites*2;i++) {
                if (cvdisp[i][i] < SIGSQTOL) {
                        if (i <= nsites) {
                                printf("  Variance of  %s:E increased from %10.3e to
%10.3le\n",sitename[i],
cvdisp[i][i],SIGSQMIN);
                        }
                        else {
                                printf("  Variance of  %s:N increased from %10.3e to
%10.3le\n",
sitename[i-nsites],cvdisp[i][i],SIGSQMIN);
                        }
                        cvdisp[i][i] = SIGSQMIN;
                }
        }

}


#undef SIGSQMIN
#undef SIGSQTOL


#include "globend.c"
*******************************************************************************
```

```
*******************************************************************************
/* subroutine to get coordinates, displacements and sitenames from a file of
   the format used for psvelomeca ie one site per line, arranged as:
   lon lat  edisp ndisp  esig nsig dummy  sitename
   if the global flag deltaz_in is 1, then the format expected is:
   lon lat  edisp ndisp  esig nsig dummy  udisp usig  sitename

   pjc jun '95
*/

#include <stdio.h>
#include <string.h>

#include "global.c"


void getpsvm(filename,excoord,eycoord,sitename)

char filename[],sitename[][32];
double *excoord,*eycoord;

{
        FILE *psvmfile;
        int i,n,status;
        double cx,cy,dx,dy,dz;

/* open file */
        printf("\nReading displacements ... ");
        psvmfile=fopen(filename,"r");
        if (psvmfile == NULL) {
                printf("\n*** error opening displacement file: '%s'\n", filename);
                exit();
        }

/* read all lines from file */
        n=0;
        status=0;
        while (status != EOF) {
                n++;
                if (deltaz_in != 0) status = fscanf(psvmfile,"%lf %lf %lf %lf %*f
%*f %*f %lf %*f %s\n", &cx,&cy, &dx,&dy,&dz, sitename[n]);
                else status = fscanf(psvmfile,"%lf %lf %lf %lf %*f %*f %*f %s\n",
&cx,&cy, &dx,&dy, sitename[n]);
                if (n > MAXSITE) {
                        printf("\n*** maximum number of sites exceeded\n");
                        printf("    number of sites >= %d\n", n);
                        exit();
                }
                excoord[n] = cx;
                eycoord[n] = cy;
                xdisp[n] = dx;
                ydisp[n] = dy;
                if (deltaz_in != 0) zdisp[n] = dz;
        }

/* close and tidy up */
        printf("%d sites found\n",--n);
        fclose(psvmfile);
        nsites = n;
}


#include "globend.c"
*******************************************************************************
```

```
*******************************************************************************
/* global.c

  global variables for okinv.c and subroutines */


/* get macro defs (array sizes etc) */
#include "globdef.c"


/* function prototypes */
int amoeba();
int comp_disps();
int dc3d3();
double drand48();
void ell2xy();
int findstnum();
void getcovs();
void getpsvm();
void globres();
void globfree();
void invprint();
void lubksb();
void ludcmp();
void makesimp();
void mprove();
double monte();
void nrutil();
double penfunc();
double pen_L1dis();
double pen_L2dis();
double pen_Linfdis();
void readinps();
void var1crdpar();
void var1par();
void xy2ell();


/* Global variables */

double *xcoord;         /* coordinates of stations in km or geod form */
double *ycoord;

double orig_x;          /* coordinates of centroid of network, used as origin */
double orig_y;          /* always in same units as input file */

double *xdisp;          /* observed displacements of stations /m */
double *ydisp;
double *zdisp;

double **wgt;           /* weight matrix of observations */

double *comp_xdisp;     /* computed displacements of stations /m */
double *comp_ydisp;
double *comp_zdisp;

double u0;              /* translations of network in x,y,z /m */
double v0;
double w0;

double rigrot;          /* rigid rotation of network in microradians */
double scale;           /* scale change of network in ppm */

double *fltx;           /* coordinates of centre of fault break in km or geod */
double *flty;

double *strike;         /* strike, dip, rake of eq /degrees */
double *dip;            /* rake=0 is left-lateral, 90 is thrust, */
```

```
double *rake;           /* -90 is normal, 180 is right-lateral (for 0<dip<90) */

double *slp;            /* slip magnitude /m */

double *ud;             /* dip- and strike-slip components of motion /cm */
double *us;             /* us +ve is left-lateral for 0<dip<90 */

double *hmin;           /* min and max depths of rupture plane /km */
double *hmax;

double *fltlen;         /* length of fault break */

double *al1;            /* half-length of fault break */
double *al2;            /* al2[i] === - al1[i] */

double *aw1;            /* width down fault plane to top of rupture */
double *aw2;            /* width down fault plane fo bottom of rupture */

double *dumx;           /* temp store for coords of end of fault plane in km */
double *dumy;

double *end1x;          /* coords of end of fault break in km or geod form */
double *end1y;

double *end2x;          /* coords of other end of fault break */
double *end2y;

double *ap_strike;      /* a priori strike and uncertainty /degrees */
double *apsig_strike;

double *ap_dip;         /* a priori dip and uncertainty /degrees */
double *apsig_dip;

double *ap_rake;        /* a priori rake and uncertainty /degrees */
double *apsig_rake;

double *ap_slp;         /* a priori slip magnitude /m */
double *apsig_slp;

double *ap_fltx;        /* a priori x-coord of centre of break in km or geod */
double *apsig_fltx;     /* uncertainty  in km or geod form */
double *apsigkm_fltx;   /* uncertainty  in km */

double *ap_flty;        /* a priori y-coord of centre of break in km or geod */
double *apsig_flty;     /* uncertainty  in km or geod form */
double *apsigkm_flty;   /* uncertainty  in km */

double *ap_fltlen;      /* a priori fault length and uncertainty /km */
double *apsig_fltlen;

double *ap_hmin;        /* a priori min depth of rupture and uncertainty /km */
double *apsig_hmin;

double *ap_hmax;        /* a priori max depth of rupture and uncertainty /km */
double *apsig_hmax;

double ap_u0    ;       /* a priori translation in lon(x) and uncertainty /m */
double apsig_u0;

double ap_v0    ;       /* a priori translation in lat(y) and uncertainty /m */
double apsig_v0;

double ap_w0    ;       /* a priori translation in hgt(z) and uncertainty /m */
double apsig_w0;

double ap_rigrot;       /* a priori rigid rotation and uncertainty /microrad */
double apsig_rigrot;

double ap_scale;        /* a priori scale factor and uncertainty /ppm */
double apsig_scale;
```

```
int norm;               /* norm of penalty function (0=Linf, 1=L2, 2=L2) */

int nfaults;            /* no of faults */

int nsites;             /* no of obs stations */

int covtype;            /* type of covariance matrix (0=full, 1=unit, 2=diag) */

int deltaz_in;          /* =1 if height changes are input */
int use_dz;             /* =1 if height changes are to be used for inversion */

int inv_u0;             /* =1 if translations, rotation or scale to be found */
int inv_v0;
int inv_w0;
int inv_rigrot;
int inv_scale;

int *inv_strike;        /* =1 if strike, dip or rake of eq are to be found */
int *inv_dip;
int *inv_rake;

int *inv_slp;           /* =1 if slip magnitude of eq to be found */

int *inv_fltx;          /* =1 if coords of centre of fault break to be found */
int *inv_flty;

int *inv_fltlen;        /* =1 if fault break length of eq to be found */

int *inv_hmin;          /* =1 if min, max depths of rupture to be found */
int *inv_hmax;

int *inv_flags;         /* index for parameters to be found */
*****************************************************************************


*****************************************************************************
/* globdef.c

  macro definitions for okinv.c and subroutines */


/* Constants and array sizes */
#define BIGNUM  1.0e30          /* (almost) largest representable number */
#define DEG2RAD 0.017453293     /* degrees to radians factor */
#define ECCEN   0.081819191     /* eccentricity of GRS-80 ellipsoid */
#define FTOL    1.0e-5          /* tolerance for fractional change in penalty */
#define INVTOL  1.0e-12         /* tolerance for goodness of vcm inversion */
#define LAMBDA  3.23e10         /* elastic constant lambda */
#define MAXDIM  3               /* max no. of dimensions of observations */
#define MAXFLT  4               /* max no. of faults to solve */
#define MAXITER 5000            /* max no. of iterations */
#define MAXSITE 250             /* max no. of sites or grid points */
#define MU      3.23e10         /* elastic constant mu */
#define NMAX    5000            /* max no. of fn calls permitted in amoeba.c */
#define REFFAC  0.1             /* factor for reflecting simplex at boundary */
#define RADIUS  6378137.0       /* radius of GRS-80 ellipsoid */


/* indices for parameter types, used when communicating with amoeba routine */
#define NFLAGS  9               /* no. of flags for params for each fault */
#define NOFFS   5               /* no. of flags for offset params etc */
#define U0      1               /* U0..HMAX are hardwired array indices */
#define V0      2
#define W0      3
#define RIGROT  4
#define SCALE   5
#define STRIKE  1
```

```
#define DIP     2
#define RAKE    3
#define MO      4
#define FLTX    5
#define FLTY    6
#define FLTLEN  7
#define HMIN    8
#define HMAX    9
*******************************************************************************


*******************************************************************************
/* globend.c
  clean up macro definitions from globdef.c at end of okinv or subroutine
/*

#undef BIGNUM
#undef DEG2RAD
#undef ECCEN
#undef FTOL
#undef LAMBDA
#undef INVTOL
#undef MAXDIM
#undef MAXFLT
#undef MAXGRID
#undef MAXITER
#undef MAXOBS
#undef MAXSITE
#undef MU
#undef NMAX
#undef REFFAC
#undef RADIUS

#undef NFLAGS
#undef NOFFS
#undef UO
#undef VO
#undef WO
#undef RIGROT
#undef SCALE
#undef STRIKE
#undef DIP
#undef RAKE
#undef MO
#undef FLTX
#undef FLTY
#undef FLTLEN
#undef HMIN
#undef HMAX
*******************************************************************************


*******************************************************************************
/* subroutine to free memory for global arrays declared in globres.c
   for comments and explanation of variable names see global.c

   pjc jun '95
*/

#include "nrutil.h"

#include "global.c"


void globfree()


{
        free_dvector(xcoord,1,MAXSITE);
        free_dvector(ycoord,1,MAXSITE);
        free_dvector(xdisp,1,MAXSITE);
        free_dvector(ydisp,1,MAXSITE);
        free_dvector(zdisp,1,MAXSITE);
/*      free_dmatrix(wgt,1,MAXSITE*MAXDIM,1,MAXSITE*MAXDIM);    */
        free_dvector(comp_xdisp,1,MAXSITE);
        free_dvector(comp_ydisp,1,MAXSITE);
        free_dvector(comp_zdisp,1,MAXSITE);
        free_dvector(fltx,1,MAXFLT);
        free_dvector(flty,1,MAXFLT);
        free_dvector(strike,1,MAXFLT);
        free_dvector(dip,1,MAXFLT);
        free_dvector(rake,1,MAXFLT);
        free_dvector(slp,1,MAXFLT);
        free_dvector(ud,1,MAXFLT);
        free_dvector(us,1,MAXFLT);
        free_dvector(hmin,1,MAXFLT);
        free_dvector(hmax,1,MAXFLT);
        free_dvector(fltlen,1,MAXFLT);
        free_dvector(al1,1,MAXFLT);
        free_dvector(al2,1,MAXFLT);
        free_dvector(aw1,1,MAXFLT);
        free_dvector(aw2,1,MAXFLT);
        free_dvector(dumx,1,2);
        free_dvector(dumy,1,2);
        free_dvector(end1x,1,MAXFLT);
        free_dvector(end1y,1,MAXFLT);
        free_dvector(end2x,1,MAXFLT);
        free_dvector(end2y,1,MAXFLT);
        free_dvector(ap_fltx,1,MAXFLT);
        free_dvector(ap_flty,1,MAXFLT);
        free_dvector(ap_strike,1,MAXFLT);
        free_dvector(ap_dip,1,MAXFLT);
        free_dvector(ap_rake,1,MAXFLT);
        free_dvector(ap_slp,1,MAXFLT);
        free_dvector(ap_hmin,1,MAXFLT);
        free_dvector(ap_hmax,1,MAXFLT);
        free_dvector(ap_fltlen,1,MAXFLT);
        free_dvector(apsigkm_fltx,1,MAXFLT);
        free_dvector(apsigkm_flty,1,MAXFLT);
        free_dvector(apsig_fltx,1,MAXFLT);
        free_dvector(apsig_flty,1,MAXFLT);
        free_dvector(apsig_strike,1,MAXFLT);
        free_dvector(apsig_dip,1,MAXFLT);
        free_dvector(apsig_rake,1,MAXFLT);
        free_dvector(apsig_slp,1,MAXFLT);
        free_dvector(apsig_hmin,1,MAXFLT);
        free_dvector(apsig_hmax,1,MAXFLT);
        free_dvector(apsig_fltlen,1,MAXFLT);
        free_ivector(inv_strike,1,MAXFLT);
        free_ivector(inv_dip,1,MAXFLT);
        free_ivector(inv_rake,1,MAXFLT);
        free_ivector(inv_slp,1,MAXFLT);
        free_ivector(inv_hmin,1,MAXFLT);
        free_ivector(inv_hmax,1,MAXFLT);
        free_ivector(inv_fltlen,1,MAXFLT);
        free_ivector(inv_fltx,1,MAXFLT);
        free_ivector(inv_flty,1,MAXFLT);
        free_ivector(inv_flags,1,NOFFS+MAXFLT*NFLAGS);
}


#include "globend.c"
*******************************************************************************
```

```
*******************************************************************************
/* subroutine to reserve memory for global arrays.
   For comments and explanation of variables see global.c

   pjc jun '95
*/

#include "nrutil.h"

#include "global.c"


void globres()


{
        xcoord = dvector(1,MAXSITE);
        ycoord = dvector(1,MAXSITE);
        xdisp = dvector(1,MAXSITE);
        ydisp = dvector(1,MAXSITE);
        zdisp = dvector(1,MAXSITE);
/*      wgt = dmatrix(1,MAXSITE*MAXDIM,1,MAXSITE*MAXDIM);       */
/* weight matrix now allocated in okinv.c after nsites is known */
        comp_xdisp = dvector(1,MAXSITE);
        comp_ydisp = dvector(1,MAXSITE);
        comp_zdisp = dvector(1,MAXSITE);
        fltx = dvector(1,MAXFLT);
        flty = dvector(1,MAXFLT);
        strike = dvector(1,MAXFLT);
        dip = dvector(1,MAXFLT);
        rake = dvector(1,MAXFLT);
        slp = dvector(1,MAXFLT);
        ud = dvector(1,MAXFLT);
        us = dvector(1,MAXFLT);
        hmin = dvector(1,MAXFLT);
        hmax = dvector(1,MAXFLT);
        fltlen = dvector(1,MAXFLT);
        al1 = dvector(1,MAXFLT);
        al2 = dvector(1,MAXFLT);
        aw1 = dvector(1,MAXFLT);
        aw2 = dvector(1,MAXFLT);
        dumx = dvector(1,2);
        dumy = dvector(1,2);
        end1x = dvector(1,MAXFLT);
        end1y = dvector(1,MAXFLT);
        end2x = dvector(1,MAXFLT);
        end2y = dvector(1,MAXFLT);
        ap_fltx = dvector(1,MAXFLT);
        ap_flty = dvector(1,MAXFLT);
        ap_strike = dvector(1,MAXFLT);
        ap_dip = dvector(1,MAXFLT);
        ap_rake = dvector(1,MAXFLT);
        ap_slp = dvector(1,MAXFLT);
        ap_hmin = dvector(1,MAXFLT);
        ap_hmax = dvector(1,MAXFLT);
        ap_fltlen = dvector(1,MAXFLT);
        apsigkm_fltx = dvector(1,MAXFLT);
        apsigkm_flty = dvector(1,MAXFLT);
        apsig_fltx = dvector(1,MAXFLT);
        apsig_flty = dvector(1,MAXFLT);
        apsig_strike = dvector(1,MAXFLT);
        apsig_dip = dvector(1,MAXFLT);
        apsig_rake = dvector(1,MAXFLT);
        apsig_slp = dvector(1,MAXFLT);
        apsig_hmin = dvector(1,MAXFLT);
        apsig_hmax = dvector(1,MAXFLT);
        apsig_fltlen = dvector(1,MAXFLT);
        inv_strike = ivector(1,MAXFLT);
        inv_dip = ivector(1,MAXFLT);
        inv_rake = ivector(1,MAXFLT);
        inv_slp = ivector(1,MAXFLT);
        inv_hmin = ivector(1,MAXFLT);
        inv_hmax = ivector(1,MAXFLT);
        inv_fltlen = ivector(1,MAXFLT);
        inv_fltx = ivector(1,MAXFLT);
        inv_flty = ivector(1,MAXFLT);
        inv_flags = ivector(1,NOFFS+MAXFLT*NFLAGS);
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to print out results of inversion to specified file.

   pjc jun '95
*/

#include <stdio.h>
#include <math.h>
#include "nrutil.h"

#include "global.c"


void invprint(outfile, ndims,nmonte, niter,nfunks, finalpen)

FILE *outfile;
double finalpen;
int ndims,nmonte,niter,nfunks;

{
        int i;
        double w;


/* check ndims is ok */
        if (use_dz == -1) ndims = 1;


/* first print out parameters for each fault */
        for(i=1; i<=nfaults; i++) {
                w = (hmax[i]-hmin[i])/sin(dip[i]*DEG2RAD);
                fprintf(outfile,"\nValue           Sigma           Invert (1/0)
Parameters of fault %1d\n", i);
                fprintf(outfile,"****.****       ***.****        *
-------------------------------\n");
                fprintf(outfile,"%9.4lf       %8.4lf        %1d               Strike
(degrees)\n", strike[i],apsig_strike[i],inv_strike[i]);
                fprintf(outfile,"%9.4lf       %8.4lf        %1d               Dip
(degrees)\n", dip[i],apsig_dip[i],inv_dip[i]);
                fprintf(outfile,"%9.4lf       %8.4lf        %1d               Rake
(degrees)\n", rake[i],apsig_rake[i],inv_rake[i]);
                fprintf(outfile,"%9.4lf       %8.4lf        %1d               Slip
(m)\n", slp[i], apsig_slp[i], inv_slp[i]);
                fprintf(outfile,"%9.4lf       %8.4lf        %1d               lon(x)
} of projected centre of\n", fltx[i],apsig_fltx[i],inv_fltx[i]);
                fprintf(outfile,"%9.4lf       %8.4lf        %1d               lat(y)
} surface break\n", flty[i],apsig_flty[i],inv_flty[i]);
                fprintf(outfile,"%9.4lf       %8.4lf        %1d               length
of scarp (km)\n", fltlen[i],apsig_fltlen[i],inv_fltlen[i]);
```

```
                fprintf(outfile,"%9.4lf       %8.4lf        %1d              minimum
depth of dislocation\n", hmin[i],apsig_hmin[i],inv_hmin[i]);
                fprintf(outfile,"%9.4lf       %8.4lf        %1d              maximum
depth of dislocation\n", hmax[i],apsig_hmax[i],inv_hmax[i]);
                fprintf(outfile,"\n");
                fprintf(outfile,"%9.4lf                                       Moment
(x10^18 N m)\n", MU*w*fltlen[i]*slp[i]/pow(10.0,12.0));
                fprintf(outfile,"%9.4lf                                       dip-
slip dislocation (m)\n", ud[i]/100.0);
                fprintf(outfile,"%9.4lf                                       strike-
slip dislocation (m)\n", us[i]/100.0);
                fprintf(outfile,"%9.4lf                                       fault
plane width (km)\n", w);
fprintf(outfile,"%9.4lf                                 fault
plane area (km^2)\n", w*fltlen[i]);
                fprintf(outfile,"%9.4lf                                       lon(x)
} of 1st end of\n", end1x[i]);
                fprintf(outfile,"%9.4lf                                       lat(y)
} surface break\n", end1y[i]);
                fprintf(outfile,"%9.4lf                                       lon(x)
} of 2nd end of\n", end2x[i]);
                fprintf(outfile,"%9.4lf                                       lat(y)
} surface break\n", end2y[i]);
        }


/* finally print out general parameters and penalty values */
        fprintf(outfile,"\n\nValue           Sigma           Invert (1/0)    General
parameters\n");
        fprintf(outfile,"****.****       ***.****        *
--------------------------------\n");
        fprintf(outfile,"%9.4lf       %8.4lf        %1d               translation in
lon(x) (m)\n", u0,apsig_u0,inv_u0);
        fprintf(outfile,"%9.4lf       %8.4lf        %1d               translation in
lat(y) (m)\n", v0,apsig_v0,inv_v0);
        fprintf(outfile,"%9.4lf       %8.4lf        %1d               translation in
hgt(z) (m)\n", w0,apsig_w0,inv_w0);
        fprintf(outfile,"%9.4lf       %8.4lf        %1d               rigid-body
rotation (microrad)\n", rigrot,apsig_rigrot,inv_rigrot);
        fprintf(outfile,"%9.4lf       %8.4lf        %1d               scale
factor (ppm)\n", scale,apsig_scale,inv_scale);
        fprintf(outfile,"\n");
        fprintf(outfile,"%9.4lf                                       lon(x) } of
centroid\n", orig_x);
        fprintf(outfile,"%9.4lf                                       lat(y) } of
network\n", orig_y);
        fprintf(outfile,"%9d                                        no. of
observations\n", nsites*ndims);
        fprintf(outfile,"\n");
        fprintf(outfile,"  %11.4le                                    total
penalty\n", finalpen);
        fprintf(outfile,"  %11.4le                                    mean penalty\n",
finalpen/((double)(nsites*ndims)));
        if (norm == 2) fprintf(outfile,"  %11.4le
root mean penalty\n", sqrt(finalpen/((double)(nsites*ndims))));
        fprintf(outfile,"\n");
        fprintf(outfile,"%9d                                        no. of Monte-Carlo
restarts\n", nmonte);
        fprintf(outfile,"%9d                                        no. of simplex
moves\n", niter);
        fprintf(outfile,"%9d                                        no. of function
calls\n", nfunks);
}
```

```
#include "globend.c"
*******************************************************************************


*******************************************************************************
/* lubksb.c
  Taken directly from NR in C, section 2.3
/*
*******************************************************************************


*******************************************************************************
/* ludcmp.c
  Taken directly from NR in C, section 2.3
/*
*******************************************************************************


*******************************************************************************
/* subroutine to get set up simplex in array params[1..nparams+1][1..nparams].
  All parameters are taken from the a priori value +/- the search width, then
  each parameter in turn is perturbed by +/- the preturbation width.

   pjc jun '95
*/

#include <stdio.h>
#include <math.h>
#include "nrutil.h"

#include "global.c"
double monte();


void makesimp(nparams,params,pertfac,searchnsig,nmonte)

int nparams,nmonte;
double **params;
double pertfac,searchnsig;


{
        int i,ii,j;
        int pmone();


        for(i=1; i<=NOFFS+MAXFLT*NFLAGS; i++) inv_flags[i] = 0;
        j=1;

/* first set up (nparams+1)th vertex of simplex with initial parameters */
        u0 = ap_u0;
        if (inv_u0 == 1) {
                u0 = monte(ap_u0,apsig_u0,searchnsig,nmonte);
                params[nparams+1][j] = u0;
                inv_flags[U0] = j++;
        }
        v0 = ap_v0;
        if (inv_v0 == 1) {
                v0 = monte(ap_v0,apsig_v0,searchnsig,nmonte);
                params[nparams+1][j] = v0;
                inv_flags[V0] = j++;
        }
        w0 = ap_w0;
        if (inv_w0 == 1) {
                w0 = monte(ap_v0,apsig_v0,searchnsig,nmonte);
                params[nparams+1][j] = w0;
```

```
                inv_flags[W0] = j++;
        }
        rigrot = ap_rigrot;
        if (inv_rigrot == 1) {
                rigrot = monte(ap_rigrot,apsig_rigrot,searchnsig,nmonte);
                params[nparams+1][j] = rigrot;
                inv_flags[RIGROT] = j++;
        }
        scale = ap_scale;
        if (inv_scale == 1) {
                scale = monte(ap_scale,apsig_scale,searchnsig,nmonte);
                params[nparams+1][j] = scale;
                inv_flags[SCALE] = j++;
        }

        for(i=1; i<=nfaults; i++) {
                ii = (i-1)*NFLAGS+NOFFS;
                strike[i] = ap_strike[i];
                if (inv_strike[i] == 1) {
                        params[nparams+1][j] = monte(ap_strike[i], apsig_strike[i],
searchnsig,nmonte);
                        inv_flags[ii+STRIKE] = j++;
                }
                dip[i] = ap_dip[i];
                if (inv_dip[i] == 1) {
                        params[nparams+1][j] = monte(ap_dip[i], apsig_dip[i],
searchnsig,nmonte);
                        inv_flags[ii+DIP] = j++;
                }
                rake[i] = ap_rake[i];
                if (inv_rake[i] == 1) {
                        params[nparams+1][j] = monte(ap_rake[i], apsig_rake[i],
searchnsig,nmonte);
                        inv_flags[ii+RAKE] = j++;
                }
                slp[i] = ap_slp[i];
                if (inv_slp[i] == 1) {
                        params[nparams+1][j] = monte(ap_slp[i], apsig_slp[i],
searchnsig,nmonte);
                        inv_flags[ii+M0] = j++;
                }
                fltx[i] = ap_fltx[i];
                if (inv_fltx[i] == 1) {
                        params[nparams+1][j] = monte(ap_fltx[i], apsigkm_fltx[i],
searchnsig,nmonte);
                        inv_flags[ii+FLTX] = j++;
                }
                flty[i] = ap_flty[i];
                if (inv_flty[i] == 1) {
                        params[nparams+1][j] = monte(ap_flty[i], apsigkm_flty[i],
searchnsig,nmonte);
                        inv_flags[ii+FLTY] = j++;
                }
                fltlen[i] = ap_fltlen[i];
                if (inv_fltlen[i] == 1) {
                        params[nparams+1][j] = monte(ap_fltlen[i], apsig_fltlen[i],
searchnsig,nmonte);
                        inv_flags[ii+FLTLEN] = j++;
                }
                hmin[i] = ap_hmin[i];
                if (inv_hmin[i] == 1) {
                        params[nparams+1][j] = monte(ap_hmin[i], apsig_hmin[i],
searchnsig,nmonte);
                        inv_flags[ii+HMIN] = j++;
                }
                hmax[i] = ap_hmax[i];
                if (inv_hmax[i] == 1) {
                        params[nparams+1][j] = monte(ap_hmax[i], apsig_hmax[i],
searchnsig,nmonte);
                        inv_flags[ii+HMAX] = j++;
                }
        }


/* copy parameters over to other vertices of simplex */
        for(i=1; i<=nparams; i++) {
                for(j=1; j<=nparams; j++) {
                        params[i][j] = params[nparams+1][j];
                }
        }


/* now perturb parameters by fraction +/-pertfac of a priori sigma */
        j=1;
        if (inv_u0 == 1) params[j][j++] += apsig_u0*pertfac * pmone();
        if (inv_v0 == 1) params[j][j++] += apsig_v0*pertfac * pmone();
        if (inv_w0 == 1) params[j][j++] += apsig_w0*pertfac * pmone();
        if (inv_rigrot == 1) params[j][j++] += apsig_rigrot*pertfac * pmone();
        if (inv_scale == 1) params[j][j++] += apsig_scale*pertfac * pmone();
        for(i=1; i<=nfaults; i++) {
                if (inv_strike[i] == 1) params[j][j++] += apsig_strike[i]*pertfac *
pmone();
                if (inv_dip[i] == 1) params[j][j++] += apsig_dip[i]*pertfac *
pmone();
                if (inv_rake[i] == 1) params[j][j++] += apsig_rake[i]*pertfac *
pmone();
                if (inv_slp[i] == 1) params[j][j++] += apsig_slp[i]*pertfac *
pmone();
                if (inv_fltx[i] == 1) params[j][j++] += apsigkm_fltx[i]*pertfac *
pmone();
                if (inv_flty[i] == 1) params[j][j++] += apsigkm_flty[i]*pertfac *
pmone();
                if (inv_fltlen[i] == 1) params[j][j++] += apsig_fltlen[i]*pertfac;
                if (inv_hmin[i] == 1) params[j][j++] += apsig_hmin[i]*pertfac *
pmone();
                if (inv_hmax[i] == 1) params[j][j++] += apsig_hmax[i]*pertfac *
pmone();
        }
}


int pmone()

{
        if (drand48() < 0.5) return -1;
        else return 1;
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* function to compute amount of dither when setting up Monte-Carlo start
  points ie return a value randomly selected from the interval (x-2*sig,x+2*sig)
  with uniform probability.

   pjc jun '95
```

```
*/

#include <stdio.h>
#include <math.h>
#include "nrutil.h"

#include "global.c"


double monte(x,sig,searchnsig,nmonte)

double x,sig,searchnsig;
int nmonte;


{
        double value;

        if (nmonte > 0) value = x-searchnsig*sig + 2.0*searchnsig*sig*drand48();
        else value = x;

        return value;
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* mprove.c
  Taken directly from NR in C, section 2.5
/*
*******************************************************************************


*******************************************************************************
/* nr.h
  Taken directly from NR in C, appendix A
/*
*******************************************************************************


*******************************************************************************
/* nrutil.c
  Taken directly from NR in C, appendix B
/*
*******************************************************************************


*******************************************************************************
/* nrutil.h
  Taken directly from NR in C, appendix B
/*
*******************************************************************************


*******************************************************************************
/* function to compute general penalty function.  routines pen_L1dis, pen_L2dis
  and pen_Linfdis are used to compute the actual value, but first, parameter
  values must be unwrapped to sensible numbers and bounds checking is performed.
  If a parameter hits a boundary, BIGNUM is returned and the inversion is
  abandoned.

   pjc jun '95
*/

#include <stdio.h>
#include <math.h>
#include "nrutil.h"


#include "global.c"


double penfunc(param_vals)

double *param_vals;

{
        int i,ii,j;
        int iret;
        double pen_val;
        double rrake,u,w;

/* extract the parameters from the array param_vals, and check bounds */
        if (inv_flags[U0] > 0) u0 = param_vals[*(inv_flags+U0)];
        if (inv_flags[V0] > 0) v0 = param_vals[*(inv_flags+V0)];
        if (inv_flags[W0] > 0) w0 = param_vals[*(inv_flags+W0)];
        if (inv_flags[RIGROT] > 0) rigrot = param_vals[*(inv_flags+RIGROT)];
        if (inv_flags[SCALE] > 0) scale = param_vals[*(inv_flags+SCALE)];
        for(i=1; i<=nfaults; i++) {
                ii = (i-1)*NFLAGS+NOFFS;
                if (inv_flags[ii+STRIKE] > 0) strike[i] = param_vals[*(inv_flags+ii+
STRIKE)];
                if (inv_flags[ii+DIP] > 0) {
                        dip[i] = param_vals[*(inv_flags+ii+DIP)];
                        if (dip[i] < 0.0) {
/*                              printf("*** dip[%1d] hits zero\n",i); */
                                return BIGNUM;
                                param_vals[*(inv_flags+ii+DIP)] = -dip[i] * REFFAC;
                                dip[i] = -dip[i] * REFFAC;
                        }
                        if (dip[i] > 180.0) {
/*                              printf("*** dip[%1d] hits 180\n",i); */
                                return BIGNUM;
                                param_vals[*(inv_flags+ii+DIP)] = (dip[i]-180.0) *
REFFAC;
                                dip[i] = (dip[i]-180.0) * REFFAC;
                        }
                }
                if (inv_flags[ii+RAKE] > 0)
                        rake[i] = param_vals[*(inv_flags+ii+RAKE)];
                if (inv_flags[ii+M0] > 0) {
                        slp[i] = param_vals[*(inv_flags+ii+M0)];
                        if (slp[i] < 0.0) {
/*                              printf("*** slp[%1d] hits zero\n",i); */
                                return BIGNUM;
                                param_vals[*(inv_flags+ii+M0)] = -slp[i] * REFFAC;
                                slp[i] = -slp[i] * REFFAC;
                        }
                }
                if (inv_flags[ii+FLTX] > 0)
                        fltx[i] = param_vals[*(inv_flags+ii+FLTX)];
                if (inv_flags[ii+FLTY] > 0)
                        flty[i] = param_vals[*(inv_flags+ii+FLTY)];
                if (inv_flags[ii+FLTLEN] > 0) {
                        fltlen[i] = param_vals[*(inv_flags+ii+FLTLEN)];
                        if (fltlen[i] < 0.0) {
/*                              printf("*** fltlen[%1d] hits zero\n",i); */
                                return BIGNUM;
                                param_vals[*(inv_flags+ii+FLTLEN)] = -fltlen[i] *
REFFAC;
                                fltlen[i] = -fltlen[i] * REFFAC;
                        }
                }
                if (inv_flags[ii+HMIN] > 0) {
```

```
                        hmin[i] = param_vals[*(inv_flags+ii+HMIN)];
                        if (hmin[i] < 0.0) {
/*                              printf("*** hmin[%1d] hits zero\n",i); */
                                return BIGNUM;
                                param_vals[*(inv_flags+ii+HMIN)] = -hmin[i] * REFFAC;
                                hmin[i] = -hmin[i] * REFFAC;
                        }
                }
                if (inv_flags[ii+HMAX] > 0) {
                        hmax[i] = param_vals[*(inv_flags+ii+HMAX)];
                        if (hmax[i] < 0.0) {
/*                              printf("*** hmax[%1d] hits zero\n",i); */
                                return BIGNUM;
                                param_vals[*(inv_flags+ii+HMAX)] = -hmax[i] * REFFAC;
                                hmax[i] = -hmax[i] * REFFAC;
                        }
                }
                if (hmax[i] < hmin[i]) {
/*                      printf("*** hmax[%1d] hits hmin[%1d]\n",i,i); */
                        return BIGNUM;
                        param_vals[*(inv_flags+ii+HMAX)] = hmin[i] * (1.0+REFFAC);
                        hmax[i] = hmin[i] * (1.0+REFFAC);
                }

        /* compute derived parameters for each fault */
                rrake = (rake[i]+90.0) * DEG2RAD;
                w = (hmax[i]-hmin[i])/sin(dip[i]*DEG2RAD);
                ud[i] =  100.0*slp[i]*cos(rrake);
                us[i] = -100.0*slp[i]*sin(rrake);
                al1[i] = -fltlen[i]/2.0;
                al2[i] = fltlen[i]/2.0;
                aw1[i] = hmin[i]/sin(dip[i]*DEG2RAD);
                aw2[i] = hmax[i]/sin(dip[i]*DEG2RAD);
        }

/* compute site displacements, return BIGNUM if error */
        iret = comp_disps();

        if (iret == 1) return BIGNUM;

/* compute penalty function */
        if (norm == 0) pen_val = pen_Linfdis();
        else if (norm == 1) pen_val = pen_L1dis();
        else if (norm == 2) pen_val = pen_L2dis();
        else return BIGNUM;

        return(pen_val);
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* function to compute penalty function (weighted L1 norm of displacement
  residuals).
  use_dz governs the observation types: -1=dz only; 0=dx,dy only; 1=dx,dy,dz
  covtype governs the weighting: 1=unit; other=use variances

   pjc jun '95
*/

#include <stdio.h>
#include <math.h>
#include "nrutil.h"

#include "global.c"


double pen_L1dis()


{
        register int i,j;
        double pen_val=0.0;
        double xtmp,ytmp,ztmp;

        if (covtype == 1) {
                if (use_dz != -1) for(i=1; i<=nsites; i++) {
                        pen_val += fabs(xdisp[i]-comp_xdisp[i]);
                        pen_val += fabs(ydisp[i]-comp_ydisp[i]);
                }
                if (use_dz != 0) for(i=1; i<=nsites; i++) {
                        pen_val += fabs(zdisp[i]-comp_zdisp[i]);
                }
        }
        else {
                if (use_dz != -1) for(i=1; i<=nsites; i++) {
                        pen_val += fabs(xdisp[i]-comp_xdisp[i]) * sqrt(wgt[i][i]);
                        pen_val += fabs(ydisp[i]-comp_ydisp[i]) *
sqrt(wgt[nsites+i][nsites+i]);
                }
                if (use_dz != 0) for(i=1; i<=nsites; i++) {
                        pen_val += fabs(zdisp[i]-comp_zdisp[i]) *
sqrt(wgt[2*nsites+i][2*nsites+i]);
                }
        }

        return pen_val;
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* function to compute penalty function (weighted L2 norm of displacement
  residuals).
  use_dz governs the observation types: -1=dz only; 0=dx,dy only; 1=dx,dy,dz
  covtype governs the weighting: 1=unit; 2=use variances; other=use full vcm

   pjc jun '95
*/

#include <stdio.h>
#include <math.h>
#include "nrutil.h"

#include "global.c"


double pen_L2dis()


{
        register int i,j;
        double pen_val=0.0;
        double xtmp,ytmp,ztmp;
        double xres,yres,zres;

        if (covtype == 1) {
                if (use_dz != -1) for(i=1; i<=nsites; i++) {
                        pen_val += (xdisp[i]-comp_xdisp[i]) *
(xdisp[i]-comp_xdisp[i]);
                        pen_val += (ydisp[i]-comp_ydisp[i]) *
```

```
(ydisp[i]-comp_ydisp[i]);
                }
                if (use_dz != 0) for(i=1; i<=nsites; i++) {
                        pen_val += (zdisp[i]-comp_zdisp[i]) *
(zdisp[i]-comp_zdisp[i]);
                }
        }
        else if (covtype == 2) {
                if (use_dz != -1) for(i=1; i<=nsites; i++) {
                        pen_val += (xdisp[i]-comp_xdisp[i]) *
(xdisp[i]-comp_xdisp[i]) * wgt[i][i];
                        pen_val += (ydisp[i]-comp_ydisp[i]) *
(ydisp[i]-comp_ydisp[i]) * wgt[nsites+i][nsites+i];
                }
                if (use_dz != 0) for(i=1; i<=nsites; i++) {
                        pen_val += (zdisp[i]-comp_zdisp[i]) *
(zdisp[i]-comp_zdisp[i]) * wgt[2*nsites+i][2*nsites+i];
                }
        }
        else {
                if (use_dz == 0) for(i=1; i<=nsites; i++) {
                        xtmp = 0.0; ytmp = 0.0;
                        for(j=1; j<=nsites; j++) {
                                xres = xdisp[j] - comp_xdisp[j];
                                yres = ydisp[j] - comp_ydisp[j];
                                xtmp += wgt[i][j] * xres + wgt[i][nsites+j] * yres;
                                ytmp += wgt[nsites+i][j] * xres +
wgt[nsites+i][nsites+j] * yres;
                        }
                        pen_val += (xdisp[i]-comp_xdisp[i]) * xtmp;
                        pen_val += (ydisp[i]-comp_ydisp[i]) * ytmp;
                }
                else if (use_dz == 1) {
                        for(i=1; i<nsites; i++) {
                                xtmp = 0.0; ytmp = 0.0; ztmp = 0.0;
                                for(j=1; j<=nsites; j++) {
                                        zres = zdisp[j] - comp_zdisp[j];
                                        xtmp += wgt[i][2*nsites+j] * zres;
                                        ytmp += wgt[nsites+i][2*nsites+j] * zres;
                                        ztmp += wgt[2*nsites+i][j] *
(xdisp[j]-comp_xdisp[j]) + wgt[2*nsites+i][nsites+j] *
(ydisp[j]-comp_ydisp[j]) + wgt[2*nsites+i][2*nsites+j] * zres;
                                }
                                pen_val += (xdisp[i]-comp_xdisp[i]) * xtmp;
                                pen_val += (ydisp[i]-comp_ydisp[i]) * ytmp;
                                pen_val += (zdisp[i]-comp_zdisp[i]) * ztmp;
                        }
                }
                else if (use_dz == -1) {
                        for(i=1; i<nsites; i++) {
                                ztmp = 0.0;
                                for(j=1; j<=nsites; j++) {
                                        zres = zdisp[j] - comp_zdisp[j];
                                        ztmp += wgt[2*nsites+i][2*nsites+j] * zres;
                                }
                                pen_val += (zdisp[i]-comp_zdisp[i]) * ztmp;
                        }
                }
        }
        return pen_val;
}

#include "globend.c"
*******************************************************************************
```

```
*******************************************************************************
/* function to compute penalty function (weighted L-infinity norm of
  displacement residuals).
  use_dz governs the observation types: -1=dz only; 0=dx,dy only; 1=dx,dy,dz
  covtype governs the weighting: 1=unit; other=use variances

   pjc jun '95
*/

#include <stdio.h>
#include <math.h>
#include "nrutil.h"

#include "global.c"


double pen_Linfdis()


{
        register int i,j;
        double pen_val=0.0;
        double tmp;

        if (covtype == 1) {
                if (use_dz != -1) for(i=1; i<=nsites; i++) {
                        tmp = fabs(xdisp[i]-comp_xdisp[i]);
                        if (tmp > pen_val) pen_val = tmp;
                        tmp = fabs(ydisp[i]-comp_ydisp[i]);
                        if (tmp > pen_val) pen_val = tmp;
                }
                if (use_dz != 0) for(i=1; i<=nsites; i++) {
                        tmp = fabs(zdisp[i]-comp_zdisp[i]);
                        if (tmp > pen_val) pen_val = tmp;
                }
        }
        else {
                if (use_dz != -1) for(i=1; i<=nsites; i++) {
                        tmp = fabs(xdisp[i]-comp_xdisp[i]) * sqrt(wgt[i][i]);
                        if (tmp > pen_val) pen_val = tmp;
                        tmp = fabs(ydisp[i]-comp_ydisp[i]) *
sqrt(wgt[nsites+i][nsites+i]);
                        if (tmp > pen_val) pen_val = tmp;
                }
                if (use_dz != 0) for(i=1; i<=nsites; i++) {
                        tmp = fabs(zdisp[i]-comp_zdisp[i]) *
sqrt(wgt[2*nsites+i][2*nsites+i]);
                        if (tmp > pen_val) pen_val = tmp;
                }
        }

        return pen_val;
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to get input parameters for okinv from file "okinv.inp" (v7).

   pjc jun '95
*/

#include <stdio.h>
```

```
#include <math.h>
#include "nrutil.h"

#include "global.c"


void readinps(inpname, psvmname,covarname, outname, residname,dispsta,dispgrid,
scratchname,scseedname,scvarname, varnstep,varnsig, coortype,deltaz_out, norm,
pertfac,searchnsig, minx,miny, xstep,ystep, nxstep,nystep, nmonte, nparams)

char *inpname,*outname;
char *psvmname,*covarname,*residname;
char *dispsta,*dispgrid;
char *scratchname,*scseedname,*scvarname;
int *varnstep;
double *varnsig;
int *coortype,*deltaz_out,norm;
double *pertfac,*searchnsig;
double *minx,*miny,*xstep,*ystep;
int *nxstep,*nystep;
int *nmonte;
int *nparams;


{
        FILE *inpfile;
        char dummystring[255];
        int i,j,invtmp;
        double tmp,sigtmp;

        inpfile = fopen(inpname,"r");
        if (inpfile == NULL) {
                printf("*** error opening input file '%s'\n",inpname);
                exit();
        }

        printf("[Expecting v7 format]\n");

        for(i=1;i<=5;i++) fgets(dummystring,255,inpfile);

/* get file names and controlling parameters */
        fscanf(inpfile,"%s",psvmname);
        fgets(dummystring,255,inpfile);
        fscanf(inpfile,"%s",covarname);
        fgets(dummystring,255,inpfile);
        fscanf(inpfile,"%s",outname);
        fgets(dummystring,255,inpfile);
        fscanf(inpfile,"%s",residname);
        fgets(dummystring,255,inpfile);
        fscanf(inpfile,"%s",dispsta);
        fgets(dummystring,255,inpfile);
        fscanf(inpfile,"%s",dispgrid);
        fgets(dummystring,255,inpfile);
        fscanf(inpfile,"%s",scratchname);
        fgets(dummystring,255,inpfile);
        fscanf(inpfile,"%s",scseedname);
        fgets(dummystring,255,inpfile);
        fscanf(inpfile,"%s",scvarname);
        fgets(dummystring,255,inpfile);
        fscanf(inpfile,"%d",varnstep);
        fgets(dummystring,255,inpfile);
        fscanf(inpfile,"%lf",varnsig);
        fgets(dummystring,255,inpfile);
        fscanf(inpfile,"%d",coortype);
        fgets(dummystring,255,inpfile);
        fscanf(inpfile,"%d",&deltaz_in);
        fgets(dummystring,255,inpfile);
        fscanf(inpfile,"%d",&use_dz);
        use_dz *= deltaz_in;
        fgets(dummystring,255,inpfile);
        fscanf(inpfile,"%d",deltaz_out);
        fgets(dummystring,255,inpfile);
        fscanf(inpfile,"%d",&nfaults);
        fgets(dummystring,255,inpfile);
        fscanf(inpfile,"%d",norm);
        fgets(dummystring,255,inpfile);
        fscanf(inpfile,"%lf",pertfac);
        fgets(dummystring,255,inpfile);
        fscanf(inpfile,"%lf",searchnsig);
        fgets(dummystring,255,inpfile);
        fscanf(inpfile,"%lf",minx);
        fgets(dummystring,255,inpfile);
        fscanf(inpfile,"%lf",miny);
        fgets(dummystring,255,inpfile);
        fscanf(inpfile,"%lf",xstep);
        fgets(dummystring,255,inpfile);
        fscanf(inpfile,"%lf",ystep);
        fgets(dummystring,255,inpfile);
        fscanf(inpfile,"%d",nxstep);
        fgets(dummystring,255,inpfile);
        fscanf(inpfile,"%d",nystep);
        fgets(dummystring,255,inpfile);
        fscanf(inpfile,"%d",nmonte);
        fgets(dummystring,255,inpfile);


/* load up a priori translation/rotation/scale parameters */
        for(j=1; j<=3; j++) fgets(dummystring,255,inpfile);
        fscanf(inpfile,"%lf %lf %d",&tmp,&sigtmp,&invtmp);
        fgets(dummystring,255,inpfile);
        ap_u0 = tmp; apsig_u0 = sigtmp; inv_u0 = invtmp;
        if (use_dz == -1) inv_u0 = 0;
        fscanf(inpfile,"%lf %lf %d",&tmp,&sigtmp,&invtmp);
        fgets(dummystring,255,inpfile);
        ap_v0 = tmp; apsig_v0 = sigtmp; inv_v0 = invtmp;
        if (use_dz == -1) inv_v0 = 0;
        fscanf(inpfile,"%lf %lf %d",&tmp,&sigtmp,&invtmp);
        fgets(dummystring,255,inpfile);
        ap_w0 = tmp; apsig_w0 = sigtmp; inv_w0 = invtmp;
        inv_w0 *= abs(use_dz);
        fscanf(inpfile,"%lf %lf %d",&tmp,&sigtmp,&invtmp);
        fgets(dummystring,255,inpfile);
        ap_rigrot = tmp; apsig_rigrot = sigtmp; inv_rigrot = invtmp;
        if (use_dz == -1) inv_rigrot = 0;
        fscanf(inpfile,"%lf %lf %d",&tmp,&sigtmp,&invtmp);
        fgets(dummystring,255,inpfile);
        ap_scale = tmp; apsig_scale = sigtmp; inv_scale = invtmp;
        if (use_dz == -1) inv_scale = 0;

        *nparams = inv_u0 + inv_v0 + inv_w0 + inv_rigrot + inv_scale;


/* for each fault load in the a priori parameters etc */
        for (i=1; i<= nfaults; i++) {
                for(j=1; j<=3; j++) fgets(dummystring,255,inpfile);
                fscanf(inpfile,"%lf %lf %d",&tmp,&sigtmp,&invtmp);
                fgets(dummystring,255,inpfile);
                ap_strike[i] = tmp; apsig_strike[i] = sigtmp; inv_strike[i] = invtmp;
                fscanf(inpfile,"%lf %lf %d",&tmp,&sigtmp,&invtmp);
                fgets(dummystring,255,inpfile);
```

```
                ap_dip[i] = tmp; apsig_dip[i] = sigtmp; inv_dip[i] = invtmp;
                fscanf(inpfile,"%lf %lf %d",&tmp,&sigtmp,&invtmp);
                fgets(dummystring,255,inpfile);
                ap_rake[i] = tmp; apsig_rake[i] = sigtmp; inv_rake[i] = invtmp;
                fscanf(inpfile,"%lf %lf %d",&tmp,&sigtmp,&invtmp);
                fgets(dummystring,255,inpfile);
                ap_slp[i] = tmp; apsig_slp[i] = sigtmp; inv_slp[i] = invtmp;
                fscanf(inpfile,"%lf %lf %d",&tmp,&sigtmp,&invtmp);
                fgets(dummystring,255,inpfile);
                ap_fltx[i] = tmp; apsig_fltx[i] = sigtmp; inv_fltx[i] = invtmp;
                fscanf(inpfile,"%lf %lf %d",&tmp,&sigtmp,&invtmp);
                fgets(dummystring,255,inpfile);
                ap_flty[i] = tmp; apsig_flty[i] = sigtmp; inv_flty[i] = invtmp;
                fscanf(inpfile,"%lf %lf %d",&tmp,&sigtmp,&invtmp);
                fgets(dummystring,255,inpfile);
                ap_fltlen[i] = tmp; apsig_fltlen[i] = sigtmp; inv_fltlen[i] = invtmp;
                fscanf(inpfile,"%lf %lf %d",&tmp,&sigtmp,&invtmp);
                fgets(dummystring,255,inpfile);
                ap_hmin[i] = tmp; apsig_hmin[i] = sigtmp; inv_hmin[i] = invtmp;
                fscanf(inpfile,"%lf %lf %d",&tmp,&sigtmp,&invtmp);
                fgets(dummystring,255,inpfile);
                ap_hmax[i] = tmp; apsig_hmax[i] = sigtmp; inv_hmax[i] = invtmp;
                *nparams += inv_strike[i] + inv_dip[i] + inv_rake[i] + inv_slp[i] +
inv_fltx[i] + inv_flty[i] + inv_fltlen[i] + inv_hmin[i] + inv_hmax[i];
        }

        fclose(inpfile);
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to vary one inverted coordinate parameter about the 'best' value
   and compute penalty function, which is output to file).
   This routine differs from var1par.c in that conversion from lat/lon to km
   may need to be done.

   pjc may '96
*/

#include <stdio.h>
#include <math.h>
#include "nrutil.h"

#include "global.c"


void var1crdpar(param_vals,parnum, nfault,xy,coortype, sig,varnsig, varnstep,
outfile)

double *param_vals, sig, varnsig;
int parnum,nfault,xy,coortype,varnstep;
FILE *outfile;

{
        double vp,vp0;
        double r;


        vp0 = param_vals[parnum];
        r = varnsig*sig;

        for(vp=vp0-r; vp<=vp0+r; vp+=2.0*r/((double)varnstep)) {
                param_vals[parnum] = vp;
                dumx[1] = fltx[nfault]; dumy[1] = flty[nfault];
                if (xy == 1) dumx[1] = vp;
                else dumy[1] = vp;
                xy2ell(dumx,dumy, dumx,dumy, 1, coortype);
                if (xy == 1) fprintf(outfile, "%9.4lf   %11.4le\n", dumx[1],
penfunc(param_vals));
                else fprintf(outfile, "%9.4lf   %11.4le\n", dumy[1],
penfunc(param_vals));
        }

        param_vals[parnum] = vp0;
        fprintf(outfile, "\n");
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to vary one inverted parameter (not a coordinate parameter) and
   compute penalty function, which is output to file).

   pjc may '96
*/

#include <stdio.h>
#include <math.h>
#include "nrutil.h"

#include "global.c"


void var1par(param_vals,parnum, sig,varnsig, varnstep, outfile)

double *param_vals, sig, varnsig;
int parnum,varnstep;
FILE *outfile;

{
        double vp,vp0;
        double r;


        vp0 = param_vals[parnum];
        r = varnsig*sig;

        for(vp=vp0-r; vp<=vp0+r; vp+=2.0*r/((double)varnstep)) {
                param_vals[parnum] = vp;
                fprintf(outfile, "%9.4lf   %11.4le\n", vp,penfunc(param_vals));
        }

        param_vals[parnum] = vp0;
        fprintf(outfile, "\n");
}


#include "globend.c"
*******************************************************************************


*******************************************************************************
/* subroutine to convert coords in the arrays x[],y[] into coords in the global
  arrays xx[],yy[].  If coortype==1, conversion from km to lat/lon must be done,
  otherwise the coords are just copied over, plus the offset for the centroid.
  The centroid of the network is in the global variables orig_x,orig_y and all
  input coords are relative to this.
```

```
   pjc jun '95
*/

#include <stdio.h>
#include <math.h>
#include "nrutil.h"

#include "global.c"


void xy2ell(x,y,xx,yy,ncoords,coortype)

double *x,*y,*xx,*yy;
int ncoords,coortype;

{
        int i;
        double rprimev,rmerid,tmp;

        if (coortype==1) {
                tmp = 1 - pow((ECCEN*sin(orig_y*DEG2RAD)),2.0);
                rprimev = RADIUS / sqrt(tmp);
                rmerid = RADIUS*(1 - ECCEN*ECCEN) / (tmp*sqrt(tmp));
        }

        for(i=1; i<=ncoords; i++) {
                xx[i] = x[i];
                yy[i] = y[i];
                if (coortype==1) {
                        xx[i] /= (rprimev/1000.0)*cos(orig_y*DEG2RAD) * DEG2RAD;
                        yy[i] /= (rmerid/1000.0) * DEG2RAD;
                }
                xx[i] += orig_x;
                yy[i] += orig_y;
        }
}


#include "globend.c"
```

```
*******************************************************************************
Input option file for okinv elastic dislocation inversion program             v7
-------------------------------------------------------------------------------

Option                                  Description
**********                              -----------
test.psvm                               input displacement (.psvm) filename
PSVM                                    covariance (.CEN) file / "UNIT" / "PSVM"
NONE                                    parameter output filename / "NONE"
test.res                                residual output filename / "NONE"
NONE                                    station displacement outfile / "NONE"
NONE                                    displacement grid output filename/"NONE"
NONE                                    scratch file (for M-C/bootstrap output)
NONE                                    scratch file (for M-C seed points)
test.var                                scratch file (for single-param var)
100                                     no of steps for single-param var
2.0                                     search width (sigmas) for 1-param var
1                                       coordinate type (0=km, 1=lat/lon)
0                                       input dU too? (1=yes, 0=no)
0                                       use dU? (1=yes, 0=no, -1=dU only)
0                                       output dU too? (1=yes, 0=no)
1                                       no. of faults
2                                       norm of penalty (0=Linf, 1=L1, 2=L2)
0.1                                     perturbation factor for simplex init
2.0                                     search width (sigmas) for M-C seeds
21.2                                    lon(X) grid left coord
39.6                                    lat(Y) grid bottom coord
0.1                                     lon(X) grid step size (units)
0.1                                     lat(Y) grid step size (units)
9                                       lon(X) no. of grid steps
10                                      lat(Y) no. of grid steps
0                                       No. of Monte-Carlo restarts

Value          Sigma          Invert (1/0)    General parameters
****.****      ***.****       *               --------------------------------
   0.0000        0.1000       0               translation in lon(x) (m)
   0.0000        0.1000       0               translation in lat(y) (m)
   0.0000        0.1000       0               translation in hgt(z) (m)
   0.0000        1.0000       0               rigid-body rotation (microrad)
   0.0000        1.0000       0               scale factor (ppm)

Start value    Sigma          Invert (1/0)    Parameters of fault 1
****.****      ***.****       *               --------------------------------
 240.00         10.00         1               Strike (degrees)
  31.00         10.00         0               Dip (degrees)
 -98.00         10.00         0               Rake (degrees)
   0.606         0.1000       1               Slip (m)
  21.68          0.10         0               lon(x) } of projected centre of
  40.08          0.10         0               lat(y) } surface break
  20.00         10.00         0               length of scarp
   0.00          5.00         0               minimum depth of dislocation
  10.00          5.00         0               maximum depth of dislocation

Start value    Sigma          Invert (1/0)    Parameters of fault 2
****.****      ***.****       *               --------------------------------
 240.00         20.00         0               Strike (degrees)
  30.00         20.00         0               Dip (degrees)
 -90.00         10.00         0               Rake (degrees)
   7.60         10.00         0               Moment (x10^18 N m)
  21.61          0.10         0               lon(x) } of projected centre of
  39.97          0.10         0               lat(y) } surface break
  20.00         10.00         0               length of scarp
   0.00          5.00         0               minimum depth of dislocation
  10.00          5.00         0               maximum depth of dislocation

*******************************************************************************
```