COMBINATION OF ALTIMETRY DATA FROM DIFFERENT SATELLITE MISSIONS

HENNO JOLAND BOOMKAMP

Doctor of Philosophy

ASTON UNIVERSITY

September 1998

Aston University

COMBINATION OF ALTIMETRY DATA FROM DIFFERENT SATELLITE MISSIONS

Henno Joland Boomkamp

Doctor of Philosophy

1998

Substantial altimetry datasets collected by different satellites have only become available during the past five years, but the near future will bring a variety of new altimetry missions, both parallel and consecutive in time. The characteristics of each produced dataset vary with the different orbital heights and inclinations of the spacecraft, as well as with the technical properties of the radar intrument. An integral analysis of datasets with different properties offers advantages both in terms of data quantity and data quality. This thesis is concerned with the development of the means for such integral analysis, in particular for dynamic solutions in which precise orbits for the satellites are computed simultaneously.

The first half of the thesis discusses the theory and numerical implementation of dynamic multi-satellite altimetry analysis. The most important aspect of this analysis is the application of dual satellite altimetry crossover points as a bi-directional tracking data type in simultaneous orbit solutions. The central problem is that the spatial and temporal distributions of the crossovers are in conflict with the time-organised nature of traditional solution methods. Their application to the adjustment of the orbits of both satellites involved in a dual crossover therefore requires several fundamental changes of the classical least-squares prediction / correction methods.

The second part of the thesis applies the developed numerical techniques to the problems of precise orbit computation and gravity field adjustment, using the altimetry datasets of ERS-1 and TOPEX/Poseidon. Although the two datasets can be considered less compatible than those of planned future satellite missions, the obtained results adequately illustrate the merits of a simultaneous solution technique.

In particular, the geographically correlated orbit error is partially observable from a dataset consisting of crossover differences between two sufficiently different altimetry datasets, while being unobservable from the analysis of altimetry data of both satellites individually. This error signal, which has a substantial gravity-induced component, can be employed advantageously in simultaneous solutions for the two satellites in which also the harmonic coefficients of the gravity field model are estimated.

Keywords : ERS-1, TOPEX/Poseidon, precise orbit determination, gravity field

Acknowledgements

Many individuals have contributed directly or indirectly to the project that has resulted in this thesis.

The gravity field work that is described in Chapter 8 has been continued by Dr. Phil Moore and Dr. Stuart Carnochan during the period in which I had already exchanged Birmingham for the warmer surroundings of Madrid. My gratitude goes to these two colleagues, for offering me the opportunity to include some interesting results that illustrate the application of dual crossover data for gravity field refinement.

In addition I would like to thank all other staff and students at Astons' Space Geodesy Group, for continuous discussions and remarks - many of which were useful. Apart from the two already mentioned, these people are Simon Ehlers, Chris Murphy, Rob Cullen, Matthew Reynolds, Russ Walmsley who works on PRARE research with Faust, and various others, who have been of help via advice and criticism or just by being tolerant about a sometimes disproportionate burden on computer resources.

Furthermore, I would like to thank

- The Scientific Committee of the European Commission
  for financial support under the Human Capital and Mobility Programme,
  and in relation to this :

  Prof. ir. K.F. Wakker (DUT/DEOS) and Dr. J.M. Dow (ESA/ESOC)
  for supporting the funding application to the European Commission

- Dr. J.A. Marshall (NASA/GSFC)
  for providing software and calibrated post-launch parameters for the TOPEX box-wing model, as well as an improved SLR centre of mass correction algorithm for this satellite

- Ir. R. Scharroo and colleagues from DUT/DEOS
  Dr. B.D. Tapley and his team from UT/CSR
  for publising high quality ERS-1 orbits that have been used as external reference

- Dr. L. Daniel (IGN Paris),
  for kindly restoring and providing DORIS data for early TOPEX/Poseidon cycles which were not available on-line

Their contributions have all been crucial in one way or another.

List of Contents

Tables and Figures

List of Acronyms

CSR	Centre for Space Research (UT)
DEOS	Delft Institute for Earth-Oriented Science
DORIS	Doppler Orbitography and Range Interferometry System
DUT	Delft University of Technology
DXO	Dual satellite crossover
ECF	Earth-Centered Fixed reference frame
ERS	European Remote sensing Satellite
ESA	European Space Agency
ESOC	European Space Operations Centre (Darmstadt)
FORTRAN	Formula translation
GDR	Geodetic Data Record
GFO	Geosat Follow-On mission
GMST	Greenwich Mean Solar Time
GMT	Generic Mapping Tool
GPS	Global Positioning System
GSFC	Goddard Space Flight Centre
IERS	International Earth Rotation Service
IAU	International Astronomical Union
IGN	Institut Geographique National
ITRF	International Terrestrial Reference Frame
J2000	Inertial reference frame of equator and equinox at 01/01/2000
JGM	Joint Gravity Model
LRA	Laser Retroreflector Assembly
MJD	Modified Julian Date
MSGM	Multi-Satellite Gravity Model
MSIS	Mass Spectrometer and Incoherent Scatter
NASA	National Aeronautics and Space Administration
NOAA	National Oceanographic and Atmospheric Agency
OSU	Ohio State University
PRARE	Precise Range and Range-rate Equipment
RA	Radar Altimetry
RAM	Random Access Memory
RAS	Royal Astronomical Society
RGO	Royal Greenwich Observatory
RMS	Root Mean Square
SATAN	Satellite Analysis
SLR	Satellite Laser Range data
SPOD	Skynet Orbit Determination Program
SRP	Solar Radiation Pressure
SXO	Single satellite crossover
TEC	Total Electron Content
TOD	True Of Date
TOPEX	Topographic Experiment
UT	University of Texas
UTC	Universal Time Coordinated
VLBI	Very Long Baseline Interferometry

Introduction Chapter 1

On the third of June 1769 the absolute size of the solar system was first determined, when the transit of Venus over the solar disk was timed both by captain Cook at Tahiti and by the Royal Observatory in Greenwich. The instrument used in this observation extended well beyond Cook's telescope, and included the entire constellation of Earth, Venus and Sun as a measurement device of opportunity, with a precision that was essentially determined by that with which the baseline between Tahiti and Greenwich was known. As a modern day equivalent, a space-borne radar instrument is only a minor element within the overall altimeter observation : the orbit of the satellite, used as a geometrical basis for the radar range, can be seen as an integral part of the adopted measurement device and must therefore be known with great precision.

The past decade has shown rapid progress in the field of precise orbit determination for radar altimetry missions. This has resulted in a situation where mission requirements have been comfortably met and exceeded, as shown by Dow *et al.* (1993) for ERS-1 and by Tapley *et al.* (1994) for TOPEX/Poseidon. In particular the improvement in the radial orbit accuracy of the ERS missions to a level compatible to that of TOPEX/Poseidon enables applications of radar altimetry that were hardly foreseen during the planning phases of the respective missions. As one such application, this thesis will describe how altimetry data from different satellite missions can be combined in dynamic solutions for the orbits of all involved satellites, conceiving a 'measurement device' formed by the constellation of two or more altimetry satellites in orbit around the Earth.

An outline of the short history of satellite radar altimetry is given in Figure 1.1. The first mission that provided the scientific community with a substantial radar altimetry dataset was ERS-1, and only since the launch of TOPEX/Poseidon in 1992 have altimeters been flown on different satellite platforms at the same time. Continuity of multi-mission altimetry data was provided by the launch of ERS-2 in 1995, which later succeeded ERS-1 as the primary platform after a nine month ERS tandem mission. As the Figure shows, the near future will provide an abundance of parallel or overlapping altimeter datasets, by means of the Geosat follow-on mission, the Jason satellites, the Envisat-1 mission and later ESA Polar Platforms.

In addition, laser altimeters will be flown on the Space Shuttle and on a dedicated GLASS mission. The arrival of such large quantities of altimetry data presents the scientific community both with the possibility and with the necessity to develope techniques for an integral analysis of datasets from different satellites, in order to explore the added value from unified datasets as opposed to the analysis of each mission separately.

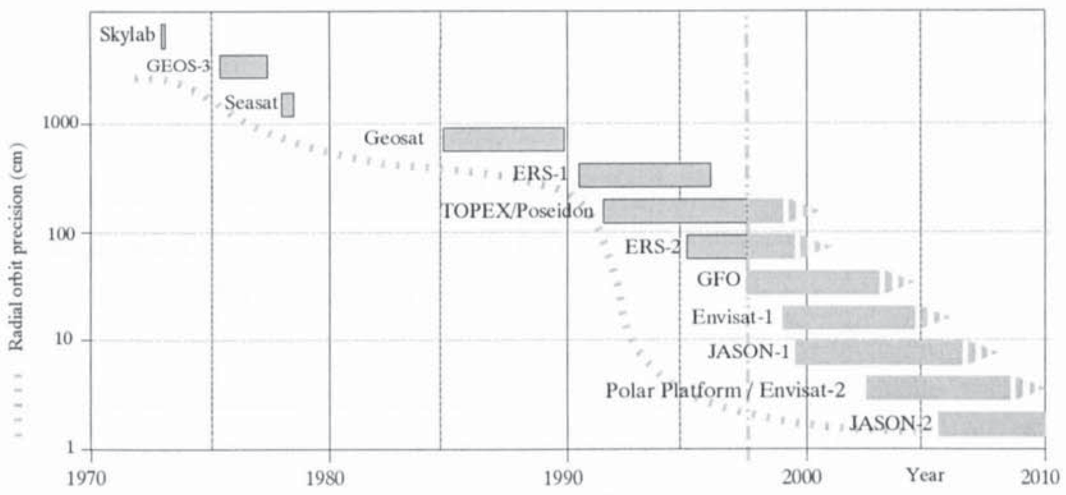


Figure 1.1 Survey of present and future satellite radar altimetry datasets. Starting with ERS-1 and TOPEX/Poseidon, parallel altimetry missions will be available until well into the next century, while compatibility between missions in terms of instrument precision and orbital accuracy is steadily increasing.

1.1 Simultaneous solutions

If the orbital parameters of two or more satellites are computed within a single mathematical process, the solution need not be any different from the outcome of single-satellite processes. Only if the orbital parameters of one satellite are mathematically correlated with those of another throughout the solution procedure, the parameter adjustment scheme can optimise the overall solution and create the integrated measurement device described above. In non-dynamical unified solutions, or in dynamic solutions without correlations between the various orbits, the only advantage in combining altimetry data from different satellite missions is an increase in data quantity, and - depending on the orbital characteristics of the involved satellites - a denser global data coverage. In such uncorrelated cases the process will be referred to in this thesis as a parallel solution.

If correlations between the various orbits are introduced, each satellite senses the presence of the other satellite arcs in the solution process. The orbit solution of each satellite will then not only be adjusted with respect to the terrestrial reference frame in which its own Earth-based tracking data is defined, but also - through correlations with orbits of other satellites - with respect to the tracking data of the other satellites. In that case the solutions will be called simultaneous. The crucial difference between parallel solutions and simultaneous solutions is the availability of a form of satellite-to-satellite tracking data that enters additional information into the system to correlate different satellite arcs. In the case of current altimetry missions this satellite-to-satellite tracking (or rather : arc-to-arc tracking) is provided by dense global networks of dual satellite crossover points. Future missions may exploit additional tracking methods, and in particular simultaneous tracking by the GPS network is likely to offer interesting contributions.

Crossover height differences are available above every point on the globe where the groundtrack of a satellite intersects with its earlier groundtrack (single satellite crossovers) or with the groundtrack of another satellite (dual satellite crossovers), as long as altimetry data is available for both crossings. In general this only excludes the points over land or ice. Geographically correlated errors in the altimetry observation model are identical for both crossings in the crossover, and are conveniently eliminated if the difference between the two heights is computed (e.g. Moore and Ehlers, 1993). This difference is therefore equal to the actual orbital height difference between the crossings, corrupted by the *differences* between all error components in the modelled observation that are not fully geographically correlated. Much of this geographically uncorrelated modelling error is directly related to the temporal variability of tides, ocean currents, interactions between the ocean surface and the atmosphere or the atmospheric conditions themselves. This causes the reliability of crossover height differences to decrease with increasing temporal separation between the two crossings. As a result, crossover datasets for orbit determination tend to be generated with an upper limit to the time interval between crossings of five days or less, in order to keep the described temporal variability within acceptable limits.

In simultaneous orbit solutions for two or more altimetry satellites it is essential that both epochs involved in a dual crossover form part of an orbit arc that is actually computed within that solution, so that the crossover error can be truly minimised by adjusting the orbital parameters of both arcs. In combination with the described restriction to the time between crossings, this condition imposes crucial demands upon the structure of the solution process.

The time interval for which the orbit of a satellite can be solved as one continuous arc is restricted by the stability of the numerical integration process that is used within the computation scheme, as will be discussed in later Chapters. For a given arc length, the dual crossover data density in a simultaneous solution would be significantly reduced in comparison to solutions in which one of the two crossings is allowed to be outside the interval of the orbit arc. An example of this effect, related to crossovers between the ERS-1 orbit and the orbit of TOPEX/Poseidon is shown in Figure 1.2. For arc lengths of five days and a maximum upper limit to the crossover interval of five days, the condition that both crossings must be within the limited solution interval reduces the crossover data density for simultaneous solutions to less than half that of a case with unlimited arc lengths. As such a reduction in data density would compromise simultaneous solutions based on dual crossovers in an unacceptable way, it inevitably imposes the requirement that true simultaneous solutions must also be dynamic *multi-arc* solutions, for all satellites between which crossovers are included in the process.

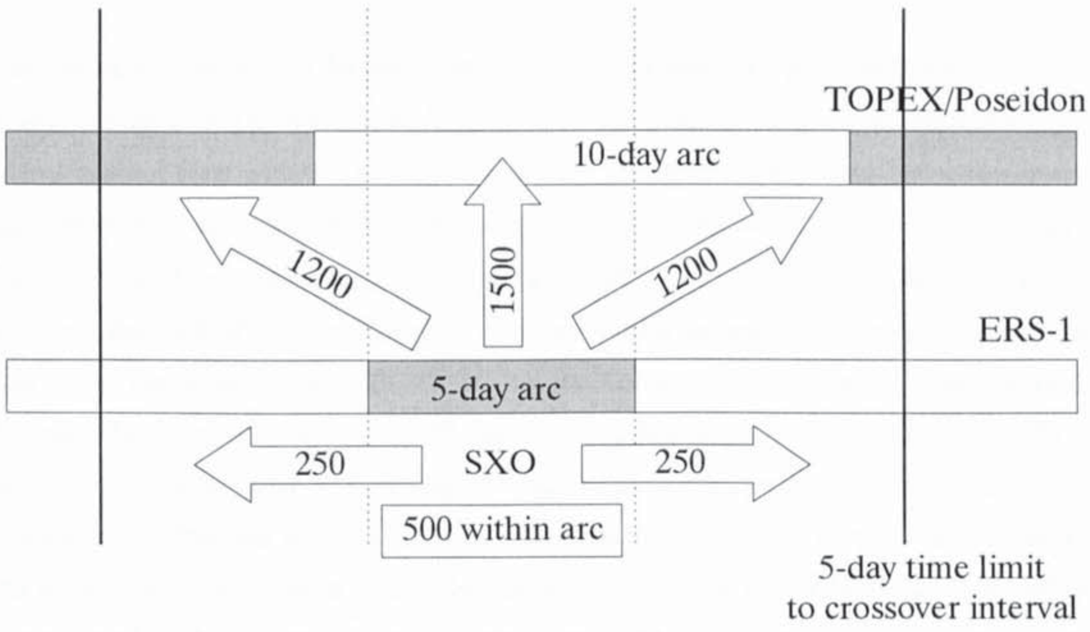


Figure 1.2 Typical quantities of single satellite crossovers for ERS-1 and dual crossovers with TOPEX/Poseidon, comparing a 5-day solution for ERS-1 with a full simultaneous multi-arc solution. The amount of single crossovers is doubled if those with the previous and next arc are included, while the amount of dual satellite crossovers in a simultaneous solution is no longer limited by arc lengths.

1.2 Formulation of study targets

Until recently the dual satellite crossovers between ERS-1 and TOPEX were only used - either in dynamic or non-dynamic solutions - for the improvement of the less accurate ERS-1 orbit, while keeping the TOPEX orbit fixed (e.g. Carnochan *et al.*, 1993; Le Traon *et al.*, 1993). As discussed above, this solution approach is neither simultaneous nor parallel, and implies that arc length restrictions do not affect the data density : the invariant TOPEX crossing can be allowed to occur outside the time interval covered by an ERS arc. However, the recent improvements in the orbit precision of ERS-1 make it unrealistic to subscribe the main residual error component in the dual crossover dataset to the radial orbit error of ERS-1 only, and future parallel altimetry missions will become even more compatible in terms of orbital precision and altimeter data quality. Consequently, the aforementioned unilateral application of dual crossovers will have to be gradually abandoned in favour of true simultaneous analysis of the orbits of altimeter satellites. This transition towards simultaneous dynamic altimetry analysis forms one of the main hurdles on the way to truly unified altimetry analysis, and constitutes the central subject of this study.

Most software systems used for orbit determination and altimetry processing were developed for analysis of a single satellite arc at a time or for other datatypes than altimetry. These systems can not cope with the specific demands of simultaneous solutions that either involve single satellite crossovers between consecutive orbit arcs, or dual satellite crossovers used to adjust both orbits simultaneously. A substantial part of the project covered by this thesis therefore consisted of the development of a new orbit determination programme - called Faust - for the dynamic analysis of crossovers between different arcs. Faust processes crossovers between orbit arcs of different satellites and between consecutive arcs for all relevant satellites, in order to guarantee the highest possible crossover data density at given practical limits (imposed by the lengths of individual arcs, or by the tolerated maximum time span between crossings). With Faust advantages and problems of simultaneous multi-satellite altimetry analysis have been analysed by applying the new solution technique to precise orbit determination and to gravity field adjustment. The latter was chosen as a classical application of satellite altimetry that could be expected to gain from simultaneous solutions for altimetry missions with different orbital characteristics. At present, this is only possible for analysis of the ERS and TOPEX/Poseidon missions.

1.3 Structure of this thesis

Because Faust is expected to be used by many others in the future, it was felt necessary to provide adequate documentation with the new programme. Such documentation serves the double purpose of introducing new users to the subject of precise orbit determination by providing sufficient theoretical background, and of explaining the essential structures of the software to enable future extensions or modifications. Initially, the documentation to Faust was started independent of this thesis, as the two documents serve different purposes. It soon became clear that this effort would result in continuous referencing from one to the other, as well as in substantial overlap. More disturbingly, it also resulted in undesired fragmentation and interruption of the natural line of thought. Because this was neither very efficient nor practical it was decided to fully integrate the documentation to Faust with the thesis. The advantage of this integral approach has been a clear systematic structure of the two-in-one thesis and documentation, although at several points this may have resulted in unusual extensions to each of the separate entities.

At first, the thesis contains several theory Sections to explain the mathematics behind the adopted solution processes. As some of these Sections are merely recapitulations of existing mathematical principles, they are primarily part of the documentation to Faust and would normally be left out of a thesis, or be referenced. Now that they are added to the main text, however, they form a useful introduction to the argument of the thesis. Second, the overall size of this thesis has become larger than commonly advised. This is also caused by the inclusion of various appendices that contain essential software engineering details of Faust, a discussion of which would fall outside the framework of the thesis itself. These Appendices can now directly refer to relevant Sections of the main thesis, which may help future users to gain quick access to parts of the code of their interest. Furthermore, the entire thesis has been organised in two main parts that are most concisely described as 'theory' and 'practice', although the separation between the two is not absolute but rather shows a gradual progression from 'mainly computational theory' to 'mainly applied research'. The first part covers the theoretical basis of simultaneous solutions, as well as some essential details of their numerical implementation for as far as innovative or crucial to the solution principle. The second part describes several applications of the simultaneous solution technique to the analysis of multi-mission altimetry, and discusses results in the area of precise orbit determination and geophysical applications of simultaneous solution techniques.

In particular, the next four Chapters each discuss one of the fundamental aspects of dynamic satellite tracking data analysis in Faust, in a systematic top-down approach. An overview of the embedded parameter estimation techniques, forming the top-level solution principle, is given in Chapter 2. The simulation of observation data forms a central aspect of the orbit determination method, and involves the numerical integration of various differential equations. Chapter 3 documents the incorporated numerical integration process in sufficient detail to explain the adopted enhancements for multi-satellite processing. Increasing the level of detail further, Chapter 4 documents the implemented force models that together form the basis for the numerical integration of the equations of motion of the satellites. In particular, the implementation in Faust of force model parameters and geophysical model parameters that are studied in later Chapters is documented. Finally, Chapter 5 describes aspets of tracking data modelling in Faust, as required for the parameter estimation scheme. This is obviously done with special emphasis on the numerical processing of crossover data and altimetry in simultaneous solutions.

From Chapter 6, the aspect of software documentation is no longer apparent in the thesis, unless the practical applications are interpreted as examples. Chapters 6 and 7 analyse the effects of simultaneous solutions on radial orbit precision, first for single satellite dynamic multi-arc solutions and then for various combinations of truly simultaneous multi-satellite / multi-arc solutions. Characteristics of precise orbits for ERS-1 and TOPEX/Poseidon, obtained from simultaneous solutions, will be compared with single-arc solutions as well as with external orbit solutions. Also, the practical implications of balancing data from one satellite in a simultaneous proces with respect to the other will be studied. Chapter 8 describes how the principle of simultaneous altimetry analysis has been applied to gravity field tailoring via a discussion of the followed solution method, the included data, and some results and practical problems that were encountered during the process. A final Chapter summarises the output of the entire project, also listing some recommendations for future work that could not be covered by the present study.

Among existing parameter estimation programmes some classical examples can be found of large scientific software packages that have become practically unmanagable, due to modifications by many different people who may not always count software maintainability among their top priorities. Three factors that contribute to such a situation are a lack of software structure, a lack of systematic documentation, and a poor or unsystematic user interface. As mentioned above, the danger of a lack of documentation is encountered by systematically covering the theory and its implementation in the first part of this thesis. Some additional theory, related to reference frames and time definitions used throughout this thesis, is included in Appendix A. Concerning a possible lack of structure, it has been an advantage that Faust was written from scratch, with a clear perception of what the final package should look like. At the time of this report the structure of the software is therefore no source of concern, as the code has been constructed in managable subroutines that are all developed in a similar coherent style. Some remarks regarding the structure of Faust are included in Appendix B. In particular, this Appendix includes two lists of subroutines, one sorted in alphabetical order to assist analysis of the code by highlighting the task of a particular subroutine, and one organised in functionally coherent groups in order to explain which routines relate to particular aspects of the solution process.

The third problem, related to the way in which the programme is used in practice and in which future enhancements may be implemented, has been recognised from the very start of the project. Substantial effort has therefore been invested in the design of a flexible high-level language interface to Faust. A basic understanding of its working will be essential for future programmers as well as for users, but the treatment of this natural language interface would somewhat interrupt the logical line of the thesis if it were to be included as a separate Chapter. This is therefore done in Appendix C.

Finally, Appendices D and E offer direct assistance to users and programmers of Faust respectively, in the form of 'cookbook' discussions of various situations that may occur in practice.

1.4 Faust

The existing orbit determination software at Aston University was known as the SATAN-A package, referring to Satellite Analysis, with an -A extension to indicate amendments of the original set added at Aston. After initial development at the Royal Greenwich Observatory (Sinclair and Appleby, 1986) this programme had been extended and modified by many different people, which had gradually turned the code into an arrangement of FORTRAN statements that stood up to its diabolic name.

Although the code of the new programme has been written from scratch, it has several obvious roots in the SATAN-A suite, in particular concerning the mathematics behind most acceleration models. The name of doctor Heinrich Faust, the ruthless scientist in Goethe's play, imposed itself therefore as a natural choice for the name of a programme that derives its force from Satan. Matching its namesake in ambition, the programme stretches the limits of an area of computational science that already operates at the boundaries of computer capacities. For normal orbit determination tasks Faust will settle for any of the SUN Sparc 4 workstations available to Aston's Space Geodesy Group, each with a modest 24 Mb of RAM. However, to reach his full unprecedented multi-satellite and multi-arc power, Faust prefers to embrace the several hundreds of Megabytes of a SUN Sparc 10 server that also happens to be called Satan. Only then will Faust defy gravity, laughing at the billions of partial derivatives that aimlessly try to prevent him from obtaining the knowledge he strives for.

Rumour goes that Goethe - wary of the upcoming analytical science of the early nineteenth century - based the character in his play on one of his illustrious contemporaries, sometimes called the Prince of Mathematicians. In order to make sure that people would not overlook the innuendo, he merely had to shift the first and last letter of his name one place in the alphabet to arrive at Faust, a doctor described by Johann Speiss in 1587 as having sold his soul to Satan in exchange for eternal youth. By irony of fate, practically all mathematical techniques that are used in modern satellite geodesy, from least squares parameter estimation to numerical integration, happen to be based upon fundamental work by the same mathematician that Goethe chose to disparage. The name of the programme may therefore also be interpreted as an hommage to Karl Friedrich Gauss who made it all possible - barely two decades after Cook's expedition to Tahiti, but almost two centuries before computer technology finally managed to support his visions.

Parameter estimation — Chapter 2

The documentation of the solution techniques in Faust will start with an overall look at the computational methods for extracting geophysical information from satellite tracking data. Relations between geophysical models and the orbital motion of a satellite are in general non-linear, and although it is possible to compute the orbit of a satellite on the basis of such models, the inverse problem of computing model parameters from measured satellite positions can only be solved by indirect data reduction methods. This Chapter describes the least squares parameter estimation process in Faust, starting with a brief summary of the theory and then moving to a more practical discussion of its numerical aspects, in particular in relation to simultaneous multi-arc solutions. Subjects that will be treated include the adopted internal organisation of parameters, partitioning and compression of the normal matrix as used to reduce demands with respect to working memory and CPU time, as well as the general structure of the differential equations involved in the construction of the solution.

2.1 Review of essential theory

A tracking observation can be simulated numerically on the basis of mathematical models. These include models for the forces that act upon the satellite and determine its position as a function of time, as well as models for the geometry of the measurement. The simulated equivalent of the observation matches its physically observed counterpart to an extent that depends on modelling detail, measurement accuracy, and, which is most relevant in this context, on the accuracy of the a priori values for various parameters within the models. The following notations are used in this thesis :

O_i	An observed measurement value at a time t_i
$\underline{P}$	The vector formed by all parameters of interest
$C_i = C_i(\underline{P})$	The calculated equivalent of O_i, being a function of $\underline{P}$
$R_i(O_i,\underline{P}) = O_i - C_i(\underline{P})$	The measurement residual at t_i

(2.1)

Observations and mathematical models only have a finite precision, and the problem of finding the most realistic parameter values is handled statistically by making the amount of observations substantially larger than the number of parameters. The vector $\underline{P}$ is then overdetermined and can not be solved uniquely. A second problem in optimising the parameter vector is formed by the non-linear character of the functions $R_i(O_i, \underline{P})$, which prevents the derivation of explicit analytical expressions $\underline{P} = \underline{P}(\underline{O})$ for the parameters in terms of the observations. Despite of this, each measurement residual R_i contains some information about the accuracy of the model parameters. The task of the parameter estimation process is to transform this information into corrections to the a priori parameter values, in such a way that the adjusted parameters will lead to smaller residuals if the simulated measurements are recomputed. This correction process is then iterated to convergence.

In order to compute adequate corrections, the influence of arbitrary parameter changes $\underline{\Delta P}$ upon the residuals can be formulated. The first order effect of parameter corrections upon a calculated measurement is found from the linearised series expansion

$$(C_i)_{n+1} = (C_i)_n + \begin{pmatrix} \frac{\partial C_i}{\partial P_1} & \frac{\partial C_i}{\partial P_2} & \cdots & \frac{\partial C_i}{\partial P_k} \end{pmatrix}_n \cdot \begin{pmatrix} \Delta P_1 \\ \Delta P_2 \\ \vdots \\ \Delta P_k \end{pmatrix}_n \tag{2.2}$$

in which index n is used to denote step n of the iterative correction process described above. With (2.2) the residuals for iteration $n+1$ can now be expressed directly in terms of the corrections $\underline{\Delta P}$ from the previous iteration n :

$$\underline{R}_{n+1} = \underline{O} - \underline{C}_{n+1} = \underline{O} - \left(\underline{C}_n + A \cdot \underline{\Delta P}_n\right) = \underline{R}_n - A \cdot \underline{\Delta P}_n \tag{2.3}$$

Matrix A in (2.3) contains the partial derivatives of all calculated measurements with respect to each parameter, having a column for each parameter and a row (2.2) for each observation :

$$A = \begin{pmatrix} \frac{\partial C_1}{\partial P_1} & \frac{\partial C_1}{\partial P_2} & \cdots & \frac{\partial C_1}{\partial P_k} \\ \frac{\partial C_2}{\partial P_1} & \frac{\partial C_2}{\partial P_2} & \cdots & \frac{\partial C_2}{\partial P_k} \\ \vdots & \vdots & & \vdots \\ \frac{\partial C_m}{\partial P_1} & \frac{\partial C_m}{\partial P_2} & \cdots & \frac{\partial C_m}{\partial P_k} \end{pmatrix} \tag{2.4}$$

Assuming that all measurements are independent and equally reliable, the preferred solution for the parameter correction vector $\underline{\Delta P}_n$ will be the one for which the norm of vector $\underline{R}_{n+1}$ is minimised, which is equivalent to minimising the sum of the squares of its elements. This sum of squares is given by the scalar product of vector (2.3) with itself:

$$\begin{aligned} S_{n+1} &= \underline{R}^T_{n+1}\,\underline{R}_{n+1} = \left(\underline{R}_n - A\,\underline{\Delta P}_n\right)^T \left(\underline{R}_n - A\,\underline{\Delta P}_n\right) \\ &= \underline{R}^T_n\,\underline{R}_n - \underline{R}^T_n A\,\underline{\Delta P}_n - \underline{\Delta P}^T_n A^T\,\underline{R}_n + \underline{\Delta P}^T_n A^T A\,\underline{\Delta P}_n \end{aligned} \tag{2.5}$$

In order to minimise S_{n+1} the partial derivatives of the matrix products in (2.5) with respect to $\underline{\Delta P}_n$ are determined, using the readily verifiable product rule notation

$$\frac{\partial(X^T Y)}{\partial Z} = \left(\frac{\partial(X^T)}{\partial Z}\right) Y + \left(\frac{\partial(Y^T)}{\partial Z}\right) X \tag{2.6}$$

Applying this relation to the four parts of sum S in (2.5) provides

$$\begin{aligned} \frac{\partial(R^T R)}{\partial \Delta P} &= 0 \\ \frac{\partial(R^T A\,\Delta P)}{\partial \Delta P} &= \frac{\partial(R^T A)}{\partial \Delta P}\Delta P + \frac{\partial(\Delta P^T)}{\partial \Delta P}(R^T A)^T = 0 + I A^T R = A^T R \\ \frac{\partial(\Delta P^T A^T R)}{\partial \Delta P} &= \frac{\partial(\Delta P^T)}{\partial \Delta P}(A^T R) + \frac{\partial((A^T R)^T)}{\partial \Delta P}\Delta P = I A^T R + 0 = A^T R \\ \frac{\partial(\Delta P^T A^T A\,\Delta P)}{\partial \Delta P} &= 2\left(\frac{\partial \Delta P^T}{\partial \Delta P}\right) A^T A\,\Delta P = 2\,A^T A\,\Delta P \end{aligned} \tag{2.7}$$

The minimisation condition for S_{n+1} follows by imposing that its derivative must be zero :

$$\frac{\partial S}{\partial \Delta P} = 0 - A^T R - A^T R + 2\,A^T A\,\Delta P \equiv 0 \tag{2.8}$$

Because S is quadratic in $\underline{\Delta P}$, and because (2.2) shows that the residuals become infinitely large for parameter corrections that approach positive or negative infinity, it is apparent that (2.8) indicates a minimum rather than a maximum. Rearranging terms leads to an expression - the normal equation - for a vector $\underline{\Delta P}$ that will ensure the smallest possible residuals in the next process iteration :

$$(A^T A)\underline{\Delta P} = A^T \underline{R} \tag{2.9}$$

The parameter corrections are of course found explicitly by multiplying both sides of (2.9) with the inverted normal matrix $(A^T A)^{-1}$.

2.2 Weights and constraints

In the above two important assumptions have been made. At first, the adopted concept of minimising S_{n+1} assumes that all observations are independent and equally reliable, so that the smallest sum of squares of all residuals indeed corresponds to the best set of parameter corrections. Furthermore, the truncation of the series in (2.2) assumes that the observability of the parameter corrections from linearised relations is guaranteed. In practice these assumptions are not always realistic, for which reason additional a priori information is taken into account related to the quality of the observations and to the quality of the initial parameter estimates respectively. A priori information about the reliability of the observations is entered into the process by multiplying each of the observations with a weight factor that will give that observation an increased or decreased influence on the result of the estimation process. This changes (2.3) into

$$w\underline{R}_{n+1} = w\underline{R}_n - wA\underline{\Delta P} \tag{2.10}$$

in which w is a diagonal weight matrix. If (2.10) is substituted into (2.5) and the differentiations are repeated, the weighted normal equation becomes

$$(A^TWA)\Delta P = A^TWR \tag{2.11}$$

Here, matrix $W = w^Tw$ so that a single element is $W_j = w_j^2$. An element j,k of the weighted normal matrix is therefore given by the weighted sum of partial products j and k :

$$(A^TWA)_{jk} = \sum_{i=1}^{N_{obs}} \left(\frac{\partial C_i}{\partial P_j}\right) w_i^2 \left(\frac{\partial C_i}{\partial P_k}\right) \tag{2.12}$$

Because of the direct application to satellite tracking data processing, the description of the least-squares solution in the previous Section used the approach of minimising measurement residuals within an iterative parameter correction process. More generally, Eykhoff (1974) derived the normal equation as a maximum likelihood solution in regression analysis. It is then assumed that the post-solution residuals are distributed as white noise :

$$\underline{R} - A\underline{\Delta P} = \underline{\epsilon} \tag{2.13}$$

in which $\underline{\epsilon}$ is a noise vector that has elements with zero mean value and variance σ^2. For the elements of the weight matrix w we can of course always adopt the value

$$w_i = \frac{1}{\sigma_i} \tag{2.14}$$

where σ_i is an a priori standard deviation for the measurement. This changes the weighted residuals w_iR_i into dimensionless numbers, which is useful if different types of measurements are to be combined within a single parameter estimation process : it generalises the physical problem into a purely mathematical one. Following this statistical approach, the inverted normal matrix is found to be the covariance matrix (e.g. Ljung *et al.*, 1983) for the parameter solutions. The diagonal elements of the inverted normal matrix V contain the parameter variances, while the non-diagonal elements form the parameter covariances, used to compute correlations ρ between the parameters according to

$$\begin{aligned} \sigma_j &= \sqrt{V_{jj}} \\ \rho_{jk} &= \frac{V_{jk}}{\sigma_j \sigma_k} \end{aligned} \tag{2.15}$$

The interpretation of post-solution residuals as noise with a Gaussian distribution also supports the principle of data editing : any observation that produces a residual larger than a certain multiple of the standard deviation σ for that datatype can be rejected as an unreliable measurement (i.e. we only use a central band of the Gaussian distribution). The need for this practice can be appreciated by inspection of the basic normal equation (2.9), which shows that one large residual for a wide point may have more influence on the solution than many small ones related to accurate measurements. Practical implications of data editing in Faust will be discussed in more detail in Chapter 5.

A priori information related to the quality of the initial parameter values is entered into the system by using the parameter corrections as additional observations, with corresponding partials and weights :

$$R = \begin{pmatrix} R_O \\ R_P \end{pmatrix}, \qquad A = \begin{pmatrix} A_O \\ A_P \end{pmatrix}, \qquad W = \begin{pmatrix} W_O \\ W_P \end{pmatrix} \tag{2.16}$$

in which index O refers to the residuals for the observations, and index P to the added residuals for the parameters. The vector R_P^n for process iteration n is defined as the difference between the actual parameter values and the a priori parameter values (the latter indicated by superscript 0) :

$$R_P^n = \begin{pmatrix} P_1^{\,0} - P_1^{\,n} \\ P_2^{\,0} - P_1^{\,n} \\ \vdots \\ P_{N_{par}}^{\,0} - P_{N_{par}}^{\,n} \end{pmatrix} \tag{2.17}$$

If the parameters are independent, the matrix with the partials for the parameters is an identity matrix :

$$\frac{\partial P_j}{\partial P_k} = \delta_{j,k} = \begin{cases} 1, & (j = k) \\ 0, & (j \neq k) \end{cases} \qquad \rightarrow \qquad A_P = I \tag{2.18}$$

The matrices (2.16) can be substituded into (2.11), with A_P replaced by an identity matrix :

$$\left(A_O{}^T W_O A_O + W_P\right)\Delta P = A_O{}^T W_O R_O + W_P R_P \tag{2.19}$$

From a point of view of statistics, this corresponds to a conditional (i.e. Bayesian) problem : we compute the parameter corrections *given* a certain likelihood for the initial values. Expression (2.19) also shows in a practical way how the constraints W_p are added to the standard normal matrix, namely by modifying each of the diagonal elements of the normal matrix as well as the elements of the right-hand side of the equation. For large geophysical applications, like the computation of a gravity field solution, it is common practice to keep the post-solution covariance matrix and use it to constrain later solutions. In that case, the inverted covariance matrix (i.e. the original normal matrix) is added entirely to the corresponding part of the new normal matrix, so that the later process includes the earlier one while adding new information to it.

2.3 Matrix partitioning

To ensure a reliable outcome of the statistical parameter adjustment process, the number of observations will substantially outnumber the amount of estimated parameters, implying that the number of rows of the matrix A from (2.4) is significantly larger than the number of columns. In practice the size of the observation matrix A will often be too large to allow storing and processing of the matrix as a whole. A first reduction in processing demands can be derived from equations (2.2) and (2.12) which show that the normal matrix is in fact a sum of matrices, one for each observation :

$$\begin{aligned} A^T A &= \sum_{i=1}^{N_{obs}} A_i{}^T A_i \\ A_i &= \begin{pmatrix} \dfrac{\partial C_i}{\partial P_1} & \dfrac{\partial C_i}{\partial P_2} & \cdots & \dfrac{\partial C_i}{\partial P_k} \end{pmatrix} \end{aligned} \tag{2.20}$$

From (2.20) it follows that it is not necessary to store all partials for each observation if contributions of individual observations are added to the normal matrix directly. Even in that case, the dimension of the normal matrix equals the amount of estimated parameters, which can be in the order of several thousands for multi-arc solutions or for the computation of detailed geophysical models. To reduce the demands with respect to working memory and CPU-time for storing and inverting such a large matrix, effective use is made of the fact that many parameters are known to be totally independent from other parameters, causing large parts of the normal matrix to contain zeroes.

As can be seen easily from (2.20), two parameters are independent - i.e. the corresponding element $(A^TA)_{jk}$ remains zero - if for each observation $C_i(\underline{P})$ at least one of the partial derivatives with respect to these parameters is zero. Such parameters cause part of the normal matrix to be a block-diagonal matrix, in which case the corresponding section of the covariance matrix will also be diagonal : all correlations (2.15) are zero. This can for instance be the case with range bias parameters from two different tracking stations. Faust distinguishes between three Categories of parameter correlations :

Category 1 Parameters that may be correlated with any other parameter within the solution. A corresponding normal matrix is a full matrix.

Category 2 Parameters that appear in correlated pairs, while no correlations exist between parameters from different pairs. These parameters cause a normal matrix to be a diagonal matrix with additional upper and lower diagonals :

$$\begin{pmatrix} D_1 & E \\ E^T & D_2 \end{pmatrix}$$

with diagonal sub-matrices D_1, D_2 and E $(= E^T)$.

Category 3 Parameters that are totally uncorrelated with each other, causing a normal matrix to be a diagonal matrix.

If these three categories of parameters occur simultaneously, the normal matrix can be organised into the following structure of submatrices :

$$\begin{pmatrix} A & B & C & F \\ B^T & D_1 & E & 0 \\ C^T & E^T & D_2 & 0 \\ F^T & 0 & 0 & D_3 \end{pmatrix} \qquad (2.21)$$

The diagonal submatrix D_3 corresponds to parameters of category 3, while the diagonal submatrices D_1, D_2 and E are related to parameters of category 2. The submatrix A relates to all parameters of category 1, and because these parameters may still be correlated with any of the parameters of category 2 or 3, additional non-zero matrices B, C and F occur. It is assumed that the parameters of category 2 and 3 are not correlated with each other : if correlations between these categories would appear, they effectively become parameters of category 1. It will be obvious how the righthand side of the normal equation (2.11) can be partitioned in four subvectors (referred to as $\underline{p}$, $\underline{q}$, $\underline{r}$ and $\underline{s}$ respectively) corresponding in sizes with those of the submatrices of the partitioned normal matrix (2.21).

All diagonal submatrices can now be stored in one-dimensional arrays rather than in rectangular matrices, while the submatrices that only contain zeroes are not stored at all. The full matrix[1] A is symmetrical due to the way in which the normal matrix is constructed from equation (2.20). Faust therefore stores only one triangular half of the matrix in a much smaller one-dimensional array, using a packed format with a mapping function $A_{ij} = a_k$ like

$$\begin{array}{c|ccccc|}
 & j=1 & 2 & 3 & 4 & \dots \\
\hline
i=1 & a_1 & a_2 & a_4 & a_7 & \\
2 & & a_3 & a_5 & \vdots & \\
3 & & & a_6 & & \\
\vdots & & & & \ddots & \\
 & & & & & a_N
\end{array} \quad \rightarrow \quad A(i,j) = a\left(\frac{j^2 - j}{2} + i\right) \qquad (2.22)$$

The introduced partitioning and packing reduce the required storage space for the normal matrix in most cases to only a small fraction of the size required for the full normal matrix. In other words : within the physical limits of the available memory a much larger amount of parameters can be solved for simultaneously than would be the case if no partitioning would be applied. This is particularly relevant for multi-satellite or multi-arc solutions, that will obviously contain more parameters than a single arc process.

Apart from the savings in storage space, the partitioned matrix will also require substantially less CPU-time for its inversion than would be needed for inverting an unpartitioned matrix of the same dimensions. Equation (2.11) is essentially a linear vector equation of the form

$$M\underline{x} = \underline{y} \qquad (2.23)$$

With the above partitioning this changes into a set of simultaneous equations :

[1] note that this is the *partition* A from (2.21), not the observation matrix from (2.3) onwards

$$\begin{cases} A\,\underline{x}_1 + B\,\underline{x}_2 + C\,\underline{x}_3 + F\,\underline{x}_4 = \underline{p} \\ B^T\underline{x}_1 + D_1\,\underline{x}_2 + E\,\underline{x}_3 = \underline{q} \\ C^T\underline{x}_1 + E\,\underline{x}_2 + D_2\,\underline{x}_3 = \underline{r} \\ F^T\underline{x}_1 + D_3\,\underline{x}_4 = \underline{s} \end{cases} \tag{2.24}$$

The vectors $\underline{x}_1 \dots \underline{x}_4$ can be found from a staged elimination process. Starting with the last of the four equations in (2.24) and moving upwards, we write

$$\underline{x}_4 = D_3^{-1}\left[\underline{s} - F^T\underline{x}_1\right] \tag{2.25}$$

$$\underline{x}_3 = D_2^{-1}\left[\underline{r} - C^T\underline{x}_1 - E\underline{x}_2\right] \tag{2.26}$$

$$\underline{x}_2 = D_1^{-1}\left[\underline{q} - B^T\underline{x}_1 - E\,\underline{x}_3\right] \tag{2.27}$$

$$\underline{x}_1 = A^{-1}\left[\underline{p} - B\underline{x}_2 - C\underline{x}_3 - F\underline{x}_4\right] \tag{2.28}$$

Combining (2.26) with (2.27) results in a relation between $\underline{x}_1$ and $\underline{x}_2$:

$$\underline{x}_2 = \left(D_1 - ED_2^{-1}E\right)^{-1}\ \left[\left(\underline{q} - ED_2^{-1}\underline{r}\right) - \left(B^T - ED_2^{-1}C^T\right)\underline{x}_1\right] \tag{2.29}$$

In a similar way (2.25), (2.27) and (2.28) combine into

$$\begin{gathered} \underline{x}_1 = A^{-1}\left[\ \underline{p} - B\underline{x}_2 - CD_2^{-1}\left(\underline{r} - C^T\underline{x}_1 - E\underline{x}_2\right) - FD_3^{-1}\left(\underline{s} - F^T\underline{x}_1\right)\ \right] \quad \leftrightarrow \\ \left(A - CD_2^{-1}C^T - FD_3^{-1}F^T\right)\underline{x}_1 = \left(\underline{p} - CD_2^{-1}\underline{r} - FD_3^{-1}\underline{s}\right) - \left(B - CD_2^{-1}E\right)\underline{x}_2 \end{gathered} \tag{2.30}$$

Finally, equation (2.29) can be substituted into (2.30) to obtain an expression for $\underline{x}_1$ alone :

$$\begin{aligned} &\left[A - CD_2^{-1}C^T - FD_3^{-1}F^T + \left(B - CD_2^{-1}E\right)\left(D_1 - ED_2^{-1}E\right)^{-1}\left(B^T - ED_2^{-1}C^T\right)\right]\underline{x}_1 \\ &= \left(\underline{p} - CD_2^{-1}\underline{r} - FD_3^{-1}\underline{s}\right) - \left(B - CD_2^{-1}E\right)\left(D_1 - ED_2^{-1}E\right)^{-1}\left(\underline{q} - ED_2^{-1}\underline{r}\right) \end{aligned} \tag{2.31}$$

From this, $\underline{x}_1$ can be solved and substituted in (2.29) to solve $\underline{x}_2$, after which $\underline{x}_3$ and $\underline{x}_4$ are solved from (2.26) and (2.25) respectively. Although this process looks more complicated than solving the simple linear relation (2.23), many of the occurring terms are diagonal matrices that are inverted by simply inverting their individual elements (note that these should therefore all be non-zero). The largest full matrix to be inverted has the size of the partitioning A, i.e. the part related to fully correlated parameters. As will be discussed in Chapters 4 and 5, the dimensions of the diagonal matrices D are often large in comparison to that of the partitioning A, which means that the inversion of the partitioned matrix is significantly less demanding in terms of processing time than that of a full normal matrix of the same dimensions.

step	calculated expression	data used from array												store in array
		A	B	C	D 1	D 2	D 3	E	F	p	q	r	s	
M1	$A - CD_2^{-1}C^T - FD_3^{-1}F^T$	O		O		O	O		O					A
M2	$B - CD_2^{-1}E$		O	O		O		O						B
M3	$D_1 - ED_2^{-1}E$				O	O		O						D_1
M4	$\underline{p} - CD_2^{-1}\underline{r} - FD_3^{-1}\underline{s}$			O		O	O		O	O		O	O	$\underline{p}$
M5	$\underline{q} - ED_2^{-1}\underline{r}$					O		O			O	O		$\underline{q}$
M6	$M_1 + M_2M_3^{-1}M_2^T$	R	R		R									A
M7	$M_4 - M_2M_3^{-1}M_5$		R		R									$\underline{p}$
$\underline{x}_1$	$M_6^{-1}M_7$	R								R				$\underline{x}_1$
$\underline{x}_2$	$M_3^{-1}(M_5 - M_2^T\underline{x}_1)$		R		R						R			$\underline{x}_2$
$\underline{x}_3$	$D_2^{-1}(\underline{r} - C^T\underline{x}_1 - E\underline{x}_2)$			O		O		O				O		$\underline{x}_3$
$\underline{x}_4$	$D_3^{-1}(\underline{s} - F^T\underline{x}_1)$						O		O				O	$\underline{x}_4$

Table 2.1 Succession of computation steps during the inversion of the partitioned normal matrix. A marking 'O' indicates that the original contents of the corresponding matrix is used, while an 'R' indicates that an intermediate result is used, having replaced the original contents of the corresponding matrix in one of the previous computation steps.

The savings in storage space are of course meaningless if additional memory would be required for performing the inversion process itself. Table (2.1) shows the carefully deviced sequence of computations in Faust, and the arrays in which the intermediate results are stored without any need for additional storage space in RAM memory or temporary disk files. As indicated by markings 'O' and 'R', the original contents of any matrix or vector is only overwritten if it is no longer needed for the computation of the solution vectors $\underline{x}_1 \dots \underline{x}_4$, which follows from the fact that no 'R' appears higher than an 'O' within the same column of the table.

2.4 Choleski decomposition of the packed matrix

If the packed format (2.22) for the partitioning A of the normal matrix is to be maintained during the inversion process, some special measures must be taken. A standard Gauss elimination process as described e.g. by Salvadori & Baron (1961) converts a matrix equation into a triangular system by multiplying each row i with a factor

$$m_{ij} = \frac{a_{ji}}{a_{ii}} \tag{2.32}$$

and then subtracting row i from the lower row j. By repeating this for each lower row (i.e. j > i) the column below diagonal element a_{ii} will only contain zeroes, and by repeating this straightforward procedure for each column i, an upper-triangle matrix arises :

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \rightarrow \begin{pmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ 0 & u_{22} & \cdots & u_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & u_{nn} \end{pmatrix} \tag{2.33}$$

The right-hand side of the matrix equation (2.19) has to follow the same row operations, for which reason all multipliers m_{ij} must be kept if a solution for a particular set of equations is to be determined. In general, the multipliers are conveniently stored in the lower triangle matrix, but in the packed format (2.22) of Faust there simply is no memory available for a lower triangle matrix. However, a special case - the Choleski decomposition - appears if a triangular root-matrix R can be found such that the normal matrix can be written as

$$A = R^T R \tag{2.34}$$

which can be worked out as

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & \ddots & \vdots \\ a_{n1} & & \cdots & a_{nn} \end{pmatrix} = \begin{pmatrix} r_{11} & 0 & \cdots & 0 \\ r_{12} & r_{22} & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ r_{1n} & & \cdots & r_{nn} \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & \cdots & r_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & r_{nn} \end{pmatrix} \tag{2.35}$$

Assuming that the triangular matrix R exists, the rules of matrix multiplication provide

$$a_{11} = r_{11}^2 \quad \rightarrow \quad r_{11} = \sqrt{a_{11}} \tag{2.36}$$

and subsequently

$$a_{1j} = r_{11} r_{1j} \quad \rightarrow \quad r_{1j} = \frac{a_{1j}}{\sqrt{a_{11}}} \tag{2.37}$$

Continuing with the second row of (2.35), we find

$$a_{22} = r_{12}r_{12} + r_{22}r_{22} \rightarrow r_{22} = \sqrt{a_{22} - r_{12}r_{12}} \qquad (2.38)$$

in which r_{12} can now be substituted directly, having been computed from (2.37). More generally, the diagonal element of row i is found from the expression

$$r_{ii} = \sqrt{a_{ii} - \sum_{j=1}^{i-1} r_{ij}^2} \qquad (2.39)$$

after which the non-diagonal elements of row i are found from

$$r_{ij} = \frac{a_{ij} - \sum_{k=1}^{j-1} r_{ik}r_{kj}}{r_{ii}} \qquad (2.40)$$

From (2.39) it follows that in order for a root-matrix R to exist, all diagonal elements a_{ii} must be greater than the sum of squares of the r_{ij} (hence each diagonal element a_{ii} must at least be positive). James *et al.* (1977) showed that a sufficient condition for the existence of R is that the original matrix is positive definite, meaning that $\underline{x}^T A \underline{x} > 0$ for any non-zero vector $\underline{x}$. This condition is indeed always satisfied here, because of the way in which the normal matrix is constructed from (2.20) :

$$\underline{x}^T(A_i^T A_i)\underline{x} = (\underline{x}^T A_i^T)(A_i \underline{x}) = (A_i \underline{x})^T (A_i \underline{x}) = (\underline{v}, \underline{v}) = \sum v_i^2 > 0 \qquad (2.41)$$

Because in the Choleski decomposition the lower triangle is by definition equal to the transposed of the upper triangle matrix, it is unnecessary to store any multipliers and the packed matrix structure from (2.22) can be maintained throughout the decomposition process, if the latter is coded carefully. Once that the triangulation (2.35) is determined by applying (2.39) and (2.40) for each row, the inverted normal matrix is computed from

$$M = R^T R \rightarrow M^{-1} = R^{-1}(R^T)^{-1} \qquad (2.42)$$

while the inversion P of matrix R can be determined directly from back-substitution :

$$\begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & p_{2n} \\ \vdots & & \ddots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{nn} \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & \cdots & p_{1n} \\ 0 & r_{22} & \cdots & r_{2n} \\ \vdots & \ddots & \ddots & \vdots \\ 0 & \cdots & o & r_{nn} \end{pmatrix} = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & 1 \end{pmatrix} \qquad (2.43)$$

which leads to relations like

$$p_{11}r_{11} = 1 \rightarrow p_{11} = \frac{1}{r_{11}} \qquad (2.44)$$

$$p_{11}r_{12} + p_{12}r_{22} = 0 \quad \Rightarrow \quad p_{12} = \frac{-p_{11}r_{12}}{r_{22}} \tag{2.45}$$

In general, the equations for computing the inverse of R have the form

$$i > j \quad \Rightarrow \quad p_{ij} = 0 \tag{2.46}$$

$$i = j \quad \Rightarrow \quad p_{ii} = \frac{1}{r_{ii}} \tag{2.47}$$

$$i < j \quad \Rightarrow \quad p_{ij} = \frac{-\sum_{k=1}^{j-1} p_{ik}r_{kj}}{r_{jj}} \tag{2.48}$$

As with the triangulation, the entire inversion can be performed within the packed matrix since each element of the uninverted matrix R is only needed *before* it is replaced by the element of P with identical indices. As illustrated in Figure 2.1, the matrix inversion in Faust has been coded with emphasis on processing efficiency, by avoiding unnecessary floating point operations (for instance for locating elements of A within the one-dimensional array).

```
*  COMPUTE ROOT-MATRIX R

n1 = 0
n2 = 0

do i = 1,m
   n1 = n1 + i

   Sum = A(n1)
   do j = 1, i - 1
      Sum = Sum - A(n2+j)**2
   end do
   A(n1) = DSQRT(Sum)

   n3 = n1
   do j = i+1,m
      Sum = A(n3+i)
      do k = 1,i-1
         Sum = Sum - A(n3+k) * A(n2+k)
      end do
      A(n3+i) = Sum/A(n1)
      n3 = n3 + j
   end do

   n2 = n1
end do
```

```
*  INVERT R INTO P

n1 = 0
n2 = 0

do i = 1,m
   n1 = n1 + i

   A(n1) = 1.D0/A(n1)

   n3 = 0
   do j = 1,i - 1
      Sum = 0.D0
      n4  = n3 + j
      do k = j,i-1
         Sum = Sum - A(n4) * A(n2+k)
         n4 = n4 + k
      end do
      A(n2+j) = Sum*A(n1)
      n3 = n3 + j
   end do

   n2 = n1
end do
```

```
*  COMPUTE  P x P

n0 = 0

do i = 1,m
   n1 = n0
   do j = i,m

      Sum = 0.D0
      n2  = n1
      do k = j,m
         n3  = n2 + i
         n4  = n2 + j
         n2  = n2 + k
         Sum = Sum + A(n3) * A(n4)
      end do
      A(n1+i) = Sum

      n1 = n1 + j
   end do
   n0 = n0 + i
end do
```

Figure 2.1 Extract from Faust : inversion of the packed triangular matrix of dimension m.

2.5 Partial derivatives

In satellite tracking, the observations O_i will in general provide information about either the position or the velocity of the satellite at the time t_i of the measurement. To obtain a calculated equivalent C_i it will then be necessary to compute the orbital position or velocity of the satellite at t_i, which is done by numerical integration of the equations of motion :

$$\frac{d^2 \underline{x}}{dt^2} = \underline{a}_{tot}(t, \underline{P}) \tag{2.49}$$

The numerical integration process used in Faust will be explained in Chapter 3. Initial values for the position and velocity, as required for this integration, are usually among the estimated parameters $\underline{P}$ because the entire integrated orbit arc will change - and with it the calculated observations - if the initial value of the integrated function changes. All other estimated parameters can be separated into two groups. The first group is formed by parameters within the acceleration model $\underline{a}_{tot}$, like coefficients in a gravity field model, scale factors for an atmospheric drag force, etcetera. Many of these parameters will form the central object of study in space geodesy, and their implementation will be discussed in Chapter 4. The second group is formed by various parameters that appear in the geometrical modelling of the calculated observation, like range bias corrections, tracking station coordinates, etcetera. Some of these parameters will be discussed in Chapter 5. In order to construct the observation equations (2.2) the partial derivatives of the calculated observation with respect to each relevant parameter are required. To express possible explicit relations between the calculated observation and a certain parameter, as well as possible implicit dependencies via the orbital position, the partial with respect to a parameter P_j can be written as

$$\frac{\partial C_i}{\partial P_j} = \left(\frac{\partial C_i}{\partial P_j} \right)_{expl} + \left(\frac{\partial C_i}{\partial x} \frac{\partial x}{\partial P_j} + \frac{\partial C_i}{\partial y} \frac{\partial y}{\partial P_j} + \frac{\partial C_i}{\partial z} \frac{\partial z}{\partial P_j} \right)_{impl} \tag{2.50}$$

In most cases only one of the two terms on the righthand side will be non-zero. The explicit term will be non-zero for parameters that appear directly in the model of the calculated observation, i.e. those of the second group described above. An analytical expression for this type of partial can usually be determined in a straightforward way from measurement geometry. The other term expresses the dependencies of the calculated observation upon the parameters that appear within the force model and hence influence the orbital position. In this implicit term the partials of the calculated observation with respect to x, y and z can also be obtained from the measurement geometry, but the partials that express the change in the orbital position as a result of a change in the force model parameters can not be determined analytically, as discussed in Section 2.1. Instead these *variational* partials are also obtained from numerical integration. Partial differentiation of (2.49) with respect to P_j results in

$$\frac{\partial}{\partial P_j} \frac{d^2 \underline{x}}{dt^2} = \frac{\partial}{\partial P_j} \underline{a}_{tot} \tag{2.51}$$

Because the position and the force model are continuous in the independent variable t, and the partial differentiation with respect to P_j is independent of the differentiation with respect to time, the order of the two differentiations may be swapped which leads to

$$\frac{d^2}{dt^2}\left(\frac{\partial \underline{x}}{\partial P_j}\right) = \frac{\partial}{\partial P_j}\,\underline{a}_{tot} \tag{2.52}$$

This is a second order differential equation in the required variational partials, the righthand side of which can again be written in explicit and implicit terms, similar to (2.50) :

$$\frac{\partial a_x}{\partial P_j} = \left(\frac{\partial a_x}{\partial P_j}\right)_{expl} + \left(\frac{\partial a_x}{\partial x}\frac{\partial x}{\partial P_j} + \frac{\partial a_y}{\partial y}\frac{\partial y}{\partial P_j} + \frac{\partial a_z}{\partial z}\frac{\partial z}{\partial P_j}\right)_{impl} \tag{2.53}$$

of course with corresponding terms for a_y and a_z. The explicit partial can be determined analytically from the mathematical relations within the force model. The partial derivatives of the acceleration components with respect to the satellite's position are also obtained in a relatively easy way, as will be discussed in Chapter 4. The remaining partial derivatives of the position with respect to the parameters are the variational partials themselves, and are now available from the numerical integration process. There may be exceptional cases in which parameters depend upon other parameters. In that case additional chain rule constructions will have to be added to (2.53), like

$$\frac{\partial a_x(P_i, P_j(P_i))}{\partial P_i} = \frac{\partial a_x}{\partial P_i} + \frac{\partial a_x}{\partial P_j}\frac{\partial P_j}{\partial P_i} \tag{2.54}$$

One example of such parameters, related to ocean tides, will be discussed in Chapter 4.

It should be noted that neither in (2.50) nor in (2.53) possible dependencies upon the satellite's velocity have been included. Observations that might depend upon the satellite's velocity in (2.50) - in particular doppler measurements - are in general converted into quasi-range observations that only relate to the satellite's position, as will be explained in Chapter 5. The acceleration in (2.53) is only weakly dependent on the velocity, because the main force upon the satellite is the Earth's central body gravity which depends upon position only, as will be discussed in Chapter 4. Omitting the velocity terms is therefore acceptable, especially because the partials do not have to be more accurate than to first order as a result of the linearisations that have already been introduced in (2.2).

2.6 Implications for simultaneous solutions

The need to integrate the variational equations - of which there may be many thousands - imposes several crucial demands upon the organisation of the code, especially in a multi-satellite / multi-arc parameter estimation process :

(1) Observations can depend on (positions of) more than one satellite, changing (2.50) into

$$\frac{\partial C_i}{\partial P_j} = \left(\frac{\partial C_i}{\partial P_j}\right)_{expl} + \sum_{i=1}^{N_{sat}} \left(\frac{\partial C_i}{\partial x_i}\frac{\partial x_i}{\partial P_j} + \frac{\partial C_i}{\partial y_i}\frac{\partial y_i}{\partial P_j} + \frac{\partial C}{\partial z_i}\frac{\partial z_i}{\partial P_j}\right)_{impl} \tag{2.55}$$

This is the case for any form of satellite-to-satellite tracking if both involved orbits are solved for, as well as for the altimetry crossovers that will play an important part in this thesis. The resulting interaction of variational partials from different arcs underlines the need to integrate orbits and partials for all satellites involved in the simultaneous solutions from Chapter 1.

(2) An observation may be related to variational partials at two or more different epochs, for instance in the case of an altimetry crossover. This interferes fundamentally with the natural chronological processing order of tracking data, and implies in fact that time would no longer be the obvious choice for the independent variable : the analysed function value (the calculated observation) now depends on two points in the independent variable (time).

(3) The variational partials at different epochs in (2) can either be related to a single arc at different epochs, or to different arcs at different epochs, introducing a further complication.

(4) Some of the parameters will be shared by more than one satellite arc (for instance parameters that describe geophysical quantities, like coefficients in a gravity field model), while other parameters are related to a single satellite only (for instance instrument bias parameters), or to a single arc only (for instance the initial position). It is not practical to store several copies of shared parameters for different arcs, but it is always necessary to integrate different sets of variational partials for different arcs because these only depend upon a single satellite position at a time.

(5) The amount of satellites, as well as the amounts and types of parameters and variational partials may be different for each parameter estimation task, while the total amount of parameters and partials will often be too large to allow substantially oversized normal matrix arrays for given limits in RAM memory. Especially in the case of multi-arc solutions this will require most of the parameter management tasks described above to be implemented by means of dynamic memory allocation.

The fact that it was not feasible to adapt existing software to support all of these features in a managable and efficient way formed one of the main incentives for the development of Faust, as a true multi-satellite / multi-arc parameter estimation programme. To cope with the various demands above and still allow the overall least-squares process to have the elementary form described in previous Sections, Faust uses internal dynamic memory allocation in the form of a pointer structure that allows free navigation through the memory blocks with parameters, partials and normal matrix partitions. In principle, all parameters are stored within a single one-dimensional array that represents the vector $\underline{P}$ from (2.1). Parameters of different types are stored in consecutive subvectors within this parameter array. Per arc, and per parameter type, the *amount* of parameters and *starting location* within $\underline{P}$ of the relevant subvector are stored in an additional mapping array $M_p(2, N_{partype}, N_{\mathrm{arc}})$. The first number of these pairs $M_p(2, i, j)$ provides the starting location of a particular subvector, the second provides the size. If different arcs j_1 and j_2 share a certain subset of the parameters, they will simply use the same starting position for that parameter type, implying that exactly the same memory locations are addressed by two or more arcs in the solution. This principle is illustrated in the Figures 2.2 and 2.3.

Because the index of a parameter in $\underline{P}$ corresponds to a specific row and column in the normal matrix, the same mapping array is used to navigate through the (partitioned) normal matrix. The array M_P - determining the structure of the parameter vector and normal matrix - is generated at the start of the programme, according to the amounts of parameters that are specified in the user input. Everywhere within the code individual parameters are accessed via this pointer structure. At the level of the least-squares process however, the parameter array can still be treated as the single vector $\underline{P}$. An additional advantage of the use of a mapping array is that the partitioning in parameters of category 1, 2 and 3 (as discussed in Section 2.3) can easily be implemented, because parameter subvectors of any type and related to any satellite arc may be stored at arbitrary locations within the total parameter vector, and are therefore organised per category.

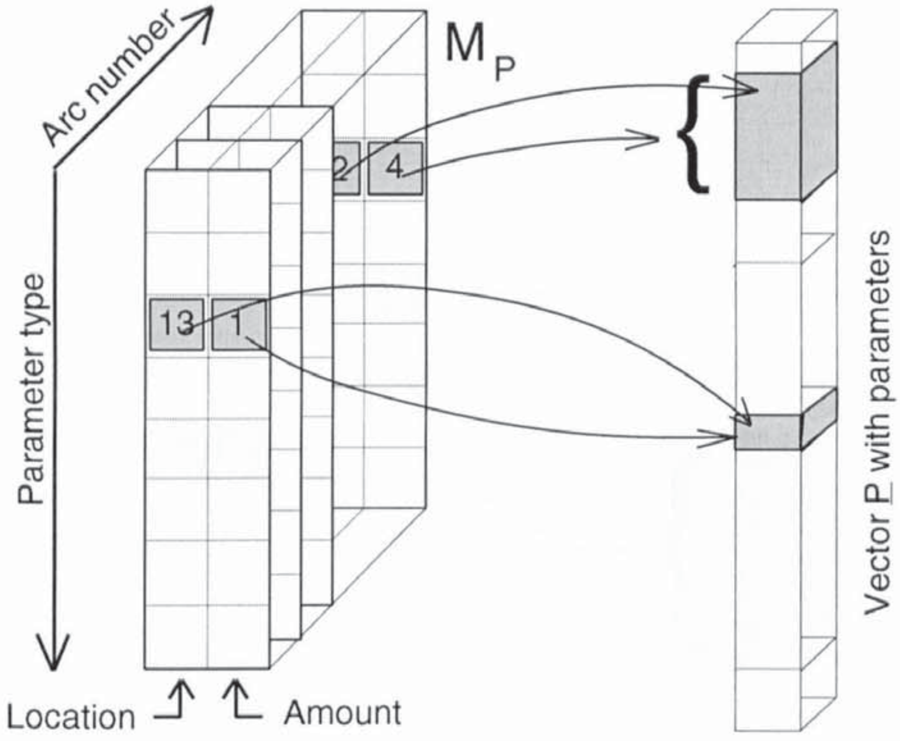


Figure 2.2 The pointer array M_P provides the starting locations and the sizes of subvectors with parameters of a certain type and related to a certain arc.

For the variational partials a similar mapping array $M_v(2, N_{partype}, N_{\mathrm{arc}})$ is used. As mentioned before, even when a parameter is shared by several satellites, the corresponding variational partials are dependent on a single satellite orbit only, and the partial derivatives from different satellite arcs are combined according to the chain rule (2.55). This means that the mapping array for the variational partials is not identical to the one for the parameter vector, although it has the same dimensions and practical application. The variational partials themselves are stored in an array of the form $E(3,\ N_{\max})$ with 3 components x, y and z for each of the $N_{\max}$ partials. The second array index is derived via the mapping array M_v. The explicit partials from (2.55) are stored in a one-dimensional array, at an offset that is provided by a similar mapping array for non-variational partials.

The use of these pointer arrays immediately solves the problems mentioned under (1), (4) and (5) above. The treatment of calculated observations that depend on vectors and variational partials at two or more different epochs inevitably requires temporary storage of the relevant partials. The details of how such vectors are stored and retrieved efficiently in Faust will be discussed in Chapter 5.

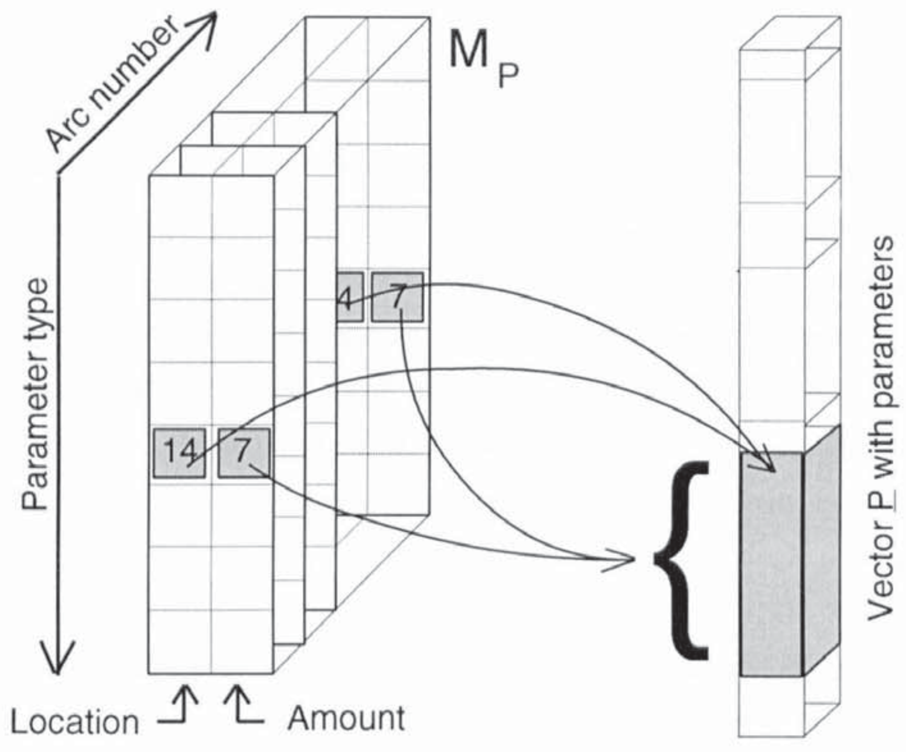


Figure 2.3 Two or more arcs in the solution can share a subvector by pointing to the same location within the parameter vector $\underline{P}$

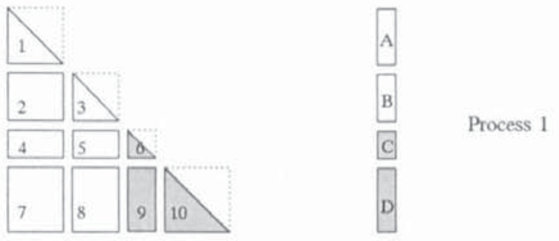
In many practical cases it is desirable to restrict the estimation process to a subset of the parameter vector only, while maintaining other parameters at a fixed value. For this reason an array with processing flags is defined in parallel to the parameter array (following the same indices, computed from the pointer array M_p), to indicate if the parameter is 'estimated', 'considered' or 'fixed'. An estimated parameter will be solved for in the iterative correction process discussed in the previous Sections. A fixed parameter, however, will not be involved in the estimation process but will be kept at its initial value throughout the correction procedure. In between these two cases a parameter may be 'considered', which implies that its variational partials will be computed, but no corrections to the initial value are applied during the iterative solution process. This creates the possibility to output the normal matrices from different processes to file and ensure that all normal matrices will still relate to identical a priori parameter values. In a later process the generated normal matrices can then be combined, resulting in a significant increase in the amount of data used for the determination of the considered parameters, as will be explained in the next Section.

2.7 Multi-run processes

Faust has default multi-arc capability, as well as multi-satellite capability because different arcs in the solution may relate to different satellites within a single parameter estimation process. Even so, there are cases in which it is useful to combine the normal matrices from several individual runs and rearrange the corresponding parameters and matrices into a larger solution process. This will be referred to as a multi-run solution, because it combines the observation data and parameters from more than one run of Faust. Examples of such processes could be the determination of a station coordinate solution (typically based on many months of tracking observations) or the computation of a gravity field solution, which would take prohibitively long computation times if performed as a single process.

Multi-run capability has been implemented in Faust in a straightforward manner. If requested in the user input, the programme will not invert the normal matrix in the usual way but rather output the parameter structure and the normal matrix to files on disk. In a later process, the input can request to do the reversed process, i.e. to read and combine several previously computed normal matrices from disk rather than compute them, and then perform the normal matrix inversion and parameter corrections for the multi-run process. Certain parameters will be shared by all individual runs (e.g. computed station coordinates, gravity field coefficients etc.) and will be described as global parameters. Other parameters will only be related to an individual arc within a single run of Faust, and will be called local parameters. In the multi-run solution, all previously generated normal matrices and submatrices that relate to global parameters are added together, while the total matrix in the multi-run solution will contain the submatrices for local parameters in unmodified form.

As an example, we can consider a case where two separate processes are combined into a multi-run solution. Each process in this straightforward example contains just two subvectors with local parameters and two with global parameters. The global parameters are the same for the two processes, and the dimensions of the corresponding sections of the normal matrix and the righthand side of the normal equation are identical. Figures 2.4 a and b below show the normal matrices and the righthand sides of (2.9) for these two individual single-run processes. The submatrices on the diagonal of the normal matrix are symmetrical, and only one triangle is available in the packed matrix structure (2.22). The other (rectangular) submatrices correspond in dimensions to the sizes of the two subvectors within $\underline{P}$ to which



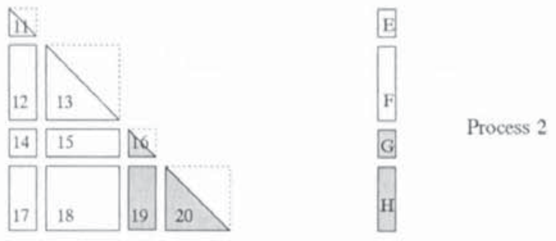


Figure 2.4 a Normal matrix and right-hand side for a single run process. Shaded submatrices relate to global parameters.

Figure 2.4 b Normal matrix and right-hand side for a second single run process.

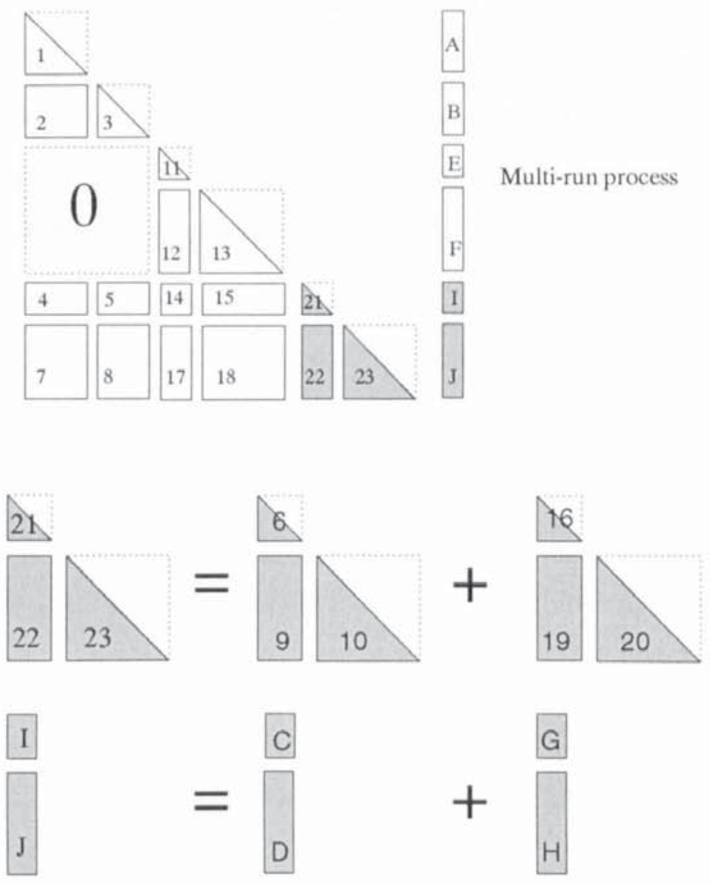


Figure 2.5 Normal matrix structure for a multi-run process, combining the earlier processes.

the particular block is related. If the two single run processes are combined in a later multi-run job, the resulting normal matrix structure will be as illustrated in Figure 2.5. To enable easy access to all submatrices in the multi-run process, Faust will store *each* submatrix produced by a single-run process in a separate file, with filenames that follow a systematic convention related to the parameter types and some processing ID numbers. Although this approach produces a large amount of files - the basic example above already contains twenty submatrices -, it allows very straightforward reconstruction of multi-run normal matrices for numerous different combinations of global and local parameter blocks, without any need to regenerate the normal matrices from individual single run processes. Obviously, the triangular matrices on the diagonal of the normal matrix are stored in a format that is different from that of the rectangular submatrices. Because the right-hand side subvectors will always be needed in combination with the corresponding triangular matrix on the diagonal, these subvectors are stored together with that triangular matrix in a single file.

In Chapters 4 and 5 it will be explained that in general the parameters of category 2 and 3 (Section 2.3) can be kept at fixed values within a multi-run solution, and for this reason the multi-run capability is only supported by Faust for the partition A from (2.21). Because this eliminates the large partitions B, C and F, the dimension of the partition A can be much larger than would be possible in a single run solution.

Orbit integration — Chapter 3

For two reasons, it was decided to base the numerical integration process in Faust on the same Gauss-Jackson algorithm that was used in the SATAN-A package. As will be explained in this Chapter, this algorithm aims specifically at the efficient integration of equations of motion, in which both the velocity and the position are required. Furthermore, there were practical advantages in modifying an available method to the demands of simultaneous solutions rather than employing a new algorithm. The mathematics behind the numerical integrator are virtually unchanged with respect to the process that was implemented in SATAN, but the numerical structures of the integrator and the initialisation process have been completely redesigned to support the multi-arc capability of Faust and the parameter structures of Chapter 2. In addition, the integration process has been enhanced with the possibility to reduce or increase the step size during integration. To compensate for a lack of proper documentation for the integrator, and to explain the above enhancements implemented in Faust, the numerical integration process will be explained in this third Chapter in some detail. In addition, the parallel integration of different satellite orbits will be discussed, as required in the simultaneous solutions described in the previous Chapter. Finally, the overall structure of the solution process in Faust will be described.

3.1 Review of essential theory

All numerical integration methods use quadrature formulae that express the integral of a function f as a weighted sum of n available values of that function :

$$\int_a^b f(x)\,dx = \sum_{i=1}^{n} \gamma_{i,n}\, f(x_i) \tag{3.1}$$

The only differences between the various algorithms are the amount of nodes n, the distribution of the nodes x_i over the axis of the independent variable, and therefore the weights γ for which the equation is valid. Independent of function f, a quadrature forms an expression for the integral of the polynomial F of degree $<n$ that fits through the sample values, such that $F(x_i) = f(x_i)$ in all nodes. This approximating polynomial follows from the well known Lagrange interpolation formula :

$$F(x) = \sum_{i=1}^{n} f(x_i) \frac{(x-x_1)\ \dots\ (x-x_{i-1})(x-x_{i+1})\ \dots\ (x-x_n)}{(x_i-x_1)\ \dots\ (x_i-x_{i-1})(x_i-x_{i+1})\ \dots\ (x_i-x_n)} = \sum_{i=1}^{n} f(x_i) \frac{p_n(x)}{(x-x_i)\bar{p}_n(x_i)} \tag{3.2}$$

The notation $\bar{p}_n(x_i)$ is used here for the polynomial $p_n(x_i)$ from which the zero for $x = x_i$ is omitted. The difference function $g(x) = f(x) - F(x)$ is obviously zero in the nodes, but may have non-zero values in any other point. The integral of the function f is then

$$\int_a^b f(x)dx = \int_a^b [F(x) + g(x)]dx = \sum_{i=1}^{n} f(x_i) \int_a^b \frac{p_n(x)}{(x-x_i)\bar{p}_n(x_i)} dx + \int_a^b g(x)dx \tag{3.3}$$

This notation shows immediately that a quadrature (3.1) will cause an integration error equal to the integral of the difference function $g(x)$, if the weights are computed according to

$$\gamma_{i,n} = \int_a^b \frac{p_n(x)}{(x-x_i)\bar{p}_n(x_i)} dx \tag{3.4}$$

In case that the function $f(x)$ is a polynomial of degree $\leq n-1$, the n sample points will suffice to retrieve the function exactly from the interpolation, hence F will be equal to f in all points, the difference function $g(x)$ is zero for all x and the quadrature turns into an exact expression[1] for the integral. For this reason the integrator is said to be of order $n-1$.

In most cases the weights γ are computed in a more straightforward way than via (3.4), by using expressions in finite differences (differences between the function values in the nodes, differences between these differences, etc. up to a certain order n). The following notations are used in this type of analysis (e.g. Levy & Lessman, 1961) :

- Derivative	$Df_k = f_k'$	(3.5)
- Backward difference	$\nabla f_k = f_k - f_{k-1}$	(3.6)
- n^{th} order difference	$\nabla^n f_k = \nabla^{n-1} f_k - \nabla^{n-1} f_{k-1}$	(3.7)
- Forward difference	$\Delta f_k = f_{k+1} - f_k$	(3.8)
- Expected value [2]	$Ef_k = f_{k+1} = (1+\Delta)f_k = (1-\nabla)^{-1} f_k = e^{hDf_k}$	(3.9)

[1] It should be noted that the finite precision of a computer will always introduce accumulative round-off errors in (3.1), even if the quadrature itself is exact.

[2] The exponential notation is introduced on the basis of a Taylor series expansion for f_{k+1} :

$$f_{k+1} = 1 + hf_k' + h^2 \frac{f_k'^2}{2!} + \dots = 1 + hDf_k + \frac{(hDf_k)^2}{2!} + \dots \equiv e^{hDf_k}$$

- Central difference $$\delta f_k = f_{k+\frac{1}{2}} - f_{k-\frac{1}{2}} = E^{\frac{1}{2}} f_k - E^{-\frac{1}{2}} f_k = 2\sinh\left(\frac{h}{2}D\right) f_k \quad (3.10)$$

- Central average $$\mu f_k = \frac{1}{2}\left(f_{k-\frac{1}{2}} + f_{k+\frac{1}{2}}\right) \quad (3.11)$$

From (3.9) we can obtain what is known as the Newton-Gregory interpolation formula :

$$\begin{aligned} f_k = E^k f_0 = (1+\Delta)^k f_0 &= \sum_{m=1}^{k} \binom{k}{m} \Delta^m f_0 \\ &= f_0 + k\Delta f_0 + \frac{1}{2}k(k-1)\Delta^2 f_0 + \frac{1}{6}k(k-1)(k-2)\Delta^3 f_0 \ldots \end{aligned} \quad (3.12)$$

If f is a polynomial of degree $n-1$, all differences of order $\geq n$ are zero which means that (3.12) becomes a finite series for any k and in fact represents the Lagrangian F in Δ that is equal to f in all points. By integrating each of the terms in this polynomial we again obtain a quadrature expression of the form (3.1), although this time in terms of finite differences :

$$\int_a^b f_k\,dk = \sum_{m=1}^{n} \Delta^m f_0 \int_a^b \binom{n}{m} dk + \int_a^b (f_k - p_n)\,dk \quad (3.13)$$

The definitions (3.6) to (3.8) subsequently enable the development of (3.13) in terms of the function values in the nodes directly, rather than in differences between them. The expressions for powers of the difference operators follow straightforward binomial relations :

$$\begin{aligned} \nabla^2 f_k &= (f_k - f_{k-1}) - (f_{k-1} - f_{k-2}) = f_k - 2f_{k-1} + f_{k-2} \quad (3.14) \\ \nabla^3 f_k &= [(f_k - f_{k-1}) - (f_{k-1} - f_{k-2})] - [(f_{k-1} - f_{k-2}) - (f_{k-2} - f_{k-3})] \\ &= f_k - 3f_{k-1} + 3f_{k-2} - f_{k-3} \end{aligned}$$

or, in general : $$\nabla^n f_k = \sum_{k=0}^{n} (-1)^k \binom{n}{k} f_{n-k} \quad (3.15)$$

Quadratures obtained from (3.13) express a function in terms of forward differences, which is not very practical : at any particular step k, only backward differences are available. Fortunately, many other formulae can be derived from (3.13) by noting that $\Delta^n = \nabla^{-n}$, and that shifting the variable k can be compensated by muliplications with polynomials in the difference operators. To this purpose, the integral over a step h is first written as

$$\frac{1}{h}\int_0^h f(x)\,dx = \frac{1}{h}\int_0^h DD^{-1} f(x)\,dx = \frac{1}{h}\left[D^{-1} f_h - D^{-1} f_0\right] \rightarrow \int_k^{k+1} f_k = \delta D^{-1} f_{k+\frac{1}{2}} \quad (3.16)$$

The final term on the right hand side forms a generalisation to equidistant nodes, implying

$h \equiv 1$. The impractical operator D^{-1} can now be removed by using (3.10) :

$$\delta f_k = 2\sinh\left(\frac{D}{2}\right) f_k \rightarrow D f_k = 2\sinh^{-1}\left(\frac{\delta}{2}\right) f_k \rightarrow$$
$$\delta D^{-1} f_{k+\frac{1}{2}} = \delta\left[2\sinh^{-1}\left(\frac{\delta}{2}\right)\right]^{-1} f_{k+\frac{1}{2}} \qquad (3.17)$$

This expression - originally due to Gauss - can subsequently be developed into a series expansion in terms of central differences and averages, as done by Todd (1962) :

$$\frac{1}{h}\int_0^h f(x)\,dx = \left(1 - \frac{1}{12}\delta^2 + \frac{11}{720}\delta^4 - \dots\right)\mu f_{\frac{1}{2}} \qquad (3.18)$$

This formulae forms the basis for a wide range of numerical integrators, with names like Gauss-Gregory, Gauss-Corelli, or indeed the Gauss-Jackson integrator used by Faust.

For the purpose of orbit determination, we need to solve the sets of second order differential equations (2.49) and (2.51), each scalar component of which has the general form

$$\ddot{y} = f(\dot{y}, y, t) \qquad (3.19)$$

From (3.19) we want to determine the value of $y\,(t)$ and/or the value of $\dot{y}\,(t)$. These values can be computed from a set of earlier values for $\ddot{y}$ if a suitable quadrature formula is used as a predictor. Lambert (1973) showed that errors caused by using Lagrangian interpolators as extrapolators just outside their enclosed interval are of the same order as the interpolation errors within the interval. In many cases the predictor will be followed by a corrector to improve integration accuracy : once that values for $y\,(t)$ and $\dot{y}\,(t)$ are known from prediction, the function $\ddot{y}(\dot{y}, y, t)$ can be evaluated in the end point of the interval, which means that this end point can then be used as a node in a new quadrature expression to obtain more accurate values for the predicted integrals $y\,(t)$ and $\dot{y}\,(t)$.

3.2 The Gauss-Jackson integrator

The exercise of computing the various weights for the Gauss-Jackson integrator will not be repeated here, but the way in which these coefficients are obtained is summarised in order to allow future users to understand the numerical process, and modify the integrator if necessary. The integration scheme was first described by Jackson (1924) notably for applications in celestial mechanics. It was developed especially for the integration of equations of motion like (2.49), and includes separate quadratures for the first integral (velocity) and for the second (position). For the single integration the Gauss series (3.18) up to order 8 is merely shifted into an expression in backward differences and then rewritten in terms of previous function values to arrive at a quadrature of the form (3.1). For the double integration the Gauss formula is applied to itself, i.e. analytical expressions derived from the first quadrature are used as the function values in the nodes of a second evaluation of the quadrature, differenced, and rearranged into a new series expansion in terms of differences. The resulting series expansion in central differences is given by Buckingham (1957) as :

$$y_{k+1} = h^2 [\, \delta^{-2} + \frac{1}{12}\delta^0 - \frac{1}{240}\delta^2 + \frac{31}{60480}\delta^4 - \frac{289}{3628800}\delta^6 + \frac{317}{22809600}\delta^8 - \dots \,]\, f_{k+1} \tag{3.20}$$

This expression is now manipulated into a quadrature formula. From (3.10) we obtain

$$\begin{aligned} \delta^{2n} f_{k+1} &= E^{-n}(E-1)^{2n} f_{k+1} = E^{-n}(E-1)^{2n} E^1 f_k \\ &= E^{-n+1}(E-1)^{2n} f_k = (1-\nabla)^{-n+1}(1-\nabla-1)^{2n} f_k \\ &= (1-\nabla)^{-n+1}\nabla^{2n} f_k \end{aligned} \tag{3.21}$$

With this general relation, polynomials can be developed in terms of backward differences of f_k for any power δ^{2m}. Ignoring the differences of order > 8, we find

$$\delta^0 f_{k+1} = \left(1 + \nabla + \nabla^2 + \nabla^3 + \nabla^4 + \nabla^5 + \nabla^6 + \nabla^7 + \nabla^8 + \dots\right) f_k \tag{3.22}$$

$$\delta^2 f_{k+1} = \left(\nabla^2 + 2\nabla^3 + 3\nabla^4 + 4\nabla^5 + 5\nabla^6 + 6\nabla^7 + 7\nabla^8 + \dots\right) f_k \tag{3.23}$$

$$\delta^4 f_{k+1} = \left(\nabla^4 + 3\nabla^5 + 6\nabla^6 + 10\nabla^7 + 15\nabla^8 + \dots\right) f_k \tag{3.24}$$

$$\delta^6 f_{k+1} = \left(\nabla^6 + 4\nabla^7 + 10\nabla^8 + \dots\right) f_k \tag{3.25}$$

$$\delta^8 f_{k+1} = \left(\nabla^8 + \dots\right) f_k \tag{3.26}$$

If these expressions are substituted into the series (3.20), the result is a quadrature formula in terms of backward differences, rather than central differences :

$$y_{k+1} = h^2 [\, \nabla^{-2} + \frac{1}{12}\nabla^0 + \frac{1}{12}\nabla^1 + \frac{19}{240}\nabla^2 + \frac{18}{240}\nabla^3 + \frac{4315}{60480}\nabla^4 + \frac{4125}{60480}\nabla^5 + \frac{237671}{3628800}\nabla^6 + \frac{229124}{3628800}\nabla^7 + \frac{9751299}{159667200}\nabla^8 + \dots \,]\, f_k \tag{3.27}$$

The term ∇^{-2} is the second *sum* of the function values, found by accumulating values y_k to form the first sum, and then accumulating this first sum to form the second sum.

The series expansion (3.20) is also used for an eventual corrector step, but the polynomials (3.22) to (3.26) are then written in terms of the new value f_{k+1} (as computed on the basis of the predicted values for y_{k+1}), rather than in terms of f_k. As mentioned before, shifting the time variable forward is equivalent to increasing the order of each backward difference term :

$$\begin{aligned} \delta^{2n} f_{k+1} &= E^{-n}(E-1)^{2n} f_{k+1} = E^{-n}(E-1)^{2n} E^0 f_{k+1} \\ &= E^{-n}(E-1)^{2n} f_{k+1} = (1-\nabla)^{-n}(1-\nabla-1)^{2n} f_{k+1} \\ &= (1-\nabla)^{-n}\nabla^{2n} f_{k+1} \end{aligned} \tag{3.28}$$

Due to the multiplication with ∇^{2n}, all exponentials in the polynomials (3.22) to (3.26) are increased by 2 to obtain the equivalent expansions in f_{k+1}. Substituting those polynomials in (3.20) and recombining terms results in the corrector formula

$$\begin{aligned} y_{k+1} = h^2\,\nabla^{-2} f_k + h^2\,[\frac{1}{12}\nabla^0 - \frac{1}{240}\nabla^2 - \frac{1}{240}\nabla^3 - \frac{221}{60480}\nabla^4 \\ - \frac{190}{60480}\nabla^5 - \frac{9829}{3628800}\nabla^6 - \frac{8547}{3628800}\nabla^7 - \frac{330157}{159667200}\nabla^8 + \ldots\,]\,f_{k+1} \end{aligned} \tag{3.29}$$

The corrector is only used for the position, not for the variational partials. As mentioned in Section 2.5, the latter are not needed to greater precision than in the form of a first order approximation, due to the linearisations (2.2) that take place within the least-squares process. The only reason for using a predictor of order 8 in this case is to prevent the integration process from diverging too rapidly.

A predictor for the velocity $\dot{y}_{k+1}$ can be obtained from manipulations of the Gauss formula (3.18) directly, in a way similar to that described above. If the expression for central differences is converted into one for backward differences, we obtain

$$\begin{aligned} \dot{y}_{k+1} = h\,[\nabla^{-1} + \frac{1}{2}\nabla^0 + \frac{5}{12}\nabla^1 + \frac{3}{8}\nabla^2 + \frac{251}{720}\nabla^3 + \frac{95}{288}\nabla^4 \\ + \frac{19087}{60480}\nabla^5 + \frac{5257}{17280}\nabla^6 + \frac{1070017}{3628800}\nabla^7 + \frac{2082753}{7257600}\nabla^8 + \ldots\,]\,f_k \end{aligned} \tag{3.30}$$

As already mentioned in Chapter 2, the acceleration $\ddot{y}$ is only weakly dependent on the velocity $\dot{y}$. The predictor of order 8 for the velocity is therefore assumed to be sufficently accurate by itself and not followed by a corrector step.

An estimate for the integration error can be obtained by analysis of the general expressions (3.3) and (3.12). From the Lagrangian quadrature we know that the integration error is equal to the integral of the difference function $g(x)$, i.e. the part of the integrated function f that can not be accounted for by n-point interpolation. In case of a finite difference method, ignoring differences of order ∇^{n+1} implies that the series (3.12) is truncated as from that term. The integration error in (3.13) is equal to the integral of the ignored part of the series. Because the differences ∇^n tend to decrease with increasing order n, a first order estimate for the truncation error in the Gauss-Jackson integrator of order 8 can be obtained by computing the contributions to the quadrature from differences of order 9. The coefficients for ∇^9 in each of the polynomials for δ^{2n} are directly available for the corrector (3.29), by using the conclusion from (3.28) that these coefficients are identical to those for ∇^7 in the polynomials for the predictor, that is : 1, 6, 10 and 4 in the polynomials for δ^2 to δ^8 respectively. Substituting these contributions in the expression (3.20) and recombining terms, we obtain the first omitted term of the series (3.29), i.e. the error estimate :

$$\epsilon_{k+1} = -\frac{292524}{159667200} h^2 \nabla^9 f_{k+1} \tag{3.31}$$

For the error estimate in the predictor, the polynomials (3.22) and further must be extended to the term with ∇^9, providing coefficents 1, 8, 21, 19 and 5 respectively. Analogous to (3.31) we find for the truncation error in the predictor (3.27) :

$$\zeta_{k+1} = \frac{4671}{78848} h^2 \nabla^9 f_{k+1} \tag{3.32}$$

Because the correction only (slightly) modifies the value of f_{k+1}, while all other terms in the polynomial for ∇^9 remain the same, the ratio between the error estimates for the predictor and corrector respectively is more or less a constant, as computed from (3.31) and (3.32).

The final stage in the derivation of a quadrature of the form (3.1) is now to express the relations (3.27) and (3.29) to (3.31) in terms of previous values of the function f, rather than in backward differences, using the binomial relations (3.15). The weights for these quadratures are listed in Table 3.1 below for the predictor and corrector of the double integrator, for the predictor of the single integrator, and for the error estimate (3.31). The error estimate for the predictor is not used by Faust, and therefore not tabulated.

In order to compute the state vectors and variational partials at the times of the observations, the 10 most recent vectors are kept in memory buffers, and the values at arbitrary times are obtained from a Lagrange interpolation as soon as the relevant time is positioned between the two central values of the ten buffers.

$$A = B + C . \sum_{j=-8}^{+1} D_j f_{k+j}$$

		Single integrator	Error estimate	Double integrator	
		PREDICTOR	CORRECTOR	PREDICTOR	CORRECTOR
A		y_{k+1}	ϵ_{k+1}	y_{k+1}	y_{k+1}
B		$h\nabla^{-1}f_k$	0	$h^2\nabla^{-2}f_k$	$h^2\nabla^{-2}f_k$
C		$\frac{h}{159667200}$	$\frac{292524}{159667200}h^2$	$\frac{h^2}{159667200}$	$\frac{h^2}{159667200}$
D	f_{k+1}	0	1	0	9751299
	f_k	506432234	-9	103798439	16036748
	f_{k-1}	-1803461924	36	-385853488	-34806724
	f_{k-2}	4047057036	-84	867424848	48315732
	f_{k-3}	-5955502036	126	-1274515624	-45851950
	f_{k-4}	5888502400	-126	1258146350	29482676
	f_{k-5}	-3896485164	84	-831418464	-12309348
	f_{k-6}	1661115764	-36	354064088	3017324
	f_{k-7}	-413645276	9	-88091848	-330157
	f_{k-8}	45820566	-1	9751299	0

Table 3.1 The quadrature weights for the Gauss-Jackson integrator of order 8

3.3 Initialisation of the integrator

The Gauss-Jackson integrator uses a variety of previously calculated quantities in order to compute a single integration step. In particular, to compute the position, velocity and variational partials for step $k+1$, values are required for

(1) The accelerations f_{k-8} to f_k for each satellite in the solution

(2) The positions y_{k-9} to y_k and velocities $\dot{y}_{k-9}$ to $\dot{y}_k$ for each satellite

(3) The first and second sums $\nabla^{-1}f_k$ and $\nabla^{-2}f_k$ for each satellite

(4) The variational equations (2.52) for all relevant parameters (of which there might be thousands), these for the 9 previous steps $k-8$ to k

(5) All variational partials for steps $k-9$ to k

(6) The second sums for all variational equations (hence also the first sums, which are needed to compute the second sums)

These values are stored in circular buffers in which it is not the data that is rotated in each step, but only a pointer variable that provides the starting index within the buffer. Like with all multi-step integrators, however, the first steps of the integration process do not have sufficient previous values at their disposal to apply the quadratures directly and must therefore be obtained by different means. In the orbit determination process the six elements of the initial state vector will generally be among the estimated parameters, so we can assume that values for y_1 and $\dot{y}_1$ are given (hence f_1 can be evaluated). This Section will summarise how accurate values for the 9 successive steps are derived.

In analogy to (3.22) and further, each term δ^{2n} of (3.20) can be expressed in terms of forward differences. Combining (3.9) and (3.21) provides :

$$\begin{aligned}\delta^{2n}f_{k+1} &= (1-\nabla)^{-n+1}\nabla^{2n}f_k = (1+\Delta)^{+n-1}(1+\Delta-1)^{2n}f_k \\ &= (1+\Delta)^{n-1}\Delta^{2n}f_k\end{aligned} \tag{3.33}$$

It will be clear that in this way we can derive the equivalents of the predictors and correctors from Section 3.2 in terms of forward differences; e.g. the double intregrator becomes

$$\begin{aligned}y_{k+1} &= h^2\Delta^{-2}f_{k+1} + h^2\,[\,\frac{1}{12}\Delta^0 - \frac{1}{240}\Delta^2 + \frac{31}{60480}\Delta^4 - \frac{31}{60480}\Delta^5 \\ &\quad - \frac{1571}{3628800}\Delta^6 - \frac{1282}{3628800}\Delta^7 + \frac{45911}{159667200}\Delta^8 + \dots\,]\,f_{k+1}\end{aligned} \tag{3.34}$$

Approximated values for the first and second sums in step 2 of the integration are obtained from the available values for step 1, by truncating the series after the very first terms :

$$\begin{aligned}\dot{y}_1 &\approx h\left(\Delta^{-1}f_2 - \frac{1}{2}f_1\right) \quad \Rightarrow \quad \Delta^{-1}f_2 \approx \frac{\dot{y}_1}{h} + \frac{1}{2}f_1 \\ y_1 &\approx h^2\left(\Delta^{-2}f_2 + \frac{1}{12}f_1\right) \quad \Rightarrow \quad \Delta^{-2}f_2 \approx \frac{y_1}{h^2} - \frac{1}{12}f_1\end{aligned} \tag{3.35}$$

With these sums, we can subsequently compute first order approximations for all values y_k,

$\dot{y}_k$, f_k, $\Delta^{-1} f_k$ and $\Delta^{-2} f_k$ for the steps 2 to 9 by applying the following scheme 8 times :

$$\begin{aligned} \Delta^{-2} f_{k+1} &= \Delta^{-2} f_k + \Delta^{-1} f_k \\ \dot{y}_k &\approx h \Delta^{-1} f_k \\ y_k &\approx h^2 \Delta^{-2} f_{k+1} \\ f_k &= f\left(t_k, y_k, \dot{y}_k\right) \\ \Delta^{-1} f_{k+1} &= \Delta^{-1} f_k + f_k \end{aligned} \tag{3.36}$$

Unfortunately the above first order values for y_k and $\dot{y}_k$ are far less accurate than those of the 8-order Gauss-Jackson integrator itself, and as the errors induced by this low accuracy would inevitably propagate through the entire integration process, the resulting orbits and partials would suffer similar low precision. Improved accuracy can be obtained by applying quadrature formulae of the form (3.1) for each of the points 2 to 9 (using the points 1 to 9 as nodes) now that initial estimates for all f_k are available. To start with, the values for the first and second sums in step 2 are improved by using the quadrature expressions for step 1 in a reversed form, namely by computing the sums from

$$\begin{aligned} y_k &= h^2 \Delta^{-2} f_{k+1} + h^2 \sum_{i=1}^{9} W_{i,k} f_i \quad \rightarrow \quad \Delta^{-2} f_2 = \frac{y_1}{h^2} - \sum_{i=1}^{9} W_{i,1} f_i \\ \dot{y}_k &= h \Delta^{-1} f_{k+1} + h \sum_{i=1}^{9} V_{i,k} f_i \quad \rightarrow \quad \Delta^{-1} f_2 = \frac{\dot{y}_1}{h} - \sum_{i=1}^{9} V_{i,1} f_i \end{aligned} \tag{3.37}$$

Starting from these new sums for step 2 the scheme (3.36) is repeated, with the only difference that the position and velocity are now obtained from quadratures of order 8 rather than just of order 1. This more accurate computation scheme will result in better values for all y_k, $\dot{y}_k$, and f_k for steps 2 ... 9, and by iterating this improvement process until sufficient convergence is reached we end up with values that apparantly satisfy the various 8-order quadratures in the scheme, and hence are compatible with the main integrator.

The 8 required quadratures for the double integrator (position and variational equations) are all obatained from (3.20) by expressing the terms $\delta^{2n} f_k$ in terms of forward differences $\Delta^n f_1$, this for every step from 2 to 9. The series expansions in $\Delta^n f_1$ are then converted to standard quadratures of the form (3.1) by developing polynomials like (3.22) to (3.26) but now in terms of *forward* differences, and substituting these polynomials into the series for central differences. Such manipulations have been illustrated in the previous Section where the Gauss-Jackson integrator was described, and will not be repeated here for each of the 8 quadratures; the occurring weights are listed in Table 3.2 below.

i	1	2	3	4	5	6	7	8	9
$W_{i,1}$	9751299	16036748	- 34806724	48315732	-45851950	29482676	-12309348	3017324	- 330157
$W_{i,2}$	- 330157	12722712	4151096	- 7073536	6715950	- 4252168	1749488	- 423696	45911
$W_{i,3}$	45911	- 743356	14375508	294572	- 1288750	931164	- 395644	96692	- 10497
$W_{i,4}$	- 10497	140384	- 1121248	15257256	- 1028050	33872	49416	- 17752	2219
$W_{i,5}$	2219	- 30468	220268	- 1307644	15536850	- 1307644	220268	- 30468	2219
$W_{i,6}$	2219	- 17752	49416	33872	- 1028050	15257256	- 1121248	140384	- 10497
$W_{i,7}$	- 10497	96692	- 395644	931164	- 1288750	294572	14375508	- 743356	45911
$W_{i,8}$	45911	- 423696	1749488	- 4252168	6715950	- 7073536	4151096	12722712	- 330157
$W_{i,9}$	- 330157	3017324	- 12309348	29482676	-45851950	48315732	-34806724	16036748	9751299
$V_{i,1}$	- 45820566	- 94047140	153921548	-198129492	182110720	-115111084	47557620	-11575388	1260182

Table 3.2 Numerators of the quadrature weights in the double integrators for the position and the variational partials in (3.37). All denominators are 159667200.

With respect to the variational partials it should be noted that these partials express how the integrated position changes if the parameters change (see Section 2.5). At the start of the integration, when the 'integrated position' is identical to the initial state vector, this position is independent of all other a priori parameter values, so at the start of the integration all variational partials are by definition zero (apart from those for the initial values of x, y and z, which are 1). Because of this, the buffer values and first and second sums for the variational partials are completely detached from anything that took place before the starting epoch, which will be identified as a potential problem in the next Section.

As mentioned before, we do not really need the high precision of order 8 for the integration of the velocity, and in the iterative improvement stage described above the applied quadratures for the single integrator are only of order 4. For the velocity quadrature of step 1 in (3.37) however - i.e. the one used for the improvement of the first sum in step 2, therefore being relevant to the double integrator - a full 8-order quadrature is applied, to avoid loss of precision at the very beginning of the integration process. The coefficients $V_{i,1}$ for that quadrature are listed in the last row of Table 3.2. For the other velocity quadratures - i.e. those that are used in the iterative improvement scheme - series expansions of order 4 in forward differences are derived from the basic integration formula (3.18), and subsequently converted into standard quadratures (3.1) in the usual way. The weights for these quadratures are listed in Table 3.3. Note that the bands of zeroes appear in the table because these series are of order 4, and are only tabulated in the same format as Table 3.2 to show that the nodes i in these 8 series are not the same for each quadrature.

i	1	2	3	4	5	6	7	8	9
$V_{i,2}$	27	830	-192	66	-11	0	0	0	0
$V_{i,3}$	-11	82	720	-82	11	0	0	0	0
$V_{i,4}$	0	-11	82	720	-82	11	0	0	0
$V_{i,5}$	0	0	-11	82	720	-82	11	0	0
$V_{i,6}$	0	0	0	-11	82	720	-82	11	0
$V_{i,7}$	0	0	0	0	-11	82	720	-82	11
$V_{i,8}$	0	0	0	0	11	-66	192	610	-27
$V_{i,9}$	0	0	0	0	-27	146	-336	462	475

Table 3.3 Numerators of the quadrature weights in the integrators for the velocity in (3.37). All denominators are 1440.

3.4 Changing the step size

The quadratures from Sections 3.2 and 3.3 are only valid as long as the integration step size h is constant, because of the expression (3.16) where the continuous variable x from the Lagrangian methods is replaced by the equidistant counter variable k. Although this leads to straighforward quadrature expressions in integer numbers - and efficient multi-step integrators that only require one function evaluation per step - a problem arises if for any reason it is desirable to increase or decrease the step size. One such reason is the occurrence of orbital manoeuvres, which take place regularly during most satellite missions in order to maintain the orbit within the limits prescribed by mission requirements, or to change the orbital characteristics altogether. In general, a manoeuvre involves one or more short bursts of the satellite's thrusters, with substantial impact on the velocity vector. In nominal operation (generally for low-Earth satellites in near-circular orbits) Faust uses a step size of 30 seconds, which has proved to be an adequate choice in terms of integration accuracy, force model resolutions and processing speed. A thruster pulse, however, may only last several seconds, which makes it impossible to include it realistically in the acceleration model f_k for this nominal step size. For such cases it is useful to temporarily reduce the step size to second or sub-second level, then integrate through the interval with strong fluctuations in the function f_k, and afterwards return to the nominal step size.

The main problem related to a change in step size is the fact that the integration buffers (as listed at the start of the previous Section) are based on the given constant step size, which implies that none of the Gauss-Jackson quadratures can be continued directly if the step size is reduced. At first sight, we might be tempted to simply repeat the starting routine from Section 3.3 to initialise the integration interval with a new step size. Unfortunately, this would imply that all variational equations are interrupted and no longer relate to any parameter values from before the change in step size, which interferes fundamentally with the parameter estimation process of Chapter 2. A different approach is therefore required.

As explained in Section 3.2, the state vector and partials are obtained for any arbitrary time t by a 10-point Lagrange interpolation, which is applied when the integration process has reached the first time step beyond $t_{int} + 4h$. In that way, the time t_{int} for which we want to interpolate is positioned between the two central nodes of the 10 buffer values, i.e. at the place where the Lagrange interpolator is most reliable as shown by Todd (1962). For reducing the integration step size, the same interpolation technique can be used. If the start of the interval with reduced step size is indicated by t_s, and the old and new (smaller) step size are denoted by h_1 and h_2 respectively, the required integration buffers can be obtained in the following way :

(1) We impose the condition that the integration interval with new step size h_2 always starts at integer boundaries of the old step size h_1, in particular at the value $t_s = t_{k-5}$ in terms of h_1 (if t_k corresponds to the actual time reached by the integration process). The 10 new integration buffer values are now obtained by *storing* the values for $t = t_s$ (they already exist, as the buffer values for t_{k-5}) and *interpolating* 9 additional values for $t = t_s + h_2$ ... $t = t_s + 9h_2$. To make sure that all these 9 values are within the two inner nodes of the interpolation (so that we can do all interpolations without the need for additional integration steps between the interpolations) we add a second condition, namely that h_2 is at least 10 times smaller than h_1. This is illustrated in Figure 3.1.

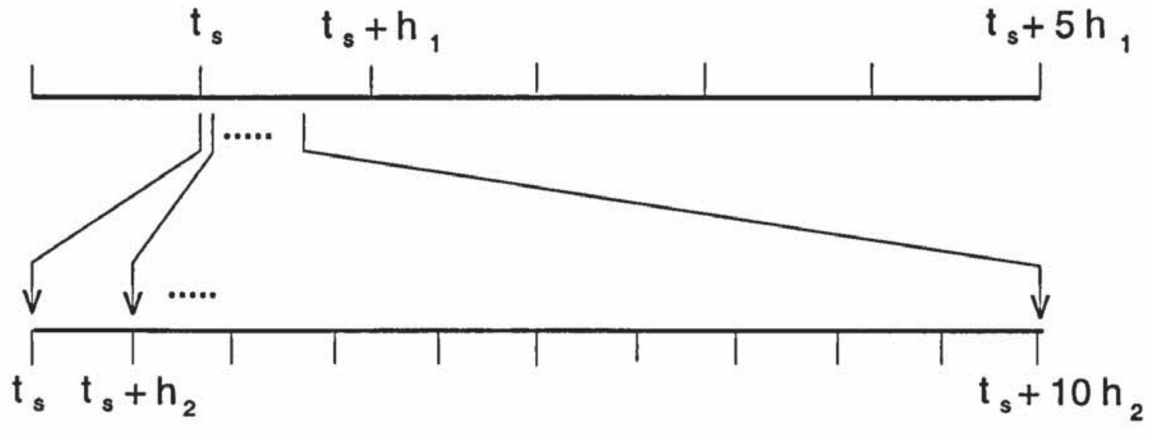


Figure 3.1 Reduction of the step size by interpolation of new integration buffers

(2) The first and second sums can not be obtained from interpolation, as there are no ringbuffers for these values. However, using the quadrature for the predictors in the Gauss-Jackson integrator, we can compute the first and second sums for t_s+9h_2 by inverting the quadrature for the next step in a way similar to (3.37) :

$$\nabla^{-1} f_k = \frac{\dot{y}_{k+1}}{h} - \sum_{i=-8}^{0} \gamma_i f_{k+i} \tag{3.38}$$

$$\nabla^{-2} f_k = \frac{y_{k+1}}{h^2} - \sum_{i=-8}^{0} \tilde{\gamma}_i f_{k+i} \tag{3.39}$$

To this purpose, we interpolate additional values for the position, velocity and variational partials at the time $t = t_s+10h_2$, to serve as the values y_{k+1} and $\dot{y}_{k+1}$ above. For the first sum of the variational equations, no expression for the single integrator is applied (although the quadrature for the velocity could of course be used), mainly because in the Gauss-Jackson integrator we do not use one either. Instead, the second sum for step t_s+8h_2 is also computed from (3.39), after which it is subtracted from the value of the second sum for t_s+9h_2. The obtained difference is the first sum $\nabla^{-1} f_{t_s+9h_2}$.

The Gauss-Jackson integration process can now be continued with the small step size, the first step to be computed being the one for time $t = t_s+10h_2$. By definition of the first and second sums in (3.38) and (3.39), the result of the predictor in that first step will be exactly identical to the values that were obtained earlier by interpolation between the buffers for the larger step size, which implies that at that point it is no longer possible to distinguish between the values obtained from interpolation (without reduction of order) and values that would have been obtained by integration with the smaller step size.

The reversed process of increasing the step size must take place in a different manner, as we can not interpolate 10 new buffer values for h_1 between nodes in the smaller step size. However, no interpolation is required at all if a method is chosen that doubles the step size in stages. Now, the time from which we wish to integrate with a new step size is denoted as t_t, and the new (larger) step size is indicated as h_3. The process is as follows :

(1) The integration with the small step size is paused as soon as the time $t_t + 9h_2$ is reached. Now all buffer values are stored for time t_t, and for each second step h_2, corresponding to the values for t_t+2h_2, t_t+4h_2, t_t+6h_2 and t_t+8h_2.

(2) The integration process is continued for 10 more steps h_2, up to the time t_t+19h_2, after which (1) is repeated for these 10 new steps. The sets of buffer values that have been stored by then contain precisely the required 10 integration buffers for the new doubled step size $h_3 = 2h_2$. This is illustrated in Figure 3.2.

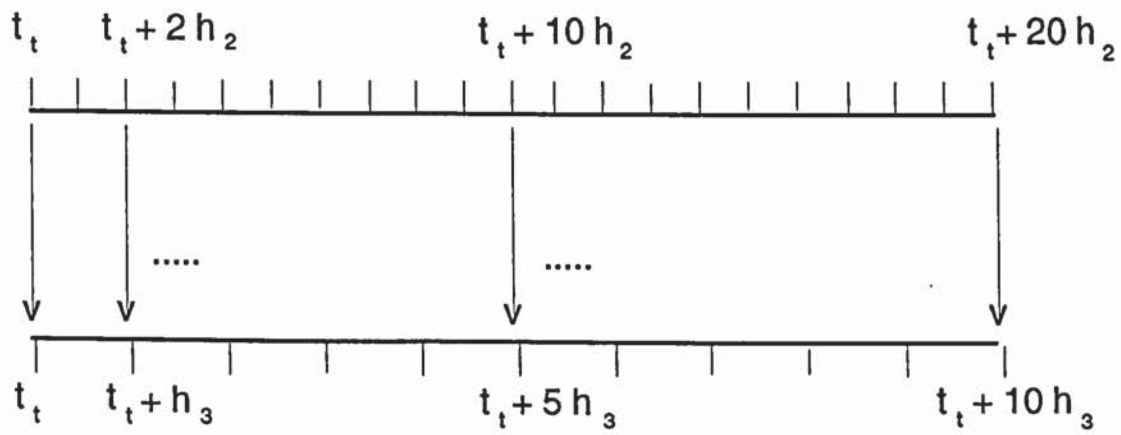


Figure 3.2 Doubling of the step size by storing each second buffer value for 20 steps h_2

(3) The integration is continued for one more step h_2, resulting in values for position, velocity and partials at time $t = t_t+20h_2 = t_t+10h_3$. With these additional values all first and second sums are computed for $t_t + 9h_3$ in exactly the same way as before, i.e. from (3.38) and (3.39).

For practical reasons, we will now impose two more demands upon process of changing the step size. At first, we require that the step size h_2 is a factor 2^m smaller than h_1, so that the step h_1 is restored exactly if the described step doubling process is repeated $m-1$ times. Furthermore, we choose the time t_t to be an integer amount of steps h_1 later than t_s, so that after restoring the original step size h_1 the step boundaries are again in phase with the original step sequence h_1. The latter condition is useful because of certain advantages of having the same step boundaries for parallel orbits, as will be discussed later in this Chapter.

3.5 Structure of the solution process

The predominantly chronological order in which tracking observations are collected, and the sequential character of a numerical integration process in which time is the independent variable, impose a primary structure upon any orbit determination programme that will be time-organised. The existing SATAN-A software package consisted of two main programmes with distinct tasks, namely an orbit integrator to produce the satellite positions and variational partials for the epochs of the observations, and a second programme that would construct and invert the normal matrix. In order to converge a parameter solution, the two programmes would be run alternately. The time span of the orbit arc in the solution is processed twice (namely first by the integrator, then by the processing of the observations), and this approach will be referred to as a double-loop structure. Because of the two separate processing loops all state vectors and partials produced in the first loop must be temporarily stored, only to be discarded again after completion of the second loop.

Two disadvantages of the single-arc / double-loop process in SATAN are the substantial demands with respect to temporary data storage and the relatively low level of automation, problems that increase proportionally to the amount of data and the amount of satellite arcs in the solution. For large parameter estimation tasks (e.g. for a full simultaneous gravity field solution, producing many thousands of variational partials for each observation) the data storage requirements make a double-loop structure practically impossible. The problem of a low level of automation will mainly result in more work for the user of the software, which will of course also become more pronounced in case of multi-arc / multi-satellite solutions.

If the two loops are combined into a single one, each observation is processed as soon as its state vector and variational partials have been determined by numerical integration, which means that they do not have to be stored at all. As mentioned above, for large parameter tasks this single-loop structure will be the only possibility, and already for that reason this approach was selected as the basic structure for Faust. To avoid repetitive runs towards convergence, the entire single-loop structure of Faust is embedded in an automatic iteration process, controlled by various convergence criteria specified by the user. In order to allow simultaneous multi-arc solutions with sufficient crossover data density - in particular, to allow partial structures like (2.55) between consecutive arcs or arcs for different satellites - the single-loop structure is not restricted to one orbit arc at a time. For calculated observations that include data from two or more epochs, the contributions of the first epoch will be temporarily stored until all required state vectors and partials are available, and as soon as the related equations have been added to the normal matrix the occupied disk space is released again to minimise the need for temporary data storage. The manipulation of these buffered state vectors and partials will be discussed in greater detail in Chapter 5.

The above description shows that, while the central process moves forward in time, many different tasks may have to be performed, like processing an observation, storing or retrieving a set of variational partials, storing or retrieving a normal matrix from disk, starting or finishing an orbit arc within the multi-arc process, etcetera. On this basis, the time-organised loop in Faust has been set up as an explicit 'event handler' that processes any of a variety of tasks as soon as they occur, i.e. in strict chronological order. The overall concept is illustrated in Figure 3.3.

The inner loop in the Figure represents the time-organised event handling step, and will be executed once for each of many thousands of 'events' that may occur within a certain solution process.

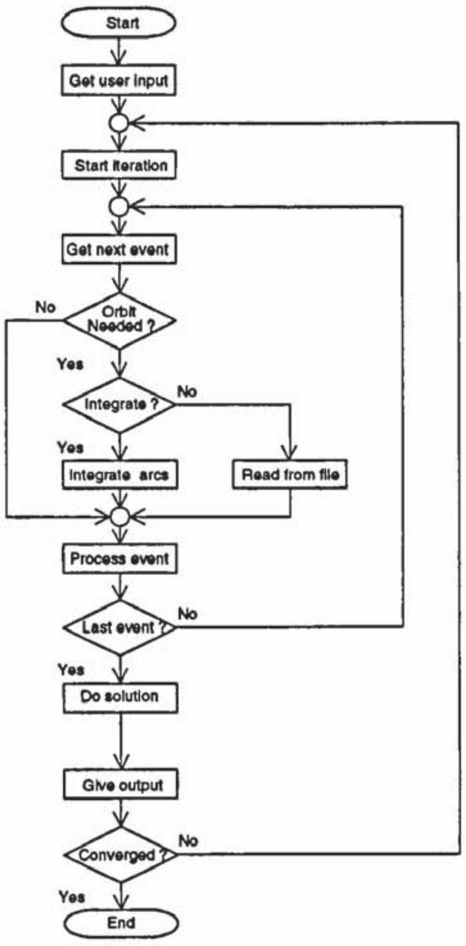


Figure 3.3 Processing sequence of Faust

The outer loop represents the iteration process towards convergence of the parameter solution, and will only be executed a limited number of times (or perhaps just once) during a particular run. Most of the occurring events will require the availability of one or more satellite state vectors and sets of variational partials. In the single loop structure, these will in principle be determined in strict chronological order while proceeding from the start to the end of the entire solution period, for several satellite orbits in parallel and / or in succession.

The event handler structure itself resembles the working of a computer programme, proceeding from one command statement (event) to the next, according to the intentions of the programme. In analogy, the user interface that is discussed in Appendix C contains the programming language to specify, at a high level, the sequences of events that should be handled for a particular application. The design of the code as an explicit event handler offers substantial freedom for future extensions without the need to modify the central structure. Faust should therefore be considered as a software platform that can be extended with arbitrary modules, rather than as a single purpose programme. Most of the main modules like input handling, the parameter structures, the orbit integrator and all normal matrix management, can be used in different combinations and for different purposes by setting up a new input file that results in a different sequence of events.

As an example, the recombination of earlier computed normal matrices into a multi-run solution is also handled by Faust. In that case, the total parameter estimation task will still be specified in the usual way, but with a few extra input commands to indicate that the programme should not compute a normal matrix internally but rather read previously generated normal matrix files. Most parts of the code, like input handling, the inversion of the matrix, the handling of weights and constraints, producing output statistics, etcetera, will be used in exactly the same way as would be done for a normal parameter estimation run.

The double-loop structure of SATAN allows one option that would not normally be included in the basic single-loop programme, namely the fine tuning of a parameter estimation task by simply generating different normal matrices that all use the *same* set of a priori parameter values, but different weights, constraints or data editing rules. In the adopted single-loop structure this would require integration of the same orbit ephemerides and variational partials for each different run, which is not very efficient. For this reason, an option is included in Faust to still generate an ephemerides file (unless, of course, such a file would become prohibitively large) and use this file as input ephemerides for several additional runs, each of which will then be very fast because the time-consuming numerical integration process is eliminated. This effectively implies that Faust can operate either in a single-loop mode or in a double-loop mode. The optional use of ephemerides files is also indicated in Figure 3.3, as an alternative to the numerical integrator for obtaining the state vectors and partials at the times of the events.

3.6 Parallel integration of satellite orbits

The strictly chronological processing sequence in Faust implies that in simultaneous orbit determination runs the orbits and variational equations for two or more satellites may have to be integrated at the same time. If the integrator is used - i.e. if the ephemerides are not available from file, or during any process iteration other than the first - the program will integrate from one event to the next. If several orbits are integrated in parallel, the integration time t_k will always be held the same for all these parallel orbits, even if only one of the orbits is affected by the event. This principle of integrating the arcs in parallel rather than sequential is illustrated in Figure 3.4.

The approach of keeping parallel arcs at exactly the same internal time, combined with the mapping arrays described in Section 2.6, has several distinct advantages. The Gauss-Jackson integrator is implemented in the form of a single subroutine (marked G-J in Figure 3.4) that will take one integration step for just one of the included satellite arcs. Together with the arrays for parameters and partials, the mapping arrays from Section 2.6 that belong to that particular arc are passed to this routine as input variables. Within the integration routine the mapping arrays are treated as a two-dimensional array, eliminating the last index of array M_P (note that in FORTRAN arrays are stored linearly in

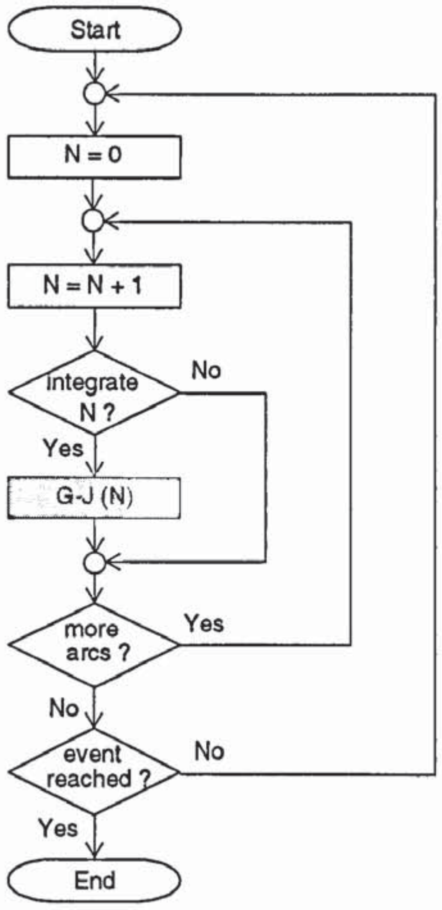


Figure 3.4 Parallel integration of all relevant arcs, with index N running from 1 to the total amount of arcs in the solution.

memory in such a way that the first index increases fastest, the last index increases slowest). This corresponds to passing a single 'sheet' of the array M_P to the integration routine, as illustrated in Figure 3.5. The routine itself, and any lower level routine, will therefore not be aware of the presence of more than one satellite arc at a given time. In order to integrate orbits and variational equations that are parallel in time, we can now call the integration subroutine several times at each time step - once for each orbit - while passing the *same* vector with parameters (of which there is only one) but *different* sheets of the mapping array. The integration routine - or any lower level routine - is organised to access parameters via the particular sheet of M_P that has been passed as input. As a consequence, each satellite arc within the total solution process uses its own ringbuffers for the accelerations, velocities and position. The ringbuffers related to the other parameters and the variational partials, however, are defined only once in the entire programme, since the mapping array will take care of navigating through these arrays.

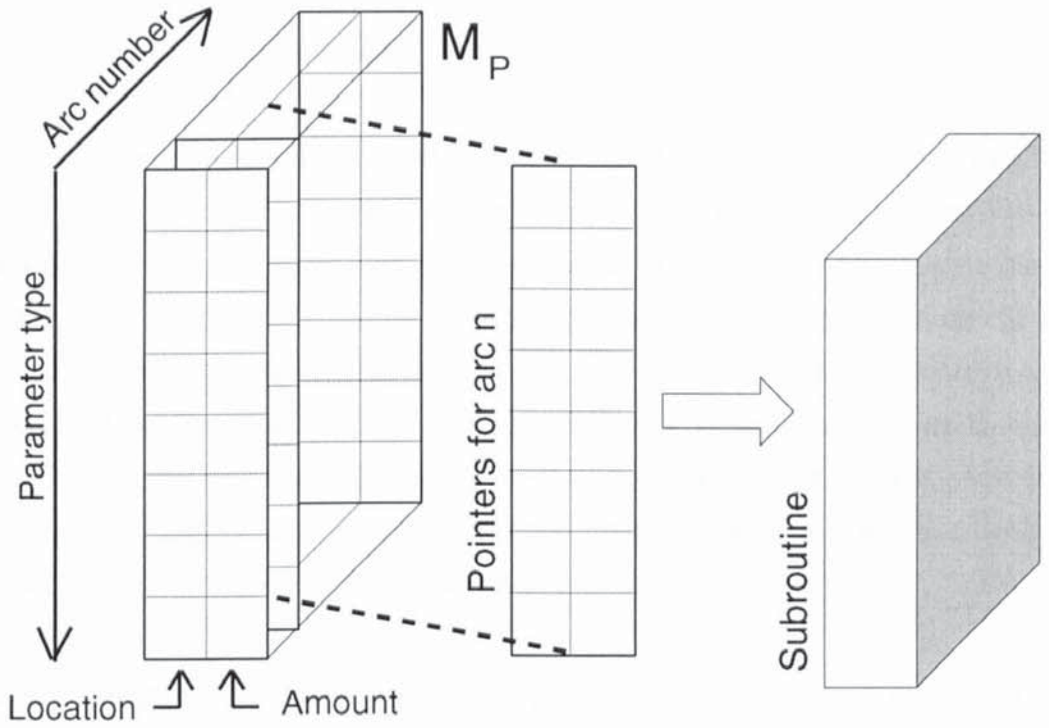


Figure 3.5 The pointer array M_p treated as a set of arrays, one for each arc in the solution.

Because all parallel orbits take the same time step within a single process step, physical quantities that are only dependent on time do not have to be computed more than once, as long as the integration steps of different arcs are in phase. Such quantities include for instance the evaluation of the position of the Sun, Moon and planets and the nutation series for the Earth's attitude in inertial space (as required to compute the employed reference frames, or Earth and ocean tides), which are described in more detail in Appendix A. Due to such centralised computations the single-loop process for a simultaneous solution becomes more efficient than a combination of separate processes for each individual satellite orbit would be. Of course, this increased efficiency only affects relatively small processes (up to several hundreds of parameters). Large parameter estimation processes will always be dominated by the need to construct the normal matrix according to (2.20), involving many millions of floating point operations for each observation O_i.

The initialisation of the integration process for each arc is implemented as one of the 'events' indicated in Figure 3.3. Like the main integrator it is implemented in the form of a one-arc only subroutine, that receives the appropriate sheets of the pointer arrays for parameters and partials as input. Ending an arc is yet another event that mainly involves various housekeeping measures, like the closing of a possible orbit output file.

This modular implementation of arc-related events makes it very easy to include parallel and / or successive arcs within a single estimation process. A transition to a new arc is characterised by a sequence of an 'end arc'-event, followed by a 'start arc'-event for the same satellite, at the same epoch. To the processing of observations that depend on two epochs in successive arcs, it does not make any difference if between these two epochs the satellite has switched from one arc to another or not : the processing of observations is just another independent event in the processing chain. In this way any combination of arcs - with or without gaps between arcs - can be combined into the solution process of Chapter 2, with a maximum that is only determined by easily modified array sizes within the code.

As an example, orbits for the ERS satellites can conveniently be converged in batches of seven 5-day arcs within a single run, covering a full 35-day repeat cycle of the multi-disciplinary mission phase. Parallel orbits for the ERS tandem mission can be converged within a single process for a full 35-day repeat cycle, creating interesting new possibilities like sharing atmospheric drag parameters or solving for differential accelerations between the two satellites. Orbits for multi-satellite constellations like GPS could easily be solved simultaneously, as could orbits for any combination of satellites that employ satellite-to-satellite tracking. It is also possible to solve satellite orbits that are far apart in time, for instance GEOSAT arcs in combination with ERS and TOPEX/Poseidon. The dense networks of crossovers between all these arcs might then help to improve the orbital quality of the GEOSAT arcs, despite of the relatively large noise level in the crossover datasets caused by the considerable sea surface variability between crossings. Within the framework of this study it is in particular the analysis of altimeter crossovers that is of interest, both between parallel orbits for ERS and TOPEX/Poseidon and between successive arcs for each of the satellites, constructing the integral 'measurement device' referred to in Chapter 1.

3.7 Orbit files and ephemeris files

Throughout this thesis, a name convention will be used that distinguishes between 'orbit files' and 'ephemeris files'. In Faust, an orbit file is an output file of the program that will be produced for each arc in the solution for which this is requested in the user input. These files contain data records with a time tag and the six elements of the orbit state vector at that time, being the 3 elements of the Cartesian position and the 3 corresponding elements of the velocity vector. Output orbits will in general be defined in the terrestrial reference frame (see Appendix A for details about reference systems in Faust) which is the most convenient frame for applications of orbit data. The typical interval between successive state vectors in the orbit files will be 30 seconds, which corresponds to the nominal integration step size adopted for the low Earth orbits of the ERS satellites and TOPEX/Poseidon. In case of a temporarily reduced step size, the orbit output records will still be produced at the same constant time intervals that are related to the nominal step size .

Ephemeris files are only of internal interest to the program, and are related to a particular solution process. These files contain the orbit state vectors *and* the variational partials (defined in the J2000 reference frame in which the integration will be performed) but only for the epochs of the events, for the current process. Ephemeris files are used in order to avoid repetitive integrations for the same combination of a priori parameter values, as discussed in Section 3.5. An ephemeris data record is therefore not related to a single *arc* within the solution processs but may contain state vectors and variational partials that relate to more than one arc, if the event in question does. The freedom to change the setup of the process for successive runs but still use the same ephemerides is limited, in the sense that it is always possible to skip certain events (for instance, we might eliminate data from one or more stations), but that it is obviously not possible to add any new events without having used the numerical integrator to obtain the corresponding state vectors and partials.

The force model Chapter 4

In the previous Chapters the equations of motion of the satellite have been mentioned as the basis of the computation of the orbital position and the variational partials, and therefore of the calculated measurements. This Chapter will describe the acceleration models in Faust that together form the function f_k in the numerical integration process. As mentioned in Chapter 1, Faust has derived most of its force models from SATAN-A, although all code has been rewritten from scratch. The coding of the accelerations in the SATAN package had become particularly blurred, as most force models were evaluated within a single subroutine. Apart from the need to rewrite the code in a more systematic way and to incorporate the new structures of parameters and partials in Faust, an overhaul of the force models was also required to upgrade all parts to double precision arithmetic.

In first order approximation the orbit of a satellite around the Earth is an elliptical Kepler orbit dominated by the central body gravity force of the Earth. Early methods for orbit determination exploited the analytical separation between a main force and several small perturbing forces, but with the arrival of fast computers most analytical or semi-analytical methods have been superseded by purely numerical ones. The numerical integration of the acceleration components in x, y and z as implemented in Faust and discussed in Chapter 2 is known as Cowell's method, and is at present the most common approach in satellite orbit determination. It has the advantages of simplicity and general applicability, at the expense of losing analytical insight in the orbit perturbations and not exploiting the knowledge that the motion of the satellite is essentially a perturbed Kepler orbit.

The force models in Faust can be categorised in the three groups of conservative force models, surface force models and empirical force models; relativistic effects of any form are not included. Each model will be discussed with special attention to the partial derivatives required in (2.53). Further emphasis will be on model elements that have been adapted to special applications in multi-arc or multi-satellite solutions, for instance for the possibility to share drag force parameters between arcs.

Physical reality can only be represented by mathematical models after the definition of adequate spatial and temporal reference frames. The systems that are used in the force models are an Earth Centered Fixed frame (ECF), the True Frame of Date (TOD) and a semi-inertial frame (J2000), with UTC as the assumed time standard. The details of these frames, most of which follow internationally standardised concepts, are not discussed in this Chapter. The way in which they are determined in Faust is summarised in Appendix A.

4.1 Gravity

As mentioned above, the dominating acceleration component is the central body gravity of the Earth, but the asymmetrical mass distribution within the planet also causes some of the largest perturbations of the orbit of Earth satellites. The gravitational acceleration that any point mass experiences towards another mass is given by the well-known Newtonian formula

$$\underline{a}_{grav} = \frac{G\,m}{\|\underline{r}\|^3}\,\underline{r} \tag{4.1}$$

with G, m and $\underline{r}$ being the universal gravity constant, the mass of the attracting body, and the vector between the two masses respectively. The relatively close distance between a satellite and the Earth implies that the inhomogeneous planet can not be considered as a point mass. The total gravitational acceleration would be obtained by integrating the accelerations (4.1) caused by all mass-elements dm of the planet, which is hardly usable as a numerical model. More pragmatically, the gravitational acceleration is computed as the gradient of a scalar potential function U :

$$\left\{ \begin{array}{l} dU = -G\dfrac{dm}{\|r\|^2} \\[2ex] U = -G\displaystyle\iiint\limits_{x\,y\,z} \dfrac{dm}{\|r\|^2} \end{array} \right. \quad \rightarrow \quad \underline{a}_{grav} = -\nabla U \tag{4.2}$$

Gravity is the definitive example of a conservative force, and the geopotential U is therefore conveniently described by the general solution of the Laplace equation $\nabla^2 U = 0$. A practical representation of the geopotential is by means of a surface around the Earth that satisfies the condition that U is constant. This surface reflects the global variability of the gravity force by means of its topography, as well as the conservative character of gravity because it is closed and continuous. Because static liquids do not support internal tangential stresses, the constant part of the topography of the Earth's ocean surfaces is approximated by a similar surface - the geoid - at which the effect of the normal acceleration due to the gravity potential U, in combination with the potential due to the Earth's rotation $U_{rot} = -\frac{1}{2}\omega^2(x^2+y^2)$, is constant.

Taff (1985) described comprehensively how the potential equation is solved by separation of variables in the spherical coordinates radius, longitude and latitude, i.e. by assuming a function of the form

$$U = R(r)\,\Lambda(\lambda)\,\Phi(\phi) \tag{4.3}$$

The matching general solution is then given by the spherical expansion series

$$U = -\frac{GM}{r}\left\{1 + \sum_{l=2}^{\infty}\sum_{m=0}^{l}\left(\frac{a_e}{r}\right)^l \left(C_{lm}\cos(m\lambda) + S_{lm}\sin(m\lambda)\right) P_l^m(\sin\phi)\right\} \tag{4.4}$$

The factors C_{lm} and S_{lm} together determine the amplitude and phase of a component of degree l and order m. The associated Legendre functions P_l^m, employed as the orthogonal functions in the expansion series (4.4), are used extensively in the numerical processing of gravity and tides. Some of their properties are used in the remainder of this Chapter and are therefore briefly recalled in Table 4.1. In this thesis, the geopotential model coefficients are always related to the *normalised* Legendre functions, even if for convenience they are simply written as P_l^m.

As is apparent from (4.4), the components of the gravity field for $m = 0$ are independent of the longitude, and are known as zonal harmonics as they divide the globe in latitudinal zones. In an alternative representation, the zonal contributions are sometimes separated from the double summation and included in their own series in l only, to omit the meaningless factors with coefficients S_{l0} by starting the second summation at $m = 1$. In first order approximation, the Earth forms a rotationally symmetrical ellipsoid with semi-major axis a_e equal to the mean equatorial radius of the Earth. From this it becomes clear that the contribution that is linear in $\sin\phi$ (i.e. the zonal C_{20}) must be substantially larger than the other contributions, in order to represent the flattening of the ellipsoid. The components with $0 < m \leq l$ are called tesseral harmonics, while for $l = m$ the part $\Phi(\phi)$ aliases into the part $\Lambda(\lambda)$: the latitude-dependent term can no longer be distinguished from the longitude-dependent part. Such components can therefore be considered independent of latitude, and are known as sectorial harmonics as they divide the globe in longitudinal sectors.

Legendre polynomial of degree l	$P_l(x) = \dfrac{(-1)^l}{2^l\, l!} \dfrac{d^l}{dx^l} (1 - x^2)^l$	(4.5)
Associated functions of order m	$P_l^m(x) = (1 - x^2)^{\frac{m}{2}} \dfrac{d^m}{dx^m} P_l(x)$	(4.6)
Recurrence relations (e.g. Engels, 1980)	$P_0^0(x) = 1$ $P_1^0(x) = x$ $P_{l+1}^m(x) = \dfrac{2l+1}{l-m+1} x P_l^m(x) - \dfrac{l+m}{l-m+1} P_{l-1}^m(x)$ $P_l^{m+1}(x) = \dfrac{1}{\sqrt{1-x^2}} \left((m-l)x P_l^m(x) + (m+l) P_{l-1}^m(x)\right)$	 (4.7) (4.8)
Normalisation	$\bar{P}_l^m(x) = \dfrac{1}{N_{lm}} P_l^m(x)$ $N_{l0} = \sqrt{\dfrac{1}{2l+1}}$ $N_{lm} = \sqrt{\dfrac{(l+m)!}{2(l-m)!\,(2l+1)}}$	(4.9) (4.10) (4.11)

Table 4.1 Properties of Legendre polynomials and functions.

The gravitational acceleration in a point (x, y, z) in the ECF frame follows from (4.2) as the gradient of the potential. In principle, the elements of this vector would be computed from

$$\frac{\partial U}{\partial x} = \frac{\partial U}{\partial r}\frac{\partial r}{\partial x} + \frac{\partial U}{\partial \phi}\frac{\partial \phi}{\partial x} + \frac{\partial U}{\partial \lambda}\frac{\partial \lambda}{\partial x} \tag{4.12}$$

with similar equations for y and z. The involved partial derivatives of spherical coordinates with respect to cartesian coordinates follow from straightforward trigonometry, while the partial derivatives of the geopotential with respect to the spherical coordinates can be computed by differentiating the corresponding components of (4.3) within (4.4). For numerical evaluation of the gravity field, however, an algorithm is commonly used that is more efficient than the standard approach of computing all partials in (4.12) explicitly. This faster algorithm by Müller (1975) creates a shortcut through the chain rule formulations by exploiting the fact that derivatives of sines and cosines are again cosines and sines, while derivatives of Legendre functions can be rearranged in other Legendre functions of lower degree. The analytical expressions for these partial derivatives can then be manipulated into new series expansions of lower degree or order, resulting in recurrence relations between terms in the acceleration components directly. This leads to a useful decrease in the amount of numerical operations for the evaluation of the geopotential. From definition (4.6) for the Legendre functions it follows that

$$P_l^m(\sin\phi) = \cos^m\phi \frac{d^m}{d(\sin\phi)^m} P_l^0(\sin\phi) = \cos^m\phi \, P_l^{(m)}(\sin\phi) \tag{4.13}$$

The notation $P_l^{(m)}$, used here for the m^{th} derivative of the Legendre polynomial $P_l\ (= P_l^0)$, should not be confused with the associated Legendre functions P_l^m of order m. Using (4.12) and the property (4.13), direct expressions for the acceleration components are derived in the form

$$\begin{cases} \ddot{x} = -Ax + B \\ \ddot{y} = -Ay - C \\ \ddot{z} = -Az + D \end{cases} \tag{4.14}$$

with

$$\begin{aligned}
A &= \frac{GM}{r^3}\left(1 - \sum_{l=2}^{\infty} J_l \left(\frac{a_e}{r}\right)^l P_{l+1}^{(1)} + \sum_{l=2}^{\infty}\sum_{m=1}^{l}\left(\frac{a_e}{r}\right)^l P_{l+1}^{(m+1)}\left(C_{lm}\alpha_m + S_{lm}\beta_m\right)\right) \\
B &= \frac{GM}{r^2}\sum_{l=2}^{\infty}\sum_{m=1}^{l}\left(\frac{a_e}{r}\right)^l m P_l^{(m)}\left(C_{lm}\alpha_{m-1} + S_{lm}\beta_{m-1}\right) \\
C &= \frac{GM}{r^2}\sum_{l=2}^{\infty}\sum_{m=1}^{l}\left(\frac{a_e}{r}\right)^l m P_l^{(m)}\left(C_{lm}\beta_{m-1} + S_{lm}\alpha_{m-1}\right) \\
D &= \frac{GM}{r^2}\left(- \sum_{l=2}^{\infty} J_l \left(\frac{a_e}{r}\right)^l P_l^{(1)} + \sum_{l=2}^{\infty}\sum_{m=1}^{l}\left(\frac{a_e}{r}\right)^l P_l^{(m+1)}\left(C_{lm}\alpha_m + S_{lm}\beta_m\right)\right)
\end{aligned} \tag{4.15}$$

In the above the zonal contributions have been separated from the other harmonics, to show the similarities between the terms A, B, C and D more clearly. The factors α and β are given by

$$\begin{cases} \alpha_m = \cos^m(\phi)\cos(m\lambda) = \alpha_{m-1}.\alpha_1 - \beta_{m-1}.\beta_1 \\ \beta_m = \cos^m(\phi)\sin(m\lambda) = \beta_{m-1}.\alpha_1 + \alpha_{m-1}.\beta_1 \end{cases} \tag{4.16}$$

As shown by the final terms on the right-hand side in (4.16), these numbers are also computed from efficient recurrence relations without the need to evaluate large amounts of trigonometrical functions, especially since the basic transformation relations between spherical and cartesian coordinates conveniently provide

$$\begin{cases} \alpha_1 = \cos\phi\ \cos\lambda = \dfrac{x}{r} \\ \beta_1 = \cos\phi\ \sin\lambda = \dfrac{y}{r} \end{cases} \tag{4.17}$$

The argument of the Legendre polynomials is $\sin\phi = \dfrac{z}{r}$, so the Müller algorithm does not require any sine or cosine function to be evaluated within the computation of $\nabla U(r,\phi,\lambda)$.

The parameters within the gravity field model that can be solved for by Faust are the main gravity parameter $\mu = GM$ (although it will usually be kept fixed at its well-established a priori value) and the coefficients C_{lm} and S_{lm} up to a desired degree and order. The partial derivatives for the central attraction parameter μ with respect to x, y and z follow directly from noting that all terms in (4.15) are proportional to this parameter :

$$\underline{a}_{grav} = \mu\,\hat{\underline{a}} \quad \Rightarrow \quad \frac{\partial\,\underline{a}_{grav}}{\partial\mu} = \hat{\underline{a}} \tag{4.18}$$

This implies that after evaluation of the acceleration components the required explicit partials are computed by simply dividing each acceleration component by μ (a principle that is obviously valid for any parameter that forms a scaling factor to an acceleration vector). The partials for the coefficients C_{lm} and S_{lm} are also obtained in a straightforward way, because within the series expansions of (4.15) only one term at a time will depend on a particular coefficient. For C_{lm} we find

$$\begin{aligned} \frac{\partial A}{\partial C_{lm}} &= \frac{GM}{r^3}\left(\frac{a_e}{r}\right)^l P_{l+1}^{(m+1)}\,\alpha_m & \qquad \frac{\partial B}{\partial C_{lm}} &= \frac{GM}{r^2}\left(\frac{a_e}{r}\right)^l m\,P_l^{(m)}\,\alpha_{m-1} \\ \frac{\partial C}{\partial C_{lm}} &= \frac{GM}{r^2}\left(\frac{a_e}{r}\right)^l m\,P_l^{(m)}\,\beta_{m-1} & \qquad \frac{\partial D}{\partial C_{lm}} &= \frac{GM}{r^2}\left(\frac{a_e}{r}\right)^l P_l^{(m+1)}\,\alpha_m \end{aligned} \tag{4.19}$$

The expressions for S_{lm} are identical, except that all α's are replaced by β's and vice versa.

For direct height measurements between the satellite and the ocean surface - i.e. radar altimetry - additional partials are computed that correspond to the explicit term in (2.50). These exploit the direct relation between the geoid undulation (the height of the geoid above the reference ellipsoid) and the local value of the gravity potential. These explicit altimetry partials will be described in Chapter 5.

4.2 Earth and ocean tides

In analogy to (4.2), the disturbing potential at the satellite's position $\underline{r}_s$, due to a point mass M_d at geocentric position $\underline{r}_d$ is given by

$$U_d = -GM_d \left(\frac{1}{\| \underline{r}_d - \underline{r}_s \|} - \frac{\underline{r}_s \cdot \underline{r}_d}{\| \underline{r}_d \|^3} \right) \tag{4.20}$$

which can be readily verified by writing out ∇U_d as the satellite's acceleration due to M_d. By writing the scalar product as $\underline{r}_s \cdot \underline{r}_d = r_s r_d \cos\psi$, with ψ being the geocentric angle between the satellite and the disturbing body, Musen (1975) developed the series expansion

$$U_d = \frac{GM_d}{r_d} \sum_{l=2}^{\infty} \left(\frac{r_s}{r_d} \right)^l P_l(\cos\psi) \tag{4.21}$$

The geocentric angle ψ is usually expressed in the more practical terms of the ECF longitude and latitude of the satellite (λ and ϕ) and the TOD ascension and declination of the disturbing body (α and δ), while the Legendre polynomials are again developed in series expansions in associated Legendre functions P_l^m, as was done in the geopotential :

$$U_d = \frac{GM_d}{r_d} \sum_{l=2}^{\infty} \sum_{m=0}^{l} c_{lm} \left(\frac{r_s}{r_d} \right)^l P_l^m(\sin\delta)\, P_l^m(\sin\phi) \cos mH \tag{4.22}$$

Here, the hour angle $H = (\lambda + \theta_G) - \alpha$ is the difference in longitude between the satellite's meridian and that of the tide-generating body; the Greenwich angle θ_G connects the ECF frame with the TOD frame (see Appendix A). For satellites around the Earth the ratio between the geocentric distances r_s to the satellite and r_d to the disturbing body is about 0.015 for the Moon and 0.000045 for the Sun. This means that the contributions to (4.22) rapidly decrease with increasing degree l. In most cases the series are therefore truncated for $l > 3$, leaving just seven terms in the series (4.22).

The gradient of U_d provides the *direct* perturbation of the satellite's motion due to the disturbing mass (i.e. the Sun or the Moon), while the redistribution of the mass of the Earth under the influence of the disturbing body results in time-dependent variations of the geopotential. This redistribution is expressed by the dimensionless Love number k :

$$U_{tot} = U + (1+k)U_d \tag{4.23}$$

in which the static geopotential is given by U, the direct disturbing potential (4.22) by U_d, and the increment of the geopotential, as induced by the disturbing body, by the factor kU_d. The influence of the tidal redistribution of the Earth's mass upon the orbit of a satellite is now most practically modelled as a set of time-dependent corrections $\Delta C_{lm}(t)$, $\Delta S_{lm}(t)$ to the static geopotential of Section 4.1. The total tide at any time is the result of a combination of separate tidal constituents, each determined by a specific frequency and amplitude. Every partial tide gives rise to a range of corrections $\Delta C_{lm}(t)$, $\Delta S_{lm}(t)$ up to a relevant degree and order that may vary for different constituents. The total correction to an individual geopotential coefficient is then found by accumulating the contributions of the tidal constituents for a given degree and order.

Doodson (1921) expressed specific tidal frequencies $\dot{\theta}_s$ as a linear combination of astronomical angles D_j that are closely related to the fundamental arguments of the Earth's nutation series (Appendix A) :

$$\theta_s(t) = \sum_{j=1}^{6} n_j D_j(t) \tag{4.24}$$

in which :

$D_1 = \tau = \theta_G + \pi - s$	Time angle in lunar days since descending node of the Moon.
$D_2 = s = F + N$	Mean ecliptical longitude of the Moon
$D_3 = h = s - D$	Mean ecliptical longitude of the Sun
$D_4 = p = s - L$	Longitude of the mean perigee of the Moon
$D_5 = N' = -N$	Longitude of the mean ascending node of the Moon
$D_6 = p_1 = s - D - L'$	Longitude of the mean perigee of the Sun

The six (small) integers n_j are usually given in the form of the Doodson argument

$$D_s = n_1/5 + n_2/5 + n_3/./5 + n_4/5 + n_5/5 + n_6 \tag{4.25}$$

allowing a range from -5 to $+5$ for the digits n_2 to n_6, which is sufficient for all relevant tidal frequencies. The principal semi-diurnal solar tide for example (Darwin notation S_2) has

Doodson argument $D_s = 273.555$, meaning that $n_1 = 2$, $n_2 = 2$, $n_3 = -2$, $n_4 = n_5 = n_6 = 0$. As an alternative to (4.22), the tidal potential may now be expressed in tidal frequencies :

$$U_d = \sum_s H_s \cos\left(\zeta_s + \theta_s(t)\right) \tag{4.26}$$

The factor H_s expresses the amplitude of the tidal constituent, while the phase angle ζ_s is either 0 or $-\frac{\pi}{2}$, in other words it turns the cosine into a sine for certain constituents. Starting from this, Eanes *et al.* (1983) modelled the changes in the geopotential coefficients due to solid Earth tides as

$$\begin{aligned}
\Delta C_{lm} &= A_m \sum_s k_s H_s \cos\theta_s && (l-m\ even)\\
&= A_m \sum_s k_s H_s \sin\theta_s && (l-m\ odd)\\
\\
\Delta S_{lm} &= A_m \sum_s k_s H_s (-\sin\theta_s) && (l-m\ even)\\
&= A_m \sum_s k_s H_s \cos\theta_s && (l-m\ odd)
\end{aligned} \tag{4.27}$$

with the amplitude factor A_m defined as

$$A_m = \frac{(-1)^m}{a_e\sqrt{4\pi(2-\delta_{0m})}} \tag{4.28}$$

in which δ_{0m} is the Kronecker delta. The Love numbers k_s in (4.27) are in principle also dependent upon the tidal frequency (e.g. Matthews *et al.*, 1995); hence the subscript. In practice, part of the tidal effect is not included in the form of corrections to the geopotential coefficients via (4.27) but rather taken into account in the form (4.23) directly, using the main terms in the expansion for the tidal potential (4.22) with a nominal Love number k_{const}. The acceleration due to this frequency independent part is (e.g. Dow, 1988)

$$\nabla U_d = k_{const}\frac{GM_d a_e^5}{2 r_d^3 r_s^4}\left[(3-15\cos^2\psi)\frac{\underline{r}_s}{r_s} + 6\cos\psi\,\frac{\underline{r}_d}{r_d}\right] \tag{4.29}$$

The Love numbers in (4.27) are therefore smaller factors k_s that represent the frequency-dependent part only. As a result of this approach many of the remaining contributions become so small that they can be neglected altogether, thereby reducing the amount of numerical operations in the evaluation of the tides. Following MERIT standards, Faust only includes six remaining diurnal and two semi-diurnal solid Earth tides in the summations of (4.27). For the frequency independent Love number k_{const} the standard value of 0.30 is adopted.

The modelling of the ocean tides is more complicated than that of solid Earth tides, mainly because the interactions between the liquid oceans and the land masses induce complicated patterns of tidal phase lags that depend on the tidal frequencies in a way that can not just be incorporated in the form of small variations in Love numbers. The expressions for the corrections to the geopotential in Faust follow the model by Cartwright and Taylor (1971) :

$$\Delta C_{lm} = B_{lm} \sum_s \left[c_{lm} \cos \theta_s + s_{lm} \sin \theta_s\right] \tag{4.30}$$

$$\Delta S_{lm} = B_{lm} \sum_s \left[s_{lm} \cos \theta_s - c_{lm} \sin \theta_s\right] \tag{4.31}$$

Lower case coefficients c and s are used here for the ocean tides, to avoid confusion with the geopotential coefficients. The amplitude factor B_{lm} in (4.30) and (4.31) is given by

$$B_{lm} = \frac{4\pi G \rho_w}{g_0} \frac{1 + k_s}{2l + 1} \tag{4.32}$$

in which ρ_w is the density of seawater and g_0 is the nominal gravity acceleration at the geoid equipotential surface. An ocean tide model then consists of ranges of coefficients c_{lm} and s_{lm} for various tidal constituents, each defined by their Doodson argument (4.25) and Love number k_s.

The tidal parameters that can be solved for by Faust are the coefficients c_{lm} and s_{lm} for the ocean tides, for as far as resulting from a specified range of constituents. The partial derivatives for these parameters are obtained from an expression in the gravity partials :

$$\frac{\partial \ddot{x}}{\partial c_{lm}} = \frac{\partial \ddot{x}}{\partial C_{lm}} \frac{\partial C_{lm}}{\partial c_{lm}} + \frac{\partial \ddot{x}}{\partial S_{lm}} \frac{\partial S_{lm}}{\partial c_{lm}} \tag{4.33}$$

in which the partial with respect to the geopotential coefficients on the right hand side are described in Section 4.1, while the other partials follow from (4.30) and (4.31) directly. For c_{lm} we find for instance

$$\frac{\partial \Delta C_{lm}}{\partial c_{lm}} = B_{lm} \cos \theta_s \qquad\qquad \frac{\partial \Delta S_{lm}}{\partial c_{lm}} = -B_{lm} \sin \theta_s \tag{4.34}$$

Because of the dependency between the gravity coefficients and the tidal parameters of the same degree and order, the variational partials for the tides follow a construction like (2.54) in terms of the implicit partials for the geopotential coefficients. This leads to slightly adjusted relations within the Gauss-Jackson integrator as well as in the initialisation routine. Precise details of these differences are documented by comments in the code at the relevant locations. Within Faust, the tidal partials form the only ones that relate to mutually dependent parameters, and obviously the tides can only be solved for if at the same time the gravity field parameters are included as estimated or considered parameters up to the required degree and order, as the gravity partials in (4.33) would be zero otherwise.

4.3 Third-body attractions

Apart from the Sun and the Moon, several planets cause non-negligible perturbations of the satellite orbit and must therefore be included in the acceleration model. Because of the large distances to the planets they can be modelled as point masses, producing Newtonian attractions of the form (4.1). As discussed in Appendix A the J2000 frame in which the equations of motion of the satellite are described is only a semi-inertial geocentric frame, and the effect of gravitational attractions of the other masses in the Solar system upon the satellite is therefore given by the *difference* between the planetary attractions of the satellite and of the Earth respectively, as expressed by the gradient of the potential (4.20) :

$$\frac{\partial U_d}{\partial x_s} = -GM_d \left(\frac{x_d - x_s}{\| \underline{r}_d - \underline{r}_s \|^3} - \frac{x_d}{\| \underline{r}_d \|^3} \right) \tag{4.35}$$

with similar expressions for y and z. The only six bodies in the solar system for which the acceleration (4.35) is non-negligible to Earth satellites (as determined by Melbourne *et al.* (1983) from the ratio of their mass with respect to the third power of their distance) are - in order of importance - the Moon, the Sun, Venus, Jupiter, Mars and Mercury. The required luni-solar and planetary ephemerides for the time of interest are obtained from interpolation in astronomical tables. For numerical efficiency this is done only once at each time step of the numerical integrator, namely at the same moment that the rotation matrix from the J2000 frame to the TOD frame is determined (see Appendix A). For the Moon and the Sun, the frequency independent luni-solar tide (4.29) is also added at that point.

The mass M_d of the planets is not entered into the equation (4.35) directly, but rather expressed as a ratio to the mass of the Earth according to

$$GM_d = GM_e \cdot \frac{M_d}{M_e} = \mu \frac{M_d}{M_e} \tag{4.36}$$

This creates the possibility to observe the main gravity parameter μ of the Earth from the third-body acceleration, by dividing its three Cartesian components by μ.

4.4 Atmospheric drag

The first surface force that will be described is the decellerating effect of the upper layers of the atmosphere, which is mainly of importance to LEO satellites. The atmospheric drag is modelled as the dynamic pressure on the surface of the satellite, projected in a direction opposite to its velocity with respect to the atmosphere :

$$\underline{a}_D = -C_D \cdot \frac{1}{2} \rho V_a^2 S \hat{\underline{v}} \tag{4.37}$$

Each of the components of (4.37) will be shortly described, with some references to a more elaborate treatment in literature.

For the atmospheric density ρ various models have been introduced over the years, but none of them can claim to be very precise. Ultimately, the poor quality of density models results from a lack of detailed knowledge of the behavior and composition of the upper layers of the atmosphere. In Faust, three alternative thermospheric models have been implemented, being the CIRA model by Rees *et al.* (1990), the DTM model by Barlier *et al.* (1979) and finally the MSIS model by Hedin (1983 & 1987) which will be used in most of the applications in later Chapters. All these models use current and time-averaged values of the solar flux at 10.7 cm wavelengths, the 3-hourly geomagnetic index K_p, and the latitude and height of the satellite to compute number densities for separate gases in the upper atmosphere. Because the subroutines for the three models in Faust have all been made available by their respective authors, their own descriptions are recommended for the precise details of each model. Schidlovsky (1976) and Herrero (1985) demonstrated that even sophisticated thermospheric models may regularly produce values that are an order of magnitude incorrect. Improvement of density models must therefore still be considered as an important subject of study in precise orbit determination for Earth satellites.

The velocity relative to the atmosphere V_a is computed in a straightforward way from the vectorial sum of the satellite's orbital velocity in the ECF frame and the velocity vector of an atmosphere that is assumed to rotate with the planet. The unit vector $\hat{\underline{v}}$ in the direction of V_a follows immediately from the same trigonometry.

The reference surface S is satellite dependent, and will only be time independent for spherical satellites like LAGEOS, Starlette, GFZ, etcetera. For other satellites, especially for those that follow intricate attitude control algorithms (e.g. for solar panel pointing), the computation of this surface is not straightforward. For ERS-1 and ERS-2, Faust uses the GUESS-model tables that are described in a comprehensive way by Ehlers (1993). These tables provide pre-computed values for an effective visible surface area $A_S \cos\theta_i$ in which θ_i is the particle flux incident angle, itself determined by two angles in space, and A_S is the nominal surface of any of 29 plates in a macro-model for the ERS geometry. The actual surface value will be interpolated as a function of the two directional angles, taking into account the time-dependent attitude of the ERS solar panel. In comparison with elaborate ray-tracing methods the GUESS-models are efficient in terms of required CPU-time - the geometrical projections and angle-dependent interactions of the particle flow with the surface have been computed only once, when the tables were created - at the expense of more substantial RAM requirements for storing the relatively large table.

For TOPEX/Poseidon, Faust uses the NASA box-wing model by Marshall *et al.* (1995). Each of the plates in the box-wing model is projected towards the direction of the velocity relative to the atmosphere, and contributions of all plates that are visible in the direction of projection are accumulated. The related subroutines implemented in Faust are virtually unmodified copies of routines used in GEODYN, as made available by NASA/GSFC.

The drag scale factor C_D in (4.37) is used to absorb the imperfections in any of the other model factors, and is among the parameters that can be solved for by Faust. An important feature of the GUESS models for ERS and the Box-Wing model for TOPEX is that they reduce the inaccuracy in the modelling of the surface S to a level that is much lower than that in the density ρ, while the velocity relative to the atmosphere is known with sufficient accuracy throughout the integration process. This implies that the scale factors C_D - having a nominal value of 1 - mainly absorb the errors in the employed density model. To account for short-term fluctuations in density and to suppress accumulation of errors in the numerical integration process, the total length of an integrated satellite arc is usually divided in shorter intervals with a separate parameter C_D defined for the times of the boundaries between intervals. The actual value at a certain time is found by linear interpolation between the values for the start and the end of the interval. With $t_n \leq t < t_{n+1}$, we find

$$\begin{cases} C_D(t_n) = \mathcal{C}_n \\ C_D(t_{n+1}) = \mathcal{C}_{n+1} \end{cases} \rightarrow \quad C_D(t) = \frac{t_{n+1} - t}{t_{n+1} - t_n}\,\mathcal{C}_n + \frac{t - t_n}{t_{n+1} - t_n}\,\mathcal{C}_{n+1} \tag{4.38}$$

Obviously this requires one more parameter than there are intervals, so that each interval has a parameter at the start and at the end. The evaluation of the drag force then produces explicit partials for the two C_D-parameters just before and just after the time of interest :

$$\underline{a}_{drag} = C_D(t) \cdot \underline{a}_{unscaled} \quad \rightarrow \quad \begin{cases} \dfrac{\partial \underline{a}_{drag}}{\partial C_{D_n}} = \dfrac{t_{n+1} - t}{t_{n+1} - t_n} \, \underline{a}_{unscaled} \\[2ex] \dfrac{\partial \underline{a}_{drag}}{\partial C_{D_{n+1}}} = \dfrac{t - t_n}{t_{n+1} - t_n} \, \underline{a}_{unscaled} \end{cases} \tag{4.39}$$

with $\underline{a}_{unscaled}$ being the unscaled part in the drag acceleration (4.37).

Because of Faust's inherent multi-arc / multi-satellite approach to orbit determination, two important options have been included in the programme. At first, the above modelling of drag scale factors implies that the very first and last parameters in an arc only receive information from one adjacent interval, while all others receive data from two intervals. This means that the first and last scale factor of the arc are less well-determined than the others. To prevent a resulting deterioration of orbital precision in these intervals, the corresponding drag parameters often receive stronger constraints than the central parameters, or the first and last interval in the arc are given a longer duration than the others.

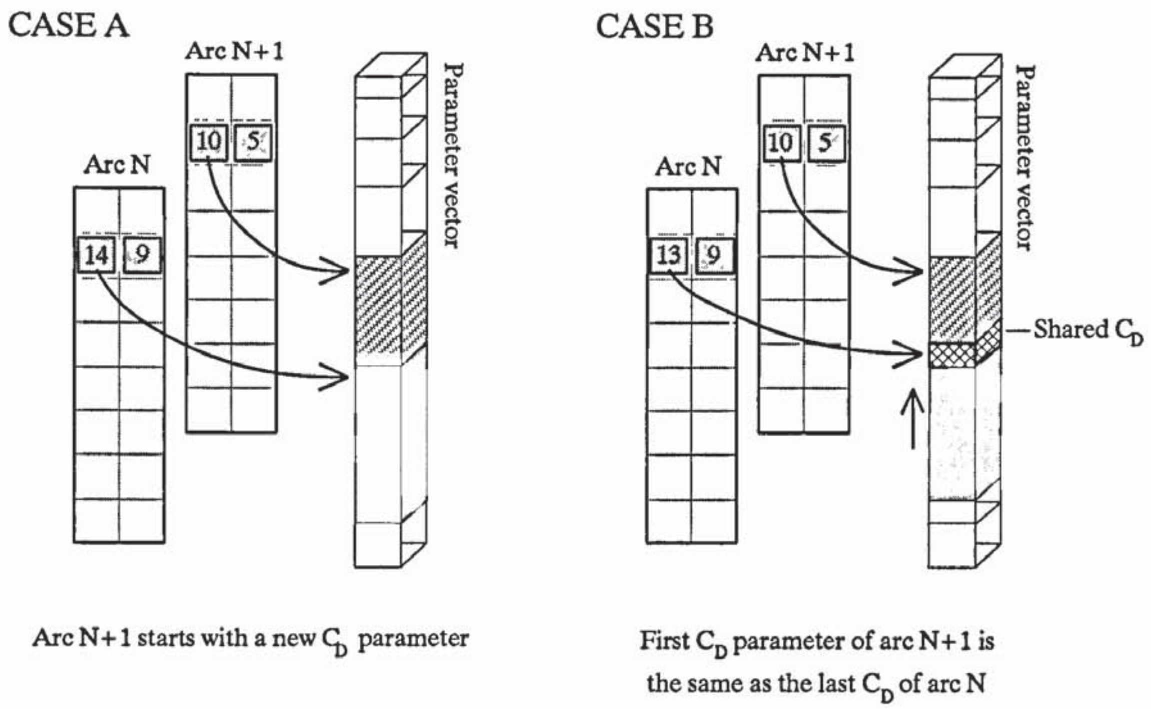

CASE A
Arc N+1
Arc N
10
5
14
9
Parameter vector
Arc N+1 starts with a new C_D parameter
CASE B
Arc N+1
Arc N
10
5
13
9
Parameter vector
Shared C_D
First C_D parameter of arc N+1 is the same as the last C_D of arc N

Figure 4.1 Consecutive arcs can be made to share the drag parameter at the arc transition, by shifting the subvector with C_D-parameters for the second arc and make it overlap with that of the first. All that needs to be done is modify the element of the array M_P (see Section 2.6) that provides the offset of the second subvector within the parameter array.

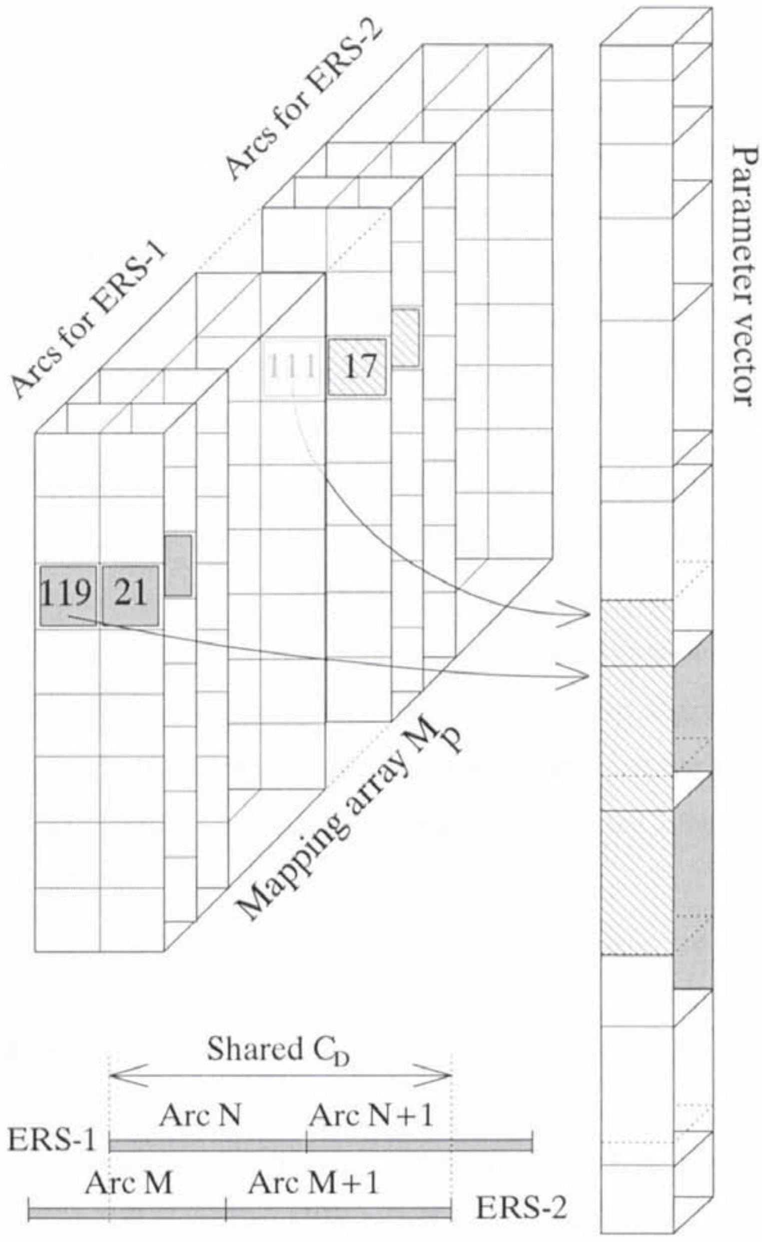


Figure 4.2 During the ERS tandem mission, arcs for the two satellites that overlap in time can be made to share the C_D parameters during the overlap period, by manipulating the pointer array M_P

However, if several consecutive orbit arcs for one satellite are computed within a single least-squares process, the time tag of the last drag scale parameter of one arc coincides with that of the first parameter of the next arc. In that case, it makes sense to use the same parameter for both cases, rather than solve for two separate values. This improves the observability of the corresponding parameters and improves continuity between consecutive arcs. In addition, it may help to decouple the first drag scale factor in the arc from the elements of the initial state vector, with which it is often highly correlated. Because of the pointer structures discussed in Section 2.6, the option to share drag parameters between consecutive arcs can be easily implemented by slightly modifying the mapping arrays, as illustrated in Figure 4.1.

A second interesting option is related to simultaneous solutions for the ERS-1 and ERS-2 tandem mission, during which the two satellites are separated by 60 degrees in orbit or about 18 minutes in time. The Earth's rotation during the interval between crossings of the satellites over a given latitude will be less than 5 degrees. This means that the two satellites are essentially passing through identical atmospheric conditions, especially because the drag scale factors always relate to an interval of several hours. As a consequence, it is realistic to let the two satellites share their drag scale parameters, if we recall from the above that the main task of these parameters is to absorb the atmospheric density errors (note that systematic errors in the surface S will also be almost identical for the two satellites, as both satellites are using the same GUESS model tables and follow identical attitude control algorithms). This is implemented by letting the subvectors with drag parameters for the two satellites overlap completely (see Figure 4.2), possibly with a shift to allow arcs for the two satellites to start at different epochs. The effects of combined drag parameters in the ERS tandem mission will be investigated in Chapter 8.

4.5 Radiation pressure

Faust accounts for the usual three forms of radiation pressure (see e.g. Haley, 1973), as caused by direct solar radiation, solar radiation reflected by the Earth (albedo), and infra-red radiation produced by the Earth's black-body temperature. Although albedo radiation is physically related to direct solar radiation, it is much more similar to infra-red radiation from a geometrical point of view. For this reason, the direct solar radiation is modelled separately while the Earth albedo and infra-red radiation are grouped together into a single force model. As both these force models are neither new nor highly relevant to the remainder of this thesis, they will only be described here for sake of completeness.

The modelling of direct solar radiation pressure is fairly similar to the modelling of atmospheric drag, with the obvious difference that the particle flow is related to the satellite's position and attitude towards the sun rather than towards the Earth's atmosphere, and that it disappears when the satellite enters the Earth's shadow. The basic formula is given by

$$\underline{a}_{SRP} = - C_R \, \psi \, F \, S \, \underline{e}_{SS} \tag{4.40}$$

and all terms will again be briefly explained.

The scale factor C_R is used in a similar way as the C_D parameters for drag, and can be solved for by Faust. Unlike with drag scale factors, it is unusual to solve for more than one parameter C_R because the uncertainties in the components of (4.40) are much smaller than those in the components of the drag model. In fact, for the sun-synchronous ERS satellites the factor C_R will often be fixed at its neutral value of 1, while solving for cross-track empirical accelerations that will be described later in this Chapter. Because there is only one scale factor, the arc is not divided in different intervals in the way that is done for atmospheric drag. The partial derivatives therefore follow directly from (4.40) as the unscaled acceleration.

The factor ψ is a shadow-function that is 1 if the satellite passes over the day-side of the Earth and 0 in the full Earth shadow, while varying according to a sinusoidal smoothing function during the penumbra transition. F is the solar flux per unit area at the position of the satellite, derived from tabulated values. The projected surface S of the satellite is again computed from GUESS tables for the ERS satellites and from the NASA Box-Wing model for TOPEX/Poseidon.

Earth albedo and infra-red radiation are effectively modelled in the same way as was done in the SATAN package and described by Ehlers (1993). The only difference is the value of the projected satellite surface used for these forms of radiation pressure. In SATAN, the same value S was used as the one in (4.40) for the direct solar radiation, which is obviously unrealistic, because the incident direction is always different. Faust therefore uses a constant value for this surface, specified in the user input. Although this may not be much more accurate than the values used by SATAN, its errors are now only due to tolerated simplification rather than to blatant mismodelling, which is at least a moral improvement.

4.6 Thrust forces

New in Faust is the possibility to integrate over manoeuvres, as described in Chapter 2.

Manoeuvre information is available from ESA and NASA mission management for the ERS satellites and for TOPEX/Poseidon respectively, tabulated in the form of effective velocity changes in along track, radial and cross-track directions, and manoeuvre start and end epochs. Although the tabulated velocity changes will usually be calibrated post-manoeuvre values, they will to a certain extent be system dependent and are therefore multiplied in Faust by scale factors with a nominal value of 1.0, which can be solved for in order to fine-tune the manoeuvre, and absorb unmodelled effects like radiation pressure due to post-pulse nozzle cooling. Initially, the accelerations were included in the form of straightforward block functions within an interval of reduced integration step size that extended sufficiently before and after the actual manoeuvre. This approach proved to result in fairly large scale factors, apparently indicating substantial modelling errors. It was concluded that this was mainly due to numerical problems caused by the discontiuities of the block function - with derivatives that will be impulse functions - which interfere with finite-difference numerical integrators like the Gauss-Jackson algorithm in a fundamental way.

To avoid this problem, the thrust force accelerations were replaced by a sinusoidal function with the same surface area as the original block pulse, and centered in time around the centre of the block pulse (see Figure 4.3). The formula for the thruster acceleration in any of direction is therefore

$$a_T = c_T A_T \left[1 - \cos\left(2\pi \frac{t - t_0}{t_1 - t_0} \right) \right] \tag{4.41}$$

in which A_T is the a priori amplitude of the original block pulse in that direction, and c_T the scale factor that can be solved for in analogy to various scale factors described earlier in this Chapter. The partial derivatives for the thrust forces are again equal to the unscaled acceleration, because a_T is directly proportional to the parameter.

Because the cosine is continuous and can be differentiated to any desirable order, its behavior is better suited for numerical integration as long as the step size is sufficiently small to follow the sinusoidal shape. The model according to (4.41) has appeared to work well for relatively short pulses - with durations up to ten seconds or so - but for large manoeuvres the difference between the sinusoidal curve and the original block pulse tends to cause notable problems, at least by requiring unrealistic scale factors. For this reason Faust will at present still have some difficulty to integrate accurately over large manoeuvres, like those used to change the repeat period of ERS-1. This is not a severe problem (it could in fact be solved easily by using a more sophisticated model, like a block pulse with sinusoidal transitions at the start and the end), because most orbits will simply be restarted just after a large manoeuvre. Furthermore, altimetry data obtained around manoeuvres - if any - will in general be rejected because of its unreliable orbital reference.

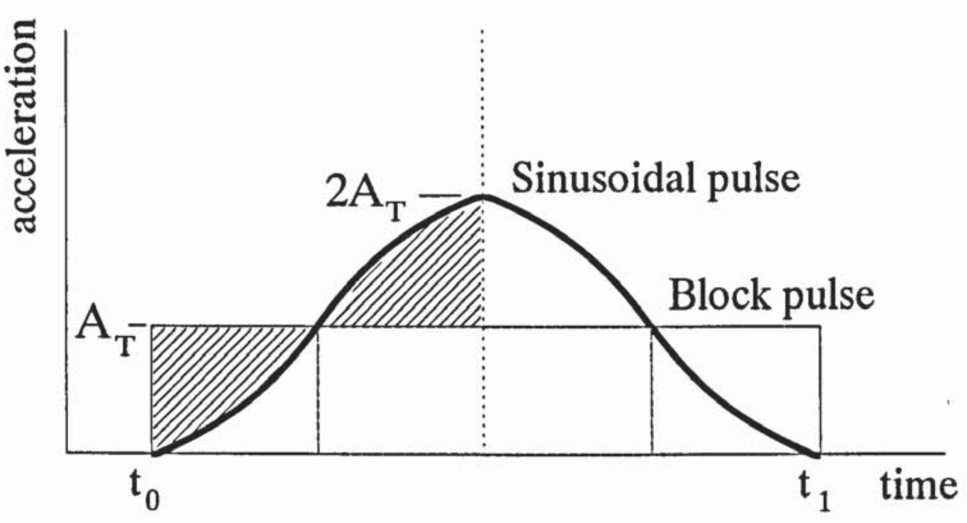


Figure 4.3 Sinusoidal thruster pulse

In Faust, the tolerated maximum integration step size h_2 during the manoeuvre interval (see Section 3.4) is computed in such a way that at least eleven steps will fall within the duration of the pulse - so that finite differences up to order 10 are non-zero - after which the real step size h_2 is found by repetitive divisions of the nominal step size by two, until it is smaller than the tolerated maximum.

4.7 Empirical cyclic accelerations

Despite of the effort invested in accurate models for the conservative and non-conservative accelerations of the satellite, no model can perfectly take into account the complexity of physical reality. Each model suffers either from a lack of detailed knowledge (as for atmospheric density models, or for precise values of many geopotential coefficients of high degree and order) or from simplifications that are inevitable if a model is to be evaluated within a reasonable time (as by simplifying the geometry of the satellite to a box-wing model, or by truncating the geopotential at a finite degree). However, the remaining nett acceleration error can be encountered by analysing its signature either in the time-domain or in the frequency domain. Many authors have computed error budgets for the precise orbit determination of altimetry missions or studied the spectral power of the error signal derived from the variability of the measurement residuals (2.1), for instance Scharroo *et al.* (1993a) for ERS-1 or Tapley *et al.* (1994) for TOPEX/Poseidon. In terms of spectral power, peaks occur for frequencies that form an integer multiple of the orbital revolution, with by far the largest component for a once per revolution frequency. Such peaks can be explained from the combined effects of gravity field mismodelling, of the satellite's attitude during it's orbital revolution, and of various other model components with a direct relation to the satellite's position relative to the Earth.

To reduce this error peak it has become common practice to explicitly model a periodic acceleration of the satellite with a frequency that corresponds to one cycle per orbital revolution. In principle such an acceleration could be applied in along-track, cross-track and radial direction, but because the effect on the orbital elements due to a radial acceleration can not be separated from that due to an along-track acceleration - as follows directly from the Lagrange planetary equations - only along-track and cross-track accelerations are modelled. This is done in the form of a sinusoidal acceleration of the satellite in these directions, each determined by two parameters (amplitude and phase, or coefficients for a sine and a cosine with zero phase) that are solved for. The straightforward formula used in Faust is

$$\underline{a}_{cpr} = \left(c_1 \sin u + c_2 \cos u\right)\underline{e}_{AT} + \left(c_3 \sin u + c_4 \cos u\right)\underline{e}_{CT} \tag{4.42}$$

in which u is the orbital anomaly of the satellite with respect to the equator, and $\underline{e}_{AT}$ and $\underline{e}_{CT}$ are the unit vectors in the along-track and cross-track directions respectively. The parameters c_1, c_2, c_3 and c_4 are estimated, from a priori values of zero.

As with drag, the integrated arc will sometimes be divided in intervals with independent sets of such parameters in order to account for the variability of the once-per-revolution error over the duration of the arc. Unlike the drag scale factors, the once-per-revolution empirical parameters are not found by interpolation between two values but are assumed constant over their interval of validity. The related variational partials are therefore only produced for the set of four parameters that relate to the time of interest. Although these parameters, if solved for, must always be defined in groups of four, it is of course possible to fix one pair and only solve for the other two.

4.8 Empirical along-track accelerations

Faust also contains the possibility to solve for empirical along-track accelerations, as an alternative to the drag scale factor parameters discussed above which will then usually be fixed at their nominal value of 1. Empirical along track acceleration parameters are again defined at boundaries of several intervals within the total arc length, and the acceleration is modelled via interpolation as

$$\underline{a}_{AT} = -\left(\frac{t_1 - t}{t_1 - t_0} \alpha_0 + \frac{t - t_0}{t_1 - t_0} \alpha_1 \right) \underline{e}_{AT} \tag{4.43}$$

in which $\alpha_0(t_0)$ and $\alpha_1(t_1)$ are the empirical parameters at the start and the end of the current interval, while $\underline{e}_{AT}$ is the unit vector in the along track direction. The partial derivatives for (2.53) are easily found in analogy to (4.39).

It should be noted that any form of empirical acceleration can only be solved for realistically if sufficient measurement data is available to justify their application. If insufficient observations are available for the modelled amount of parameters, they will tend to absorb a substantial part of the measurement signal, resulting in misleadingly small observation residuals even though the quality of the overall parameter solution will be very poor.

4.9 Time tags

As discussed in various previous Sections, many of the force model parameters are only valid during a specific time interval within the overall solution period, the latter spanning from the earliest starting epoch of all arcs in the solution until the latest end epoch. For this purpose, an array with time tags is defined in parallel to the parameter array $\underline{P}$ from Chapter 2, accessed via the same pointer structure M_P as the parameters themselves. Although some parameter types - like gravity field coefficients - may not be linked to any particular epoch, the generalised structure of time tags will enable easy implementation of future time-dependent parameter types without the need to define individual time tags for each parameter type separately. Most of the housekeeping code of Faust (input modules, the structure of the numerical integrator and force models) is automatically capable of handling time tags for any parameter type.

The calculated observation — Chapter 5

This fifth Chapter completes the discussion of the solution process in Faust by describing the way in which the tracking data is modelled and entered into the parameter estimation scheme in the form of the measurement residuals R_i in (2.1) and the geometrical partials $\frac{\partial C_i}{\partial x}$ in (2.50). After a summary of some general aspects of tracking data, the various datatypes supported by Faust will be briefly reviewed. Furthermore, the handling of partial derivatives of the form (2.53) will be discussed, related to observations that depend on state vectors and variational partials at different epochs or from different satellites. This concerns not only the two supported types of range-rate measurements, but in particular the crossover differences that have been identified in Chapter 1 as the backbone of the simultaneous solution process for altimetry missions.

As follows from the chosen time-organised processing structure in Section 2.8, Faust requires all observations to be presented to the program in strict chronological order. In order to sort and merge files of different data types, a separate data preprocessing programme - Wagner - has been written that also includes some elementary possibilities for filtering or sampling. This programme will be shortly described in order to explain the generic data format used by Faust. A final Section discusses the details of data editing in Faust, and how the influence of different satellites, stations or datatypes on the solution process is controlled in order to balance a multi-satellite solution.

5.1 General aspects of tracking data modelling

All relevant forms of satellite tracking are based on propagation models for radiowaves or optical signals that travel between reference points on the satellite and on the Earth's surface. Measurement modelling therefore always involves the accurate modelling of the spatial position of the instrument on the satellite, of the position of a ground station or other terrestrial measurement target - e.g. the ocean surface for altimetry - and of the behavior of the signal along the travel path through the Earth's atmosphere. In order to avoid unnecessary repetition later in this Chapter, this Section will summarise some aspects of tracking data that

1	Origin of the ECF frame
2	Coordinates of a local marker point
3	Tectonic plate motion
4	Station uplift due to Earth tides
5	Local eccentricity vector
6	Instrument bias
7	Tropospheric effects
8	Ionospheric effects
9	Instrument centre-of-mass offset
10	J2000 velocity of the satellite
11	J2000 velocity of the station
12	Two-way range return path

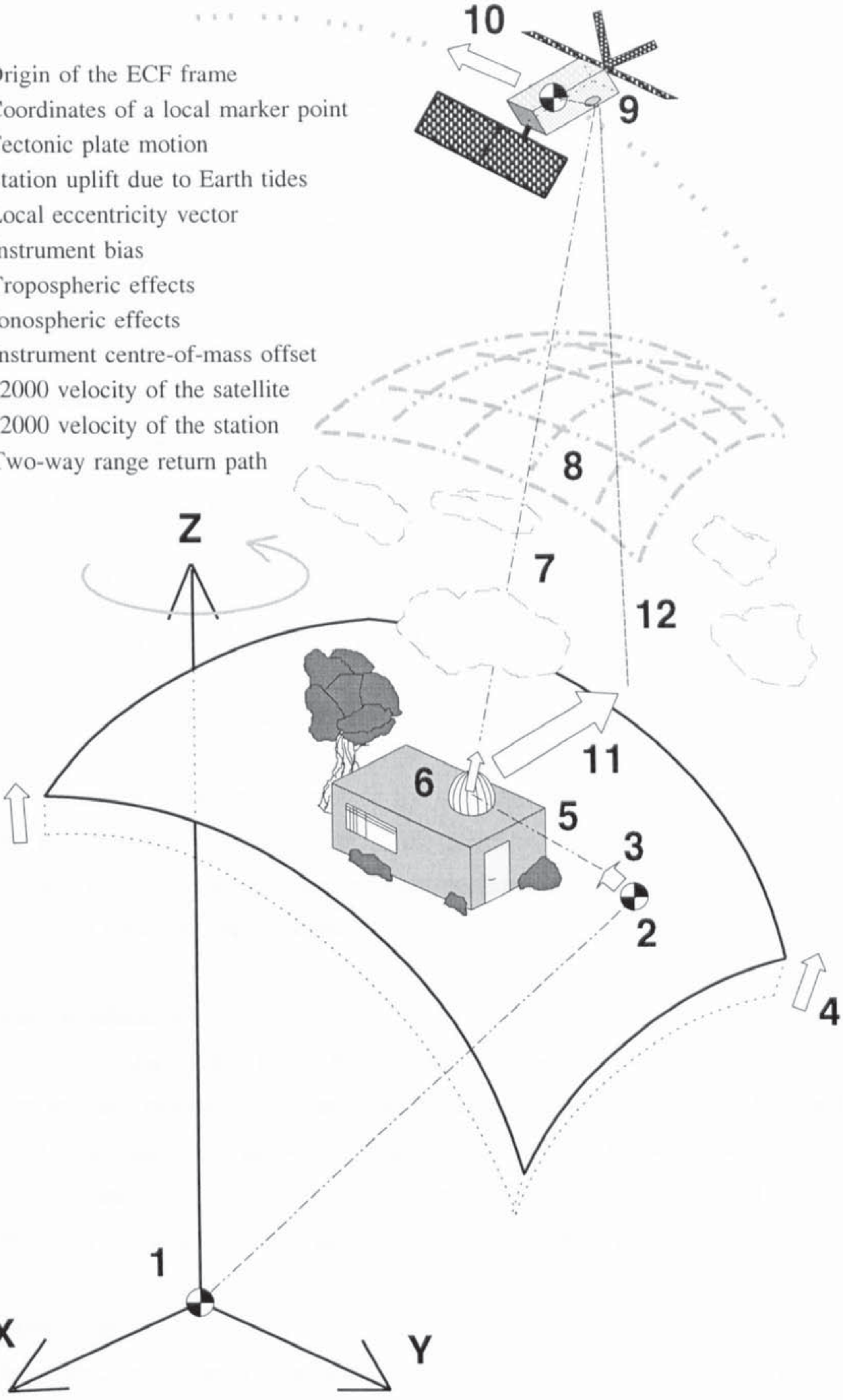


Figure 5.1 Satellite tracking configuration

are common to practically all data types. To this purpose the mathematical model for the tracking signal will be followed from the origin of the reference frame to the centre of mass of the satellite, for which the orbit is determined in the way described in Chapters 2 and 3. The general configuration of the tracking signal is illustrated in Figure 5.1.

5.1.1 Station coordinates

The origin of all reference frames described in Appendix A coincides with the centre of mass of the Earth. Sets of ECF coordinates for tracking stations - implicitly defining the frame itself - are available from sources like the IERS or various other space geodesy groups throughout the world. The coordinates themselves are in general also obtained from parameter estimation processes, because they are easily observable. For example, the coordinate in the x direction is observed from

$$\begin{aligned} C_i(x,y,z) &= \| \underline{r}_{sat} - \underline{r}_{sta} \| = \sqrt{(x-x_{sta})^2 + (y-y_{sta})^2 + (z-z_{sta})^2} \\ \rightarrow \left(\frac{\partial C_i}{\partial x_{sta}} \right)_{expl} &= \frac{-2(x-x_{sta})}{2\sqrt{(x-x_{sta})^2 + (y-y_{sta})^2 + (z-z_{sta})^2}} \\ &= -\frac{x - x_{sta}}{\| \underline{r}_{sat} - \underline{r}_{sta} \|} \end{aligned} \qquad (5.1)$$

This shows that the three explicit partials for the three station coordinate parameters are given by the elements of a unit vector in the direction from the satellite to the station. Because these coordinates are independent from the satellite orbit, the positional partials are zero and with it the entire implicit term.

5.1.2 Station velocities

Tectonic plate motion causes the ECF coordinates to gradually change over the years. A correction for plate motion is computed from linear station velocities that are tabulated with the station coordinate data, applied over the period between the reference epoch of the coordinate set and the time of interest. Station velocities may be computed from trends between station coordinate solutions that are sufficiently far apart in time.

5.1.3 Station tides

A further correction to station coordinates is formed by the uplift of the station due to solid Earth tides. Faust uses the standardised model described by Melbourne *et al.* (1985).

5.1.4 Local eccentricities

Many station coordinates do not provide the actual ECF position of the tracking instrument's reference point, but rather relate to a local marker point in the vicinity of the instrument at the ground station. For this reason, a local eccentricity vector may have to be added to the tabulated coordinates. These station eccentricities are also provided by sources like IGN, with the periods for which they are valid. However, eccentricity data is notoriously unreliable in the sense that physical movements of the tracking instrument may only be reported with long delays. for instance through annual reports.

5.1.5 Range bias

Tracking systems involve electronic instruments that tend to produce a non-zero systematic delay of the measurement signal, appearing as a more or less constant range bias. For many tracking data types it is common practice to solve for a range bias parameter. A range bias may either be associated with a groundstation instrument - in which case there may be one per station, or possibly even one per pass over a station - or to a satellite instrument, in which case there will be just one. The explicit partial in (2.50) for a range bias parameter P_j can be obtained in a very straightforward way as

$$C_i = \| \underline{r} + \underline{\Delta r}_{bias} \| = r + P_j \quad \rightarrow \quad \left(\frac{\partial C_i}{\partial P_j} \right)_{expl} = 1 \tag{5.2}$$

but even without this formal computation it will be obvious that corrections to range bias parameters correspond to the mean value of all residuals, which is effectively what is expressed by (5.2).

5.1.6 Timing bias

The UTC time tag for the observation data record will be generated by a clock at the ground station or at the satellite that may be suffering from a systematic timing bias. To compensate, it is possible to solve for a time tag bias parameter in any type of observation by considering

$$\begin{gathered} C_i(t+\Delta t) = C_i(t) + \frac{dC_i}{dt}\Delta t \\ \frac{\partial C_i}{\partial \Delta t} = \frac{dC_i}{dt} \end{gathered} \tag{5.3}$$

The partial derivative for a time tag bias parameter $P_j = \Delta t$ is apparantly given by the time derivative of the observations themselves, i.e. for range observations the partial is given by the range rate.

5.1.7 Tropospheric correction

For the modelling of the travel path through the atmosphere we distinguish between tropospheric effects - related to the atmosphere up to a height of 60 km or so - and ionospheric effects at heights where free electrons are present in such high densities that they influence radiowaves. The troposphere causes variations in the index of refraction along the travel path, and a compensating range correction can be expressed as

$$\Delta R_n = \int_r N\,dr = \int_r (n - 1).10^6\,dr \tag{5.4}$$

in which n is the index of refraction - typically with a value close to one, like 1.00035 - and N a more practical variable that is usually denoted as the 'refractivity'. A well-known expression for the tropospheric refractivity of signals at radio frequencies is the one given by Smith and Weintraub (1953) :

$$N = N_d + N_w = 77.6\,\frac{p}{T} + 3.73*10^5\,\frac{e}{T_2} \tag{5.5}$$

in which the tropospheric correction is separated in a dry component N_d and a wet component N_w, the latter being dependent on the water vapor content e of the atmosphere; p and T are the atmospheric pressure and temperature as measured by the ground station. Black and Eisner (1984) showed that the dry tropospheric correction - which makes up 85 to 90 percent of the total correction - can be accurately computed from these surface conditions, but that the substantial variability of the water vapor content along the travel path makes it difficult to model the wet term from surface data only. The tropospheric corrections applied to data types used by Faust are in fact all slightly different, and will therefore be treated in more detail in later Sections.

5.1.8 Ionospheric correction

The ionosphere does not affect frequencies in the optical bands, but causes phase and group delays - and is therefore dispersive - for radiowaves. As dispersion is by definition a frequency dependent effect, it can be adequately computed from the observably different behavior of signals at different carrier waves as shown by Imel (1994). Many tracking systems therefore employ dual-frequency instruments to derive an accurate ionospheric correction. Systems that do not have this option - notably the altimeter instruments of the ERS satellites - use a ionospheric correction that is computed from

$$\Delta R_{ion} = 40.3\,\frac{TEC}{f^2} \tag{5.6}$$

in which TEC is the total electron content in m^{-2} along the line of sight and f the signal frequency in Hz. Bent *et al.* (1973) showed that the diurnal variations in the total electron content typically span three orders of magnitude, which makes it difficult to accurately model the TEC. The ionospheric correction for single-frequency instruments is therefore less reliable than that for dual-frequency instruments, which could result in systematic errors in the dual crossover dataset between ERS-1 and TOPEX/Poseidon.

5.1.9 Instrument centre-of-mass offset

The equations of motion for the satellite are defined for the centre of mass of the spacecraft. The observed reference point on the satellite is in general a point within an instrument, which will not coincide with the centre of mass of the satellite. This means that for tracking instruments that produce range observations we will have to take into account a centre-of-mass offset, being the projected distance along the line of sight between the instrument reference point and the centre of mass of the satellite. This also implies that it is necessary to take into account the exact attitude of the spacecraft at the time of the measurement, because a rotation of a non-spherical spacecraft will change the centre-of-mass offset.

5.1.10 Transit time compensation

Because both the satellite and the ground station - attached to a rotating Earth - move with a considerable velocity relative to the J2000 reference frame, it is necessary to account for the displacement of the reference points during the travel time of the signal. As an example, we will look at the configuration of a two-way range measurement from station to satellite and back to the station, as in Figure 5.1; the analogy to other configurations will be obvious.

Let the UTC time of pulse firing be given in the measurement data record. The position and velocity of the satellite at this time are obtained from numerical integration, as described in previous Chapters. If t_0 is the pulse fire time, t_1 is the time of reflection at the satellite, and c is the speed of light, the exact travel time Δt_1 (with initial estimate 0) and corresponding range $c\Delta t_1$ follow from a straightforward iterative scheme :

$$\begin{aligned} t_1 &= t_0 + \Delta t_1 \\ \underline{x}_{sat}(t_1) &= \underline{x}_{sat}(t_0) + \underline{v}_{sat}(t_0).\Delta t_1 \\ \Delta t_1 &= \frac{\| \underline{x}_{sat}(t_1) - \underline{x}_{sta}(t_0) \|}{c} \end{aligned} \tag{5.7}$$

Note that in each iteration step various range corrections may have to be applied, as described in other subsections. A similar iterative scheme is used to find the travel time Δt_2 for the return path, but now it is the station position that is varied rather than the satellite position. The calculated one-way range $\| \underline{r} \|$ is computed as half the sum of the two paths.

5.1.11 Geometrical partials

Finally, the vector of geometrical partials for any range observation - as required to construct the implicit term of (2.50) - is found to be equal to the unit vector in the range direction, in analogy to (5.1). For instance, for the x direction we find :

$$\begin{aligned} C_i(x,y,z) &= \| \underline{r}_{sat} - \underline{r}_{sta} \| = \sqrt{(x - x_{sta})^2 + (y - y_{sta})^2 + (z - z_{sta})^2} \\ \rightarrow \quad \left(\frac{\partial C_i}{\partial x} \right) &= \frac{2\,(x - x_{sta})}{2\sqrt{(x - x_{sta})^2 + (y - y_{sta})^2 + (z - z_{sta})^2}} \qquad (5.8) \\ &= \frac{x - x_{sta}}{\| \underline{r}_{sat} - \underline{r}_{sta} \|} \end{aligned}$$

In case that an observation is processed in the ECF frame, it will be necessary to rotate these geometrical partials into the J2000 frame in which the variational equations are defined.

5.2 Satellite laser range observations

The laser observation is a two-way range measurement between a laser / telescope system on the ground and a passive retroreflector on the spacecraft. Applied corrections include those for Earth tides and local eccentricities, a centre-of-mass offset for the retro-reflector as well as a short range correction for the travel path *within* the reflector. The most commonly used tropospheric correction for optical signals is the one given by Marini and Murray (1973) :

$$\Delta R_{MM} = \frac{f(\lambda)}{F(\phi,H)} \cdot \frac{A + B}{\sin E + \dfrac{\left[\dfrac{B}{A + B}\right]}{\sin E + 0.01}} \qquad (5.9)$$

in which A and B are empirical functions of atmospheric pressure, temperature and relative humidity as measured by the ground station, f is a function of the wavelength λ, F is a function of the station's latitude ϕ and height above the ellipsoid H, and E is the elevation of the spacecraft as seen by the station. The wavelength and atmospheric quantities required to evaluate (5.9) are provided in the input data record. According to Schwarz (1990), the

Marini-Murray correction has a precision that is better than 1 cm for zenith measurements, although at lower elevations the accuracy decreases due to the sharply increasing travel path through the atmosphere. For this reason, a cut-off elevation may be applied below which no tracking data is accepted.

Although it is common practice to solve for a range bias and possibly a timing bias for less accurate stations, Faust supports neither of these bias parameters for laser observations : both parameters are notorious for absorbing orbit errors rather than data errors, resulting in misleadingly small observation residuals without actually having a good orbit solution.

The laser retro-reflector array on TOPEX/Poseidon forms a ring around the base of the altimeter antenna, and its projection on a plane perpendicular to the line of sight will be a section of an ellipse. Not each part of this ellipse is equally distant from the observer, and an intricate centre-of-mass correction procedure is therefore included that takes into account the visible projection of the retroreflector array and the technical characteristics of the type of laser used by the station of interest, to determine the expected distribution and intensity of reflected photons. The algorithm used by Faust for this correction was made available by NASA/GSFC (J.A. Marshall; personal communication) and has been implemented without significant modifications.

SLR tracking is potentially very accurate, but several disadvantages imply that with laser tracking alone the orbit solutions of ERS and TOPEX/Poseidon would not reach the highest possible precision. In the first place, the global coverage of the SLR ground stations is poor, especially for the ERS satellites that have much smaller visibility masks than the higher TOPEX/Poseidon orbit. Most laser stations are on the North American continent and in Europe, while in the rest of the world - in particular on the entire southern hemisphere - only a very small amount of SLR tracking stations are available. Furthermore, the land-based character of SLR stations implies that the sections of the orbit that are most relevant to radar altimetry analysis, i.e. those over the oceans, are hardly covered at all.

5.3 DORIS range rate

The DORIS system was developed by CNES especially for the precise tracking of satellites like the SPOT family and TOPEX/Poseidon (see Nouël *et al.*, 1988). In comparison with SLR tracking, the global data coverage is substantially better and covers the TOPEX orbit almost completely, providing a crucial contribution to achieving its precise orbit targets. The DORIS system consists of a dense network of ground beacons and an on-board receiver. Each beacon transmits radio signals at two frequencies - allowing correction for ionospheric effects - that are accurately derived from an ultra-stable oscillator (USO). The signal is divided into regular measurement intervals by interruptions at intervals of about 10 seconds to transmit a data package that contains atmospheric data (temperature, pressure and humidity) at the ground beacon, required for a tropospheric correction.

5.3.1 Measurement geometry

Due to the component of the satellite's velocity along the line towards the beacon, the radio frequency received by the satellite will be Doppler-shifted with respect to its nominal value. The peaks in the interference signal between the received frequency and a known reference frequency are counted during a certain time interval, to provide what is known as the beat-count. In the DORIS system the reference frequency is chosen in such a way that for any possible range rate it will be larger than the received frequency, so that the beat-count is defined unambiguously.

If the transmitted and received frequency are given by f_t and f_r, the satellite's range rate by v_r, and the speed of light by c, the Doppler shift can be written (e.g. Ehlers, 1993) as

$$f_r = \frac{1 - \frac{v_r}{c}}{\sqrt{1 - \left(\frac{v_r}{c}\right)^2}} f_t \approx \left(1 - \frac{v_r}{c}\right) f_t \tag{5.10}$$

The beat count over a measurement interval Δt follows from

$$\begin{aligned} N_{bc} &= \int_{t_0}^{t_0+\Delta t} \left(f_{ref} - f_r\right) dt = \int_{t_0}^{t_0+\Delta t} \left(f_{ref} - f_t + \frac{v_r}{c} f_t\right) dt \\ &= \left(f_{ref} - f_t\right)\Delta t \quad + \quad \frac{f_t}{c}\int_{t_0}^{t_0+\Delta t} v_r\, dt \\ &= \left(f_{ref} - f_t\right)\Delta t \quad + \quad \frac{f_t}{c}\left[\rho(t_0+\Delta t) - \rho(t_0)\right] \end{aligned} \tag{5.11}$$

in which $\rho(t)$ is the momentary range between the beacon and the satellite. The observation values given in a DORIS data record are t_0, Δt, and an observed quasi range rate value that is *constructed* according to

$$O_i = \frac{c}{f_t}\left(\frac{N_{bc}}{\Delta t} - f_{ref} + f_t \right) \tag{5.12}$$

As follows from expression (5.11), this construction now allows the calculated range rate to be conveniently modelled as

$$C_i = \frac{\rho(t_0 + \Delta t) - \rho(t_0)}{\Delta t} \tag{5.13}$$

This is the easily computed finite difference between range values at the start and the end of the Doppler-count interval, divided by the given duration Δt of the interval. The two ranges $\rho(t)$ are computed in an iterative scheme as described in Section 5.1.10.

5.3.2 Parameters solved for

A nominal tropospheric correction term $\Delta\dot{\rho}_{trop}$, based upon the atmospheric conditions broadcasted by the station, is given in each DORIS data record. This correction has already been included in the input data value O_i. In order to compensate for inaccuracies in the wet component, an additional correction is modelled as a term that is proportional to this nominal correction :

$$C_i = (C_i)_{nom} + K_{trop}\Delta\dot{\rho}_{trop} \tag{5.14}$$

The parameter K_{trop} is solved for within the orbit determination process, applying one parameter for each DORIS pass over a station. From (5.14) it follows directly that

$$\left(\frac{\partial C_i}{\partial K_{trop}} \right)_{expl} = \Delta\dot{\rho}_{trop} \tag{5.15}$$

This value is again the nominal tropospheric correction as given in the data record. Like with laser observations, the reliability of the entire tropospheric correction term reduces if the travel path through the atmosphere increases, i.e. if the elevation of the satellite decreases. For this reason a cut-off elevation is specified in the input to Faust with a typical value of 20 degrees above the local horizon.

Another correction is included for errors in the reference frequencies f_t and f_{ref}, to compensate for the fact that in practice the USO, though highly stable, will still show some frequency drift. From (5.11) and (5.12) it is evident that we can not distinguish between a drift in the on-board frequency f_{ref} and a drift in the beacon frequency f_t, because only their difference is observed. It is therefore assumed that the reference frequency is perfect and that any USO drift is only present in the beacon frequency f_t. If the nominal and true (i.e. drifted) beacon frequency are given by f_t^n and f_t^d respectively, the corresponding range rates can be written as

$$O_i^n = \frac{c}{f_t^n}\left(\frac{N_{bc}}{\Delta t} - f_{ref} + f_t^n\right)$$

$$\begin{aligned} O_i^d &= \frac{c}{f_t^d}\left(\frac{N_{bc}}{\Delta t} - f_{ref} + f_t^d\right) \\ &= \frac{f_t^n}{f_t^d}\frac{c}{f_t^n}\left(\frac{N_{bc}}{\Delta t} - f_{ref} + f_t^n - f_t^n + f_t^d\right) = \frac{f_t^n}{f_t^d}\left(O_i^n - c\,\frac{f_t^n - f_t^d}{f_t^n}\right) \end{aligned} \tag{5.16}$$

The frequency drift is therefore equal to

$$O_i^d - O_i^n = \frac{f_t^n - f_t^d}{f_t^d}\,O_i^n - c\,\frac{f_t^n - f_t^d}{f_t^d} \tag{5.17}$$

The value given in the data record is inevitably the drifted value O_i^d, because O_i^n is only a fictive, theoretical value. The calculated observation C_i will therefore also have to be modelled as a drifted one - to allow comparison with O_i^d - using the expression (5.17) :

$$\begin{aligned} C_i^d &= C_i^n + \left(O_i^d - O_i^n\right) \\ &= C_i^n + \frac{f_t^n - f_t^d}{f_t^d}\,O_i^n - c\,\frac{f_t^n - f_t^d}{f_t^d} \\ &\approx C_i^n + \frac{\beta}{c}C_i^n - \beta \end{aligned} \tag{5.18}$$

Note that here the - fictive - nominal value O_i^n from (5.11) had to be approximated by the calculated value C_i^n, trusting that any error caused by this practice will be absorbed by solving for the frequency drift parameter β, the definition of which follows from (5.18). Like with K_{trop}, a separate value for β is used for each pass over a DORIS beacon. For its partial derivative we find

$$\frac{\partial C_i}{\partial \beta} = \frac{C_i}{c} - 1 \tag{5.19}$$

Because the range rate for Earth orbiting satellites can hardly be expected to approach the speed of light, the first term on the right-hand side in (5.19) can be neglected.

The two pass-dependent parameters K_{trop} and β influence the calculated observation C_i directly and will therefore be correlated with each other, although they can be assumed to be totally independent from similar pairs for other passes. These pairs therefore form parameters of Category 2 in Section 2.3. In practice there will be about 1500 DORIS passes within a single 10-day arc for TOPEX/Poseidon, producing an equal amount of parameter pairs. Given the fact that there will only be twenty to thirty orbit parameters of Category 1 in such a 10-day arc, the relevance of the matrix partitioning adopted in Chapter 2 will be appreciated.

5.3.3 Construction of partial derivatives

The followed treatment of quasi range-rate signals - involving a finite range difference - is much more straightforward than evaluation of the integral in (5.11) directly. However, it implies that the partial derivatives of the DORIS observations have a format like (2.55), involving state vectors and partials at two different epochs. From (5.13) it follows that :

$$\frac{\partial C_i}{\partial P} = \frac{1}{\Delta t}\left(\frac{\partial \rho(t_0+\Delta t)}{\partial P} - \frac{\partial \rho(t_0)}{\partial P}\right) \tag{5.20}$$

The partials for each of the two epochs are then computed from

$$\frac{1}{\Delta t}\frac{\partial \rho}{\partial P} = \frac{1}{\Delta t}\left(\frac{\partial \rho}{\partial x}\frac{\partial x}{\partial P} + \frac{\partial \rho}{\partial y}\frac{\partial y}{\partial P} + \frac{\partial \rho}{\partial z}\frac{\partial z}{\partial P}\right) \tag{5.21}$$

Because the numerical integration process will not produce all required variational partials at once, those for the epoch t_0 will have to be stored in some way until the entire measurement can be processed at the epoch $t_0 + \Delta t$ when the integrator has produced the second set of partials. The arrangement of DORIS stations over the globe is such that it is not possible for the satellite to be within measurement range of more than 4 ground stations at any time. For this reason, the amount of vectors with variational partials that may have to be buffered within the computation process will never exceed 4 and will in fact rarely be more than one or two at a time. Fortunately, the size of a vector of variational partials is insignificant in comparison to the size of the normal matrix itself, so Faust can store these few vectors of partials internally. The geometrical partials - i.e. the unit vector in the direction of the range, as computed from (5.8) - are divided by the factor Δt as required by (5.21), and also stored.

With respect to the internal organisation of the code, storing the partials at the start of a DORIS interval is implemented as a separate 'event' in Figure 3.3. This event involves producing the required partials, storing them in one of four available slots for such partials, and submitting an internal request for another event that will take place at the epoch of the end of the Doppler count interval. If that second event time is reached, the actual DORIS observation is processed and added to the normal matrix, while the slot for the involved partials is released for future use.

5.4 PRARE range and range-rate

Although the PRARE equipment on ERS-1 failed shortly after launch, its successor ERS-2 is carrying a similar but functioning instrument, and after some initial problems with the PRARE ground segment the system is now fully operational for ERS-2. Although this thesis focuses in particular on simultaneous solutions between ERS-1 and TOPEX/Poseidon, the PRARE tracking data has also been implemented in Faust, mainly in support of other research at Aston. As the acronym suggests, PRARE provides both range and range-rate measurements. The two-way range observations are processed in a way almost identical to SLR range, while the treatment of the range-rate data is very similar to the DORIS observations described above. The fundamental computational difference in both cases is that the direction of the two-way ranges is opposite, i.e. the space segment forms the active transmitter / receiver while the ground station acts as transponder. In order to avoid interference between signals from different ground stations, the system uses four different dual-frequency microwave signals, allowing up to four stations to track simultaneously.

The range data is corrected by a range bias per station, solved for in analogy to (5.2). Because these range biases should not vary significantly in time, only one such parameter is solved for per 5-day arc during which each station may provide several PRARE range passes. In addition, both the range and the range-rate observations include a tropospheric correction term that is provided on each input data record. As with DORIS, this correction is fine-tuned by means of a tropospheric correction parameter K_{trop} per pass, modelled in the form of a scale factor to the nominal tropospheric correction. The tropospheric correction ΔR to the range observation is then

$$R_{tot} = R_{obs} + K_{trop} . \Delta R_{input} \tag{5.22}$$

from which we immediately obtain the corresponding explicit partial in (2.50) as ΔR_{input} itself. A single pass over a station usually provides both range observations and range-rate observations, but the same tropospheric scale factor K_{trop} is used to tune both data types in the pass. For the range-rate observation the dimensions of the nominal correction (as obtained from the input data record) are of course different from those of a range observation. Note that for DORIS the tropospheric correction was already included in the observation value, and the scale factors $(K_{trop})_{DORIS}$ for the additional correction will therefore be close to zero. For PRARE, the correction has not yet been included in the input value, so the scale factors will be close to unity.

Given the dense global coverage by PRARE stations, the amount of scale factors K_{trop} will be in the order of 100 to 150 per 5-day arc. As a result, simultaneous solutions that include a full 35-day cycle of ERS-2 will contain a very large amount of these parameters, but fortunately we can assume that the scale factors for different passes are independent of each other. The parameters K_{trop} for the PRARE data therefore form parameters of Category 3 in Section 2.3, and because of the adopted normal matrix partitioning the memory requirements will remain within realistic limits. The treatment of partial constructions of the form (2.55) as required for PRARE range-rate is implemented in Faust in exactly the same way as was done for DORIS, and described in the previous subsection.

5.5 Radar altimetry

In general altimetry is not considered as a tracking data type for orbit determination but specifically reserved for oceanographic and geodetic modelling. This is merely a political choice, based upon the desire to avoid aliasing of geodetic signals into an orbit solution fitted to altimetry data. The relevance of altimetry to precise orbit determination will therefore be restricted to the use in crossovers, which will be discussed in the next section. Altimetry data differs from most other observation types in the sense that it is an entirely satellite based system that does not involve ground stations. The terrestrial reference point is formed by the ocean surface rather than by ground station coordinates, which means that precise geometric modelling of the ocean surface is required within the evaluation of C_i. In practice, a preprocessing step translates the observed measurement O_i into a measurement of the height of the satellite over the reference ellipsoid by correcting the raw observation for the local height of the marine geoid, for ocean tides, for the dynamic surface topography and for significant wave height. This preprocessing simplifies the computation of the calculated observation C_i, which can now easily be determined from the integrated satellite position as the instantaneous geodetic height of the satellite. This Section will describe the relevant parts of the geometrical model for the altimeter observation, in particular the modelling of the instantaneous ocean surface and that of the travel path through the atmosphere. The model for the space segment only includes an instrument range bias, and possibly a time tag bias.

5.5.1 The ocean surface

A satellite radar altimeter measurement is taken along the momentary geodetic orbital radius, i.e. along the line perpendicular to the ellipsoid within the longitudinal plane through the satellite. In a preprocessing stage, the raw measurement is translated into a value for the height above the ellipsoid (the geodetic height) rather than for the height over the instantaneous ocean surface. Apart from a correction for the significant wave height, which is provided in the GDR data record, the applied ocean surface corrections include :

(1) The height of the geoid relative to the reference ellipsoid (the geoid undulation N). The geoid is determined by the combined effects of the Earth's gravity field and its daily rotation (see Section 4.1). At present, spherical expansion models for the geoid are available up to degree and order 360 (e.g. Rapp *et al.*, 1991; Lemoine *et al.*, 1996). Apart from the gravity-induced orbit error - observed via the equations (4.19) and the resulting implicit term in (2.50) - the shape of the geoid is also observable from altimeter data, implying that for altimetry observations an explicit term can be included in (2.50). To this purpose, the gravity potential U from (4.4) is interpreted as a combination of a potential W at the ellipsoid and a disturbing potential T. The actual scalar *value* of W is chosen identical to the value of the potential U, so that W approximates the real gravity potential as closely as possible. The gravity force due to the potential W - perpendicular to the ellipsoid - is called the normal gravity, while the force due to the potential T is called the gravity anomaly :

$$-\nabla U = \underline{a}_{grav} = \underline{a}_{normal} + \underline{a}_{anomaly} = -\nabla W - \nabla T \tag{5.23}$$

We can now write the total potential at the geoid not only as $U = W + T$, but also in terms of a first order series expansion of the potential W of the ellipsoid :

$$\left.\begin{aligned} U &= W + T \\ U &= W + (\nabla W, \underline{n}_e).N = W + N\gamma \end{aligned}\right\} \rightarrow \; N \equiv \frac{T}{\gamma} \tag{5.24}$$

The unit vector $\underline{n}_e$ in (5.24) gives the direction perpendicular to the ellipsoid, so that the projection $\gamma = \left(\nabla W, \underline{n}_e\right)$ forms the component of ∇W in the direction of N. Because ∇W is *by definition* perpendicular to the ellipsoid, γ is in fact equal to the normal gravity in the point of interest. The right-hand side expression for N in (5.24) is known as the equation of Bruns, and shows that the geoid undulation is determined by the ratio of the disturbing potential and the normal gravity.

Because N is subtracted from the observed value O_i in the preprocessing stage, it is present in the calculated measurement C_i with a positive sign. The explicit partials for (2.50) are therefore available for each parameter P_j (being either C_{lm} or S_{lm}) by means of the geoid height partials :

$$\frac{\partial C_i}{\partial P_j} = \frac{\partial\left(N + h_{tides} + \dots\right)}{\partial P_j} = \frac{\partial N}{\partial P_j} = \frac{\partial}{\partial P_j}\left(\frac{T}{\gamma}\right) \qquad (5.25)$$

In practice, the disturbing potential and normal gravity are found by projecting the gravity vector due to U into directions perpendicular and tangential to the adopted ellipsoid model.

(2) The variations in the sea surface height due to tides. The total geometrical tidal uplift accounts for all *periodic* ocean surface height variations (see Section 4.2) and possibly the constant polar tide if it is not included in the geoid model. This total uplift includes the movements caused by the solid Earth tides of the ocean floor, those related to the liquid ocean tides, as well as effects like inverse barometer depression and ocean loading. In practice, the model to correct the altimetry measurements for tides may be less detailed than the model used during the orbit integration process, because only the larger tidal constituents produce tidal uplifts of a significant level, in relation to other noise components in the dataset. For the same reason, it not useful to include explicit terms like (5.25) for the tides, as the geometrical signal of most constituents is not observable from altimetry measurements. The tides are therefore only observed through the implicit term of (2.50), i.e. through their effect upon the satellite orbit.

(3) The dynamic sea surface topography. Non-periodic surface elevations that are not part of the geoid undulation can be subscribed to surface slopes caused by geostrophic ocean currents or large scale ocean eddies. The maximum height differences associated with this dynamic sea surface topography are about two metres globally, which illustrates that the geoid must be known with great precision before it becomes realistic to explicitly superimpose a dynamic surface topography. It is only since the arrival of the high precision satellite altimetry of ERS-1 and TOPEX/Poseidon and accurate gravity field models like JGM-3 that reliable global sea surface topography models can be determined (Romay-Merino *et al.*, 1993). The surface topography model itself is modelled by a straightforward geometrical expansion model in Legendre functions :

$$h_{sst} = \sum_{l=0}^{L} \sum_{m=0}^{M} \left[C_{lm} \cos m\lambda + S_{lm} \sin m\lambda \right] P_{lm}(\sin\phi) \tag{5.26}$$

The partials to solve for a sea surface topography model - as easily computed from (5.26) and Table 4.1 - are included in Faust. Note that these partials are only included in the form of an explicit term in (2.50), as changes in the gravity field due to the sea surface topography are assumed to be negligible.

5.5.2 Atmospheric corrections

The dry tropospheric correction for radar altimetry measurements is generally computed from straightforward empirical formulae like one given by Cheney *et al.* (1994) :

$$\Delta r = -2.277p.(1 + 0.0026\cos 2\phi) \tag{5.27}$$

in which p and ϕ are the local atmospheric pressure (mb) and geocentric latitude repectively.

Wet tropospheric corrections are either based upon meteorological data or upon actual measurements of the water vapor content along the line of sight, by means of on-board microwave radiometer instruments. Only the latter method is capable of following short-term variations in the tropospheric conditions, and will therefore be more reliable. For TOPEX/Poseidon and for the ERS satellites, a dry and wet tropospheric correction term are provided in the input GDR data records. In Faust, no model parameters for either the dry or wet tropospheric correction can be solved for, so errors in these corrections are present in the altimetry signal without change.

The ionospheric correction for the TOPEX altimeter is derived from the behavior of its dual-frequency signal (Imel, 1994) with an accuracy of 5 to 15 millimetres. For the single-frequency ERS altimeters such information is unavailable. Instead, a correction term based upon the model described by Bent *et al.* (1973) is applied. This model is inevitably less accurate, and will therefore introduce a noise component in the dual satellite crossovers between ERS and TOPEX.

5.5.3 Biases

The altimeter instrument introduces a non-zero electronic travel time delay which is compensated by solving for an instrument range bias according to (5.2). In addition, a time tag bias (5.3) has been identified in the ERS GDR time stamp, with a typical value of -1.2 milliseconds. For TOPEX/Poseidon the time tag bias is negligible, although it may be solved for as a precaution. Note that the time tag bias for altimetry data is in fact computed from altimetry crossovers, although once that it is available it is also applied to normal altimetry measurements that may be included in geophysical applications of Faust.

5.6 Crossover height differences

As explained in Chapter 1, crossover differences are obtained by subtracting two altimeter measurements that are taken above the same point on Earth at different epochs. This Section will discuss the two subjects of analysis of error components in the crossover datasets, and of numerical treatment of the crossover data in Faust.

5.6.1 Crossover error components

The total error in the calculated crossover difference contains components of radial orbit error, atmospheric propagation errors and sea surface variability errors. For near-circular orbits the effect of radial orbit errors in the crossings of a single satellite crossover may be approximated in first order (e.g. Jolly, 1995) by

$$\epsilon_{XO} = \epsilon_{h_1} - \epsilon_{h_2} = \left(A\cos M_1 + B\sin M_1 + C\right) - \left(A\cos M_2 + B\sin M_2 + C\right) \qquad (5.28)$$

If the mean anomaly M of the satellite is set to be zero at the point of highest latitude, the intersection of an ascending and descending pass implies the symmetry $M_1 = -M_2$, and the error term degenerates into

$$\epsilon_{XO} = 2B\sin M_1 \qquad (5.29)$$

Even if (5.28) would be extended into a series of higher order sine and cosine terms, the latitudinal symmetry will still eliminate all cosine terms $\cos nM$, including the constant bias C which may be interpreted as a zero-order cosine. At the same time, all sine terms are observable at twice their normal amplitude, as seen from (5.29). A fully geographically correlated radial orbit error will appear as an even function of M (i.e. a cosine series), and is therefore totally absent from the single satellite crossover dataset. However, geographically anti-correlated orbit errors show up as odd functions of M (i.e. sine series) and are present in the crossover differences at double intensity. For dual satellite crossovers the coefficients

A and B of (5.28) will not be identical for both orbits, which implies that the geographically correlated radial orbit error does *not* completely cancel out if the height difference is computed. This feature forms the crucial contribution of simultaneous solution techniques to gravity field refinement, as will be discussed in more detail in Chapter 8.

From the above it follows that an indication of the geographically fully anti-correlated orbit error can be obtained from the single satellite crossovers, by separately averaging the SXO residuals from ascending passes and descending passes over a certain area (to account for their differences in the sign of the mean anomaly M) and dividing by two. An indication of the geographically correlated orbit error is obtained from the even component of the *dual* crossover residuals, i.e. it follows by direct averaging of the DXO residuals over a certain area.

With respect to atmospheric propagation errors, Stum (1994) showed that the wet tropospheric corrections for ERS and TOPEX are not significantly different in quality, resulting in a noise component in the dual crossovers with an RMS below one centimetre. It has already been mentioned that the single-frequency instrument of the ERS satellites will introduce ionospheric correction errors in the dual crossover dataset. These errors will mainly show up as noise, especially because the temporal distribution of the crossovers is effectively independent from the spatial distribution.

5.6.2 Numerical processing of crossovers in Faust

The crossovers enter Faust in chronological order of the first crossing, i.e. of the first epoch for which vectors for state and variational partials are required. These vectors will have to be temporarily stored in some way until the integration process reaches the epoch of the second crossing, at which point the crossover observation can be processed and its contribution added to the normal matrix. While solving this buffering problem for DORIS and PRARE range rate data was relatively easy - because of the limited amount of data that had to be stored - the time interval between the crossings in a crossover point can be up to several days, rather than the 10 seconds or so for a Doppler count interval. The result is that the amount of data that may have to buffered at any particular time will exceed realistic RAM memory requirements for present computer systems (see Figure 5.2), especially in a network environment.

Within the present computer configuration at Aston, the only realistic option is to create the crossover buffer as a (temporary) file on disk, writing the state and partials for the first crossing to disk and reading them back at the time of the second crossing. The way in which this buffer system is organised will be briefly described. As illustrated in Figure 5.3, the main elements of the buffer system are a direct access file on disk, an array with all epochs of second crossings for the crossovers that are waiting to be processed, a parallel array with the numbers of the records in which the corresponding buffer data is stored, and an array with flags to indicate all records of the file that are currently used.

Amount of variational equations	6000
Amount of elements in triangular normal matrix	$\frac{1}{2} \times 6000^2 = 18 \times 10^6$
Amount of crossovers within 5-day maximum interval	
ERS-1 single satellite crossovers	1500
TOPEX/Poseidon single satellite crossovers	2500
Dual satellite crossovers	4000
TOTAL	8000
Required buffer capacity	± 0.5 x 8000 = 4000
Storage space	
Size of triangular normal matrix in RAM memory	8 x 18 MB = 144 MB
Variational partials for one parameter	3 x 8 = 24 Bytes
Variational partials for one crossing	6000 x 24 B = 144 kB
Total crossover buffer size for all sets of variational partials	4000 x 144 kB = 576 MB

Figure 5.2 Example computation for required memory sizes in a simultaneous solution process for ERS-1 and TOPEX/Poseidon, if partials for gravity field and tides are included.

An initial crossover event will occur when the integrator reaches the epoch of the first crossing. At that time, the state vector and partials will be written to the buffer file in the first available record, as indicated by the flags. The epoch of the second crossing as well as the used record number will be stored in the first available elements of the respective memory arrays. The crucial design principle is that the first time tag in the array must always be the earliest of all, i.e. it will indicate the next crossover that has to be processed as an observation. When Faust arrives at this epoch, the state and partials of the first crossing will be read back from the buffer record that is indicated by the first element of the array with record numbers. The state vector and partials for the second epoch are at that moment available from the integration process, so that the entire equation (2.55) can be constructed.

When a record is read back from the buffer file, its record number is released for future use by toggling the corresponding element of the array with flags. This ensures that the buffer file will never grow larger than required for storing the maximum amount of records that is ever buffered simultaneously. At the start of the processing sequence in Faust, the rate at which buffer records are created will be much larger than the rate at which the second crossover epochs are reached, so that the buffer file will grow in size. At a certain moment the rate of processing will catch up with the rate at which new records are written, typically after integrating over the tolerated maximum time span between crossings.

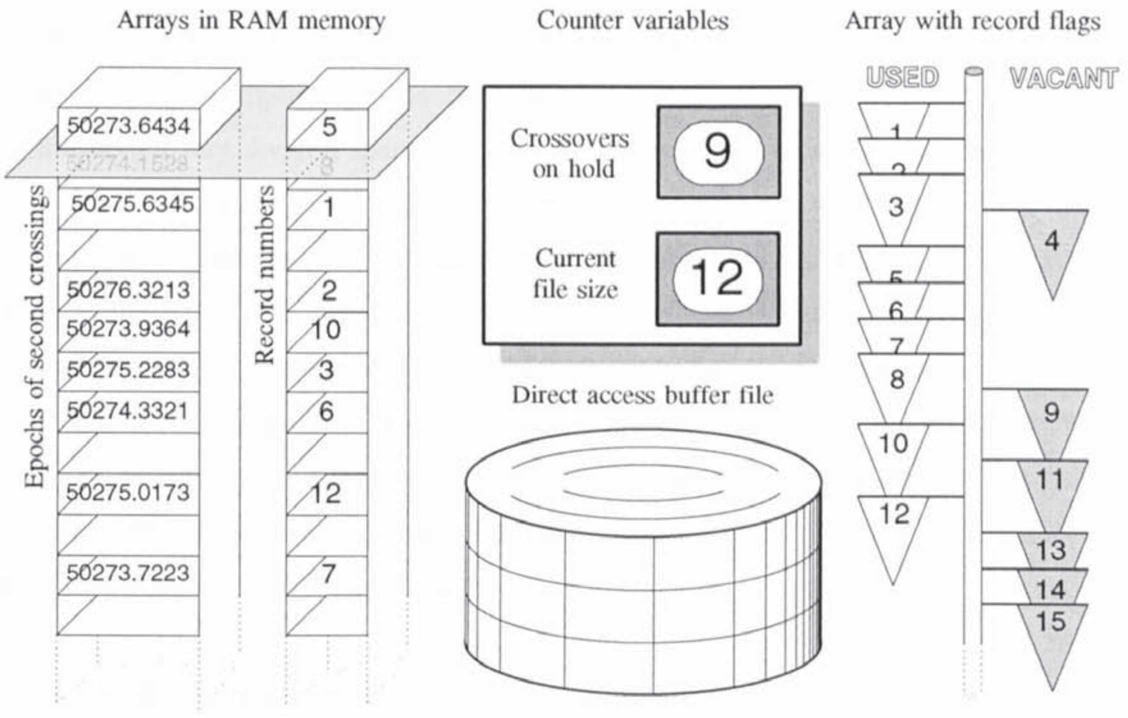


Figure 5.3 The elements of the crossover buffer system. Only the first time tag and record index are relevant to the event handling sequence of Faust.

From that moment on a steady state situation will be maintained, in which it is no longer necessary to create new records at the end of the buffer file. Without this recycling of released records the buffer file would eventually grow to a size that contains as many records as there are crossover observations in the entire solution, which will often exceed realistic limits for disk usage.

After processing a crossover observation at the epoch of the second crossing, the entire array with time tags is scanned to find the earliest of all elements, and this element is then moved into the first array position that has become vacant, both for the array with time tags and for the array with record numbers. Also, when a new record is added to the buffer, a test is performed to make sure that the corresponding second epoch is not earlier than the time tag for the first array elements. These two measures will ensure that at any time the first array elements correspond to the earliest of all waiting crossover observations, and no further sorting of the buffer data is required.

5.7 Wagner

The choice to design Faust as an event handler that processes observation data in strict chronological order implies that the observations have to be offered to the main programme in a time-sorted way. Even if individual tracking datasets are available in chronological order, which is not always the case, it will be necessary to merge the different tracking datatypes into a single coherent dataset that can be used as time-sorted input to Faust. The format of this input observation file must be chosen carefully in order to allow different data types to be stored within a single file in an unambiguous way, however without requiring excessive record lengths. To this purpose, a preprocessor programme has been written - named Wagner, after Faust's assistant - that also allows some other manipulations of the tracking data, like eliminating observations from one or more stations or satellites, removing duplicate records if overlapping files are merged, etcetera.

The internal data format used by Faust is very straightforward, and allows future extensions for new tracking data types without losing downward compatibility. The file is unformatted and sequential, and each record contains :

- A UTC time tag
- A measurement type identifier number

- n_1 satellite identification numbers	$0 \le n_1 \le n_{\max}$
- n_2 station identification numbers	$0 \le n_2 \le n_{\max}$
- n_3 measurement components	$1 \le n_3 \le n_{\max}$
- n_4 real data numbers	$0 \le n_4 \le n_{\max}$
- n_5 integer data numbers	$0 \le n_5 \le n_{\max}$

The numbers n_1 to n_5 determine the total record size, and are stored in a header record. After opening the observation file, Faust - or any other programme using these files - will first read the five numbers n_j to find out what data is stored in each record. At the location in the code where data is read from the file, implied loops are used to read 'zero or more' fields for satellite numbers, station numbers, etcetera. The limit $n_{\max}$ only exists in the *programme*, not in the data, and can therefore easily be increased in the future without rendering older data files useless. The fields with tracking type dependent real or integer arguments may contain any data that can not be stored in the other fields. As an example, the second time tag of a crossover observation may be stored as the first real data argument, as long as Wagner and Faust both know what this argument represents for a crossover observation. For DORIS, this same record field might contain the duration of the count interval.

Each data type will require a certain minimum record size that is essentially determined by series of the numbers n_1 to n_5, hardcoded for each datatype in Wagner. If different datatypes are to be combined into a single datafile for Faust, Wagner will compute the smallest possible numbers n_i that can hold all data types, and create the input file to Faust in that format. If laser data (one satellite and one station per record) is to be combined into a single file with crossover data (two satellites, no station) each record in the combined data file will have two satellite fields and one station field. As a result, records for both data types will have certain fields that contain zeroes, and this general data format is therefore *not* optimal in terms of storage (at least not for a file that contains several data types with different record formats). Permanently stored data, if stored in this format, should therefore be separated into files per tracking data type. The combined observations file can then be produced just before running Faust, given the fact that even for very large files Wagner will require a runtime of seconds rather than minutes.

The way in which wagner operates is illustrated in the Figures 5.4 and 5.5. The temporary files that are produced for each input file - indicated by the block **Save to tmp** file in Figure 5.5 - are not stored in the format of the final output file, but rather in the smallest possible format for that particular datatype. While reading a raw data input file and copying it into the temporary file, Wagner will perform several data validation tests, if requested in the user input. At this point - shown as the block **filter or edit** in Figure 5.5 - Wagner will for instance reject data records that should be filtered out. In addition a simple test is performed to see if the UTC time tag of the record is not earlier than that of the previous record, so that sorting of datafiles can be avoided when an input file is already sorted on time. Finally, Wagner will merge all temporary files into one single output file, making sure that all records are in chronological order and that no duplicates are written to the output. For convenience, a report file will be produced that contains statistics of the data contents listed per data type, per satellite and per station.

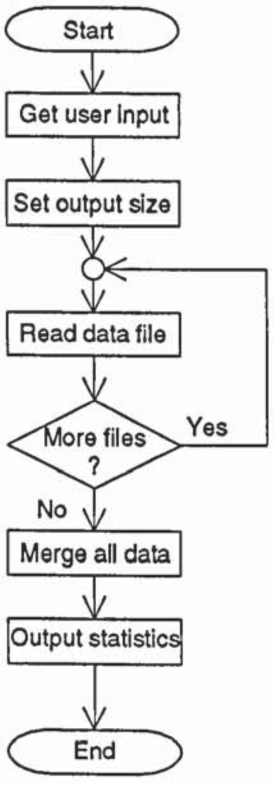


Figure 5.4 Wagner

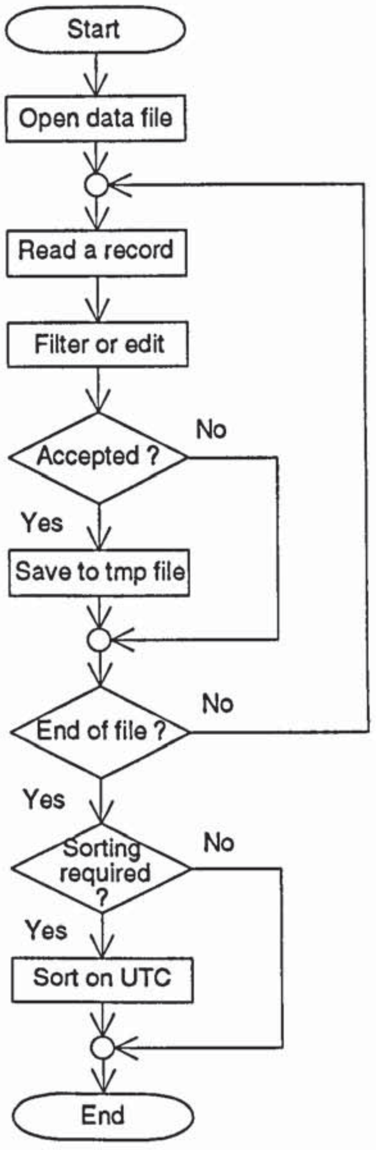


Figure 5.5 'Read data file'

The UTC time tag on which the records will be sorted must always be the earliest epoch involved in a possible partial construction like (2.55), in order to allow all involved state vectors and partials to be determined by the integration process in strict chronological order. For doppler measurements this time tag will therefore relate to the start of the doppler count interval, and for crossovers it will be the time of the earliest crossing.

Wagner uses a natural language interface that is identical to the one that is used by Faust (see Appendix C), although it will obviously use a different control language database. One of the reasons for using the same interface - apart from convenience for the user - was to allow for the possibility that in the future Wagner may be fully integrated with Faust into a single programme that will then share a single user input module. At present, the time saved by separating the two still seems to be more efficient, although especially for large multi-arc runs the very short runtime of Wagner is negligible in comparison to that of Faust.

5.8 Numerical aspects of observation processing

Apart from the computation of residuals and geometrical partials, which is done differently for each tracking data type, many aspects of observation processing, like data weighting, editing or adding a contribution to the normal matrix, are identical for all observations. Many of these computational aspects are therefore centralised in Faust. This also makes it very straightforward to implement new tracking data types in Faust, as most of the processing infrastructure will already be present in the code. Figure 5.6 shows the handling sequence of the 'event' that is formed by the occurrence of a single observation during the chain of events depicted in Figure 2.6. For observations that depend on different epochs (DORIS, PRARE doppler and crossovers) this happens at the last epoch for which state vectors and partials had to be computed, so that at the start of this event all required elements for (2.55) are available. In a preparatory subroutine, a variety of housekeeping variables within the code will be set, like the number of the arc to which the observation is related, the index of satellites and stations that are involved in the observation, the related standard deviation as defined in the user input, a pass number, etcetera. This preparation step is identical for all data types.

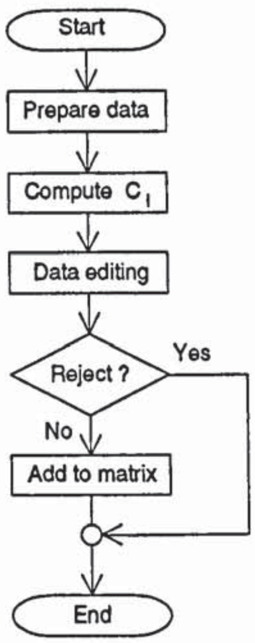


Figure 5.6 Sequence for each observation

For each data type supported by Faust, the computation of residuals and geometrical partials is different and therefore performed in separate subroutines. If a new data type is to be included in the programme, it is in particular this step - indicated as the block **Compute $\mathbf{C}_i$** in Figure 5.6 - that will have to be implemented. Its output is formed by the measurement residual and the partials $\frac{\partial C_i}{\partial x}$ for the construction of (2.55).

It is important to have the possibility to control the influence that individual data types have upon the outcome of the least squares process. In principle this is done by means of the a priori standard deviations σ_i discussed in Section 2.2. However, known differences in data quality or achievable orbital precision will make it desirable to have various other control variables, to allow weighting of different satellites, stations or data types with respect to each other. In Faust this is done by different types of standard deviations and scale factors.

A nominal standard deviation for (2.15) is defined in the user input for each tracking data type. For land-based tracking types, it may be desirable to have different a priori standard deviations for different stations, because the data quality from some stations may be known to be systematically better or worse than that from other stations. For that reason, it is possible to define a standard deviation per station, which - if defined in the input - will overrule the default standard deviation for that data type. As an example, we could define a standard deviation of 20 cm for SLR data, but set a larger standard deviations for some new stations of which the station coordinates are not yet known with great precision. Stations that are not defined explicitly will then adopt the nominal value of 20 cm.

It can happen that of two satellites in a simultaneous solution process one will always have a better orbital precision than the other, for instance because of substantial differences in orbital height. For that reason the standard deviation for a certain observation is multiplied with a scale factor *per satellite*, which is a real number with a nominal value of one. This scale factor will increase or decrease the nominal standard deviation described above, so that its square will appear in the weighted normal equation (2.12). In the example given above with a standard deviation of 20 cm for laser data, we could give TOPEX/Poseidon a scale factor of 0.75 and ERS-1 a scale factor of 1.00. The result would be that the actual standard deviation for TOPEX laser data would be 15 cm instead of 20 cm.

Finally, we may want to *emphasize* a single tracking data type within the solution, for instance if the amount of observations of this type is substantially smaller than that of another data type (e.g. SLR data versus DORIS data). To this purpose, it is possible to define a scale factor for a *tracking data type*, which is used to scale the equation (2.20) directly before adding it to the normal matrix. Note that this scale factor is *not* squared in the normal equation. In the next Chapters, scale factors will be used to balance a solution according to the amount and quality of data for each tracking type.

It is important to clearly distinguish between the four different control parameters :

(1) A *standard deviation for each data type*, which is the nominal σ from (2.15),
(2) A *standard deviation per station*, which will simply replace the σ from (1) if it is defined for specific stations,
(3) A *scale factor per satellite*, with which the σ from (1) or (2) is multiplied,
and (4) A *scale factor per data type*, used to increase the influence of a single data type within the solution by direct scaling of the normal matrix contributions.

Although at first sight this combination of different standard deviations and scale factors may seem rather inconsistent or confusing, it has in fact evolved in practice as the most efficient way to have direct control over the influence of individual satellites, stations or data types.

One of the reasons to have these different control parameters is to allow efficient automatic data editing. The data editing rules used by Faust are fairly straightforward but should always be used with caution. The three rejection criteria used for data editing are

(1) Automatic rejection of absurd values
(2) Window rejection, using an absolute rejection level that is defined *per datatype*
(3) Gaussian rejection, using a relative rejection level with respect to the RMS value of the residuals in the previous iteration, *per datatype* and *per satellite*

The first type of rejection is always applied, although the user can set the level of residuals that should be considered as 'absurd' in the input. This editing rule forms a safety net to eliminate obviously wrong observations that sometimes slip through the normal point generation sequence. The process iteration in which the two other rejection criteria will first be applied - if at all - can be set in the user input.

The window rejection is 'crude but effective' and allows a straightforward elimination of all observations with a residual above a certain absolute value, specified for each tracking data type separately. At present, Faust does not include the possibility to set different window levels for different *satellites*, as this is not really necessary for simultaneous solutions between ERS and TOPEX/Poseidon.

The third type of rejection assumes that the residuals have a Gaussian distribution, as discussed in Section 2.2, and that residuals that are larger than a value $f_G.\sigma_{RMS}$ should be rejeceted. The standard deviation σ_{RMS} is then the RMS of all residuals of a certain datatype and related to a certain satellite, as found in the previous process iteration. In a first iteration, the a priori standard deviation σ_{input} would be used. It is for this reason that *scale factors per datatype* are introduced in the above : if we would attempt to emphasize a specific datatype by defining an unrealistically small standard deviation - in order to obtain a large weight in (2.14) - the Gaussian rejection can no longer be used, because most observations would fall outside the $f_G.\sigma_{RMS}$ band. Under all circumstances, the a priori value of σ should be realistic for the specific combination of data type, station and satellite in order for the editing process to work properly.

Multi-arc orbit determination — Chapter 6

The previous Chapters have treated the full theory behind dynamic multi-arc / multi-satellite solutions for altimeter missions, and the essential details of their implementation in Faust. In three subsequent Chapters the theory will now be applied to specific parameter estimations tasks, in order to analyse the added value of simultaneous multi-arc methods for precise orbit determination, and for applications of radar altimetry data to space geodesy.

In Chapter 1 it was explained that the necessity for multi-arc solutions is a direct consequence of the choice to apply dual satellite crossovers for the simultaneous adjustment of both involved satellite orbits, if loss of data density is to be avoided. However, once that the multi-arc solution has been adopted as the default approach to precise orbit determination, even the single satellite orbit case can benefit from additional information that is available in a multi-arc solution. This Chapter will show how various inaccuracies in a single-arc orbit determination process are related to the finite length of computed arcs, and how such problems may be overcome in a multi-arc process by using knowledge related to the behavior of the satellite's orbit before or after the time interval covered by a single arc. In addition, some practical problems related to multi-arc solutions will be discussed.

6.1 Crossover data density

For altimetry satellites, the crucial difference between single-arc and multi-arc solutions is the availability of crossovers between successive arcs. As illustrated in Figures 1.2 and 6.1, the size of a crossover dataset in a multi-arc solution will be about twice that of a single-arc solution, for an arc length of five days used for the ERS satellites and a maximum time interval between crossings also set to five days. Figures 6.2 a to f visualize the spatial data coverages for ERS-1 during a five day arc, for various upper limits to the interval between crossings. For the three types of crossovers that are relevant to this study, being single satellite crossovers for ERS-1 and TOPEX as well as dual crossovers between the two, the relation between the crossover interval and the data density has been analysed.

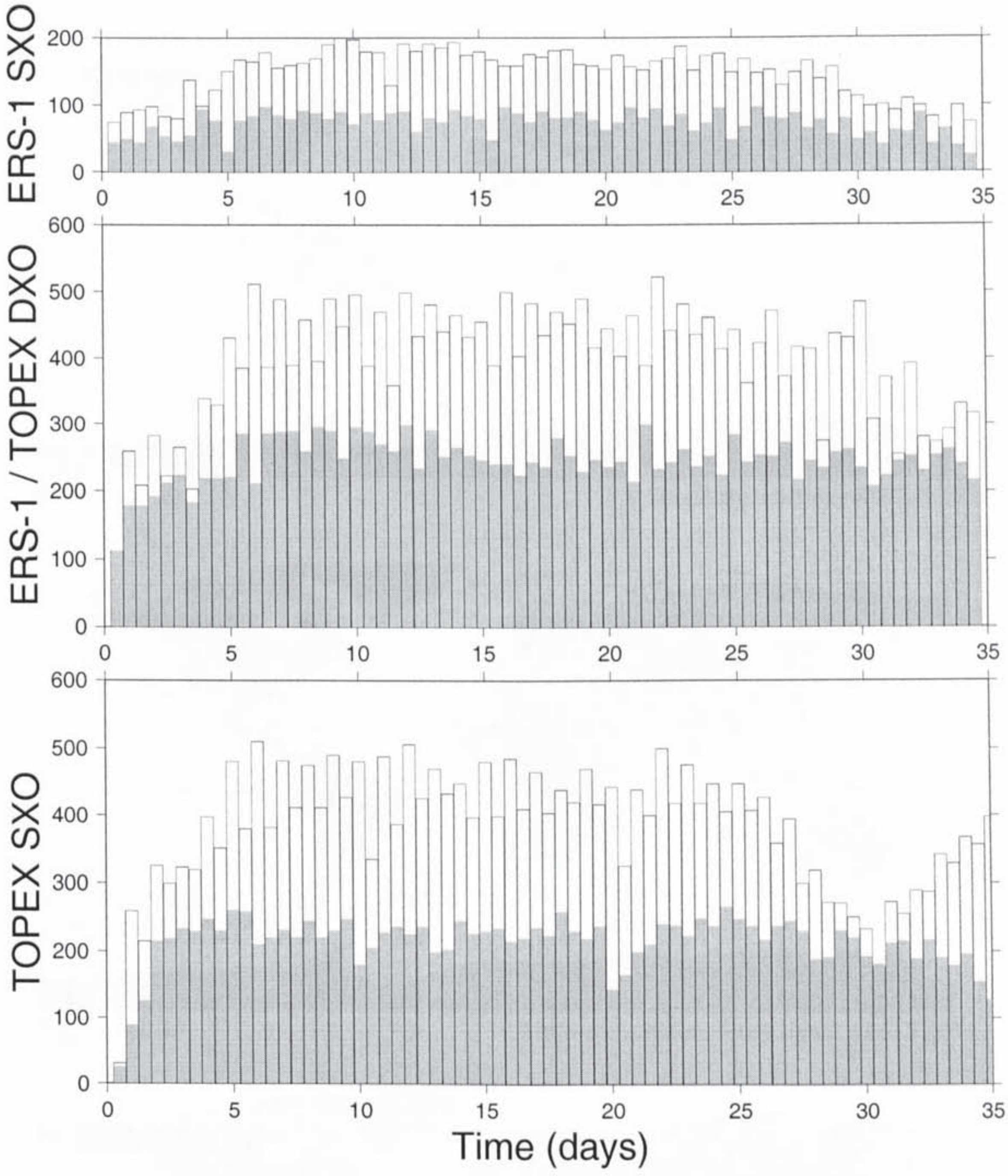


Figure 6.1 Typical amounts of crossovers during 12-hour intervals. Shaded bars represent the crossover density in single arc solutions, while the white bars represent the available crossovers in multi-arc solutions. All datasets have been generated with a 5-day upper limit to the interval between crossings.

As shown in Figure 6.3, the data density is more or less proportional to the tolerated time span between the crossings, which will be true at least until this interval approaches the groundtrack repeat period. To obtain the same data density as feasible in single arc solutions, we can reduce the tolerated time span between crossings to about half the interval used in single arc solutions. Alternatively, we can substantially increase the crossover data density if

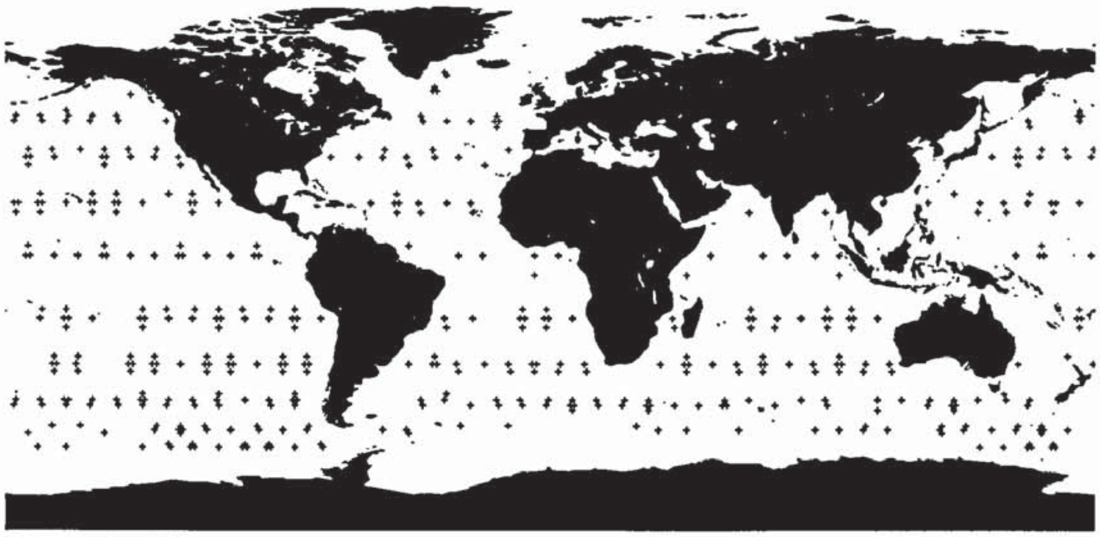


Figure 6.2 (a) Single arc solution : 489 SXO

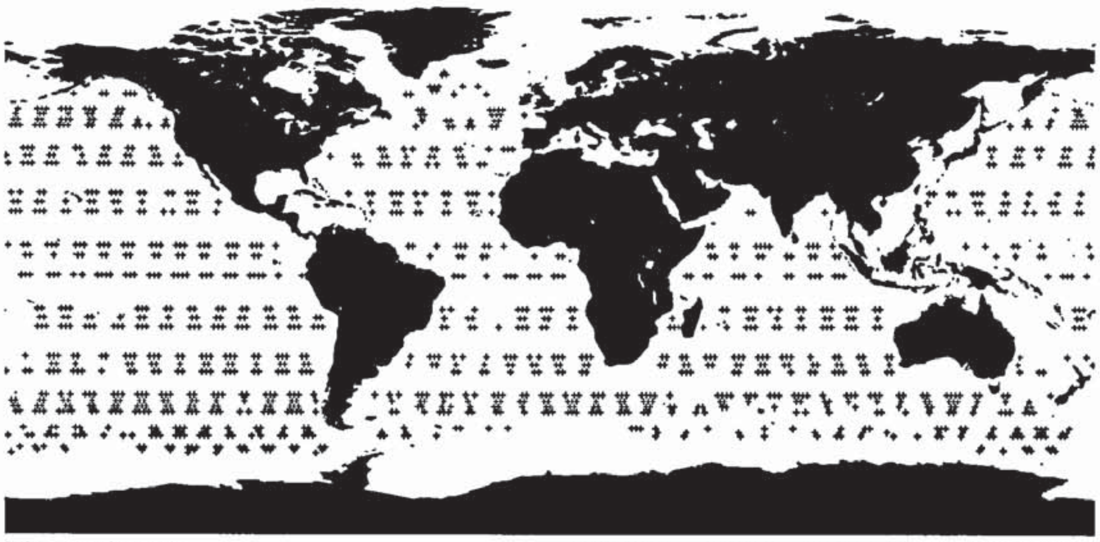


Figure 6.2 (b) Multi-arc solution, 5-day limit : 1220 SXO

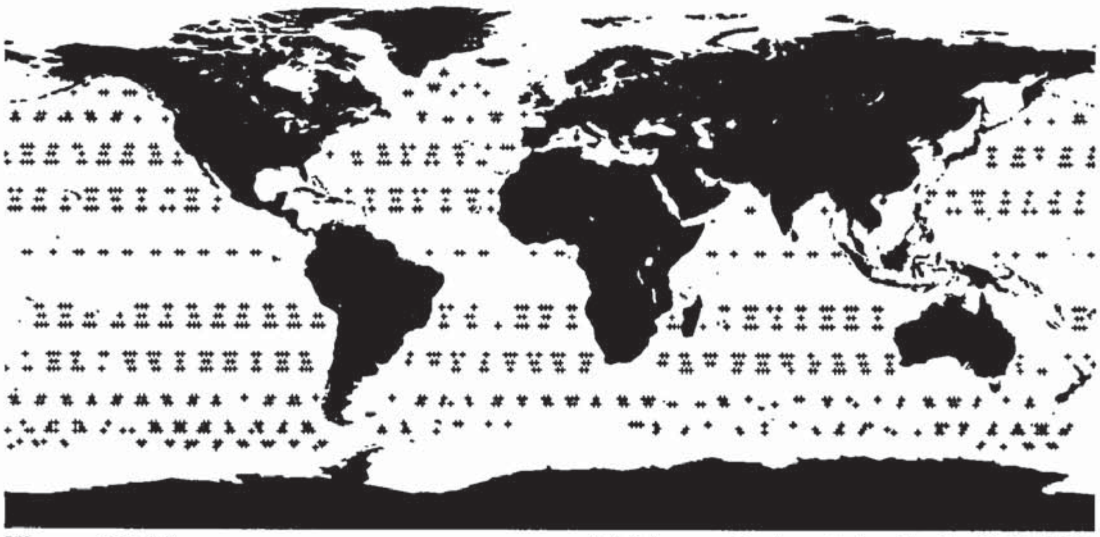


Figure 6.2 (c) Multi-arc solution, 4-day limit : 966 SXO

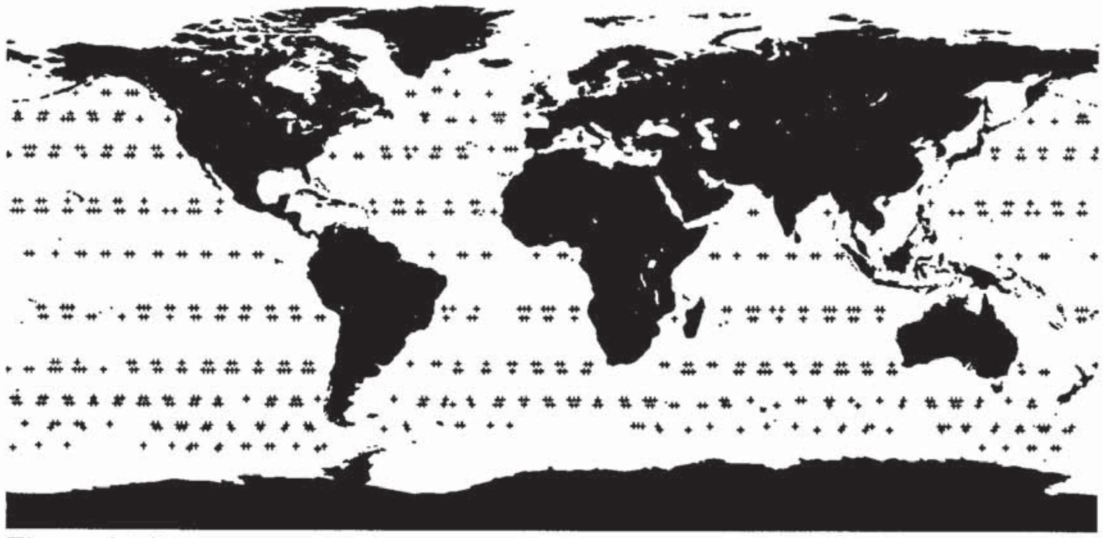


Figure 6.2 (d) Multi-arc solution, 3-day limit : 659 SXO

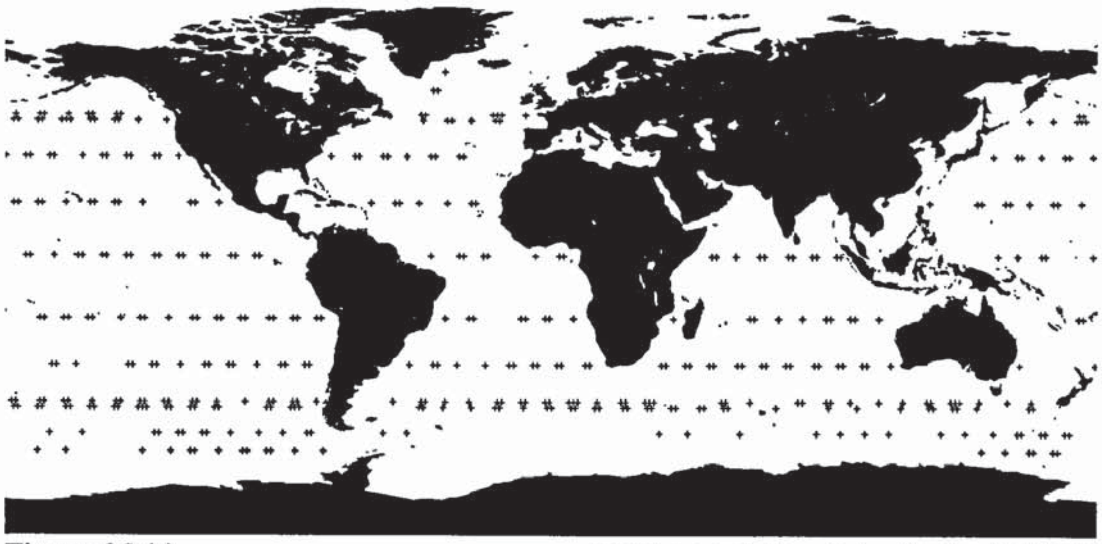


Figure 6.2 (e) Multi-arc solution, 2-day limit : 407 SXO

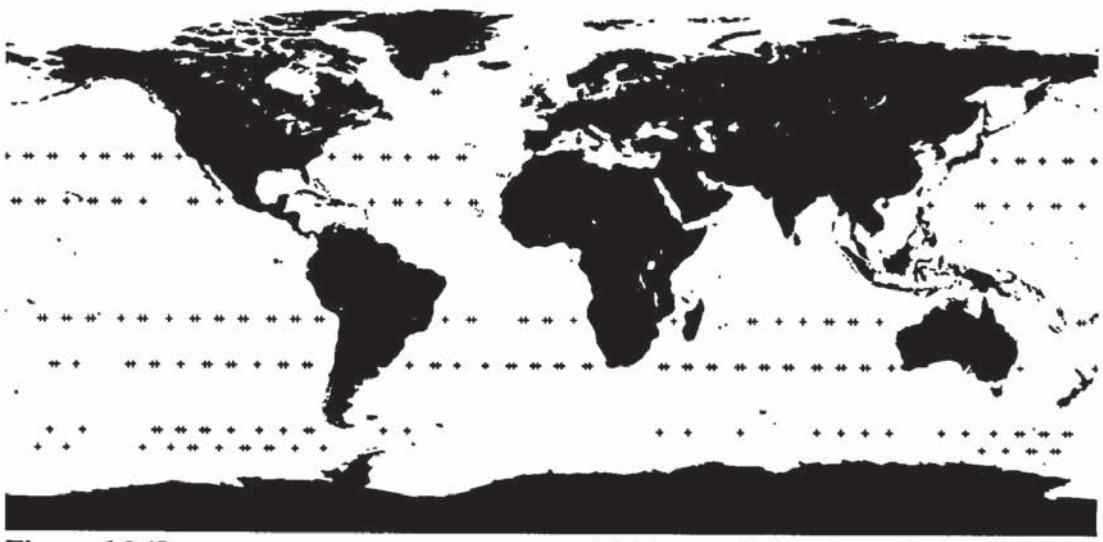


Figure 6.2 (f) Multi-arc solution, 1-day limit : 205 SXO

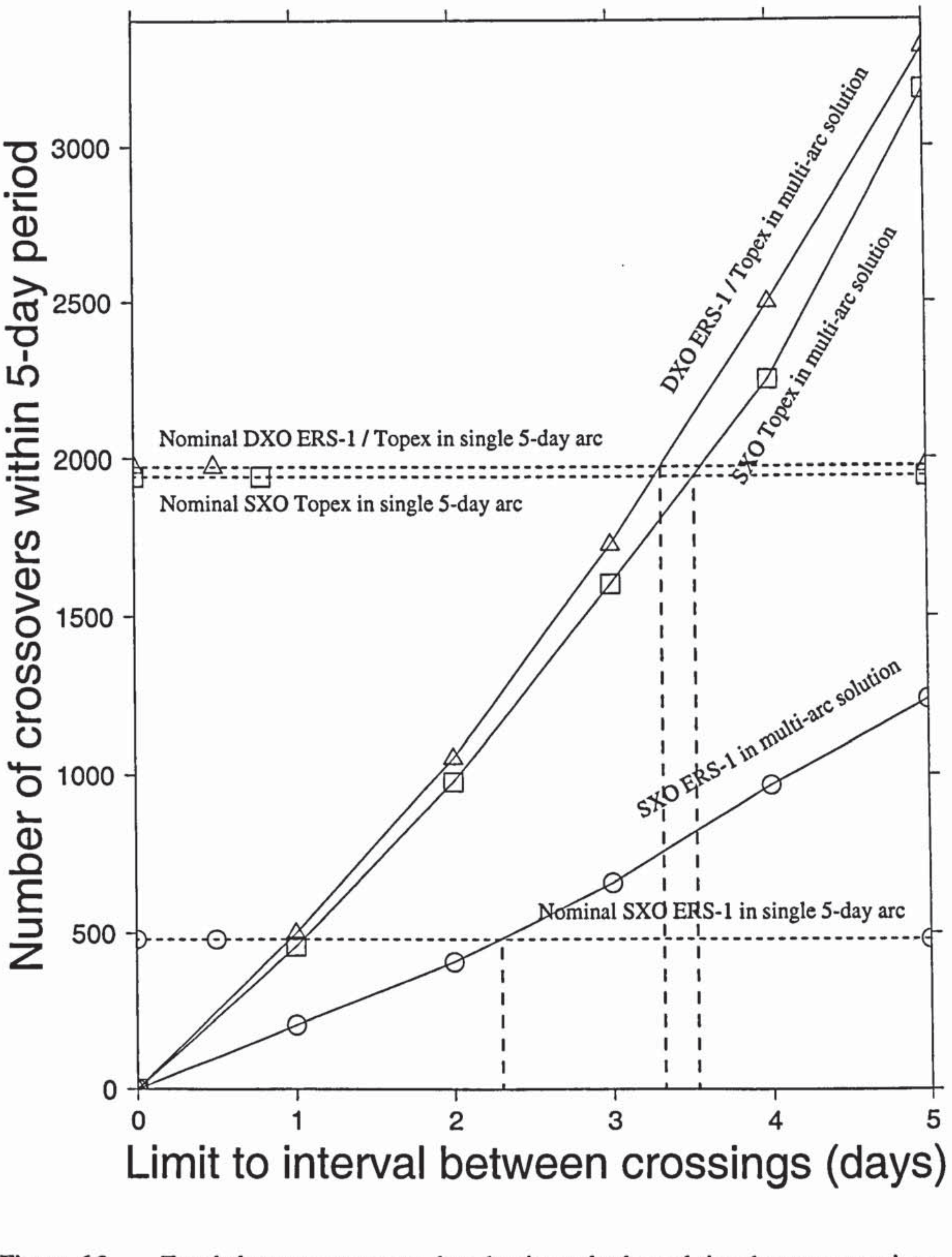


Figure 6.3 Trends between crossover data density and tolerated time between crossings

the same 5-day interval is used as in the single arc case. In between these two cases, we will obtain an improvement in both the data quantity and the data quality. In later Sections, two crossover intervals will be used, namely the generally used value of 5 days, and a smaller value of 3.5 days that should reduce sea surface variability noise in the dataset, while still maintaining a data density that is at least as good as that of the single arc solution.

6.2 Orbit quality assessment

No independent measure exists to verify the actual radial accuracy of a satellite orbit. Various complementary techniques can however provide an adequate estimate of the upper limit of the remaining orbit error. For quality assessment of the TOPEX/Poseidon orbits, Marshall *et al.* (1995) use the three criteria of agreement with tracking data, overlap tests between orbit solutions based on different subsets of the available data, and comparisons with external orbit solutions. Scharroo & Visser (1997) use similar techniques for ERS-1 and add a test based upon crossover residuals for crossings that are further apart than the time span covered by the arc itself.

Test ID	Analysis technique	ERS-1	TOPEX
	Agreement with tracking data :		
1	SLR	Yes	Yes
2	Single satellite crossovers	Yes	Yes
3	DORIS	No	Yes
4	SLR at high elevation	Yes	Yes
5	Power of crossover residuals	Yes	Yes
	Overlap tests with alternative solutions :		
6	Comparisons with 1-day short arcs	Yes	Yes
7	Comparisons with 3-day short arcs	Yes	Yes
8	Comparisons single arc vs. multi-arc solutions	Yes	Yes
	Comparisons with external orbit solutions :		
9	Comparisons with UT/CSR solutions	Yes	No
10	Comparisons with DUT/DEOS solutions	Yes	No
11	Comparison with ephemerides from altimetry GDR	No	Yes
	Long-interval crossover differences :		
12	$0 < \Delta t < 17.5$ days	Yes	No
13	$5 < \Delta t < 35$ days	Yes	No

Table 6.1 Identification of different orbit quality tests, and indication of availability for ERS-1 and TOPEX/Poseidon.

Two other quantities that can serve as an indication of the radial orbit accuracy are the SLR residuals of observations at high elevations (i.e. close to the radial direction) and the RMS of the crossover residuals divided by $\sqrt{2}$. The latter criterion assumes perfect orthogonality between crossovers, corresponding to a residual distribution with a white noise character.

Available tests are listed in Table 6.1, so that later in this Chapter each test can be identified by the ID numbers that are defined here. Obviously, the quality of any solution depends on the a priori models used in the orbit determination process, as well as on the chosen set of model parameters that are solved for. The models used in this Section for all orbits computed with Faust are summarized in Table 6.2 below.

To study the added value of a multi-arc orbit determination approach, precise orbits have been computed with Faust for a sizable period of the ERS-1 and TOPEX/Poseidon missions. For ERS-1, orbits were computed for the entire mission phase C during which the satellite is in a 35-day repeat orbit. TOPEX/Poseidon maintains the same 9.91-day repeat orbit throughout its mission. Due to various inaccuracies during the commissioning phase, the period studied for TOPEX does not start before cycle 10, and continues to cycle 46 in order to span one year. This period overlaps completely with the solution period for ERS-1, so that the overlapping period can also be used for analysis of simultaneous dual-satellite solutions in Chapter 7.

Single-arc orbit solutions for each satellite were computed for the entire period, as well as multi-arc solutions that cover a full 35-day cycle for ERS-1, or a batch of six consecutive repeat cycles (i.e. 60 days) for TOPEX/Poseidon. To these sets of orbit solutions, most of the tests listed in Table 6.1 were applied. The results of all orbit computations and accuracy tests will be presented below in various Figures and Tables.

Geometry	
Ellipsoid	R_e = 6378.1363 km
	f = 1 / 298.257
	ω = 6.300387446 rad / day
Inertial reference frame	J2000 equinox and mean pole
Terrestrial reference frame	ITRF 94 (SLR & DORIS)
Tectonic plate motion	Linear station velocities from ITRF 94
Polar motion	IERS
Nutation	Wahr model (see Appendix A)
Conservative forces	
Gravity field	μ = 0.398600441500 x 10^{15} m^3 / s^2
	JGM-3 70x70 (Tapley *et al.*, 1996)
Solid Earth tides	MERIT (Melbourne *et al.*, 1985)
Ocean tides	UT/CSR, 168 terms included
Third body attraction	IAU Planetary ephemerides
Surface forces	
Atmospheric density model	MSIS83 (Hedin, 1983)
Drag scale parameters	TOPEX : daily
	ERS : 2 parameters per day (SLR only orbits)
	or 4 parameters per day if sufficient data
Solar radiation pressure	TOPEX : 1 scale factor per arc
	ERS : no parameters solved for
	K_p and $F10.7$ from IAU
Earth albedo / infrared	Included for both satellites
Empirical accelerations	
Once per revolution cyclic	TOPEX : along track & cross track, daily
	ERS-1 : along track and cross track, 1 set per arc;
	occasionally daily [1]
Along track acceleration	Not solved for

Table 6.2 Models used in orbit determination

[1] Daily or two-daily sets of parameters were used in most arcs of the cycles 9 and 10, where atmospheric density variability was high due to extreme solar activity.

Figure 6.4 shows all post-solution tracking data residuals as a function of time, for both satellites (tests 1, 2 and 3 of Table 6.1). Figure 6.5 shows the results of the performed orbit comparisons (tests 8, 9 and 10), separated in radial and horizontal components of the total orbital differences. Note that the program used for these comparisons also computes a 7-parameters transformation between the two orbit solutions, in order to compensate for differences between reference frames (in particular for different polar motion data and / or the different ellipsoid model). The provided RMS of differences is obtained *after* transformation. To allow for a more precise analysis than is possible from the Figures alone, the information that is graphically represented in the Figures is also given numerically in the form of the Tables 6.3 and 6.4. Finally, Table 6.5 summarizes the RMS values for all involved tracking data sets and orbit comparisons, as well as some other quality indicators like the results of the tests 4 and 5.

A few short periods with clear anomalies have been excluded from the results. In particular, this concerned three periods with repetitive manoeuvres between which insufficient tracking data was available, as well as a period of two days in ERS-1 cycle 9 for which so few SLR observations were available that the differences between all available orbit solutions was at the level of several meters. Because such values do not reflect the typical orbit accuracy of the satellite, exclusion of such periods provides more relevant information.

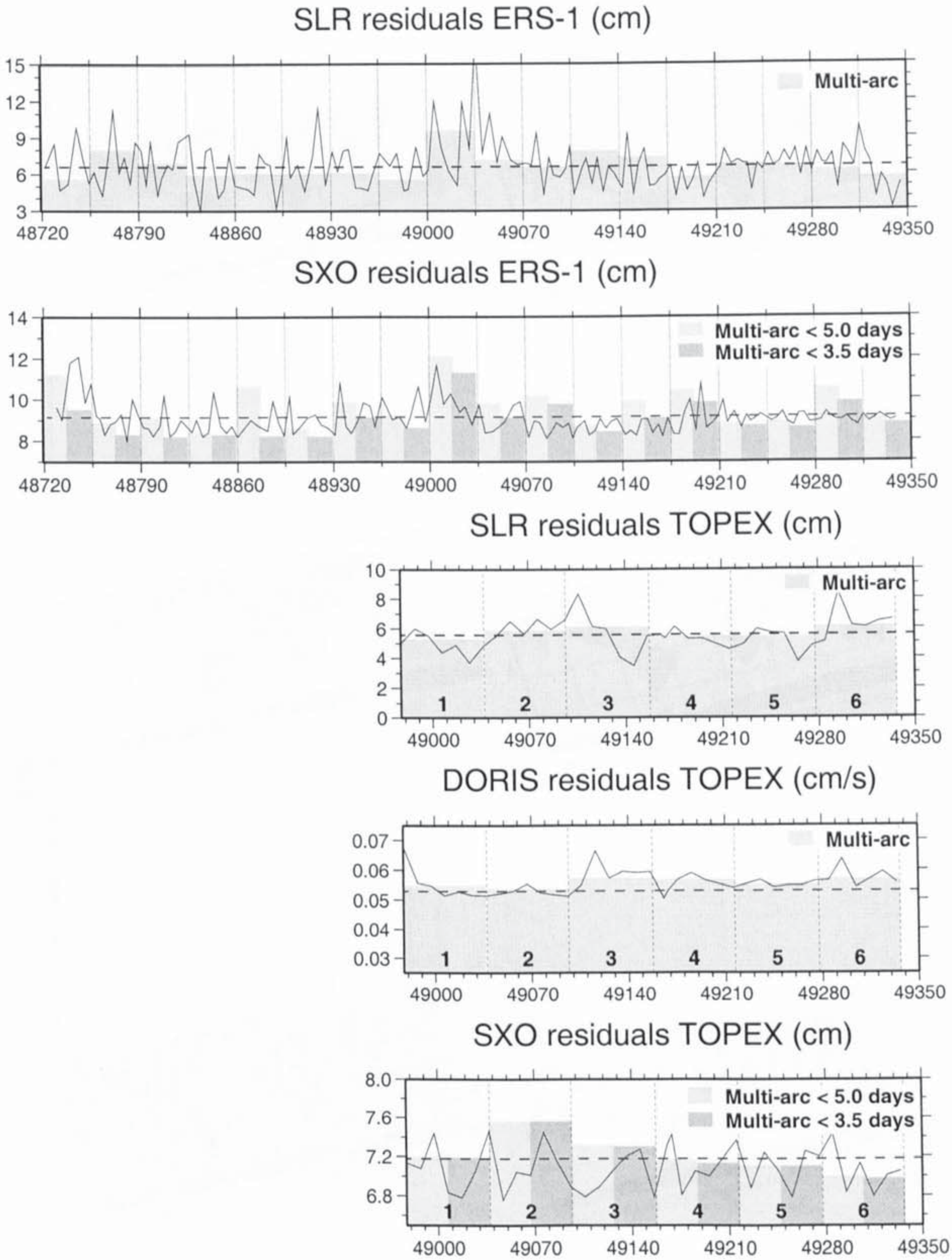


Figure 6.4 Tracking data residuals for all solutions. Continuous lines represent the RMS for single arc solutions, grey bars give the RMS values for the multi-arc solutions, which are numbered as in Tables 6.3 and 6.4. For SXO residuals, separate values are shown for data sets with 5-day and 3.5-day upper limits to the crossover interval. Note that each of the two SXO bars for a given multi-arc solution relate to the entire multi-arc period.

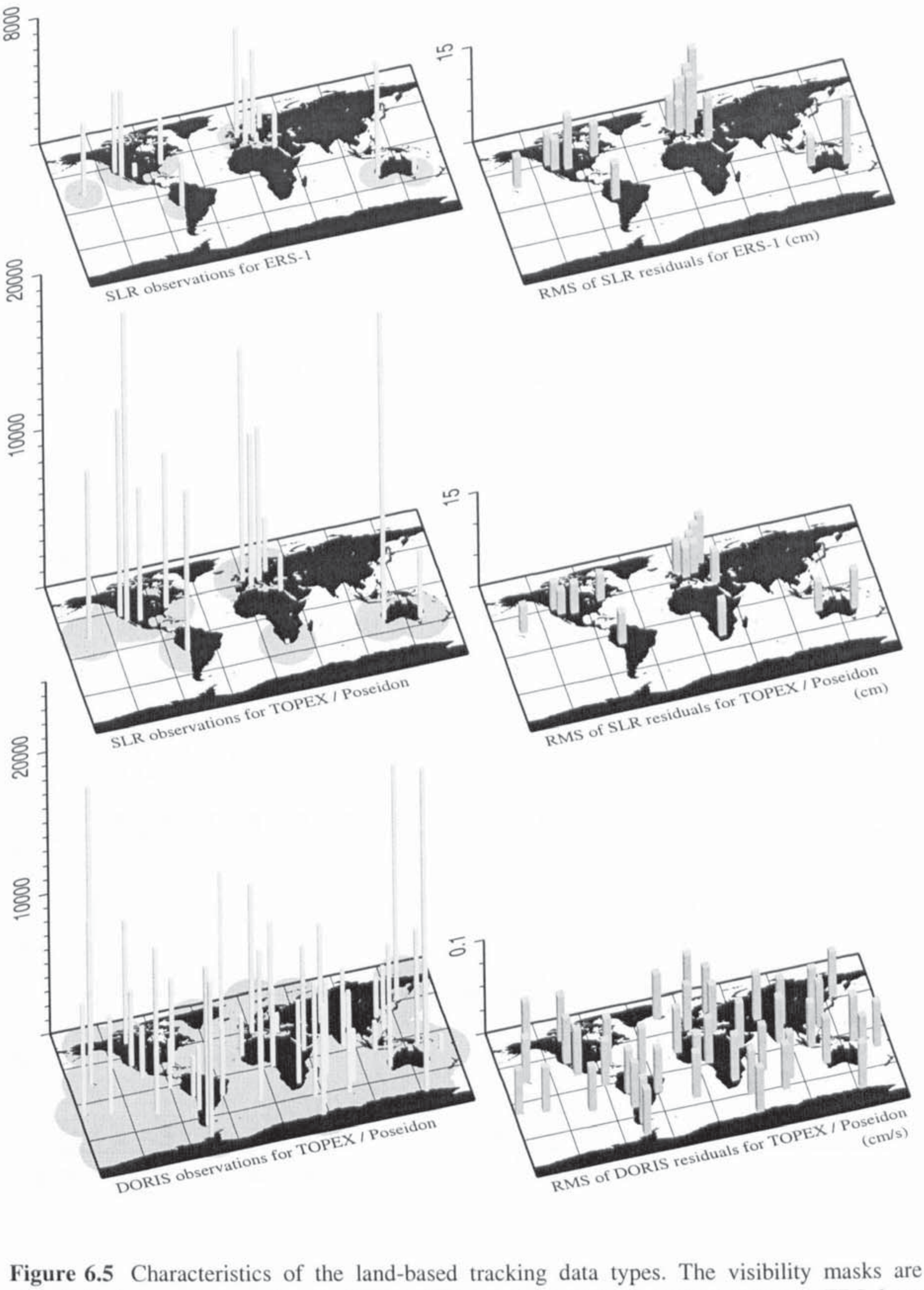
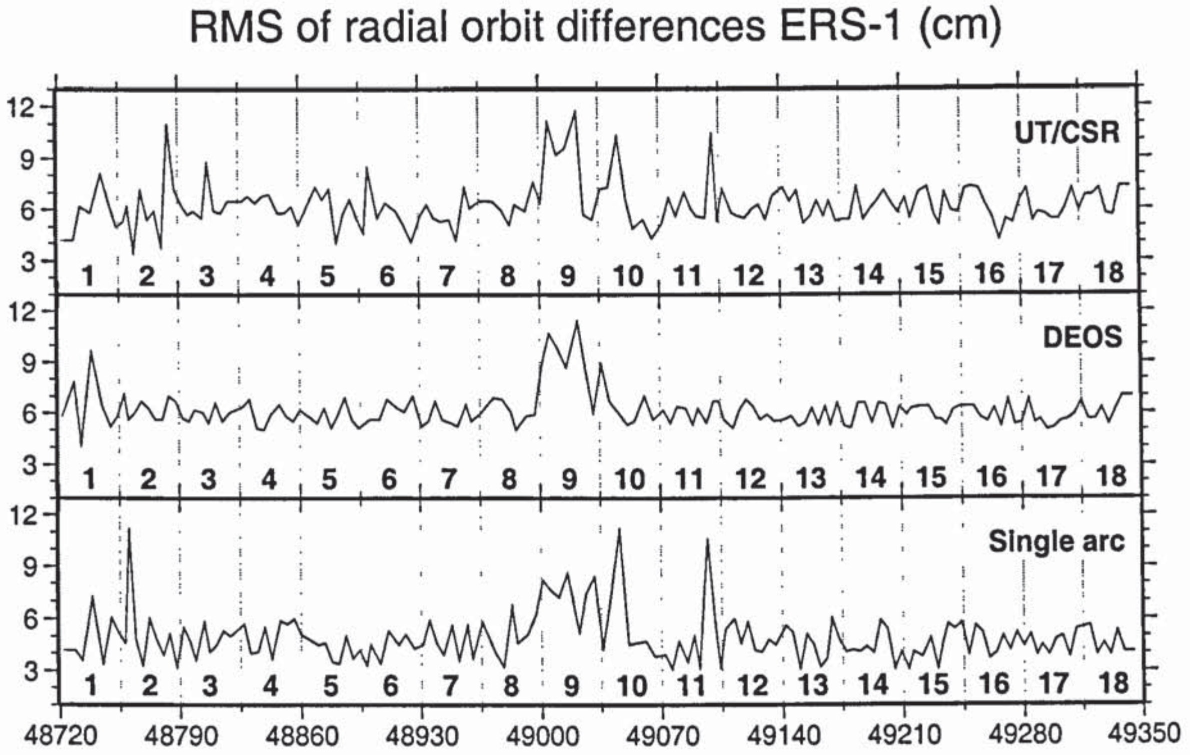


Figure 6.5 Characteristics of the land-based tracking data types. The visibility masks are also indicated, and show the significant differences in global coverage between ERS laser data, TOPEX/Poseidon laser data and DORIS data. The DORIS data is even sampled to reduce the computational loads; only one of every four available measurements is used.



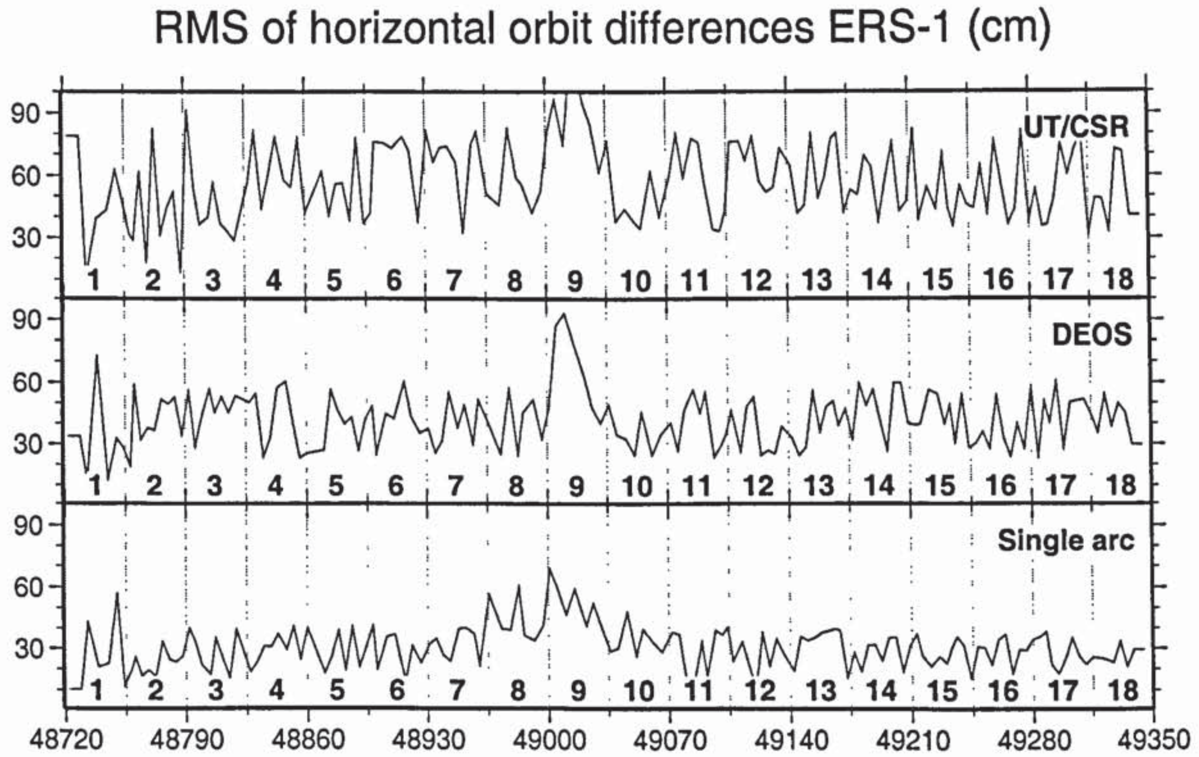


Figure 6.6 Comparisons of ERS-1 orbits from multi-arc process with other solutions.

Solution ID		Epoch MJD	RMS of residuals (cm)					Orbit comparisons with multi-arc solutions (cm)					
Multi arc	Single arc		Multi-arc solution			Single arc solution		UT / CSR		Delft DEOS		Single arc	
			SLR	SXO < 5	SXO < 3.5	SLR	SXO	Radial	Horiz.	Radial	Horiz.	Radial	Horiz.
RMS for complete period >			6.62	9.15	8.98	6.82	9.11	6.27	56.8	6,05	52.7	4.6	28.4
1	1	48729.00	5.65	11.24	9.52	8.52	9.62	4.2	78.8	15.8	33.7	4.2	10.2
	2	48733.00				4.66	9.01	6.2	111.2	4.2	12.8	3.6	42.5
	3	48739.00				5.21	11.80	11.8	38.9	9.6	72.8	7.2	21.0
	4	48745.00				9.88	12.10	8.1	42.9	6.5	12.9	3.4	22.8
	5	48750.00				7.23	9.86	6.3	62.9	3.2	32.8	6.1	57.2
	6	48754.00				5.31	10.77	5.0	48.0	3.8	28.4	5.2	12.0
	7	48758.00				6.21	9.21	5.3	31.9	7.1	19.1	4.6	19.3
2	8	48760.50	8.02	8.88	8.30	5.22	9.07	6.2	28.9	5.6	59.2	11.2	25.9
	9	48764.00				4.21	8.31	1.5	62.1	6.0	32.0	4.8	16.7
	10	48768.00				7.29	8.81	7.1	17.9	6.7	37.7	.. 3.3	19.4
	11	48772.00				11.31	8.94	5.4	82.5	6.3	36.6	6.0	16.1
	12	48776.00				6.12	9.32	5.9	31.6	5.6	51.7	4.8	33.2
	13	48780.00				7.41	8.01	3.8	44.4	5.6	49.3	3.9	24.9
	14	48784.00				5.23	10.02	11.0	51.8	7.0	52.7	5.1	23.4
	15	48788.00				8.67	9.31	7.2	12.9	6.7	34.4	3.2	26.4
	16	48792.00				7.98	8.67	6.3	91.6	5.7	55.8	5.5	39.8
3	17	48795.50	6.90	9.12	8.21	5.10	8.63	5.7	54.1	5.5	28.5	4.7	32.0
	18	48799.00				8.70	8.24	5.9	36.3	6.2	41.5	3.6	21.9
	19	48804.00				4.27	8.73	5.5	39.6	6.0	56.6	5.8	17.2
	20	48807.00				5.72	10.20	8.7	56.5	5.4	45.1	4.1	34.9
	21	48811.00				6.79	8.25	5.9	36.7	6.6	53.0	4.5	25.4
	22	48815.00				6.44	8.46	5.8	33.2	5.5	45.2	5.3	15.8
	23	48819.00				8.71	9.01	6.5	28.8	6.0	53.2	5.0	38.9
	24	48827.00				9.35	8.45	6.5	56.6	6.4	50.1	5.7	18.7
4	25	48830.50	5.96	8.42	8.30	5.56	9.04	6.8	80.7	6.8	54.4	4.0	23.0
	26	48835.00				3.01	8.23	6.4	44.1	5.1	23.6	4.1	30.9
	27	48839.00				7.96	8.42	6.8	60.5	5.0	33.1	5.5	30.6
	28	48843.00				8.22	10.38	6.9	78.1	5.9	57.4	3.7	36.9
	29	48848.00				4.06	8.72	5.8	57.9	6.5	60.5	5.9	29.4
	30	48852.00				4.74	8.40	5.8	54.3	5.8	40.5	5.7	40.7
	31	48856.00				7.61	8.73	6.2	77.7	5.5	23.4	6.0	25.1
	32	48860.00				5.08	8.21	5.1	41.6	6.2	25.6	5.1	39.8
5	33	48870.00	6.05	10.63	8.24	4.72	9.02	7.3	62.1	5.4	27.3	4.5	18.0
	34	48874.00				4.28	8.78	6.6	40.4	6.3	56.4	4.6	26.1
	35	48878.00				7.70	8.63	7.2	55.7	5.1	46.2	3.5	38.6
	36	48882.00				6.97	8.49	4.0	56.4	5.9	39.6	3.4	19.8
	37	48886.00				6.77	9.93	5.7	38.6	6.9	42.9	5.0	40.3
	38	48890.00				3.20	8.86	6.6	78.3	5.6	27.3	3.7	21.1
	39	48894.00				5.40	8.27	5.5	36.8	5.1	42.2	4.2	31.0
	40	48898.00				9.07	10.14	4.6	41.3	5.4	48.4	3.3	40.9
6	41	48900.50	6.00	8.63	8.23	5.73	8.32	5.4	75.9	5.6	25.4	4.5	20.0
	42	48906.00				6.79	8.69	5.5	75.3	5.6	44.7	3.4	35.3
	43	48911.00				4.55	9.10	6.4	72.9	6.8	42.1	5.3	36.7
	44	48917.00				7.52	9.28	5.9	78.5	6.3	60.4	4.5	14.1
	45	48921.00				11.49	8.77	5.2	71.5	6.1	43.0	5.1	31.3
	46	48926.00				5.15	8.62	4.1	37.8	7.0	35.5	4.3	23.2
	47	48931.00				7.88	8.31	5.6	80.9	5.2	37.4	4.5	32.1
	48	48935.00				6.24	10.82	6.3	66.5	5.5	26.0	5.9	34.6
7	49	48939.00	6.13	9.87	9.12	7.95	8.71	5.5	73.2	6.7	31.8	4.6	26.8
	50	48943.00				8.10	8.36	5.3	73.7	5.6	55.1	3.9	23.9
	51	48948.00				4.87	9.00	5.4	66.2	5.4	37.8	5.6	39.7
	52	48952.00				4.86	9.85	4.2	32.0	5.2	48.9	3.6	40.2
	53	48957.00				4.66	9.67	7.3	74.7	6.5	30.1	5.6	37.0
	54	48960.10				5.93	8.68	6.1	81.1	5.5	51.6	3.7	21.9
	55	48965.50				7.74	1010	6.5	51.0	6.0	40.5	5.8	16.3
	56	48973.08				6.71	9.02	6.5	45.4	6.9	25.4	4.1	39.7
8	57	48978.00	5.56	9.21	8.65	7.75	9.17	6.0	82.1	6.8	56.8	3.2	38.9
	58	48983.00				4.62	8.63	5.1	58.8	6.1	25.1	3.7	20.3
	59	48986.00				4.72	9.16	6.3	55.7	5.0	45.1	4.6	36.1
	60	48992.00				8.24	10.63	5.9	41.8	5.8	51.6	5.1	33.6
	61	48997.00				5.84	8.93	4.6	51.8	5.9	32.7	4.3	41.1
	62	49001.00				6.37	9.98	6.4	80.5	6.8	27.1	4.2	18.6

9	63	49005.50	9.62	12.1	11.31	12.00	11.68	11.1	66.3	6.7	37.1	7.6	19.5
	64	49010.50				8.18	9.77	9.2	84.8	6.8	33.0	7.2	16.7
	65	49015.50				6.32	10.28	6.6	41.2	6.7	49.2	4.4	29.3
	66	49022.00				5.06	9.39	11.7	134.2	6.4	59.0	5.2	40.9
	67	49026.00				11.96	9.64	5.7	114.2	6.9	78.0	3.4	23.3
	68	49031.00				8.07	8.78	5.4	61.6	6.0	39.8	8.4	39.9
	69	49036.00				16.89	9.72	7.2	76.3	5.9	49.0	4.3	28.6
10	70	49040.50	7.22	9.78	9.11	7.73	8.38	7.3	37.6	6.7	34.5	7.2	20.1
	71	49046.00				11.00	8.47	10.3	43.5	6.0	32.6	11.1	18.0
	72	49051.00				7.00	8.73	6.6	37.9	5.3	24.9	4.5	26.5
	73	49055.00				9.09	8.95	4.9	34.2	5.5	45.3	4.6	39.2
	74	49061.00				7.17	9.71	5.4	62.1	7.0	24.9	4.7	32.7
	75	49066.00				6.78	9.88	4.3	39.8	5.6	34.3	3.8	28.3
	76	49072.00				6.91	8.19	5.2	59.9	6.2	40.2	3.9	37.8
11	77	49076.00	6.58	10.16	9.77	6.75	8.96	6.7	79.9	5.4	26.9	3.1	36.8
	78	49080.00				9.43	8.87	5.6	58.9	6.4	46.9	4.7	14.3
	79	49085.00				4.33	8.12	7.0	77.5	6.3	56.4	3.5	15.5
	80	49089.00				7.36	8.48	6.0	75.6	5.3	44.9	5.0	33.3
	81	49092.00				5.91	8.85	5.6	57.4	6.3	55.1	3.2	16.5
	82	49097.00				5.76	8.64	5.5	34.1	5.4	23.7	10.6	38.7
	83	49101.00				6.46	8.83	10.5	33.3	6.7	29.3	5.9	36.5
	84	49104.00				8.27	8.15	5.4	41.9	6.7	34.9	3.2	40.3
	85	49107.00				4.32	8.62	7.2	76.0	5.6	45.9	5.4	23.8
12	86	49112.50	7.91	8.72	9.45	6.44	8.95	5.8	76.1	5.1	26.4	6.0	33.3
	87	49116.00				7.29	8.29	5.6	67.1	6.2	48.4	4.6	22.3
	88	49120.00				5.22	8.46	5.5	78.5	6.8	52.7	5.8	10.4
	89	49124.00				7.34	9.05	6.0	57.2	6.4	24.5	4.2	37.3
	90	49128.00				4.51	8.60	6.3	51.9	5.6	26.9	4.1	21.4
	91	49132.00				6.58	8.91	5.4	54.2	5.9	25.2	4.8	34.6
	92	49136.00				6.05	9.05	6.8	72.9	5.5	38.3	4.5	27.7
	93	49142.00				4.79	8.29	7.3	64.4	5.6	32.9	5.6	19.3
13	94	49146.00	7.33	9.89	9.10	9.38	8.49	6.5	41.5	5.8	24.4	5.2	35.7
	95	49150.00				4.35	8.25	7.1	45.3	5.2	28.6	3.1	33.7
	96	49154.00				6.88	9.01	5.2	79.4	5.4	55.5	5.1	35.2
	97	49158.00				8.10	9.03	5.6	49.0	6.3	35.7	4.5	37.5
	98	49162.00				4.98	8.36	6.5	59.2	5.3	48.1	3.2	38.4
	99	49166.00				5.07	8.76	5.6	77.7	6.4	50.8	3.7	39.6
	100	49169.00				5.56	8.20	6.5	80.2	5.3	38.8	6.0	38.7
	101	49173.00				5.87	9.08	5.3	41.9	6.6	46.8	4.8	15.8
	102	49177.00				6.71	8.30	5.4	52.9	5.2	32.2	4.1	28.0
14	103	49181.00	5.76	10.42	9.84	4.19	8.28	5.4	50.4	5.1	59.5	4.2	18.9
	104	49185.00				6.49	9.38	7.3	69.1	6.6	48.5	4.1	31.6
	105	49189.00				4.60	9.97	5.4	64.0	6.6	56.4	4.4	31.7
	106	49193.00				5.55	8.52	6.0	37.4	5.4	41.0	4.0	24.1
	107	49197.00				6.91	10.78	6.5	58.2	6.6	26.9	5.9	35.1
	108	49201.00				4.02	8.60	7.1	75.7	6.5	59.8	5.4	35.5
	109	49205.00				5.24	8.81	6.4	42.4	5.1	59.8	3.1	19.0
	110	49209.00				5.92	9.10	5.8	47.0	6.4	39.5	4.1	31.3
	111	49213.00				6.52	9.94	6.7	82.4	5.9	38.9	3.1	37.2
15	112	49216.00	6.87	8.83	8.71	8.02	8.86	5.5	38.6	6.3	39.2	4.1	26.7
	113	49221.00				6.80	9.39	7.0	54.5	6.4	56.2	3.8	21.4
	114	49226.00				7.10	8.62	7.3	43.5	6.4	54.2	4.9	26.2
	115	49230.00				6.83	9.22	5.7	70.4	5.6	39.4	3.1	23.1
	116	49233.00				6.82	8.89	5.1	44.0	5.6	48.8	4.6	30.3
	117	49236.00				7.65	9.03	7.0	35.3	5.3	30.6	5.7	35.6
	118	49240.00				6.82	9.25	5.9	55.0	6.2	53.7	5.4	31.3
	119	49244.00				6.32	9.09	5.8	45.4	6.4	28.1	5.8	15.6
	120	49248.00				7.63	8.91	7.2	43.9	6.4	30.3	3.9	31.0
16	121	49252.00	6.70	8.90	8.62	6.65	9.03	7.3	64.8	6.4	35.9	5.6	30.4
	122	49256.00				6.90	9.37	7.2	41.3	5.7	27.9	5.2	22.3
	123	49260.00				7.83	8.86	6.3	76.6	5.5	53.3	3.7	34.3
	124	49264.00				6.96	8.72	5.5	56.1	6.3	33.2	4.0	36.9
	125	49268.00				8.05	9.30	4.2	36.3	5.2	24.0	5.0	17.8
	126	49272.00				5.89	9.31	5.4	43.5	6.8	39.6	4.2	29.6
	127	49276.00				8.04	8.85	5.2	81.7	5.3	27.8	5.2	29.2
	128	49280.00				6.20	8.90	6.5	37.5	5.4	57.4	4.4	34.2
	129	49284.00				7.75	8.96	7.2	53.4	6.8	24.2	5.1	35.7

17	130	49288.00	6.28	10.54	9.86	6.83	9.38	5.3	35.3	5.4	50.8	3.9	38.3
	131	49291.00				6.82	9.16	5.8	35.9	5.6	40.1	4.5	22.3
	132	49295.00				7.65	9.00	5.7	48.4	5.0	60.2	3.9	17.8
	133	49299.00				6.82	8.98	5.4	75.5	5.1	28.1	4.8	25.0
	134	49303.00				6.32	9.16	5.4	60.6	5.5	50.2	5.0	35.1
	135	49307.00				7.63	8.88	6.1	72.0	5.6	51.1	3.8	26.5
	136	49311.00				6.65	8.74	7.2	80.2	5.9	51.7	5.4	22.5
	137	49315.00				6.90	9.22	5.9	31.6	6.7	44.3	5.5	25.9
	138	49319.00				7.83	8.89	6.8	48.8	5.6	35.3	5.6	25.6
18	139	49323.00	5.75	9.08	8.81	6.96	9.03	6.8	48.5	5.6	54.0	3.9	24.8
	140	49327.00				4.05	9.25	7.2	33.2	6.3	38.4	4.6	23.3
	141	49331.00				5.89	9.09	5.7	72.4	5.3	49.6	3.9	33.5
	142	49335.00				8.04	8.91	5.6	71.1	6.2	45.1	5.3	21.3
	143	49339.00				3.20	9.03	7.3	40.3	7.0	29.9	4.1	29.5

Table 6.3 Orbit determination results for ERS-1

Solution ID		Epoch MJD	RMS of residuals (cm, cm/s)						Orbit comparisons (cm)			
Multi arc	Single arc		Multi-arc solution			Single arc solution			GDR Ephemerides		Multi vs. Single	
			SLR	SXO	DORIS	SLR	SXO	DORIS	Radial	Horiz.	Radial	Horiz.
RMS for complete period >			5.57	7.18	0.0558	5.57	7.16	0.0558	4.7	53.8	2.8	18.2
1	10	48977.39	5.31	7.20	0.0546	5.02	7.13	0.0677	5.2	69.4	2.8	22.4
	11	48987.30				6.00	7.07	0.0556	5.8	46.3	3.2	12.5
	12	48997.22				5.52	7.44	0.0548	4.8	51.6	2.7	12.8
	13	49007.13				4.39	6.83	0.0514	5.5	49.6	3.0	24.6
	14	49017.00				4.89	6.78	0.0528	3.6	69.7	3.0	21.6
	15	49026.97				3.69	7.05	0.0515	3.8	73.7	2.8	17.5
2	16	49036.88	5.84	7.56	0.0538	4.82	7.46	0.0514	5.0	56.8	2.7	22.2
	17	49046.80				5.54	6.76	0.0521	4.1	34.9	2.7	15.7
	18	49056.71				6.50	7.04	0.0529	4.0	46.0	2.9	25.3
	19	49066.63				5.61	7.01	0.0554	4.7	53.4	2.8	23.2
	20	49076.54				6.65	7.45	0.0524	4.6	60.6	2.4	23.1
	21	49086.46				5.92	7.14	0.0517	4.7	31.0	3.0	22.5
3	22	49096.37	6.08	7.31	0.0572	6.57	6.88	0.0512	5.0	48.0	2.7	22.5
	23	49106.29				8.33	6.79	0.0551	5.3	71.3	2.5	12.1
	24	49116.21				6.13	6.89	0.0665	4.1	58.7	3.1	16.0
	25	49126.12				5.97	7.05	0.0572	5.6	55.8	2.8	12.1
	26	49136.04				4.07	7.20	0.0595	4.0	52.5	3.2	25.5
	27	49145.95				3.47	7.27	0.0591	3.5	55.5	3.0	19.0
4	28	49155.87	5.41	7.16	0.0566	5.47	6.79	0.0593	3.5	61.6	2.5	19.7
	29	49165.78				5.55	7.32	0.0504	5.9	54.7	2.7	22.7
	30	49169.00				5.35	7.43	0.0532	4.5	39.4	3.1	18.1
	31	49175.70				6.10	6.81	0.0567	4.9	33.7	2.7	20.9
	32	49185.62				5.29	7.05	0.0589	3.5	33.7	2.8	20.5
	33	49195.53				5.30	7.00	0.0563	5.6	43.8	2.6	17.5
5	34	49215.36	5.42	7,11	0.0551	4.58	7.37	0.0540	4.8	67.3	3.2	24.1
	35	49225.28				4.91	6.88	0.0552	5.1	66.8	2.9	23.6
	36	49235.19				5.92	7.24	0.0565	5.6	45.7	2.8	10.7
	37	49245.11				5.67	7.06	0.0541	4.1	42.1	2.9	19.6
	38	49255.02				5.63	6.79	0.0546	5.4	31.7	2.6	23.4
	39	49264.94				3.73	7.26	0.0546	4.6	60.9	3.2	26.1
6	40	49274.86	6.14	7,00	0.0569	4.80	7.20	0.0561	3.7	45.5	2.7	24.3
	41	49284.77				5.12	7.44	0.0564	5.8	58.7	3.0	12.6
	42	49294.69				8.44	6.84	0.0637	5.1	58.5	2.5	15.7
	43	49304.60				6.15	7.13	0.0541	5.4	57.5	2.8	20.7
	44	49314.52				6.04	6.80	0.0566	5.2	55.2	3.0	13.5
	45	49324.43				6.44	7.01	0.0593	4.0	65.0	3.2	20.7
	46	49334.35				6.57	7.05	0.0557	4.8	50.6	2.9	16.1

Table 6.4 Orbit determination results for TOPEX/Poseidon

Quality assessment test		ERS-1		TOPEX/Poseidon	
	(units : cm, cm/s)	single	multi	single	multi
Tracking data					
(1)	SLR data residuals	6.82	6.62	5.57	5.57
(2)	SXO residuals, 5.0 day limit	9.11	9.15	7.16	7.18
	SXO residuals, 3.5 day limit	-	8.98	-	-
(3)	DORIS data residuals	-	-	0.056	0.056
(4)	SLR at elevations > 70°	6.16	6.10	4.72	4.73
(5)	SXO residuals / $\sqrt{2}$	6.44	6.47	5.06	5.08
Orbit comparisons (radial component)					
(8)	Single arc vs. multi-arc		4.62		2.83
(9)	Comparison with UT/CSR orbits		6.27		-
(10)	Comparisons with DUT/DEOS orbits		6.05		-
(11)	Comparisons with GDR ephemerides		-		4.70

Table 6.5 Orbit accuracy assessment

The post-solution RMS of tracking data fit for ERS-1 turns out to be very similar to results obtained by other groups (e.g. Ries *et al.*, 1996; Scharroo & Visser, 1997). The computed orbits are also comparable to the various external solutions. This close agreement between independent solutions provides good confidence in each individual solution. In the framework of the current study, the most interesting comparison is of course that between single-arc and multi-arc solutions computed by Faust, as given in the last columns of the Tables 6.3 and 6.4.

On the basis of Table 6.5, the typical radial orbit accuracy for ERS-1 can be estimated at 6.1 cm, and for TOPEX/Poseidon at about 4.8 cm. For both satellites, we can conclude that the multi-arc approach only modifies the orbits *within* the post-solution noise bands of the single-arc solutions, while the RMS fit of the data is hardly any different from the single-arc case. At first sight this might raise questions about the advantages of multi-arc solutions : if we only modify the existing noise without reducing it, it is impossible to conclude that the orbital quality has improved. At closer inspection, however, two useful improvements can be deduced from these results.

The a posteriori residuals in a least-squares process will tend to increase if the ratio between the amount of observations and the amount of parameters solved for increases, unless the solution period is substantially longer than the decorrelation period for all possible noise components. This is a consequence of the need to satisfy a larger amount of - independent - calculated observations, each of which will have its own imperfections. In the Tables we see that *despite* the substantial increase in crossover data quantity, the post-solution noise levels for the multi-arc solutions are not larger than in the single-satellite case. This indicates that the overall RMS level of 9.15 cm as obtained for ERS-1 forms indeed the remaining noise level in the dataset, and that both the single-arc and the multi-arc solutions have converged to the lowest noise level that is feasible for the given input data. However, *because* the multi-arc solutions contain many more observations, they provide greater confidence in the quality of the resulting orbits.

Furthermore, the case of crossover intervals that are limited to 3.5-days shows a decrease in post-solution RMS of the residuals with respect to the single-arc solutions, despite of the fact that the size of the crossover dataset is larger. This indicates that there is indeed a reduction in data noise between the two cases, which then implies that the *orbit* noise in the solutions with a 5-day limit to the crossover interval is lower than the level that is suggested by the post-solution RMS of SXO residuals. However, there may also be an effect of poorer decorrelation of different observations in the 3.5-day dataset, which would not actually improve the quality of the solution, but only create the impression of a reduced noise level.

The tables show that for TOPEX/Poseidon the differences between single-arc solutions and multi-arc solutions for this satellite are negligible. The radial orbit error for TOPEX/Poseidon, as estimated at 4.8 cm above, is larger than what is found by Cheney *et al.* 1994, although it is still smaller than that for ERS-1. The relatively large orbit errors for TOPEX/Poseidon in comparison to external solutions seem to be caused by the use of less sophisticated attitude control algorithms in Faust, or by the fact that the implemented observation models were all calibrated in other software, not in Faust. In any case, the differences appear to be related to observation modelling rather than dynamic modelling, because the difference between the single-arc and multi-arc solutions suggests a much lower orbit noise level than the estimated 4.8 cm. The remainder of this and following Chapters will mainly concern improvements of the ERS orbits, for the simple reason that the radial precision of the TOPEX/Poseidon orbits is barely open to further improvement by Faust.

6.3 Improvement of the ERS-1 orbit error in multi-arc solutions

The main causes of the larger orbital error for ERS-1 is its lower altitude in comparison to TOPEX/Poseidon. This lower altitude implies greater sensitivity to (errors in) the higher degree harmonics of the gravity field model, as well as higher atmospheric densities and a resulting greater sensitivity to the substantial errors in density modelling (see Section 4.4). This Section will analyse some characteristics of the orbit error of ERS-1, after which two subsequent Sections will show examples of methods that exploit the possibilities of multi-arc solutions to improve upon the results from single-arc techniques.

In order to get a more detailled view of the nature of the orbit error, Figure 6.7 shows comparisons for a single 35-day repeat period (cycle 15 of phase C) between the Aston orbits, precise orbits computed by UT/CSR and orbits computed by DUT/DEOS. One of the features that can be clearly noticed in these comparisons is the effect of arc transitions. The arc lengths of each solution are indicated by alternating grey/white backgrounds. The Figure also shows that many of the largest differences between solutions occur around the epochs where either of the compared orbits shows a transition to a new arc. This observation is confirmed by Figure 6.8, in which the orbital differences (for all cases of Figure 6.7) are plotted as a function of their distance to the nearest arc transition in either of the compared solutions. When arc transitions in one solution can be observed as jumps in the differences with another solution, this is likely to be caused by error components in the solution with the arc transition. If the orbit errors that are apparently related to arc-transitions could be reduced, the overall quality of the solutions may improve to levels that are typical for the central parts of solution arcs, i.e. for the right end of the scatter plot in Figure 6.8. To this purpose, we can analyse the nature of orbit errors at arc transitions.

The errors related to arc transitions may be classified into two groups. The first group is caused by reduced observability of parameters that are solved for by means of linear splines, i.e. drag scale factors or possibly empirical along-track accelerations, as discussed in Sections 4.4 and 4.8. Related to this problem are errors due to increased correlations between such parameters at the start of the arc (drag scale factors, empirical accelerations) and the parameters of the initial position and velocity : an error in along-track position or velocity could be compensated by opposing errors in the drag parameters at the start of the arc, and as a result these parameters tend to show higher correlations (i.e. they are less separable).

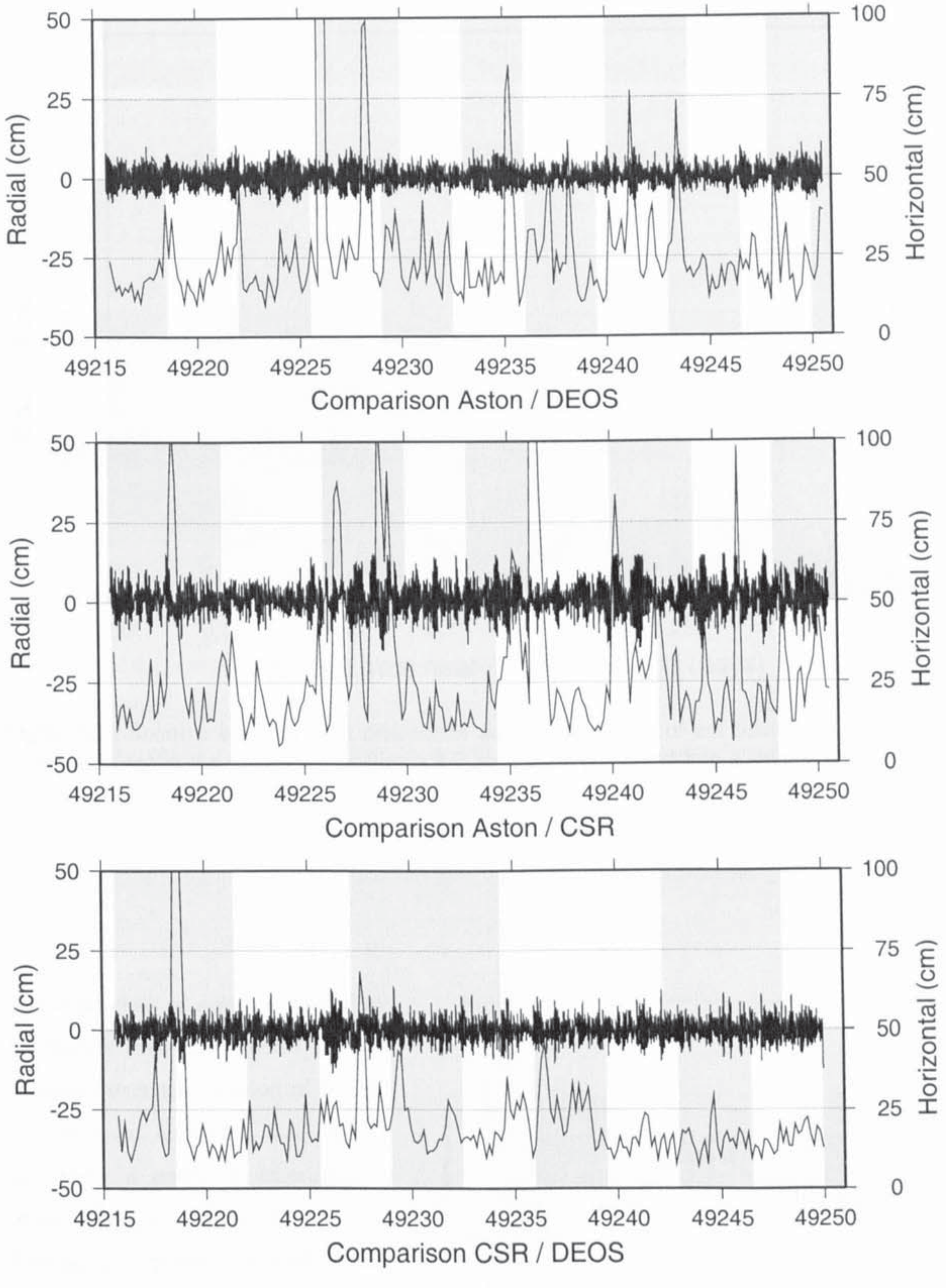


Figure 6.7 Comparisons between three precise orbit solutions for ERS-1 repeat cycle C-15. In each plot the arc boundaries are indicated by grey/white transitions, for each of the two compared solutions.

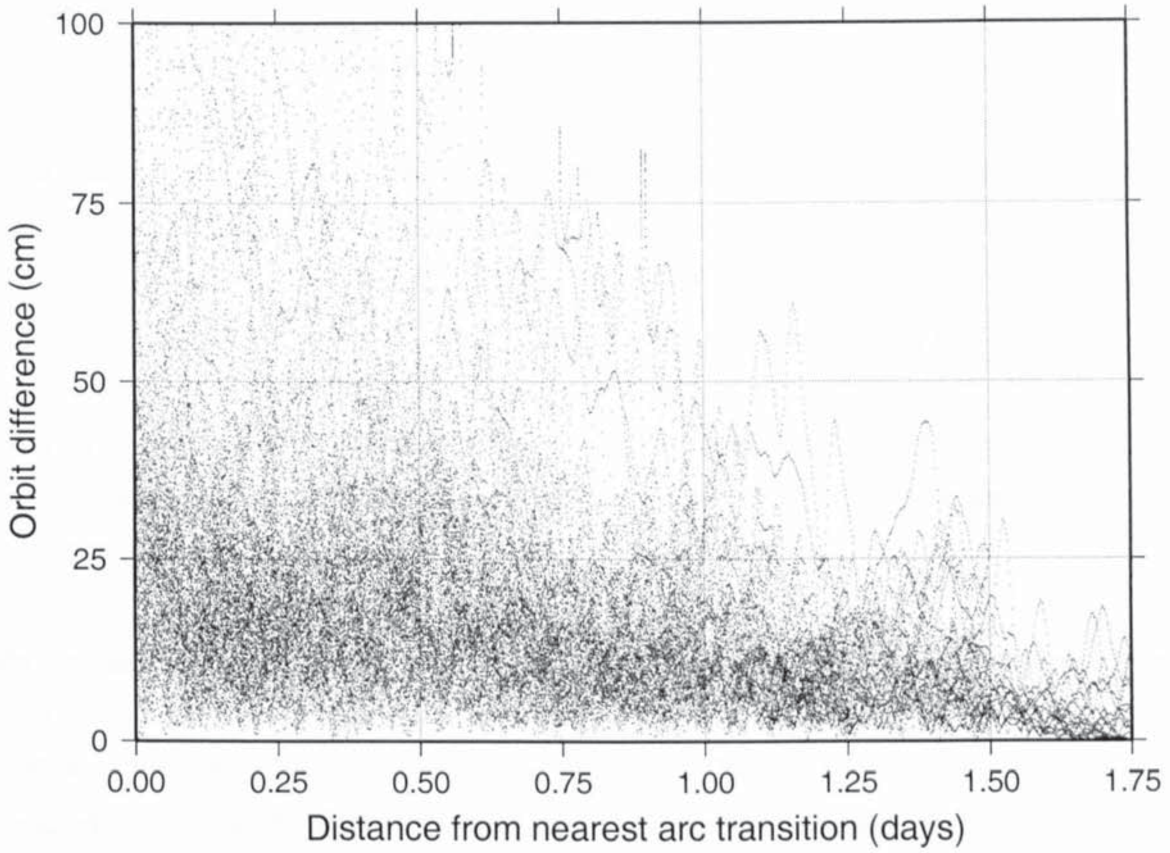


Figure 6.8 Relation between orbit differences and the proximity of arc boundaries. All data points involved in the generation of Figure 6.7 have been combined into a single scatter plot.

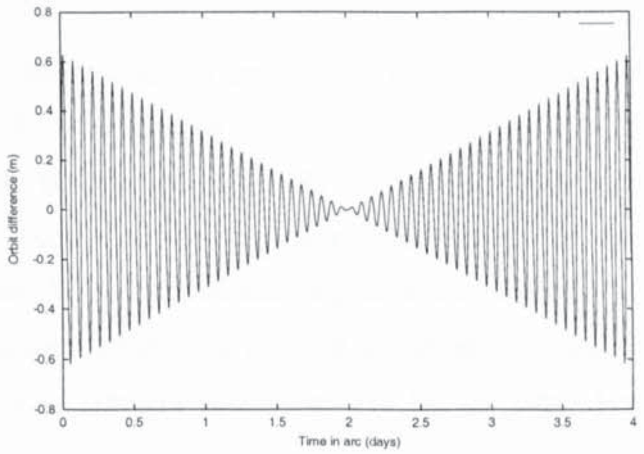


Figure 6.9 Typical error in an orbit with a first order perturbation, solved for as a constant.

A second class of errors is caused by parameters that are solved for as constants over the duration of the arc (a zero-order term) but that may in fact show a trend, or other higher-order behaviour, during the arc. The least-squares process will of course tend to converge such parameters to their average (zero-order) value. This value will be matched most closely somewhere in the central part of the arc, but will lead to errors in this parameter that are largest at the arc boundaries (the so-called 'butterfly'-effect, named after the shape of the plotted orbit error as a function of time; see Figure 6.9).

Note that the first group of errors will benefit from an increase in arc durations, simply because this will reduce the overall density of arc transitions in time. Unfortunately, the second group of errors will benefit from a *de*crease of the arc durations, because this would suppress effects of higher order variations in the parameter values during the arc period. The latter effect can be encountered by dividing the length of the arc in shorter intervals, each with its own - constant - parameter. Trends in a parameter value are hence modelled by a succession of step functions, as is for instance done regularly by using daily sets of empirical accelerations for TOPEX/Poseidon.

As explained in Section 4.4, Faust contains the possibility to use a continuous sequence of drag scale factors throughout the period covered by a multi-arc solution. This option has been applied to the same ERS-1 cycle as used in Figure 6.7. The effect of the continuation of drag parameters is twofold. At first, it eliminates the reduction in observability of the linear splines by which the drag scale factors are computed. At second, it helps to decorrelate the drag scale factors at the start of each arc from the elements of the initial state vector, because they also have to match data from the end of the previous. The resulting orbits are again compared with the external solutions, the result of which is shown in Figure 6.10. As can be observed from the differences with figure 6.7, many errors related to arc transitions in the Aston solutions have been reduced, while peaks related to arc transitions in either of the external solutions are similar to what they were before.

We can conclude that this relatively simple modification to the solution process - using one drag scale factor at the epochs of arc transitions rather than two - offers notable benefits to the orbital precision that can obviously only be obtained in a multi-arc approach. The drag scale factors are mainly observed from along-track tracking data components, and are practically unobservable from crossover data. While Section 6.2 only used the capability of multi-arc solutions to support crossovers between consecutive arcs, the discussed continuation of drag parameters at the arc transitions also correlates successive arcs for the other data types (SLR, DORIS, PRARE).

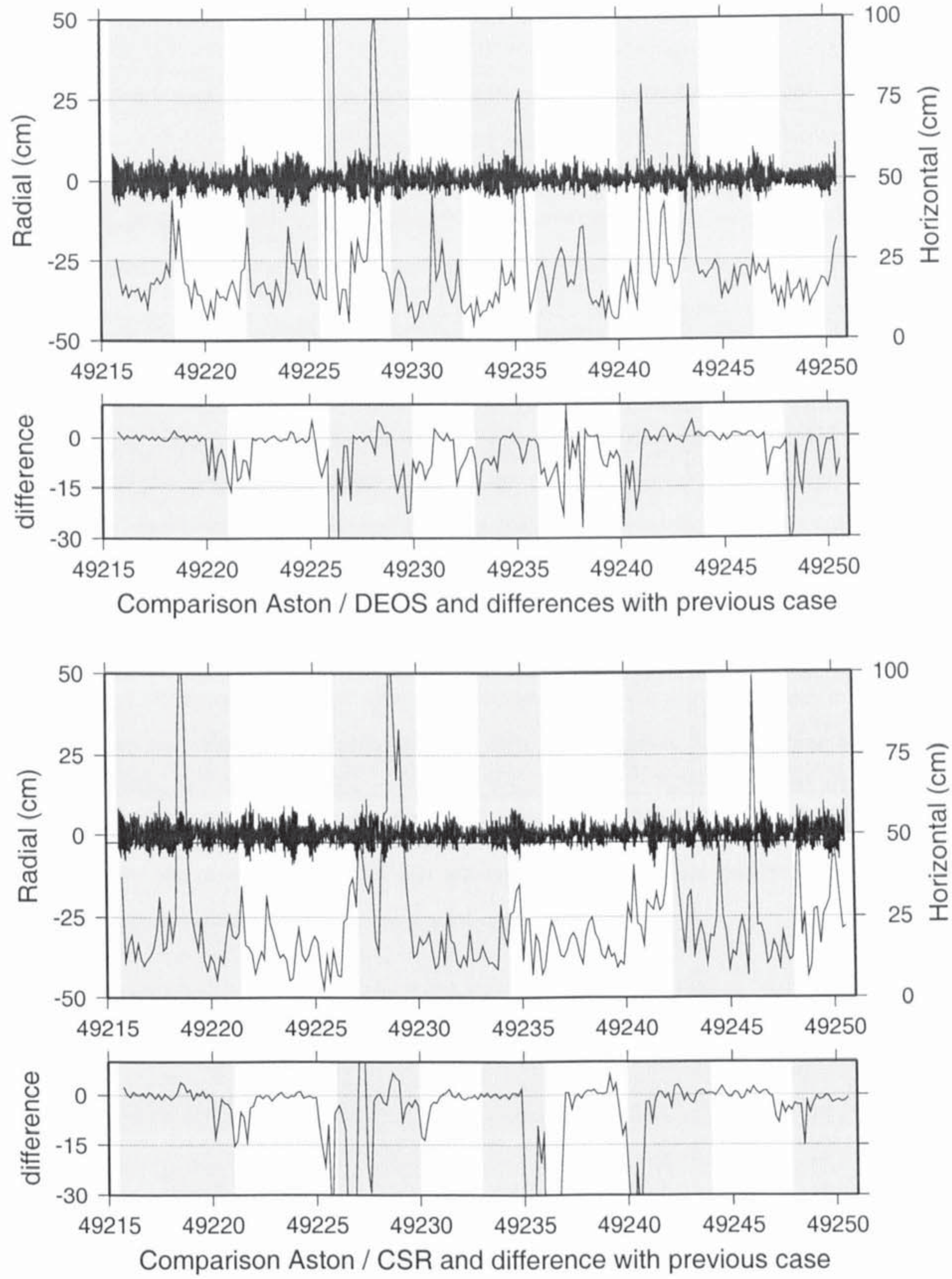


Figure 6.10 Comparisons with external solutions if continuous drag parameters are used throughout the 35-day repeat cycle. Also plotted are the differences with Figure 6.7, to indicate that errors at arc transitions in the Aston solution have been reduced.

6.4 Conclusions

The multi-arc approach to orbit determination causes a subtantial increase in crossover quantities, with typically twice as many observations for the ERS orbits. This increased data quantity allows for improvements of the data quality by reducing the maximum allowed time interval between crossings, typically from 5 days to 3.5 days. The resulting reduction in post-solution RMS of the crossover residuals can not be subscribed to changes in orbit height - as these were not observed - and must therefore be caused by reduced sea surface variability noise in the dataset.

If the same 5-day limit is used as in a single arc solution, the changes in radial orbit height are within the post-solution noise band of the individual solutions, although the double amount of observations must be satisfied. Because the post-solution RMS is not larger than that of single-arc solutions, the multi-arc solution provides greater confidence in the overall solution quality.

For ERS-1, many of the largest orbit errors occur around the arc transitions. In a multi-arc process we can connect consecutive arcs not only by crossovers between these arcs but also by decoupling the drag parameters at the arc transitions from each individual arc. The result is a reduction in correlations between the initial state vector and other parameters at the start of the arc, and a reduction of orbital discontinuities at the arc transitions. Both results improve the overall quality of the orbit solution.

Multi-satellite / multi-arc orbit determination Chapter 7

This Chapter discusses the simultaneous orbit determination process for ERS-1 and TOPEX/Poseidon. The essence of the simultaneous solution method for altimetry satellites is formed by the adjustment of the orbits to the tracking data of the other satellites in the solution, via correlations introduced by means of dual satellite crossovers. This effect will be demonstrated in this Chapter.

In simultaneous solutions, the size and reliability of the various datasets will not be identical. Making use of the a priori standard deviations and scalefactors discussed in Chapter 5 the weights attached to each observation can be tuned for each combination of tracking type, tracking station and satellite. In comparison to the single satellite solutions of the previous Chapter, this issue becomes much more relevant now that two or more satellites will be influenced by each others' tracking data. Some aspects of solution balancing and weighting strategies will therefore be discussed first, in order to obtain a sensible weighting strategy for the simultaneous solutions.

Subsequently, simultaneous multi-satellite solutions will be presented and compared with the single satellite solutions from Chapter 6. The ERS-1 orbits are compared with orbits from solutions in which crossovers with TOPEX/Poseidon are used without simultaneously solving for the orbits of TOPEX/Poseidon.

Finally, an example for the ERS tandem mission will be shown. Because this solution does not yet make use of the ERS-2 PRARE data, it is merely intended as an illustration of some interesting possibilities of the simultaneous solution approach, in particular related to modelling of atmospheric drag.

7.1 Strategy for setting weights and constraints

The mathematical aspects of weights have been treated in Chapters 2 and 5, but the way in which data weights are actually determined in practice was not discussed. It is recalled that by giving weights to observations relative to each other we add a priori information to the solution process. If this a priori information is inaccurate or incomplete, the solution process will be steered in an undesired way. It might therefore seem better not to set any weights at all, if insufficient information is available with respect to the relative quality of different datasets. However, using a standard weight for each observation (e.g. a weight of '1') already implies overweighting of the less accurate data : *un*weighted solutions do not exist. It is therefore inevitable to obtain a sensible weighting strategy before starting a simultaneous solution process. The two functions of weights are (1) to compensate for differences in data quality and (2) to compensate for differences in data quantity. These two aspects will be analysed separately.

7.1.1 Data quality

The 'quality' of a data set is usually associated with the RMS of its post-solution residuals. This is a deceptive measure. The RMS is not only determined by the inherent noise and errors in the measurements, but also by systematic errors due to mismodelling of either the satellite dynamics (Chapter 4) or observation models (Chapter 5).

As an important example, the crossover datasets in which the maximum separation between crossings is 3.5 days tends to produce lower RMS values than the 5-day dataset and may therefore seem the 'better' dataset; nonetheless, the differences are caused by modelling errors (in the combined sea surface geometry models) rather than by pure noise. Another example is mentioned in Section 5.2 in relation to the estimation of range biases for SLR data : by removing the mean value from the measurement residuals, the RMS will obviously be reduced; however, it is not likely that the entire mean value is caused by systematic errors rather than by relevant signal components. This means that the estimation of a bias does not necessarily improve either the observation model or the data 'quality'.

The above examples show that it is hazardous to attach an absolute quality figure to any dataset; fortunately all a priori information that we need for the estimation process is found in *relative* weights between different datasets. The strategy that will be followed here is based on relative ranking of datasets, in terms of factors that contribute to data quality differences :

- Tracking data type (or : estimated modelling quality for the overall signal)

- Modelling quality at the satellite-end of the tracking configuration.
- Modelling quality at the station-end of the tracking configuration.

In terms of technological quality, the adopted order of the data types is : SLR / DORIS / Altimetry crossovers / Direct altimetry. This order is consistent with the relative RMS levels of the previous Chapter, apart from the direct altimetry data which has not been used so far. It is inevitable to assign altimetry data a lower ranking than the crossover data. Note that PRARE has not been included here. Even though it should probably be inserted in the ranking between SLR and DORIS, various practical difficulties still existed at the time of this study and prevented the PRARE data from reaching its full potential. Heuristic weights applied to the different datasets are :

$$\begin{aligned} \mathbf{SLR} &= \mathbf{1.25 \times DORIS} \\ \mathbf{DORIS} &= \mathbf{1.25 \times XO} \\ \mathbf{XO} &= \mathbf{2.50 \times Altimetry} \end{aligned} \tag{7.1}$$

These factors are intended as direct multiplication factors to the normal matrices, so the associated standard deviations should be divided by the square roots of these factors. The 'translation' of a range value into a range rate value is done via the typical overall SLR and DORIS RMS values from Chapter 6, i.e using a factor 5.7 / 0.056 = 100.

Relative weighting in terms of 'station' quality is only relevant for SLR data. The DORIS stations can be considered to be of equal quality as they are technically identical and hardly affected by geographical asymmetry (see also Figure 6.5). For altimetry and/or crossovers only a single 'station' exists - namely the sea surface geometry model - and no relative station weights are applied[1]. For the SLR data, three different levels of station quality are introduced, i.e. 'good', 'average' and 'poor'. This separation in three quality categories is based on the results of SLR analysis in Chapter 6, as represented below in Figure 7.1. Although this classification is again based on the typical RMS of data residuals, it does not use the RMS values directly in order to avoid perfect aliasing of eventual systematic errors. The relative weights are quantified by setting a priori standard deviations of 5, 10 and 20 cm

[1] Note that altimetry data or crossover data is sometimes weighted according to geographic region, on the basis of the geogrophically correlated or anti-correlated orbit error. However, in this thesis these signals will be treated as output signals of the orbit computation process, rather than input signals.

respectively for the three groups. The stations of group 1 will therefore influence the normal matrix $20^2 / 5^2 = 16$ times more than group 3, because of the quadratic relations in (2.11).

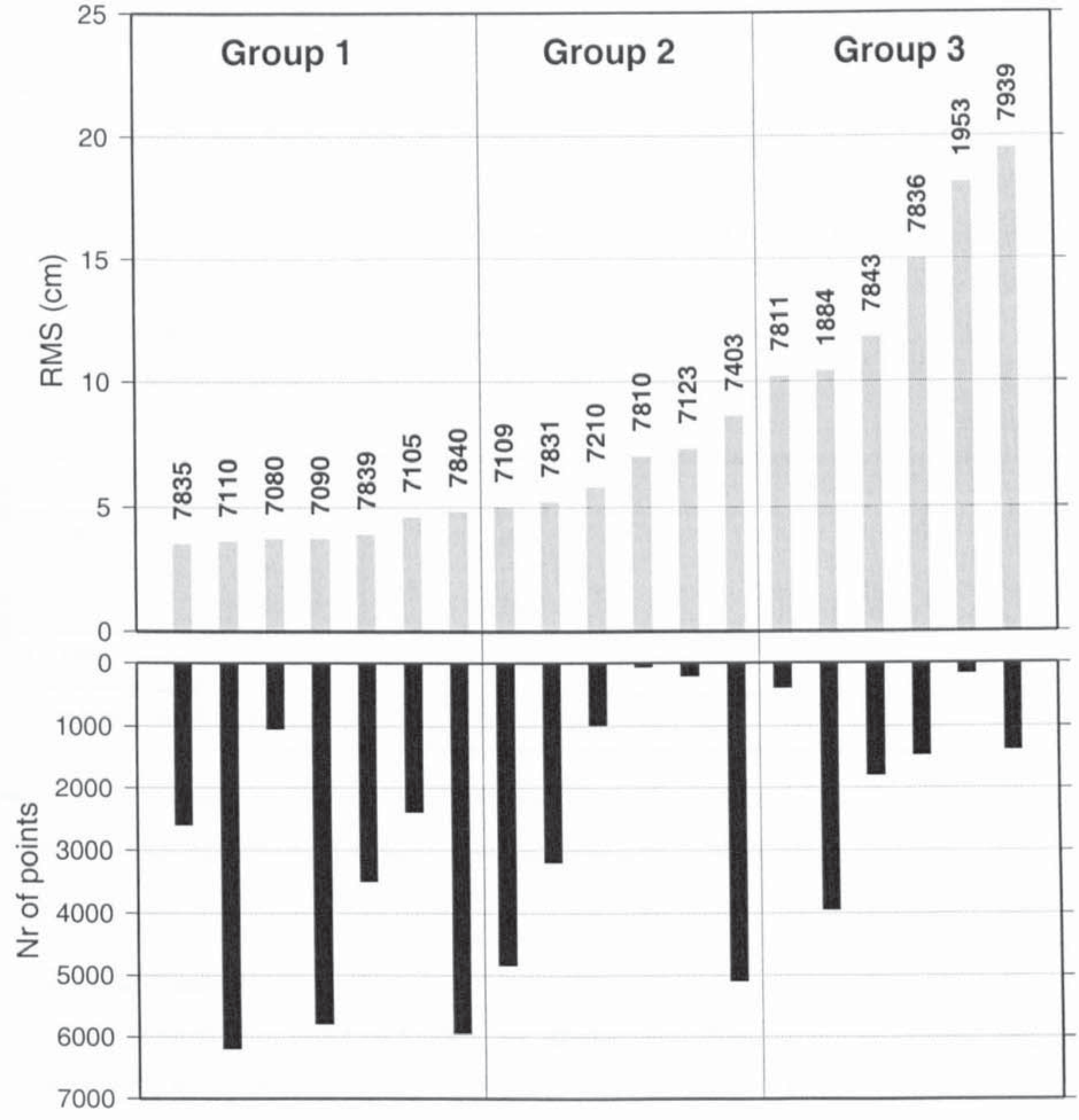


Figure 7.1 Post solution RMS and amount of observations per station. The dataset used for this analysis is the ERS-1 SLR data from Chapter 6. On the basis of these results, the SLR ground stations have been divided into three groups, with a priori standard deviations of 6 cm, 12 cm and 24 cm respectively.

The differences that exist in terms of 'satellite' quality are in fact better described as differences in orbit quality. The low orbits of the ERS satellites are more sensitive to atmospheric and gravity field perturbations than the higher TOPEX orbit, and their pass rate over a tracking station will be higher. Both effects decrease the signal to noise ratio for ERS with respect to TOPEX, in particular for land-based tracking data. The relevant 'ranking' in terms of orbit quality is therefore simply :

TOPEX/Poseidon > ERS-1 and ERS-2

In the simultaneous orbit solutions, the TOPEX data will be given a 20 % lower sigma than the ERS data. This number is derived from the overall SLR and SXO fit for the two satellites in Chapter 6, which differ by some 20 %.

The relative weights between the various datasets, as discussed in the above, are summarised in Table 7.1. To indicate that these values are based on experience rather than on unambiguous scientific analysis, the values obtained from the combined discussions above are rounded to the nearest 0.5 cm.

Data type		ERS-1	TOPEX/Poseidon
SLR	Group 1	6.0 cm	5.0 cm
	Group 2	12.0 cm	10.0 cm
	Group 3	24.0 cm	20.0 cm
DORIS		N.A.	0.065 cm/s
Single and dual crossovers		9.5 cm	8.0 cm
Altimetry data		24 cm	20 cm

Table 7.1 Quantification of a priori standard deviations for the datasets

7.1.2 Data Quantity

As clearly illustrated in Figure 6.5, there are substantial differences in the sizes of different datasets. The effect of dataset size is similar to that of relative weighting : if 1000 SLR observations and 4000 DORIS observations are combined in a solution, the DORIS data will implicitly have a four times larger impact on the normal equations. If all observations would be free of systematic error, dataset sizes would not have a relevant impact on the solution : the two datasets then define the same - perfect - solution. However, the differences in sizes of the datasets will still tend to interfere with the relative scaling in terms of measurement type (7.1) that was introduced before. A compensation is therefore introduced via scale factors that are more or less inversely proportional to the sizes of the datasets :

SLR = SXO = DXO = 4 x DORIS = Altimetry

In other words, the DORIS data is downscaled with a factor 4, to compensate for its dominant dataset size (note that the DORIS data as used in these Chapters is already a sampled subset from the total amount of available data, only one of every four observations is used). It is recalled that the altimetry data is not used for orbit determination directly, its relative weight will only be of importance in Chapter 8.

7.2 Results of simultaneous orbit solutions for ERS-1 and TOPEX/Poseidon

For the one year period of overlap between the ERS-1 and TOPEX/Poseidon arcs of Chapter 6, full simultaneous orbit solutions have been computed that each covered a 35-day repeat cycle of ERS-1 and (overlapping) four consecutive repeat cycles of TOPEX/Poseidon. The main subject of interest is the analysis of simultaneous solutions versus the single-satellite solutions from Chapter 6. To this purpose the orbit computations for the simultaneous solutions have all been run a second time, during which the TOPEX/Poseidon orbital parameters were kept fixed at their a priori values (while still using the dual satellite crossovers between ERS-1 and TOPEX/Poseidon). This second case represents the 'classical' application of dual crossovers between ERS and TOPEX/Poseidon.

The ERS-1 orbit solutions resulting from the two cases can be compared directly, and can also be compared with the single-satellite solutions from Chapter 6. For TOPEX/Poseidon, the case in which dual crossovers are applied while keeping the ERS orbits fixed has not been run, because the effect of simultaneous solutions can be studied more clearly from the ERS-1 orbits than from the insensitive TOPEX/Poseidon orbits.

To summarize, five one-year sets of orbit solutions are of interest in this Section :

(1) ERS-1 orbits from the single satellite solutions of Chapter 6 (5-day SXO)

(2) ERS-1 orbits from simultaneous solutions with TOPEX/Poseidon

(3) ERS-1 orbits using dual crossovers with TOPEX but keeping the TOPEX orbits fixed

(4) TOPEX/Poseidon orbits from the single satellite solutions of Chapter 6 (5-day SXO)

(5) TOPEX/Poseidon orbits from simultaneous solutions with ERS-1

The simultaneous solutions (2) and (5) were computed with crossover datasets limited to 5 days between crossings (nominal) as well as crossovers limited to 3.5 days between crossings [2]. The orbits (3) were only computed using the 5-day SXO and DXO datasets. For comparisons of different orbit solutions, the (larger) 5-day datasets will always be used. Some statistics for the 3.5 day datasets will be presented to again illustrate the effect of reduced sea surface variability, as was done in Chapter 6. Note also that as a result of the differences in orbital period and inclination, the spatial and temporal density of the dual satellite crossover dataset for ERS-1 and TOPEX/Poseidon is larger than that of the single crossover datasets for each satellite individually.

The results of the orbit computations are presented in several Tables and Figures below. First, Tables 7.2 and 7.3 show all numerical characteristics of the solutions by means of tracking data residuals and orbit comparisons. The orbits obtained in the simultaneous solutions have been compared with the same reference orbits that were used in Chapter 6, for the two satellites. The Tables 7.2 and 7.3 are therefore the equivalents of Tables 6.3 and 6.4, allowing direct numerical comparisons between the results from the simultaneous solutions and from the single satellite solutions. These tables also contain the results from the solutions with a 3.5 day limit between crossings.

To analyse the orbit errors in more detail, the geographically correlated and geographically anti-correlated orbit errors (see Section 5.6) were computed. The geographically anti-correlated orbit error is derived from the post-solution SXO residuals and can therefore be computed for all 5 orbit sets of interest. The geographically correlated orbit error is derived from the post-solution DXO residuals, and is therefore not available for the single-satellite solutions of Chapter 6. The intention of the Figures 7.2 to 7.7 is to visualize and compare geographical patterns in these orbit errors, and some additional processing was needed between the computation of the errors data itself and the generation of the graphical Figures. The crossover residuals are only available in grid points that include crossover locations - concentrated around certain latitude bands, see also Figure 6.2 -, so the data has been extrapolated in directions along the groundtrack, after which it was gridded / smoothed in longitudinal directions. The result is a 'global' dataset, with continuous spatial coverage between the latitude bands covered by the crossover locations as well as coverage of the diamond-shaped areas between groundtracks. This shows the geographical patterns much more clearly.

[2] Note that this can be accomplished by an input option of Faust to set lower and upper time limits to the tolerated interval between crossings, avoiding the need to generate separate crossover datasets.

Figures 7.2 and 7.3 show the geographically anti-correlated orbit errors for ERS-1 and TOPEX/Poseidon respectively, for the orbits from the single-satellite solutions from Chapter 6. For ERS-1, only the 1-year overlap period with TOPEX/Poseidon was used in all Figures. Figures 7.4 a/b and 7.5 a/b show the geographically anti-correlated and correlated orbit errors for the orbits of the simultaneous solutions (at 5-day crossover datasets), for the two satellites. For comparison, the Figures 7.6 a/b show the same for ERS-1 for the case (3) in which the TOPEX/Poseidon orbits were kept fixed. Obviously, the geographically anti-correlated orbit errors for TOPEX/Poseidon will in that case be identical to those from Figure 7.5, while the geographically correlated orbit error can not be computed in a sensible way : the DXO data has not been used to adjust these TOPEX orbits, so the post-solution DXO residuals do not provide the desired information. Also note that the geographically correlated errors for ERS-1 and TOPEX (Figures 7.4b and 7.5b) are derived from the same DXO residuals (though the sign will be opposite for one satellite relative to the other). The differences between the two Figures can therefore provide some information about the quality of the computation process that has lead to these plots, because the Figures should in principle be very similar.

To visualize the effect of the simultaneous solution process, the differences between the plots 7.4 and 7.6 are shown in Figure 7.7. Note that the scale of the plots 7.2 to 7.6 is kept the same, in order to make direct comparisons easier. However, the difference plots in Figure 7.7 use a smaller scale because the overall signal amplitude is smaller.

Solution ID		Epoch MJD	RMS of residuals (cm)					Orbit comparisons with simultaneous soutions (cm)					
			Simultaneous solution					UT / CSR		Delft DEOS		Single arc	
Multi arc	Single arc		SLR	SXO < 5	SXO < 3.5	DXO < 5	DXO < 3.5	Radial	Horiz.	Radial	Horiz	Radial	Horiz.
RMS for complete period >			6.58	9.10	8.72	8.22	7.86	6.20	54.5	5.83	54.3	5.6	34.6
8	57	48978.00	5.22	9.62	8.22	7.42	7.22	6.2	72.1	5.3	46.6	4.4	26.6
	58	48983.00						6.7	77.8	5.8	45.1	5.0	23.7
	59	48986.00						5.2	64.7	6.4	55.0	2.5	35.3
	60	48992.00						5.7	66.8	5.1	41.5	4.7	34.4
	61	48997.00						5.3	73.8	6.9	52.5	5.3	24.2
	62	49001.00						5.4	48.5	6.8	77.2	3.9	33.9
9	63	49005.50	8.43	12.7	10.26	8.63	8.91	9.6	40.3	5.4	57.0	6.8	21.6
	64	49010.50						8.3	76.8	5.5	63.5	3.4	35.4
	65	49015.50						8.4	63.2	7.7	69.3	5.1	16.3
	66	49022.00						10.8	63.8	5.3	59.8	4.0	28.0
	67	49026.00						4.4	31.6	6.5	48.8	1.8	27.6
	68	49031.00						5.6	46.3	6.9	59.6	6.4	30.3
	69	49036.00						6.3	84.8	5.3	49.1	4.2	20.6
10	70	49040.50	8.06	9.00	8.23	9.28	8.72	6.4	52.6	5.5	34.7	5.6	39.8
	71	49046.00						9.9	48.5	6.3	42.2	12.5	23.1
	72	49051.00						8.6	44.9	6.7	34.9	5.9	22.2
	73	49055.00						8.3	32.2	5.4	35.5	4.2	34.0
	74	49061.00						5.2	66.1	6.8	44.0	4.6	26.5
	75	49066.00						6.6	48.8	6.1	64.2	5.8	35.5
	76	49072.00						4.8	45.9	5.0	42.9	4.3	38.3

11	77	49076.00	6.33	9.95	9.69	8.02	7.48	3.4	73.9	5.0	23.7	4.9	30.4
	78	49080.00						7.2	81.9	4.6	37.5	5.8	22.8
	79	49085.00						4.8	89.5	5.4	53.8	3.2	33.2
	80	49089.00						3.4	44.6	5.5	55.5	3.8	23.9
	81	49092.00						5.6	66.4	6.7	29.2	4.7	22.6
	82	49097.00						2.0	68.1	5.3	64.0	8.5	20.2
	83	49101.00						8.1	52.2	5.9	31.0	7.6	25.0
	84	49104.00						3.4	43.2	6.4	36.4	7.4	34.3
	85	49107.00						4.6	43.0	6.6	49.9	2.0	36.4
12	86	49112.50	7.23	8.32	9.36	7.86	7.25	6.4	57.1	5.3	64.1	3.3	25.5
	87	49116.00						4.6	46.1	6.7	36.4	5.0	29.8
	88	49120.00						4.3	52.5	6.5	47.5	3.8	31.1
	89	49124.00						5.3	59.2	6.3	29.6	6.9	27.4
	90	49128.00						5.8	53.9	5.7	34.3	5.6	13.3
	91	49132.00						5.8	67.2	5.4	62.9	2.4	23.2
	92	49136.00						4.5	57.9	6.9	38.2	3.4	12.0
	93	49142.00						5.2	54.4	5.4	43.2	4.1	28.8
13	94	49146.00	7.61	9.28	8.76	8.28	8.02	5.5	52.5	5.7	44.0	4.4	27.5
	95	49150.00						5.8	45.3	6.4	38.3	3.5	10.7
	96	49154.00						6.4	62.4	6.3	23.8	5.0	22.7
	97	49158.00						4.3	56.0	4.0	27.8	5.2	29.4
	98	49162.00						6.7	54.2	6.1	31.6	4.8	12.6
	99	49166.00						5.0	63.7	4.5	49.2	5.7	26.2
	100	49169.00						6.1	58.2	5.3	48.8	6.5	34.5
	101	49173.00						5.8	60.9	5.9	27.4	5.6	22.0
	102	49177.00						5.4	61.9	6.5	52.3	5.5	30.5
14	103	49181.00	5.34	9.61	9.20	8.00	8.31	5.4	45.4	5.3	35.8	4.4	27.8
	104	49185.00						6.6	52.1	7.8	60.7	4.2	38.7
	105	49189.00						6.9	58.0	7.4	44.3	3.3	35.5
	106	49193.00						6.3	54.4	5.8	37.0	5.0	22.5
	107	49197.00						7.7	47.2	6.9	25.4	3.9	11.3
	108	49201.00						7.4	62.7	5.4	32.1	5.3	20.2
	109	49205.00						6.2	48.4	6.8	43.0	5.7	37.6
	110	49209.00						6.7	53.0	6.2	58.8	4.7	24.1
	111	49213.00						5.4	55.4	5.3	54.4	5.6	33.4
15	112	49216.00	6.95	8.42	8.23	7.45	7.52	6.3	43.6	5.7	32.6	5.3	28.8
	113	49221.00						6.5	48.5	6.4	28.3	4.2	22.3
	114	49226.00						5.8	52.5	7.1	34.0	4.8	39.2
	115	49230.00						4.4	62.4	6.0	40.8	5.3	28.4
	116	49233.00						6.3	45.0	5.0	52.3	4.5	24.3
	117	49236.00						6.7	40.3	5.4	53.2	5.7	34.1
	118	49240.00						5.4	31.0	4.1	43.2	3.1	26.0
	119	49244.00						5.8	40.4	6.9	37.8	3.8	32.9
	120	49248.00						6.5	53.9	5.7	28.4	5.8	19.5
16	121	49252.00	6.86	8.47	8.61	7.42	6.93	6.3	54.8	6.8	44.9	4.9	20.6
	122	49256.00						5.8	64.3	5.5	26.1	4.2	21.3
	123	49260.00						5.6	62.6	5.5	36.6	5.4	19.8
	124	49264.00						6.2	48.1	6.3	42.5	5.5	23.7
	125	49268.00						7.8	43.3	5.1	24.0	6.6	25.2
	126	49272.00						7.5	55.5	5.0	28.9	3.2	37.3
	127	49276.00						6.3	65.7	6.4	31.6	4.6	22.9
	128	49280.00						5.5	47.5	6.2	58.1	4.8	14.0
	129	49284.00						5.1	61.4	5.8	49.8	5.4	27.0
17	130	49288.00	6.01	9.38	9.09	8.17	8.12	6.6	60.3	5.3	31.3	4.7	32.2
	131	49291.00						5.3	42.9	5.1	21.5	5.2	20.4
	132	49295.00						6.2	67.4	6.8	36.2	5.1	26.6
	133	49299.00						6.7	64.5	5.9	42.0	4.0	33.2
	134	49303.00						5.4	76.6	6.3	40.9	5.2	16.8
	135	49307.00						5.3	66.0	6.7	26.7	3.5	29.6
	136	49311.00						6.8	62.2	5.3	37.5	6.8	32.7
	137	49315.00						5.4	79.6	5.6	23.2	3.2	26.1
	138	49319.00						4.3	51.8	6.4	58.7	4.4	21.5
18	139	49323.00	6.01	8.26	7.69	7.76	7.23	NO COMPARISONS WERE MADE FOR SOLUTION 18					
	140	49327.00											
	141	49331.00											
	142	49335.00											
	143	49339.00											

Table 7.2 Orbit determination results for ERS-1 in simultaneous solution

Solution ID		Epoch MJD	RMS of residuals (cm, cm/s)					Orbit comparisons (cm)			
ERS sol	TP cycles		Simultaneous solutions with ERS-1					GDR Ephemerides		Multi vs. Single	
			SLR	SXO	DORIS	DXO < 5 d	DXO < 3.5 d	Radial	Horiz.	Radial	Horiz.
RMS for complete period >			5.62	7.02	0.0560	8.22	7.86	4.7	47.6	3.2	22.2
8	10	48977.39	5.30	7.21	0.0546	7.42	7.22	4.5	48.4	3.7	14.9
	11	48987.30						4.7	32.3	2.3	20.6
	12	48997.22						5.3	63.6	2.6	14.2
9	12	as 8	5.19	6.59	0.0551	9.63	8.91				
	13	49007.13						5.2	49.6	2.6	15.3
	14	49017.00						4.2	59.7	3.2	30.9
	15	49026.97						5.8	52.7	3.9	13.5
10	16	49036.88	5.74	7.01	0.0580	9.28	8.72	4.9	42.8	2.0	24.5
	17	49046.80						5.5	41.9	2.4	23.2
	18	49056.71						5.4	88.0	3.4	28.2
	19	49066.63						4.6	43.4	2.3	16.0
11	20	49076.54	5.13	6.03	0.0563	8.02	7.48	4.2	37.6	2.5	23.2
	21	49086.46						4.1	46.0	2.5	37.7
	22	49096.37						4.1	40.6	2.2	24.1
	23	49106.29						4.0	53.0	3.3	24.2
12	23	as 11	6.22	7.52	0.0546	7.86	7.25				
	24	49116.21						3.2	53.3	3.3	22.6
	25	49126.12						4.9	38.7	3.0	23.6
	26	49136.04						4.3	48.8	2.1	17.2
13	27	49145.95	5.13	6.25	0.0549	8.28	8.02	4.1	62.6	3.9	26.0
	28	49155.87						5.2	34.7	2.6	26.2
	29	49165.78						3.9	39.4	2.5	17.3
	30	49175.70						3.9	52.7	3.5	13.5
14	30	as 13	5.61	7.02	0.0552	8.00	8.31				
	31	49185.62						4.1	49.7	2.9	19.5
	32	49195.53						4.0	33.8	3.3	15.2
	33	49215.36						5.8	43.7	2.4	16.3
15	33	as 14	5.29	6.92	0.0558	7.45	7.52				
	34	49225.28						3.7	46.8	2.5	28.3
	35	49235.19						4.2	69.7	3.8	23.0
	36	49245.11						4.1	43.1	2.3	23.9
16	36	as 15	5.23	7.11	0.0561	7.42	6.93				
	37	49255.02						4.8	44.7	2.8	17.8
	38	49264.94						4.3	25.9	3.1	12.2
	39	49274.86						5.6	52.1	3.2	20.7
	40	49284.77									
17	40	as 16	6.21	7,28	0.0552	8.17	8.12				
	41	49294.69						5.1	50.7	3.0	27.6
	42	49304.60						5.1	41.5	3.2	27.7
	43	49314.52						5.2	49.5	2.7	12.7
18	43	as 17	5.08	7.31	0.0550	7.76	7.23				
	44	49324.43						4.2	43.0	2.3	18.3
	45	49334.35						3.9	27.6	2.6	23.1

Table 7.3 Orbit determination results for TOPEX/Poseidon in the simultaneous solutions. For each ERS-1 repeat cycle of 35 days, the 4 or 5 TOPEX cycles were chosen that fully enclose the ERS-1 solution.

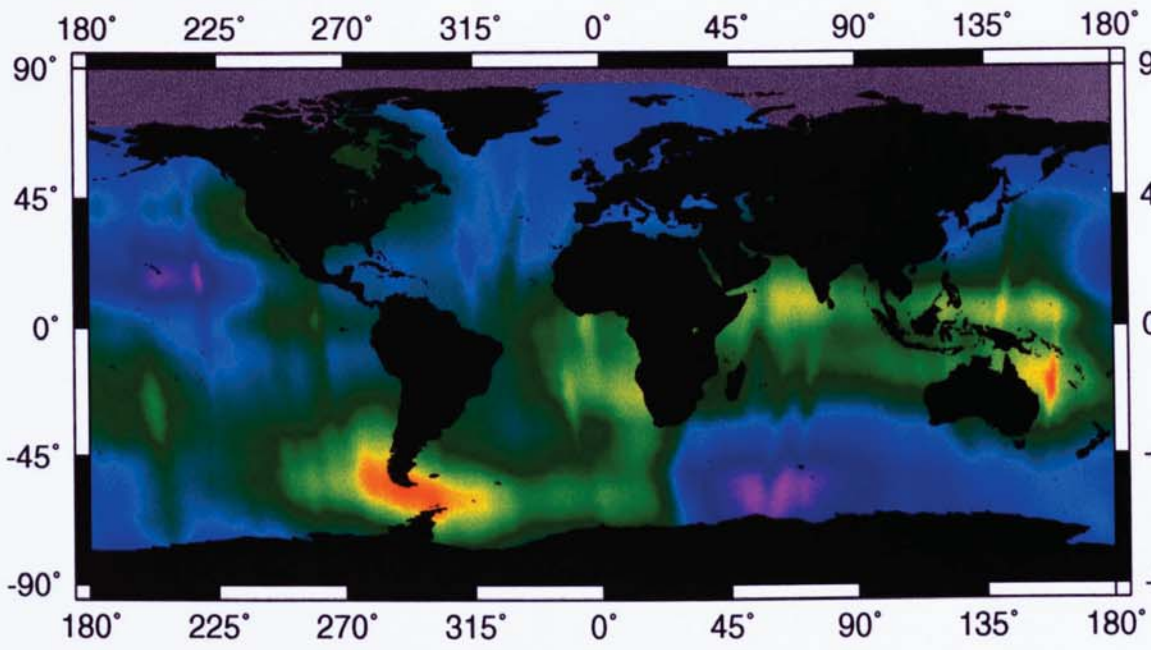
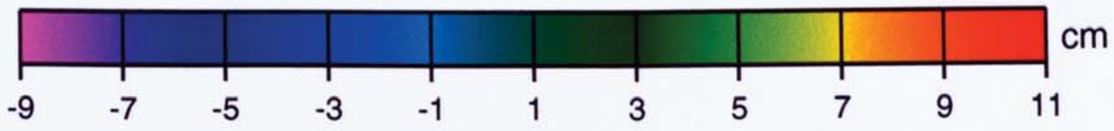


Figure 7.2 Geographically anti-correlated orbit error for ERS-1, from single-satellite solutions of Chapter 6. **RMS 6.35 cm**



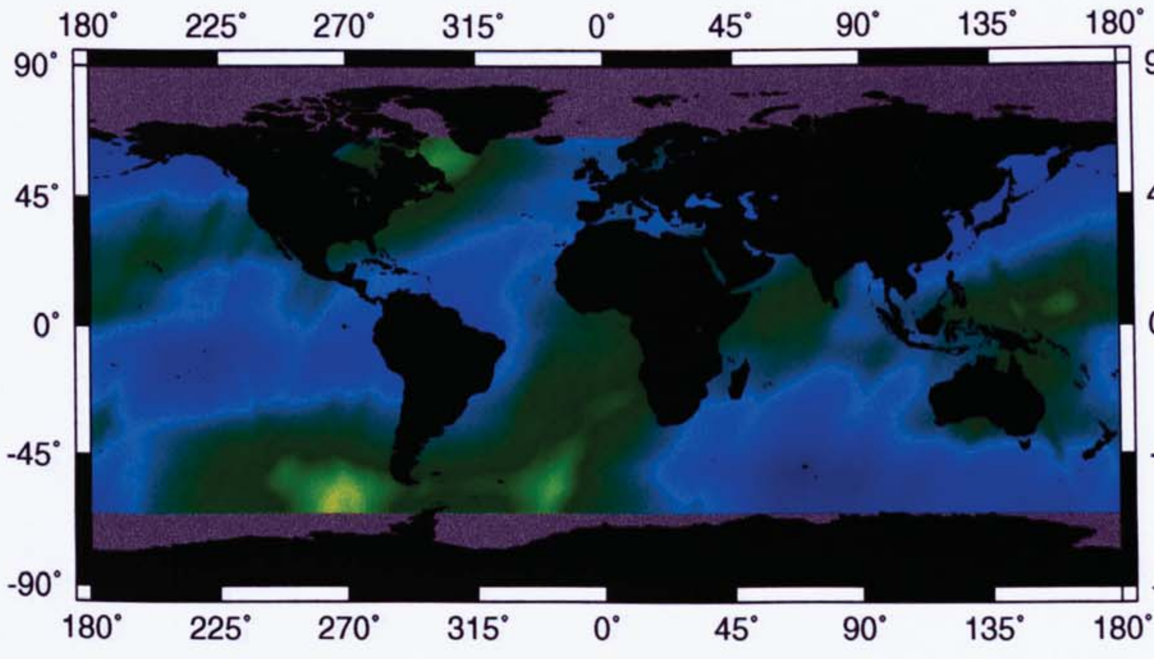


Figure 7.3 Geographically anti-correlated orbit error for TOPEX/Poseidon, from single-satellite solutions of Chapter 6. **RMS 4.92 cm**

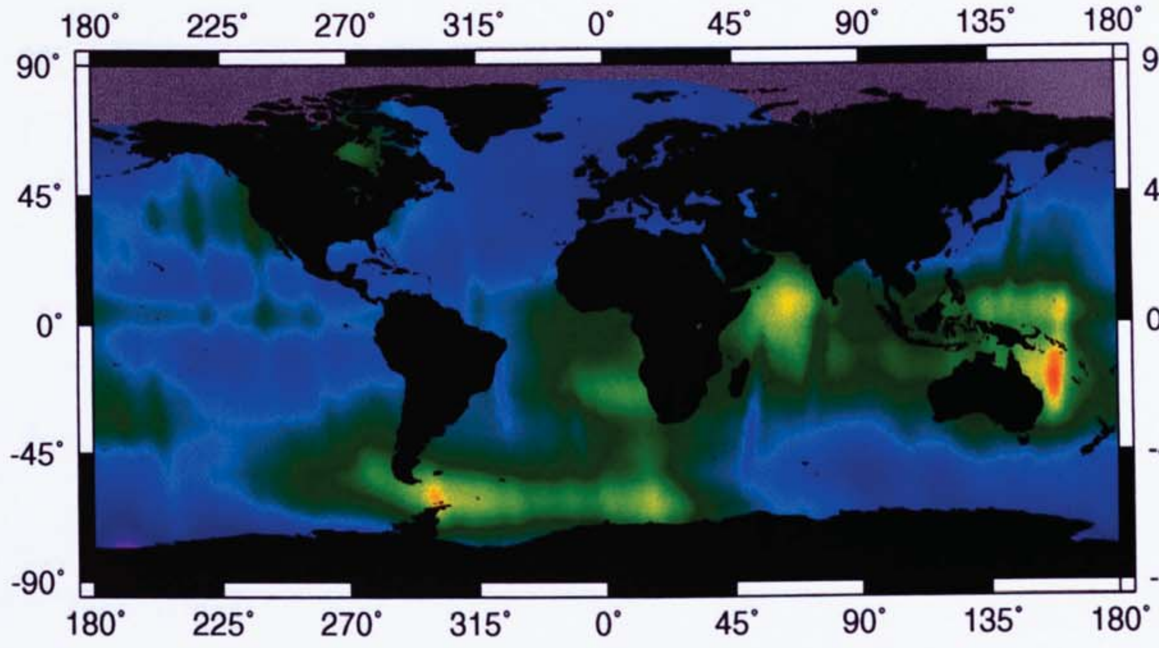
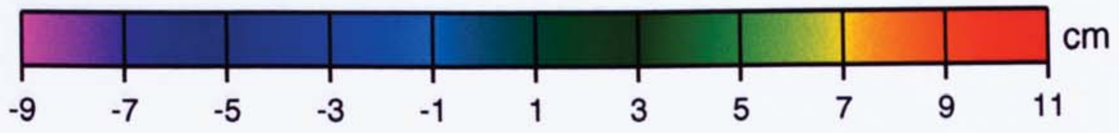


Figure 7.4a Geographically anti-correlated orbit error for ERS-1, from DXO solution in which the TOPEX/Poseidon orbit is kept fixed. **RMS 5.86 cm**



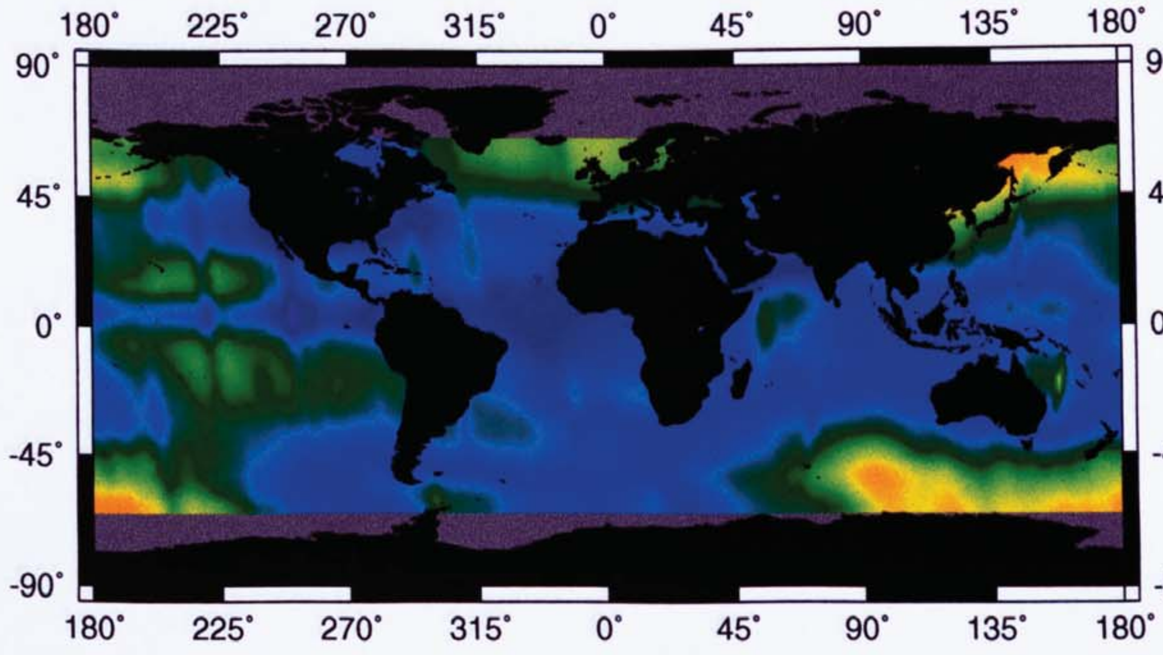


Figure 7.4b Geographically correlated orbit error for ERS-1, from DXO solution in which the TOPEX/Poseidon orbit is kept fixed. **RMS 6.13 cm**

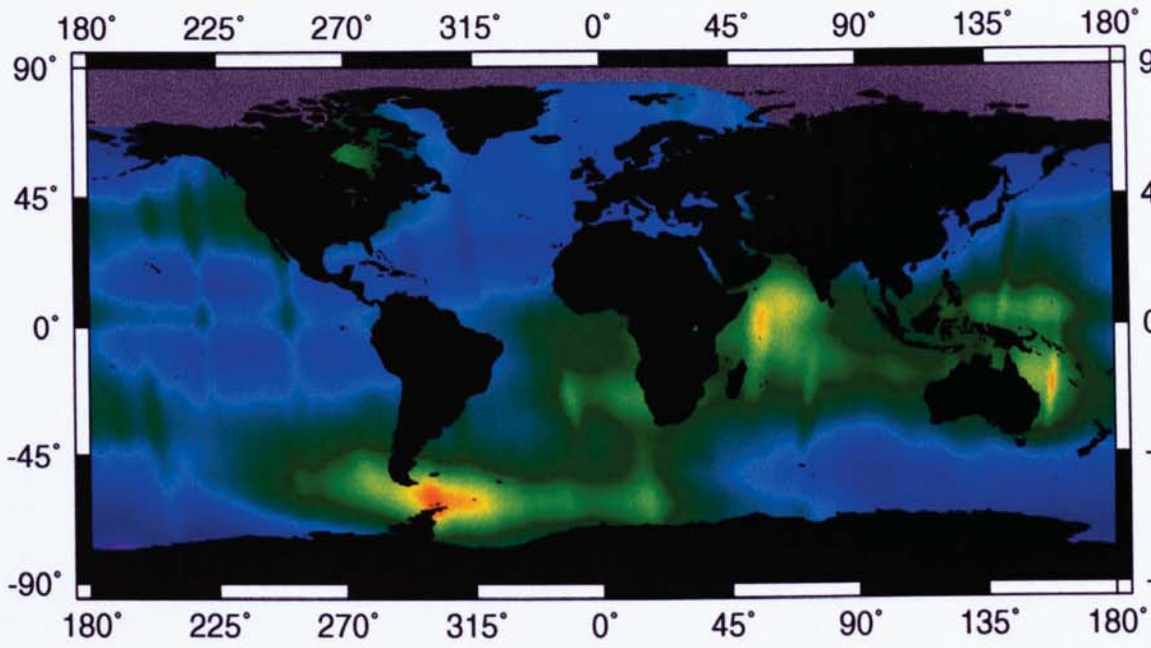
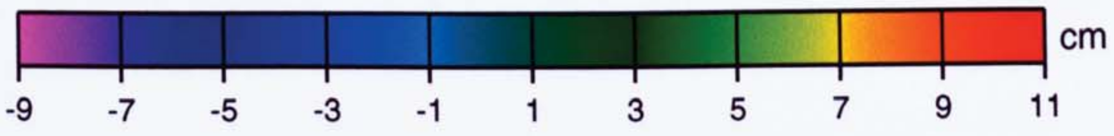


Figure 7.5a Geographically anti-correlated orbit error for ERS-1, from simultaneous solution with TOPEX/Poseidon. **RMS 5.18 cm**



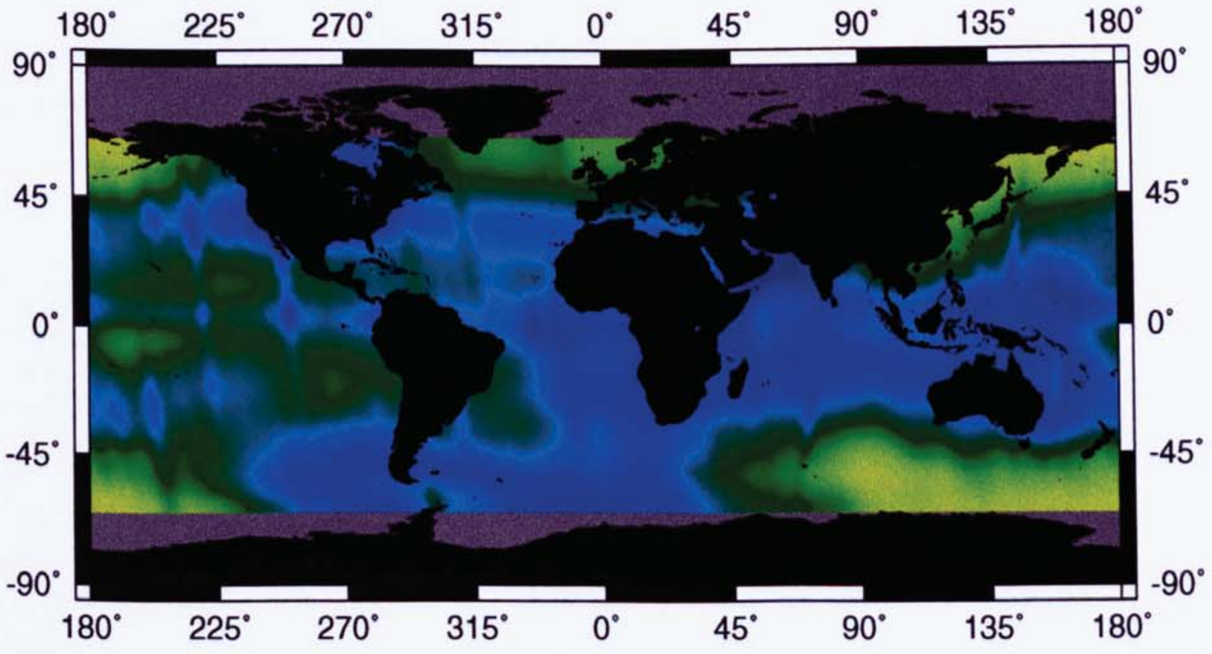


Figure 7.5b Geographically correlated orbit error for ERS-1, from simultaneous solution with TOPEX/Poseidon. **RMS 5.76 cm**

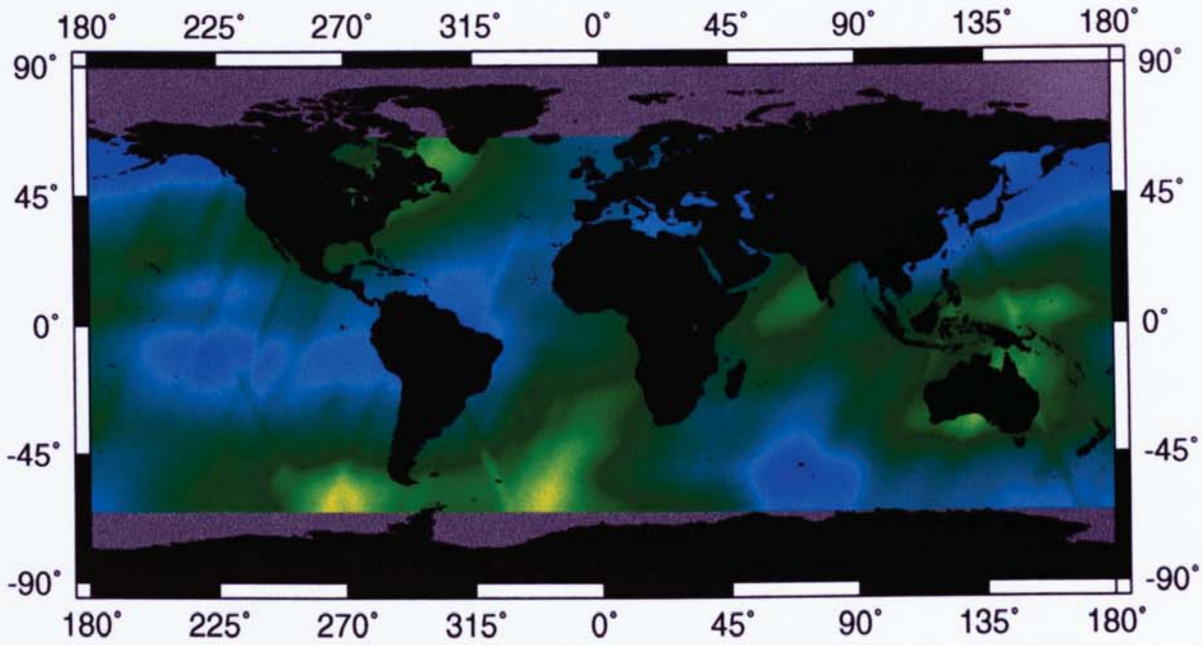
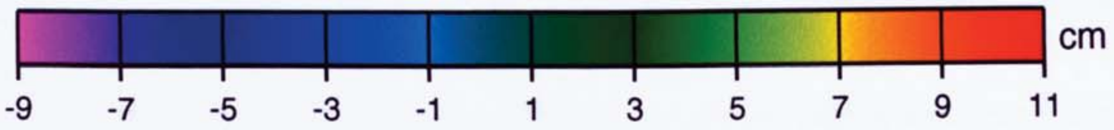


Figure 7.6a Geographically anti-correlated orbit error for TOPEX/Poseidon, from simultaneous solution with ERS-1. **RMS 4.85 cm**



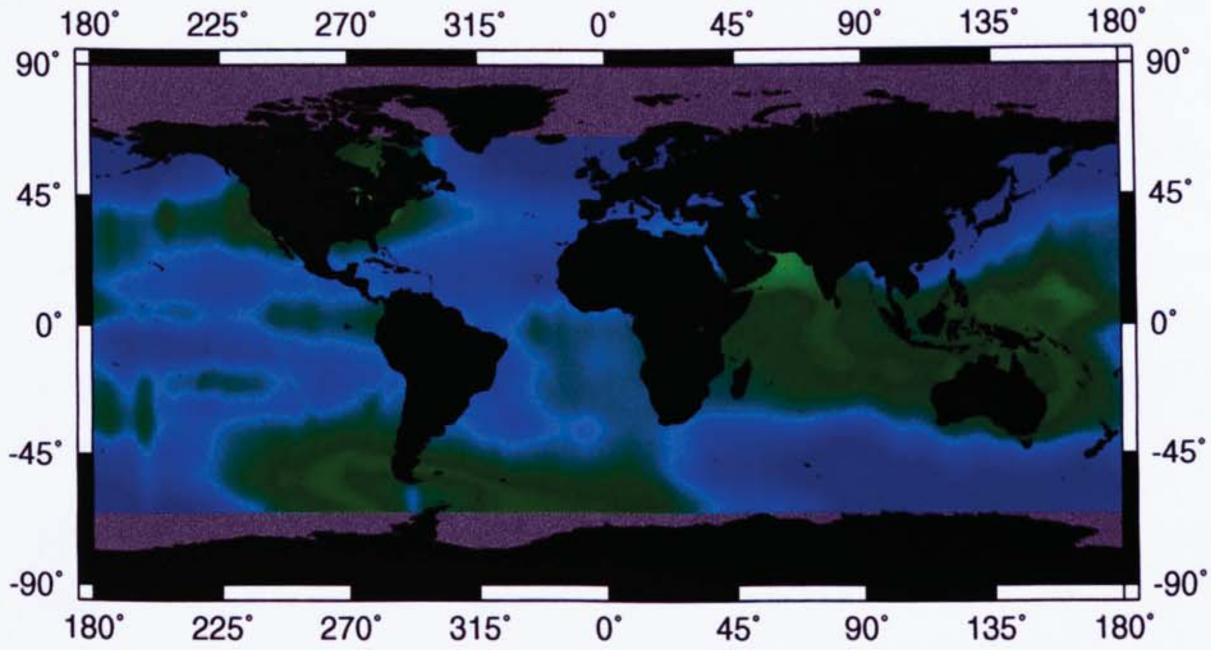


Figure 7.6b Geographically correlated orbit error for TOPEX/Poseidon, from simultaneous solution with ERS-1. **RMS 5.70 cm**

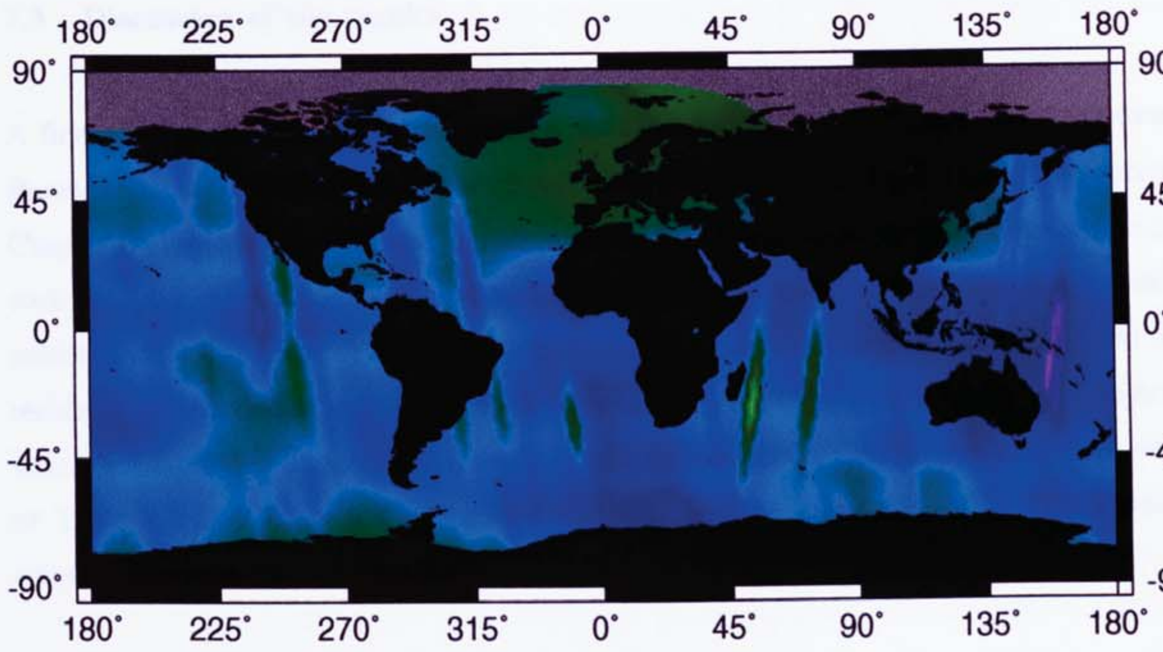
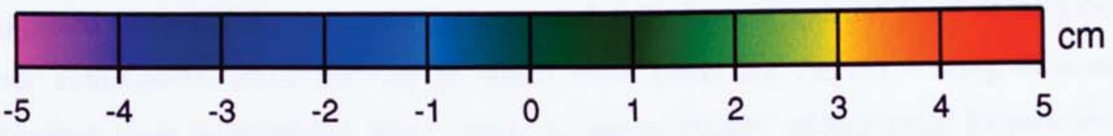


Figure 7.7a Change in geographically anti-correlated orbit error for ERS-1, induced by simultaneous solution (difference between Figure 7.5a and 7.4a). **RMS 2.12 cm**



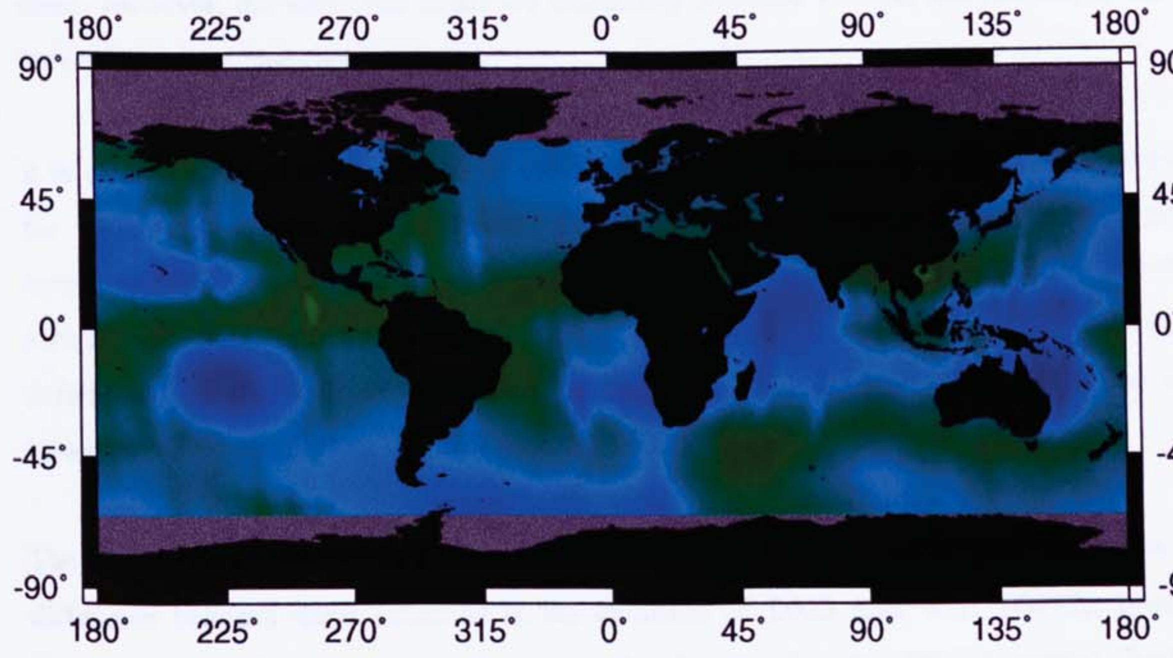


Figure 7.7b Change in geographically correlated orbit error for ERS-1, induced by simultaneous solution (difference between Figure 7.5 b and 7.4 b). **RMS 1.32 cm**

7.3 Discussion of the results

A first point of interest is that the tracking data RMS for the ERS-1 SLR and SXO data as shown in table 7.1 hardly seems to change with respect to the single satellite solutions of Chapter 6 (Table 6.3). Of course, no strong changes should occur in these datasets, because such differences would either indicate a substantial systematic error in the single satellite solutions or a systematic error in the application of the DXO dataset. The SXO and SLR residuals of individual arcs only appear to change within the RMS values of Chapter 6, which indicates that important systematic errors do not occur. Comparison of overall values of Table 7.2 and Table 6.4 shows that the same can be concluded of TOPEX/Poseidon, where differences are even smaller.

More interesting than the numeric results from the Tables are the Figures of geographically correlated and anti-correlated errors, which reveal various other details. The indications 'geographically correlated' and 'anti-correlated' should be interpreted with some care : the suggested error signals can never be separated fully from a spectral band of neighbouring error components, while the way in which these errors are derived - using an averaging technique over geographical bins - tends to merge closely related error frequencies even more. However, the employed terms are commonly accepted in orbit analysis and it was felt inevitable to continue their use here.

It is clear that these errors tend to be more substantial for ERS-1 than for TOPEX/Poseidon, for which two reasons can be found. At first, the lower orbit of ERS-1 implies greater sensitivity to mismodelling of gravity field terms (i.e. stronger geographically correlated errors) and drag, producing noisier orbits. At second, the TOPEX orbits are mainly determined on the basis of land-based tracking data (SLR and DORIS), which has much better global coverage than the SLR data for ERS-1.

The latter effect is very clear from comparison of Figure 7.2 and Figure 7.4a. The only difference between these solutions is the inclusion of DXO data with TOPEX, providing dense global tracking relative to an external reference (i.e. the TOPEX orbits). Both the geographically correlated and the anti-correlated error can be expected to have strong zonal components, not only because crossovers tend to be concentrated along bands of constant latitude but also because of the high orbital inclinations which result in 'sampling' of the globe in latitudinal directions. With the perigee more or less coinciding with the largest argument of latitude, the signals cos(n.M) and sin(n.M) will be predominantly zonal.

In Figure 7.4a such zonal patterns can be distinguished, especially over the Pacific Ocean where crossover influence will be most significant. However, Figure 7.2 is dominated by various other features that seem to confirm a poor fixation of the orbit with respect to the Earth Fixed reference frame. The ERS-1 single satellite solution is essentially connected to the terrestrial frame by means of the rather sparse SLR dataset (see Figure 6.5) while the SXO data - due to the deliberate elimination of the geographical constant term - hardly improves the positioning of the orbit relative to the planet. A strong regional feature can for instance be observed in Figure 7.2 around the Hawaii SLR station (7210), while this feature is no longer present in Figure 7.4a. Also, the maximum and minimum values noticed at latitudes around 50 degrees South are much more pronounced in Figure 7.2 than in Figure 7.4a. Looking at Figure 7.3 for TOPEX, a similar though smaller feature can be noticed around Hawaii, with opposite sign.

As these plots concern the geographically anti-correlated error signal - the component with opposite sign for ascending and descending passes - neither an error in the station coordinates for 7210, nor a range bias seem to be likely explanations. Such systematic errors would be geographically correlated rather than anti-correlated. However, the 7210 SLR data could be suffering from a timing bias, which would, in combination with Earth rotation, result in opposite errors for ascending and descending passes (both passes then seem to be shifted in the same along-track direction, i.e. in opposite geographical directions). Also, Earth rotation may cause a timing bias to show up with different signs for ERS-1 and TOPEX due to the fact that the ascending pass for TOPEX occurs from West to East, while the ascending pass of the retrograde ERS-1 orbit will pass from East to West. Errors in the UTC time tag for the SLR data will therefore pull an ascending TOPEX pass in the opposite direction of an ascending ERS-1 pass.

In Figure 7.4a the effect is suppressed thanks to the good fix of the ERS-1 orbit relative to the TOPEX orbit (the SLR data from Hawaii has smaller impact on the orbit), and via this orbit to the Earth Fixed frame. The strong maximum and minimum at high Southern latitudes (Figure 7.2) might be indicating a poor definition of the Earth's rotation pool, although this has not been investigated further. Again, this would confirm a poor fix between the orbits from the ERS-1 single satellite solution and the terrestrial reference frame.

The ERS-1 geographically correlated error in Figure 7.4b seems to share some global patterns with the anti-correlated error. This can be explained from strong noise levels in these areas, in which many error frequencies will be present (hence also the two specific error signals isolated here). Because a substantial component of the geographically correlated error for ERS-1 will be gravity-field induced, the Figure 7.4b may indicate certain weaknesses of the employed JGM-3 model. In particular, doubts about the J_3 coefficient ($C_{3,0}$) reported by Lemoine *et al.* (1996) may be confirmed from the error patterns over the Atlantic Ocean. This error is associated with the pear-shape of the Earth and may therefore also help to explain the 'polar-type' errors in the Southern oceans.

The crucial result, for as far as this thesis is concerned, is obtained from a comparison of Figure 7.5 with Figure 7.4. While the TOPEX/Poseidon orbits were held at fixed values in the solutions from Figure 7.4 (the classical application of dual crossovers), in Figure 7.5 the orbit errors after the simultaneous solution are shown. The differences (Figure 7.7a) are small but significant. The overall RMS of the errors (both correlated and anti-correlated) appears to be reduced reduced with respect to Figure 7.4. Because the only difference between the two cases is the fact that the TOPEX/Poseidon orbits are kept fixed for Figure 7.4, this reduction in RMS confirms that a component of the TOPEX/Poseidon orbit error must be present in the error signals of Figure 7.4 : the TOPEX/Poseidon orbits are of better overall quality than the ERS-1 orbits, but are not perfect either.

The other important point of interest is if the simultaneous solution process injects significant ERS-1 orbit error signals into the TOPEX/Poseidon orbits. This can be investigated to some extent by means of the Figures 7.3, 7.6a and 7.7b. The global RMS of the error signal in Figure 7.6a is in fact lower than that of Figure 7.3, although the difference is negligible. Another point of interest is that the single satellite crossovers for TOPEX/Poseidon (Table 7.2) also show a marginally lower overall RMS than in the single satellite solutions from Chapter 6. Although it is probably too optimistic to conclude that the TOPEX orbits actually gain from the simultaneous solution approach, the size of the datasets involved in this analysis is sufficiently large to conclude that the orbit precision for TOPEX does not suffer in any relevant way.

An important remark must be attached to the above results. The quality of the TOPEX/Poseidon orbits used in this thesis is still inferior to the precision claimed by NASA and NOAA (Cheney *et al.*, 1994). The main reason for this difference is suspected to be the fact that the surface force models (NASA TOPEX Box-Wing model, Marshall 1995) have been calibrated in another software environment than faust. Only within the environment in which a model is calibrated will it be possible to subsequently obtain its highest precision. This is due to the inevetable absorption of systematic errors in the model. Such absorbed systematic errors will be removed from solutions in the same environment in which the calibration took place, while they will be introduced in solutions obtained within any other environment. In addition, this other environment will contain its own systematic errors, which are not removed by the calibrated model.

The above line of reasoning implies that simultaneous solutions will at present be more advantageous to an independent solution process - as implemented in faust - than to a solution process in which all applied models have been tailored to their implementation environment. The dual crossovers certainly contain useful information for the determination of the ERS-1 orbits, but the errors resulting from relatively 'noisy' Aston orbits for TOPEX/Poseidon (at 4 cm radial RMS, compared to 2 cm RMS for the NASA orbits) will introduce stronger error signals into the ERS-1 orbits if the TOPEX orbits are not allowed to move simultaneously. Lower error levels in the TOPEX orbits can not only reduce the difference between 'classical' DXO applications and simultaneous solutions, but may also imply the injection of more substantial ERS-1 orbit errors into the TOPEX/Poseidon orbits. However, further investigation of this matter would only be possible within the calibration environment of the TOPEX models at NASA/GSFC; unfortunately the GEODYN software does not allow for simultaneous solutions involving dual satellite crossovers with ERS-1.

7.4 Example of simultaneous solution for the ERS tandem mission

Although the analysis of ERS-2 data does not form part of this study - mainly due to problems with the PRARE system -, the simulataneous multi-satellite solution offers some interesting possibilities for the combined analysis of data from the ERS tandem mission. As an illustrative example it will be demonstrated here how the atmospheric drag parameters of the two satellites can be combined, as explained in Chapter 4.

The two ERS satellites are separated over 60 degrees in their orbits, leading to a situation in which ERS-1 will pass through more or less the same atmospheric conditions as ERS-2 after less than 25 minutes. The only differences in experienced gas densities will be due to the small differences in orbital heights and longitudinal offsets between the two satellites, due to the rotation of the atmosphere (with the Earth) during the 25 minute separation interval, and due to the variability of atmospheric gas densities within this period. However, as drag scale parameters are typically kept constant during many hours (i.e typically 6 or 12 hours for ERS) the atmospheric variability during the 25 minute interval will be modest in comparison to the atmospheric variations during the full validity interval of the parameter. This means that in principle the drag scale factors for the two satellites should show very similar temporal variations, as they are supposed to absorb the same errors in atmospheric density. For a period of 35 days, the drag scale parameters of the two satellites have been analysed, as shown in Figure 7.8. The overall behaviour is indeed similar, but important differences between the satellites occur which can hardly be explained from actual variations in the atmospheric density. This suggests that through correlations with other parameters in the solution process (in particular the orbit state vector and the empirical cyclic accelerations) the drag scale parameters absorb signal components that should rather be absorbed by other orbital parameters.

In a simultaneous solution process for the two satellites the parameter values can be shared between the two satellites, which implies that about twice as many observations are available for the estimation of the parameter in comparison to the single satellite solutions. To improve the temporal resolution of the model these validity intervals can then be reduced, e.g. from 6 hours to 4 hours. In the latter example, the data density per parameter is still higher than what is feasible in a single satellite solution, while at the same time the shorter intervals will help to account for the true atmospheric variability in a better way.

The results of this example are shown in Table 7.4, where some statistics of the solutions are summarised. The Table shows that the RMS of the SLR residuals has been improved for both satellites, while the variance of the drag scale parameters has also been reduced. The statistics in the Table are related to the sigma values as obtained from the post-solution normal matrix, as explained in Section 2.2. Although the orbit differences with reference orbits have hardly been affected, the drag scale parameters appear to have been separated from the other orbital parameters in a more realistic way, which will reduce orbit noise. Although this analysis is kept rather rudimentary, it shows how simultaneous solutions can be used to the benefit of both involved orbits.

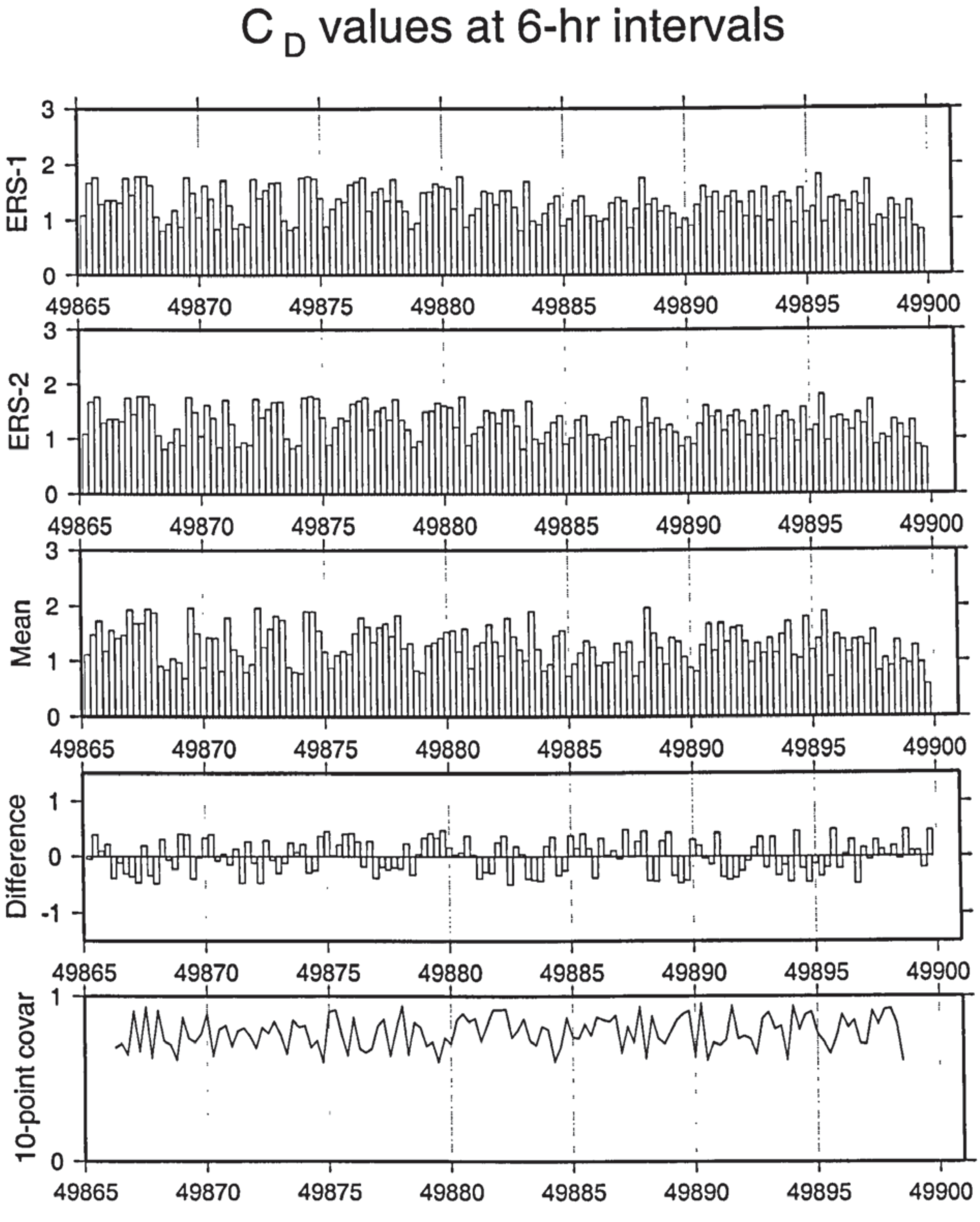


Figure 7.8 Analysis of drag scale parameters at 6-hr intervals for ERS-1 and ERS-2. From top to bottom, the plots represent (a) and (b) the estimated drag scale factors for ERS-1 and ERS-2, (c) and (d) the mean and difference between the two parameter series. Plot (e) provides the cross-correlation function over a 10-point moving frame, showing an average cross-correlation between the two series above the 0.8 level.

		ERS-1 6-hr intervals	ERS-2 6-hr intervals	Simultaneous 4-hr intervals
Tracking data residuals				
	SLR	5.82 cm	5.92 cm	5.43 cm
	SXO	9.23 cm	9.02 cm	9.08 cm
	DXO	-	-	9.22 cm
Orbit comparisons				
DEOS orbits	Radial	7.3 cm	-	7.3 cm
	Horizontal	46.7 cm	-	42.8 cm
CSR orbits	Radial	6.3 cm	-	6.0 cm
	Horizontal	58.2 cm	-	56.6 cm
Post-solution parameter sigmas from the covariance matrix				
	Amount of parameters	141	141	211
	RMS over all drag parameters	0.28	0.36	0.17
	Minimum	0.12	0.13	0.10
	Maximum	0.67	0.81	0.64

Table 7.4 Statistics for the simultaneous solution with shared drag parameters. Note that the single satellite solutions involve a total of 282 drag parameters, i.e. more than the 211 of the simultaneous solutions. Despite of this, the simultaneous solution provides better results.

7.5 Summary and conclusions

The simultaneous solutions between ERS-1 and TOPEX/Poseidon appear to perform in line with expectation. The effect of the simultaneous solution process for one year of ERS-1 and TOPEX/Poseidon orbits is summarised in the Figures 7.7a and b, which capture the essence of this thesis.

The single-satellite orbit solutions for ERS-1 seem to suffer from regional biases, induced by a poor connection between the orbits and the terrestrial reference frame. Because the land-based tracking data for TOPEX/Poseidon shows much better global coverage, the dual corssovers between the two satellites contribute substantially to the quality of the 'overall altimetry instrument' introduced in Chapter 1, even if the TOPEX orbits are not yet allowed to move. The simultaneous solutions for the both satellite orbits will further improve the ERS-1 orbits, while the TOPEX/Poseidon orbits can not be observed to degrade in any way. The simultaneous solution process therefore appears to be useful, although the relatively 'poor' quality of the Aston orbit solutions for TOPEX/Poseidon (4 cm radial RMS) may be a contributing factor to this success.

In an example for the ERS tandem mission, it was demonstrated how in a simultaneous solution process for the two satellites the drag scale parameters can be considered as satellite independent parameters. By allowing both satellites to contribute to these common parameters the data density per parameter improves, which in turn allows for a higher temporal resolution of the drag model. It is recommended to extend such analysis of the ERS tandem mission in the future, when reliable PRARE data will be able to contribute to the solution process. In that case, the ERS-1 orbits can still take advantage of the PRARE tracking system even though this will be in another way as was anticipated before its launch.

Simultaneous solutions for space geodesy — Chapter 8

Chapter 7 showed that the use of dual crossovers in simultaneous solutions can indeed provide a signal that is unavailable from single satellite solutions or from parallel (uncorrelated) multi-satellite solutions. With this outcome the thesis could in fact be concluded, as the implementation and validation of simultaneous orbit solution techniques was the main goal of the study. However, it was felt useful to spend a final Chapter on applications of the simultaneous solution process to geophysical research, in particular to the enhancement of the gravity field model. The work in this area had already started during the final phases of the current study, and has continued ever since. Some more recent results obtained with Faust will therefore also be summarized in this Chapter.

8.1 General remarks on model calibration

The levels of accuracy that are currently reached in precise orbit determination are such that even small systematic errors - model limitations in the satellite dynamics, in the measurement geometry, in the data pre-processing steps, etcetera - start to affect computed orbits. Two factors associated with this problem make it sensible to go through the substantial effort of computing a new gravity field solution, and will therefore be briefly discussed first.

8.1.1 System dependency

If we consider two different orbit determination software environments X and Y, model parameters that are estimated within environment X will almost inevitably absorb some systematic errors from X. If the model is subsequently implemented in environment Y, the result will be that Y not only has to cope with its own particular systematic errors - which have *not* been aliased into the model - but also with whatever errors from environment X are present in the model. Consequently, orbits computed within environment Y will not be able to reach the level of post-solution tracking data fit that can be obtained within environment X, despite of the fact that exactly the same model is being used. To summarise : dynamic models used in orbit determination software are becoming more and more system dependent.

The above effect has been discussed in Section 7.3 as a likely explanation for the fact that the TOPEX/Poseidon orbits produced with Faust appear to have a radial precision in the order of 4 cm, which is less precise than the 2 cm level obtained - at least over the Pacific - by NOAA (Cheney *et al.*, 1994). The employed models are virtually identical, but both the gravity field model and the TOPEX Box-Wing model have been calibrated within the same NASA software environment that is used at NOAA to compute precise orbits. Faust has to live with the combination of its own systematic errors and with any systematic errors of the NASA software that are present in the models, while neither of these two will appear in the computed NOAA orbits.

The obvious cure is to re-estimate the set of model parameters within Faust. The errors 'X' would be corrected, while the internal errors 'Y' of Faust would partially disappear in the new model parameters. Model X will continue to perform optimally within its own X environment, while the Faust model will perform optimally within the Faust environment.

It is important to keep in mind that any model that is calibrated within a single software system (and subsequently produces orbits with a better post-solution fit to the tracking data within that system) will not necessarily constitute an absolute improvement to the a priori model. The best test for orbital quality is not the post-solution data fit, but the direct comparison with orbits produced by external - independent - computation environments. The only useful test for absolute model quality is the application of the model for independent satellites (i.e. satellites that are not involved in the solution process), and preferably also within an independent software environment. Only if in such an independent case the obtained post-solution data fit is also improved, an absolute improvement to the a priori model can be confirmed.

8.1.2 Gravity field tayloring

As explained in Section 4.1, the harmonic coefficients of the spherical expansion model that describes the gravity field of the Earth are observable from their perturbing effect on the satellite orbit. Because these perturbations rapidly decrease with increasing orbital radius and degree of expansion, a low satellite like ERS will sense the gravity field in much more detail (higher degree and order of expansion) than a high satellite like TOPEX/Poseidon. In addition, differences in orbital periods and groundtrack repeat patterns esult in satellite-dependent bands of resonant frequencies, and bands of frequencies for which the satellite is hardly sensitive.

These two considerations imply that a single satellite mission can hardly expected to produce a model for 'the' gravity field, but that a particular satellite is likely to prefer a specific gravity field model. A model with general validity can only be obtained from a combination of different satellites, each sensitive to its own particular model frequencies. In practice, generic gravity field models like JGM models are computed from elaborate combination solutions (Nerem *et al.*, 1994) in which tracking data from many satellites is extended with surface measurements. Although such a solution is quite accurate for a wide range of applications, it will always remain a compromise between the gravity signals produced from different datasets : for a single satellite the generic solution is sub-optimal. Consequently, each satellite is likely to benefit from a calibration excercise in which its own tracking data is 'overweighted' in such a way that it will steer the solution towards a model that is particularly in agreement with the observations made to its own orbit. This process is known as gravity field tayloring.

For non-conservative force models like surface forces and manoeuvre calibrations, the first aspect (system dependency) will be the most important justification to perform a model re-calibration, while for generic geophysical models like the JGM3 gravity field the aspect of tayloring will be the strongest argument. However, if the model is estimated within a single software environment, it is quite likely that some system dependent errors may be absorbed in the new gravity field model, while at the same time the taylored model has little value for geophysical applications other than improved orbit determination for its 'own' satellite.

8.2 Gravity field solutions with Faust

At first sight, the simultaneous solution process seems particularly suited for the computation of generic gravity field models because it can combine various different tracking data sets, while correlating different orbits not only via the gravity field model parameters but also directly by means of dual crossovers. Nonetheless, the main purpose for the gravity field computation effort at Aston was the improvement of the ERS orbits, i.e. the elimination of possible alien systematic errors -(while probably absorbing some from Faust) and the tayloring of the gravity field towards the ERS orbit.

In recent times, various other groups have performed tayloring excercises of the JGM3 model for ERS orbit computation, notably UT/CSR (TEG3 model) and DUT/DEOS (DGM4 model). In both cases *internal* improvements to the solution process were apparent, but absolute applicability of the new solutions could hardly be consolidated. The latter is especially difficult because at the current levels of orbit precision any form of improvement will only be at the sub-centimetre level. The ever-present noise bands that surround comparisons between external orbit solutions (see Chapters 6 and 7) will subsequently make it very difficult to confirm or deny any improvements in either of the compared orbit solutions.

Neither of the two taylored solutions mentioned above includes the specific signal provided by the dual-crossovers with TOPEX/Poseidon, and it can therefore be expected that a simultaneous solution with Faust will be able to reduce the geographically correlated ERS-1 orbit errors to a greater extent than solutions that do not use the DXO information directly. It is important to realise that dual crossover data can only be applied realistically to gravity field tayloring if both involved satellite orbits are also solved for, i.e. in a true simultaneous solution process. As soon as the dual crossover dataset is introduced in the solution process, both a posteriori orbit errors become a source of noise in the taylored gravity field solution. If the TOPEX orbits are fixed, its post-solution orbits (which have not changed with respect to tha a priori orbits) would not be consistent with the computed gravity field, and the differences between orbits resulting from the initial and new gravity models would most likely be absorbed in the new model.

8.2.1 The first solution

The main effort in gravity field tayloring is related to the heavy computational load involved in the estimation of such a large amount of parameters. The DGM4 model was obtained from a more elegant method based on the single crossover residuals directly (Scharroo & Visser, 1997). However, to involve the dual crossover information, a least-squares estimation of the full 70 x 70 gravity field was performed with Faust. This Section will describe the work that resulted in a first Faust gravity field solution (Boomkamp & Moore, 1998), and discuss some suspected problems with the model. The next Section describes an improved second solution.

In the first solution, tracking data was used from a period of 40 days fully enclosing the ERS-1 repeat cycle 15 of its mission phase C. This period may seem short for the estimation of geophysical models from a stochastical process, but the amount of observations is already substantial (see Table 8.1). In addition, all information used for the computation of the JGM3 model is included by using its covariance matrix to constrain the solution. Using two or more repeat cycles would hardly introduce new ERS-1 information, although it would probably improve the signal-to-noise ratio. It was considered that such a small potential gain would hardly justify a doubling of the already considerable computational effort. The estimated parameters for this first solution are listed in Table 8.2. Apart from all orbital parameters and gravity field parameters, some expansion coefficients for the S2 and M2 tides were included, as well as a 20 x 20 dynamic sea surface topography model. The tidal model is not primarily included to try and improve its own parameters - which would hardly be possible from just 35 days of data - but mainly in an attempt to avoid absorption of long-period tidal signals in the gravity field model. The sea surface topography model is included because the model used in the altimetry pre-processing stage related to another period, and changes between that model and the actual topography at the time of the solution period would show up in the altimetry observations.

Dataset		Number of observations	Residuals	
			Before solution	After solution
ERS-1	SLR	3031	5.57 cm	4.28 cm
	Altimetry	30668	21.31 cm	20.53 cm
TOPEX	SLR	9052	4.62 cm	4.48 cm
	Altimetry	50863	23.21 cm	23.08 cm
	DORIS	20649	0.57 cm/s	0.57 cm/s
both	Dual Crossovers	12477	10.15 cm	9.68 cm

Table 8.1 Survey of observations involved in the first gravity field solution. Note that the single satellite crossovers were not used for this solution process, because their contribution to a geographically correlated signal like the gravity field was not expected to validate the considerable effort of generating normal matrices for the large SXO datasets.

The actual solution process consisted of many short-period runs with Faust, each writing its normal equations to file as discussed in Section 2.7. The triangular sub-matrix related to the gravity field coefficients (5035 parameters) has a size in excess of 100 MB, the second biggest matrix being the block matrix that connects the sea surface topography parameters with the gravity field coefficients (17 MB). After each short-arc run the normal matrices related to global parameters were therefore accumulated into a single matrix, so that only one of these large files would have to be stored. However, to keep some freedom in setting a priori weights for different tracking data types, separate solutions were generated for each of the six data types involved (see Table 8.1). The total required storage space to generate all multi-run normal matrices was therefore still more than 800 MB. This, in combination with the rather excessive CPU-time requirements for the generation of all normal matrices, forms a major practical problem that prohibits repetitive runs.

As can be seen from the data fit results in Table 8.1, the initial solution provided a reduction of the RMS of the tracking data, as could be expected if such a large number of parameters is solved for. However, for orbit arcs outside the solution period no similar improvements were noticed. Alternative matrix inversions, involving only the matrices related to various subsets of the tracking data, different estimated parameters (in particular, not solving for the tides and/or sea surface topography model) or involving different a priori data weights, did not change this situation substantially.

Analysis of the actual changes relative to the JGM3 model reveal some interesting features (see Figure 8.1). In the first place, the changes in the low degree terms seem to be rather large, which suggests geometrical biases in the tracking data. This seems to be confirmed by the North-South bias that is notable from the changes at the poles, although no tracking data is available at these high latitudes.

A second problem that appeared in all performed inversions was a large correlation between the gravity field parameters and the tidal coefficients. It was concluded that the 35 day period is too short to reliably solve for the S2 and M2 tides (note that the main tidal perturbation for TOPEX has a 60 day period; see for instance Ray & Cartwright, 1994). Solutions that did not include the tidal parameters still showed a domination of low-degree corrections, in particular the bias that seems to occur over the Indian Ocean. Because the low degree terms of the gravity model are better established than the high degree terms, the occurring changes were considered unrealistic. In the end, it was concluded that the altimetry time-tag biases that were applied to the a priori data (as estimated from the a priori orbit solutions, but not estimated during the gravity field runs) should either not have been added, or should have been allowed to move during the gravity solution. By imposing an unadjustable time-tag bias upon the altimetry data, a disagreement is introduced between the altimetry data and the land-based tracking data, which leads to inconsistencies between the variational partials and the geoid height partials for the altimetry observations.

A re-run of the altimetry and crossover solution would effectively imply a re-run of the entire solution process (the small amount of SLR observations hardly affects the total required processing time), which was not considered useful. Instead, the gravity field work with Faust was continued for a different dataset, leading to the solution that will be described in the next Section.

Parameter type	**ERS-1** **9 arcs**	**TOPEX** **4 arcs**	**Global**	Total
Initial orbit state vector	1 per arc	1 per arc	-	78
Drag scale factors	6-hr intervals, shared between continuous arcs	daily	-	138
Solar radiation pressure	Not solved for	1 per arc	-	4
Empirical harmonic	daily, along & cross track	daily, along & cross track	-	296
Relative crossover bias	-	-	1	1
Altimeter time tag bias	A priori value applied but not solved for	-	-	-
Altimeter range bias	1 per arc	1 per arc	-	13
Gravity field coefficients	-	-	full 70 x 70, except for C21 and S21	5035
Tidal coefficients	-	-	S2 and M2 expansions from CSR3	168
Sea surface topography	-	-	20 x 20 except for constant term	440
Total				6168

Table 8.2 Parameters involved in the initial gravity field solution

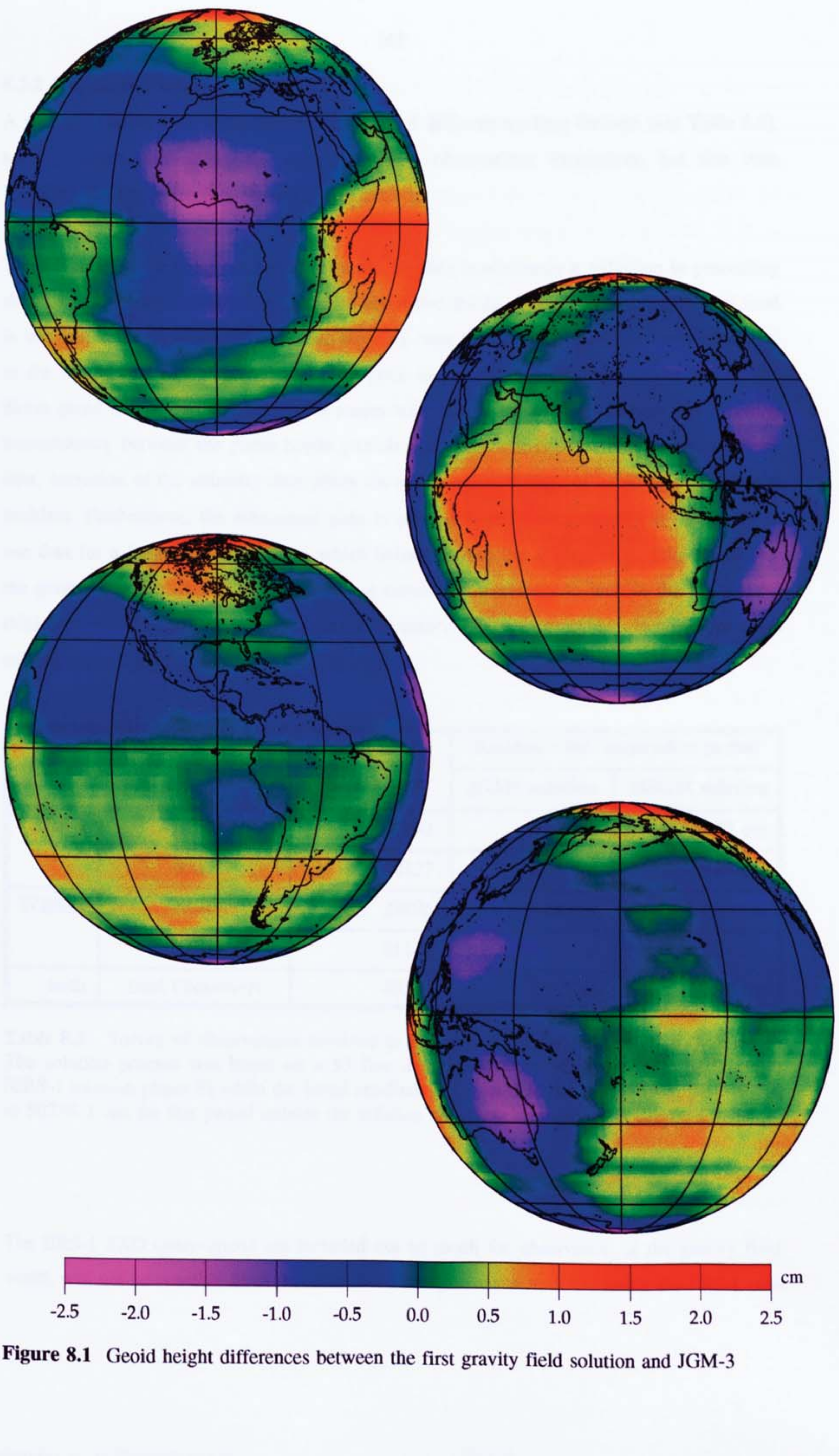


Figure 8.1 Geoid height differences between the first gravity field solution and JGM-3

8.2.2 The second solution : MSGM-1

A second solution was performed on the basis of different tracking datasets (see Table 8.3), notably without involving the direct altimetry observations themselves, but this time including the single satellite crossovers for ERS-1.

The advantage of leaving out the direct altimetry data is obviously a reduction in processing time : the altimetry observations comprise about two thirds of the total tracking dataset used in the first solution, while they are also the only ones that produce the large matrices related to the sea-surface topography model. The price to pay for this omission is the loss of the direct geoid height partials. Because a suspected problem in the first solution was a timing inconsistency between the geoid height partials and the variational partials for the altimetry data, exclusion of the altimetry data offers the additional advantage of avoiding this potential problem. Furthermore, the substantial gain in processing efficiency offers the possibility to use data for a longer period of time, which helps to avoid the absorption of tidal signal into the gravity solution. It was subsequently not considered necessary to include the M2 and S2 tidal expansions in the solution, nor was it necessary - or even possible - to solve for a sea surface topography model.

Dataset		Number of observations	Residuals for *independent* period	
			JGM3 solution	MSGM solution
ERS-1	SLR	7392	7.61 cm	5.52 cm
	SXO	10127	10.18 cm	8.88 cm
TOPEX	SLR	29896	4.59 cm	4.48 cm
	DORIS	61124	0.57 cm/s	0.55 cm/s
both	Dual Crossovers	48542	9.32 cm	7.99 cm

Table 8.3 Survey of observations involved in the second gravity field solution (MSGM-1). The solution process was based on a 55 five day period from MJD 49616 to MJD 49671 (ERS-1 mission phase E) while the listed residuals are related to the period from MJD 50202 to 50238. Even for this period outside the solution arc, clear improvements can be noticed.

The ERS-1 SXO observations are included not so much for observation of the gravity field, which will not be possible from this data, but mainly to assist in constraining the ERS-1 orbit parameters. For TOPEX this will not be necessary, because sufficient tracking data is available in the form of SLR and DORIS observations. Note also that in this solution the data was taken from the first geodetic phase of ERS-1, so that the groundtrack pattern does not repeat after 35 days. This implies that throughout the 55 day solution period the crossover data will add new spatial information. The slightly longer period spanned by the 6 enclosing TOPEX cycles (74 - 79) ensures that the dual crossover data density does not reduce substantially towards the start and end of the solution interval for the ERS-1 data, which could lead to undesired side effects for the earliest and latest days of ERS-1 orbits.

The result of the new set-up and longer data period is a simpler and more robust solution method than in the first computation. The gravity model obtained from this approach provides clear improvements of tracking data fit, also outside the solution interval itself which effectively confirms that the solution process has been successful (see Table 8.3). The tayloring aspect, as discussed in Section 8.1.1, appears to have succeeded with the MSGM-1 solution. However, it is useful to investigate if substantial systematic errors from the Faust solution environment have been absorbed in this model, as described in Section 8.1.1. To investigate this, a comparison was made between orbits computed for the GFZ satellite, first with the JGM3 gravity model and subsequently with the MSGM model. Because GFZ has an orbit that is even lower than that of ERS-1 (ca. 385 km), its sensitivity to high degree gravity harmonics is greater. Any non-gravitational signal that would have been absorbed in the MSGM solution should therefore have a notable perturbing effect on the GFZ orbits. Table 8.4 shows that this is not the case.

MJD	**JGM-3**		**MSGM-1**	
	Number of observations	RMS	Number of observations	RMS
50196 - 50201	669	49.64 cm	668	48.87 cm
50233 - 50238	1125	57.43 cm	1126	58.78 cm
50238 - 50243	2309	66.92 cm	2303	66.49 cm
50243 - 50248	1250	51.89 cm	1250	51.67 cm
50248 - 50253	979	42.09 cm	981	40.24 cm

Table 8.4 Orbit determination results for GFZ. Virtually no changes are noticed between model performances, which suggests that no substantial systematic errors were absorbed during the MSGM computation process.

8.3 Conclusions

The application of the simultaneous solution process to gravity field tayloring for ERS-1 has been discussed. Given the large computational load involved with such a process, so far only two solutions have been computed with Faust, but initial results are promising.

The first solution appeared to perform well for the orbits within the solution period but did not show significant improvements for other, independent, periods of the ERS-1 mission. Suspected problems were the length of the solution period, causing poor separability between tidal signals and gravity signals, and an inconsistency between the dynamic and geometric altimetry information caused by the included altimetry timetag bias.

A second solution avoided these potential problems by not using the direct altimetry data but only the crossover information, which also allowed for a longer solution period. This in turn reduces the risk of absorbing tidal signal in the gravity field model. Good results were obtained for independent periods of the ERS-1 mission, confirming the positive impact of the tayloring process that involves the dual crossover signal. Because orbit computations for the GFZ satellite remained perfectly neutral under the changes to the gravity field, it can be concluded that no substantial systematic errors were absorbed in this second solution.

Future work in this area may enable further improvements of the MSGM model, notably by also including ERS-2 PRARE data and dual crossovers between ERS-2 and TOPEX. It is also recommended to compute a longer period of ERS and TOPEX orbits on the basis of the model, which would then allow the computation of the post-solution geographically correlated and anti-correlated orbit errors. Comparison with the results of Chapter 7 can then provide further evidence for the impact of the DXO data upon gravity field tayloring.

Synopsis Chapter 9

This final Chapter briefly summarises the simultaneous solution approach, and some results obtained with Faust. It will add a few general considerations related to recent developments in orbit determination and altimetry processing, in particular as encountered during the project that resulted in this thesis. Some conclusions will be extrapolated towards the future, and in relation to this some practical enhancements of Faust will be proposed.

9.1 Recapitulation of results

The crucial advantage of the simultaneous solution over classical orbit computation methods is the introduction of direct correlations between different solution arcs. For altimetry satellites this is achieved by using dual satellite crossover differences in a process in which the defining parameters of both involved satellite orbits are estimated at the same time. Because the typical time span of a single solution arc for low satellites like ERS-1 is short in comparison to the longest accepted time interval between the crossings, a simultaneous solution is only sensible in a multi-arc process, in which several consecutive orbit arcs of each satellite are computed.

Although this multi-arc requirement - among other reasons - imposed the development of the new orbit computation program Faust, it also brought the advantage of a substantially increased single satellite crossover density, namely by including crossovers between successive arcs for the same satellite. Several other advantages of the multi-arc solution method - even for single satellite solutions - have been demonstrated, like the reduction of typical orbit errors associated with arc boundaries. Alternatively, the increased crossover density allows for a reduction in the limit to crossover interval, leading to reductions in data noise originating from the temporal variability of the ocean surface.

Analysis of orbits spanning a substantial period of time enabled the computation of the so called geographically correlated and anti-correlated orbit errors, both for ERS-1 and TOPEX/Poseidon. A direct comparison between results from simultaneous solutions and orbits computed by the traditional application of dual crossover data - in which the TOPEX orbits are kept fixed - showed that the orbit errors for ERS-1 are indeed reduced by the simultaneous process, while the quality of the TOPEX orbits was not notably affected. It was also shown how the simultaneous solution process enables the inclusion of dual crossover data for the purpose of gravity field tayloring, which introduces the signal that describes the geographically correlated orbit errors directly into the solution. Results obtained with the computed gravity field models already show significant reductions in post-solution RMS of tracking data residuals for the ERS satellites. Work in this area is still continuing.

9.2 General considerations

The direct results obtained within the course of this project - like quantitative improvements in precise orbit determination - have been discussed in concluding Sections of earlier Chapters. These results confirmed various advantages of simultaneous orbit computation techniques for altimetry satellites. However, some other considerations with wider implications to the field of altimetry research have not yet been discussed. This Section therefore collects a few general aspects of orbit computation and altimetry processing, building on additional experience gained during the course of the project.

9.2.1 Compatibility between datasets

In this thesis the simultaneous solution method has only be applied to the ERS-1 and TOPEX/Poseidon altimetry missions. On several occasions it has been mentioned that these two satellites are not optimally compatible in terms of orbital characteristics, tracking data coverage and instrument precision, leading to a situation in which the ERS-1 orbits will never really match the precision of the TOPEX orbits. The orbital quality of various future altimetry missions will be more similar, and with ERS-1 and TOPEX the developed simultaneous techniques will not yet have reached their full potential. An increased compatibility between satellite orbits has positive and negative impacts, some of which will be briefly summarised here.

A first positive impact of increased compatibility between satellites will be the reduced risk of degrading the more precise of the two orbits involved in a dual crossover. In the present study this effect was not really noticed, but this may be partially due to the poor quality[1] of the involved TOPEX orbits. A second advantage will be that it becomes easier to set relative weights between datasets from different satellites without a risk of unwillingly steering the solution process. Finally, there is the advantage of producing more generic models rather than taylored models for a specific satellite (i.e. typically the 'weaker' satellite). Although this latter aspect will be irrelevant to the task of orbit determination itself, it will generally be seen as the most useful aspect of radar altimetry processing.

Some drawbacks can also be foreseen from the reduction of differences between orbits. The main reason why the geographically correlated orbit error is conveniently observable from the dual crossovers between ERS and TOPEX, is the substantial difference between heights, inclinations and revolution periods of the two satellite orbits. Crossovers between ERS-1 and ERS-2 for instance are really closer to single satellite crossovers than to dual crossovers : the characteristics of crossover data depend on the orbit, more than on the spacecraft. In contrast, dual crossovers between Envisat and the ERS satellites will offer interesting possibilities for retrospective improvements of ERS products, despite of the virtually identical orbits : the dual frequency altimeter of Envisat introduces attractive *in*compatibilities between the datasets that can be exploited.

The conclusion is that there must be a balance in the quality differences of datasets involved in a simultaneous solution. Very similar datasets do not need simultaneous solution techniques, because little additional information will be gained. However, if the involved datasets are very different, the advantage will be rather one-sided and the risk of corrupting the 'better' dataset may make simultaneous processe unattractive.

9.2.2 Absolute orbit precision

State of the art levels of radial orbit precision for altimetry satellites are such that further improvements can only become decreasingly credible. The involved satellites typically have dimensions in order of magnitude of 10 meter. A variety of dynamic effects take place on board a satellite : fuel consumption, fuel sloshing, asymmetric thermal expansions, flexing of antenna beams and solar panels, attitude oscillations, and continuous movements of the solar panels. Together, these effects impose standard deviations upon the momentary position of the centre of mass and the satellites' moments of inertia that will be at least in the order of

[1] ... 'poor' meaning a radial precision of 4 cm - much has changed since the days of Seasat

several centimetres. With the possible exception of fuel consumption - which is hardly the most relevant one - none of these effects is normally modelled during the orbit determination process. This makes claims of radial orbit precision levels below 5 centimetre seem rather meaningless - nobody knows that accurately where the momentary centre of mass is located within the spacecraft. In Chapter 1 it was stated that precise orbit determination for altimetry satellites is part of the construction of an integral altimeter instrument, of which the orbit is an inseparable geometric reference element. The quantity of interest is not the orbit of the satellites' centre of mass, but that of the altimeter antenna phase centre. Without modelling the dynamic variations in the offset between the altimeter reference point and the calculated orbit (which relates to the centre of mass - wherever it may be), the position computation for the altimetry reference point of ERS-1 or TOPEX/Poseidon can not realistically be improved over the currently achieved level. The state-of-the-art precision levels can therefore almost be considered as absolute limits.

Such considerations must give rise to critical questions about the value of further gravity field tayloring efforts for ERS-1 or other altimetry satellites. The line of continuous improvements in radial orbit precision, which started around 1992 with the arrival of the first ERS-1 data, is already flattening (see Figure 1.1 : the biggest step was made during the ERS-1 mission, notably by the introduction of the JGM-2 gravity model in 1993). An inevitable conclusion must be that if the limit in orbit improvements has been reached, attention must shift towards the upgrading of generic models rather than to the production of further taylored models, which can not bring relevant improvements. Note that this conclusion is in agreement with the trend of increased compatibility between altimetry datasets discussed above : if the satellites in a simultaneous solution do not provide a useful 'difference' signal, the solution method loses its purpose for model tayloring purposes.

9.2.3 Computer resources

The SUN Sparc Ultra workstations used for all work described in this thesis have a capacity that can comfortably compete with previous generations of mainframes, at desktop sizes. Still, the computational load involved in an estimation process for the full gravity field normal matrix is almost prohibitive, requiring many days of CPU time and a multiple of that in real time. Some computational aspects of altimetry processing will be discussed here.

Since the arrival of the first computers, the trend in computer capacity has been that about every 2 year the computational capacity has doubled (Bjorstadt *et al.*, 1993).

Although this may seem an encouraging speed of development, inspection of Figure 1.1 shows that the invasion of new altimetry data that will take place in the very near future - not to mention the associated possibilities for computing crossover combinations - will not offer any form of relief, and may actually overstretch the available computer capacity if we just extrapolate the mentioned logarithmic trend. In the 6-year period between the scheduled launches of the GFO mission and of JASON-2, each two years will show more than a doubling of data, especially if combination solutions with earlier missions are considered.

This means that selections may have to be made as to which data can be processed and which data will be ignored, or that radically new processing methods must be developed - soon. In this respect, the fact that orbit precision becomes more and more 'absolute' is of great value : it will allow separation of applications of altimetry data from the orbit computation itself. This trend has in fact already started with the distribution of precise orbits via CD-ROM or via the internet. More to the point, it seems to imply that classical orbit determination and parameter estimation techniques - including the simultaneous solution technique in Faust - may loose their meaning for geodynamic applications (the application to orbit computation remains interesting, as shown in Chapters 6 and 7). For orbit determination itself, the process will of course still be useful.

Already several years ago, parallel processing techniques have been proposed for gravity field calibrations and other geophysical applications (e.g. Gallivan *et al.*, 1990.). Such techniques can offer the required processing speed, but do not cope with the associated need for large scale data storage. During the project that is covered by this thesis, data storage proved to be at least as big a problem as CPU requirements, and, once again, current developments in mass storage - though impressive - do not seem to keep pace with expected future needs in the area of altimetry data processing.

9.3 Future developments

The three considerations above represent some general engineering experience in the field of precise orbit determination and altimetry data processing, not in the last place as learned from practice during the current study. Although they imply some criticism towards the current trends of attempting to reduce tracking data residuals to zero - which will *not* improve the quality of the computed orbits in any way - they also offer some useful guidelines as to what may be a useful direction to take in future work.

In the first place, the applications of simultaneous solution techniques based on crossovers seems to remain mainly of interest to the field of precise orbit computation, rather than to geophysical applications of altimetry data. The 'absolute' precision of current orbits disconnects the two applications, because no significant correlations remain between orbit error and geophysical model imperfections. In other words : taylored gravity fields and other state of the art dynamic models have more or less reached all precision that is required for orbit computation purposes.

In the second place, the separation of orbit computation from other altimetry applications may be necessary in order to cope with the enormous input of new, high quality altimetry data in the near future. The modest residual orbit error signal that will contaminate the altimetry datasets can very likely be compensated by combination of altimetry data from different satellite missions (but in *non*-dynamic solutions. Note that this is not what is implied in the thesis title).

Furthermore, the worrying gap between computational needs and improvements in computer hardware may require a reconsideration of planned altimetry missions beyond the next generation of satellites, i.e. from around the year 2010. Continuation of altimetry data is certainly required, not only for the direct altimetry products obtained during the mission, but particularly for analysis of long-term oceanographic changes and climate studies. However, if data processing power remain the limiting factor, there is no gain in having six parallel altimetry missions rather than two. Note that in the year 2000 we may already have five.

9.4 Suggested enhancements

At the time of this writing, the PRARE data for ERS-2 has become very useful, and PRARE work with Faust has just reached a mature status. Combination solutions between the ERS satellites or with TOPEX/Poseidon will be able to consolidate the gravity field tayloring process for the ERS missions. Another study that is already in process is the improvement of atmospheric density models with Faust, because the density model is now arguably the dominant source of orbit error at the heights of the ERS satellites. These ongoing activities will soon provide Faust with its own, internally calibrated dynamic models. In this final Section, some other recommendations will therefore be listed with regard to the more short-term future of Faust.

(1) The crossovers that have been used throughout the current project were always obtained from pre-processing software, using a priori orbits that are effectively in disagreement with the subsequently computed orbit solutions. It is a small step further to re-compute the crossover locations during each process iteration of the least squares process, leading to a situation where not only the calculated observation is improved, but also the a priori crossover observation. The effect may be small, but in particular for DXO observations between ERS-1 and TOPEX a significant noise signal may be removed. This, due to existing incompatibilities between ERS-1 along track shifts - caused by drag modelling errors - and TOPEX along track shifts.

(2) The separation between tides and gravity field seemed to cause problems in the first, short-period gravity solution described in Chapter 8. In the MSGM-1 solution no tidal parameters were included, as the longer period was hoped to reduce interference between tidal and gravitational signals. Still, the explicit separation of the tides from the gravity field and sea surface topography would be poreferrable. In addition, it can improve the generic value of the computed gravity field model.

(3) A multi-satellite solution for the two ERS satellites with TOPEX/Poseidon - including all six crossover combinations - will provide a very good global constraint to the ERS orbits. Given a substantial period of data (i.e. the full tandem mission) this may allow for the estimation of a station coordinates solution for SLR and PRARE or for the analysis of polar motion parameters, even from the low - and noisy - ERS orbits. Both efforts would help to resolve geometrical inconsistencies between the DORIS station network, the SLR network and the Earth Rotation Parameters.

(4) The multi-run capability of Faust can easily be used to combine normal matrices obtained from periods that are separated considerably in time. Gravity field tayloring for ERS may therefore gain from the generation of normal matrices (in simultaneous solutions with TOPEX and / or ERS-2) for periods from all different ERS-1 mission phases that offer different groundtrack repeat periods. Because the different repeat patterns provide different geographical information, the effect may be equal to using a much longer period of data from the geodetic phases for reaching similar spatial coverage. Related to this, a comparative analysis of the geographically correlated orbit errors for all different ERS-1 repeat patterns may be of value.

The above suggestions, or any other future work, will regularly require enhancements of Faust. A variety of practical details about the Faust software is therefore collected in several Appendices that follow. In addition, future users and programmers of Faust are invited to address practical problems - if sufficiently grave - to the author. Eternal youth may occasionally require a helping hand.

List of References

Barlier, F.C.; Berger, C.; Falin, J.L.; Kockarts, G.; Thuillier, G.; 1979
A thermospheric model based on satellite drag data
Journal of Atmospheric and Terrestrial Physics, Vol. 41, pp 527-541

Bent, R.B.; Llewellyn, S.K.; Schmid, P.E.; 1973
Ionospheric Refraction Corrections in Satellite Tracking
Space Research Vol. XII, pp 1186-1194, Berlin, Akademie-Verlag

Bjorstad, P.; Manne, F.; Sorevik, T.; Vajtersic, M.; 1992
Efficient matrix multiplication on SIMD computers
SIAM Journal on Matrix Analysis and Applications,Vol. 13, No. 1, p 386

Black, H.D.; Eisner, A.; 1984
Correcting Satellite Doppler Data for Tropospheric Effects
Journal of Geodetic Research, Vol. 89, No. D2, pp 2616-2626

Boomkamp, H.J.; Moore, P.; 1997
A gravity field solution based on unified ERS-1 and TOPEX/Poseidon altimetry data
Proc. Third ERS Symposium, Florence, March 1997

Boucher, C; Altamimi, Z.; Duhem, L; 1992
ITRF 91 and its associated velocity field
Earth Rotation Service, Technical Note 12, Paris, October 1992

Buckingham, R.A.; 1957
Numerical Methods
London, Pitman

Carnochan, S; Moore, P.; Ehlerss, S; Lam, C.; Woodworth, P.; 1993
Improvement of the radial positioning of ERS-1 through dual crossover analysis with TOPEX/Poseidon
Proceedings Second ERS-1 Symposium, Hamburg, October 1993, Vol. 2, pp 753-758

Cartwright, D.E; Taylor, R.J.; 1971
New computations of the tide-generating potential
Geophysical Journal Royal Astron. Society, nr. 23, pp 45 - 74

Cheney, R.; Miller, L.; Agreen, N.; Doyle, N.; Lillibridge, J.; 1994
TOPEX/Poseidon : the 2 cm solution
Journal of Geophysical Research, Vol. 99, No. C12, pp 24,555-24,563

Doodson, A.T.; 1921
The harmonic development of tide generating potential
Proceedings of the Royal Society of London, No. 100, pp 305-329

Dow, J.M; Romay-Merino, M.M.; Piriz, R.; Boomkamp, H.J.; Zandbergen, R.; 1993
High precision orbits for ERS-1 : 3-day and 35-day repeat cycles
Proceedings Second ERS-1 Symposium, Hamburg, October 1993, Vol. 2, pp 1349-1354

Dow, J.M; 1988
Ocean Tides and Tectonic Plate Motions from Lageos
München, Verlag der Bayerischen Akademie der Wissenschaften ISBN 3 7696 9392

Eanes, R.J.; Schutz, B.E.; Tapley, B.D.; 1983
Earth and ocean tide effects on Lageos and Starlette
Proceedings 9th International Symposium on Earth Tides, New York, Aug. 1981; pp 239-249

Ehlers, S.; 1993
Various Techniques and Procedures for Refining ERS-1 Orbits
Ph.D. Thesis, The University of Aston in Birmingham

Engelis, T.; 1983
Analysis of sea surface topography using altimetry data
OSU Report No. 343, Dep. of Geodetic Science and Surveying, Ohio State University

Engels, H. 1980
Numerical Quadrature and Cubature
London, Academic Press ISBN 0-12-238850-X

Eykhoff, P. 1974
System Identification: Parameter and State Estimation
London; John Wiley

Eymard, L.; Le Cornec, A.; Tabary, L.; 1994
The ERS-1 microwave radiometer
International Journal for Remote Sensing, Vol. 15 no. 4, pp 845-857

Gallivan, K.A.; Plemmons, R.J.; Sameh, A.H.; 1990
Parallel algorithms for dense linear algebra computations
SIAM Review, Vol. 32, No. 1, pp54-135

Haley, D.; 1973
Solar radiation pressure calculations in the GEODYN program
EG&G Report 008-73, NASA GSFC

Hedin, A.E.; 1983
A revised thermospheric model based on mass spectrometer and incoherent scatter data : MSIS-83
Journal of Geophysical Research, Vol. 88, No. A12, pp 10,170-10,188

Hedin, A.E.; 1987
The MSIS-86 Thermospheric Model
Journal of Geophysical Research, Vol. 92, p 4649

Herrero, F.A.; 1985
The lateral surface drag coefficient of cylindrical spacecraft in a rarefied finite temperature atmosphere
American Institute of Aeronautics and Astronautics, Vol. 23, No. 6, pp 862-868

Herring, T.A.; 1992
Modelling atmospheric delays in the analysis of space geodetic data
Proceedings of Refraction of Transatmospheric Signals in Geodesy, Netherlands Geodetic Commission Series, Vol. 36, The Hague, Netherlands, pp 157 - 164

Imel, D.A.; 1994
Evaluation of the TOPEX/Poseidon dual-frequency ionospheric correction
Journal of Geophysical Research, Vol. 99, No. C12, pp 24,895-24,906

Jackson, J.; 1924
Monographic Notes of the Royal Astronomical Society, Nr. 84, p 602
Royal Astronomical Society, London

James, M.L.; Smith, G.M.; Wolford, J.C.; 1977
Applied Numerical Methods for Digital Computation
Harper & Brown Publishers, New York

Jolly, G.W.; 1995
Empirical orbit refinement and ocean signal recovery techniques
Ph.D. thesis, Aston University,

Kaula, W.M.; 1966
Theory of Satellite Geodesy
Waltham MA, Blaisdell

Lambert, J.D. 1973
Computational Methods in Ordinary Differential equations
New York, Wiley

Lemoine, F.G.; Smith, D.E.; Kunz, L.; Smith, R.; Pavlis, E.C.; Pavlis, N.K.; Klosko, S.M.; Chinn, D.S.; Torrence, M.H.; Williamson, R.G.; Cox, C.M.; Rachlin, K.E.; Wang, Y.M.; Kenyon, S.C.; Salman, R.; Trimmer, R.; Rapp, R.H.; Nerem, R.S.; 1996
The Development of the NASA GSFC and NIMA Joint Geopotential Model
Proceeding of the International Symposium on Gravity, Geoid and Marine Geodesy GRAGEOMAR 1996, Tokyo, Japan, October 1996

Levy, H.; Lessman, F. 1961
Finite Difference Equations
New York, Macmillan

Ljung, L.; Söderström, T. 1983
Theory and Practice of Recursive Identification
Cambridge, Mass; MIT Press

Marini, J.W.; Murray, C.W.; 1973
Correction of laser range tracking data for atmospheric refraction at elevations above 10 degrees
Goddard Space Flight Centre, Technical Memorandum X-591-73-351

Marshall, J.A.; Zelensky, N.P.; Klosko, S.M.; Chinn, D.S.; Luthcke, S.B.; Rachlin, K.E.; Williamson, R.G.
The temporal and spatial characteristics of TOPEX/POSEIDON radial orbit error
Journal of Geophysical Research, Vol. 100 C12, pp 25,331-25,352, December 1995

Marshall, J.A.; 1995
Post-launch surface parameters for the TOPEX/Poseidon Box-Wing model
Personal communication

Matthews, P.M.; Buffett, B.A.; Shapiro, I.I.; 1995
Love numbers for a rotating spheroidal Earth : new definitions and numerical values

Geophysical Research Letter, No. 22 pp. 579-582

McCarthy, D.D.; 1996
IERS Standards
IERS Technical Note 21, Observatoire de Paris

Merson, R.H.; Odell, A.W.; 1975
Skynet Orbit Determination Program SPOD-2
R.A.E. Technical Report 75093-1975, London

Melbourne, W.; Anderle, R.; Feissel, M.; King, R.; McCarthy, D.; Smith, B.; Tapley, B.; Vicente, R.; 1983; update 1, 1985
MERIT standards
Circular 167, US Naval Observatory, Washington D.C.

Moore, P.; Boomkamp, H.J.; Carnochan, S.; Walmsley, R.; 1998
Multi-satellite orbital dynamics
1998 COSPAR conference, Japan

Moore, P.; Ehlers, S; 1993
Orbital Refinement of ERS-1 using Dual Crossover Arc technques with TOPEX / Poseidon
Manuscripta Geodaetica, Vol. 18, pp 249-262

Moore, P.; Rothwell, D.A.; 1990
A study of gravitational and non-gravitational modelling errors in crossover differences
Manuscripta Geodaetica 1990 Vol. 15, pp 187 - 206

Müller, A.C.; 1975
A fast recursive algorithm for calculating the forces due to the geopotential
Note No. 75-FM-42 (JSC-09731), Johnson Space Center, Houston

Musen, P; 1975
The exterior potential acting on a satellite
Journal of Astronautical Science, Nr. 13, Part 2 pp 161-178

Nerem, R.S.; Lerch, F.J.; Marshall, J.A.; Pavlis, E.C.; Putney, B.H.; Tapley, B.D.; Eanes, R.J.; Ries, J.C.; Schutz, B.E.; Shum, B.E.; Watkins, C.K; Klosko, S.M.; Chan, J.C..; Luthcke, S.B.; Patel, G.B.; Pavlis, N.K.; Williamson, R.G.; Rapp, R.H.; Biancale, R.; Nouel, F; 1994
Gravity Model Development for TOPEX/Poseidon : Joint Gravity Models 1 and 2
Journal of Geophysical Research, Vol. 99, pp 24,421 - 24,447

Nouël, F.J.; Bardina, C.; Jayles, C.; Labrune, Y.; Troung, B.; 1988
DORIS : A precise satellite positioning Doppler system
Advances in Astronautical Science Vol. 65, pp 311-320

Papoulis, A. 1965
Probability, Random Variables, and Stochastic Processes
New York, McCraw-Hill

Rapp, R.H.; 1983
The determination of geoid undulations and gravity anomalies from SEASAT altimeter data
Journal of Geophysical Research, Vol. 88, No. C3 pp 1552 - 1562

Rapp, R.H.; Wang, Y.M.; Pavlis, N.K.; 1991
The Ohio State 1991 geopotential and sea surface topography harmonic coefficient models
Department of Geodetical Science and Surveillance Report 410, The Ohio State University, Columbus 1991

Ray, R.D.; Cartwright, D.E.; 1994
Satellite altimeter observations of the Mf and Mm ocean tides, with simultaneous orbit corrections,
Gravimetry and Space Techniques Applied to Geodynamics and Ocean Dynamics, Geophysical Monograph 82, IUGG Vol. 17, pp 69-78

Rees, D.; Barnett, J.J.; Labitzke, K.; 1990
CIRA 1986 Part I : Thermospheric Models
Advances in Space Research (COSPAR) Vol. 10, Nr 12

Renard, P.; 1990
Final Report on Satellite Skin-Force Modelling
Matra technical report number S374/NT/100/89, ESA contract number 7850/88/HGE-1, Toulouse

Ries, J.; Bordi, J.; Shum, C.; Tapley, B.; 1996
Assessment of precision orbit accuracy for ERS-1 and ERS-2
AGU Spring Meeting, Baltimore, Maryland, May 1996

Romay-Merino, M.M.; Dow, J.M.; Piriz, R.; Boomkamp, H.J.; 1993
Global ocean tide mapping and dynamic sea surface topography determination using ERS-1 altimetry
Proceedings Second ERS-1 Symposium, Hamburg, October 1993, Vol. 2, pp 1355-1360

Salvadori, M.G.; Baron, M.L.; 1961
Numerical Methods in Engineering
Prentice Hall

Scharroo, R.; Wakker, K.F.; Mets, G.J.; 1993
The orbit determination accuracy of the ERS-1 mission
Proceedings Second ERS-1 Symposium, Hamburg, October 1993, Vol. 2, pp 735-740

Scharroo, R.; Wakker, K.F.; Ambrosius, B.A.C.; Noomen, R.; Van Gaalen, W.J.; Mets, G.J.; 1993
ERS-1 precise orbit determination
AAS/GSFC international Symposium on Space Flight Dynamics, Greenbelt, Maryland, April 1993, AAS paper 93-270

Scharroo, R.; Visser, P.N.A.M.; 1997
ERS Tandem Mission Orbits : is 5 cm still a challenge ?
Proceedings Third ERS Symposium, Florence, March 1997, ESA SP-414

Schidlovsky, V.P.; 1976
Introduction to the Dynamics of Rarefied Gases
Elsevier, New York

Schwartz, J.A.; 1990
Laser ranging error budget for the TOPEX/Poseidon satellite
Applied Optics, Vol. 29, No. 25

Schwiderski, E.W.; 1980
Ocean Tides, Part I : Global ocean tidal equations.
Marine Geodesy, Nr. 3, pp 161-217

Sinclair, A.T.; Appleby, G.M.; 1986
SATAN - Programs for the determination and analysis of satellite orbits from SLR data
SLR Technical note 9, Royal Greenwich Observatory, Herstmonceaux

Smith, E.K.; Weintraub, S.; 1953
The constants in the equation for atmospheric refraction index at radio frequencies
Proceedings of the I.R.E., Vol. 41, pp 1635-1637

Stroud, A.H. 1974
Numerical quadrature and solution of ordinary differential equations
New York, Springer ISBN 0-387-90100-0

Stum, J.; 1994
A comparison between TOPEX microwave radiometer, ERS-1 microwave radiometer, and European Centre for Medium-Range Weather Forecasting derived wet tropospheric corrections
Journal of Geophysical Research, Vol. 99, No. C12, pp 24,927-24,939

Taff, L.G.; 1947
Celestial Mechanics - A Computational Guide for the Practitioner
New York, John Wiley ISBN 0-471-89316-1

Tapley, B.D.; Ries, J.C.; Davis, G.W.; Eanes, R.J.; Schutz, B.E.; Shum, C.K.; Watkins, M.M.; Marshall, J.A.; Nerem, R.S.; Putney, B.H.; Klosko, S.M.; Luthcke S.B.; Pavlis, D.; Williamson, R.G.; Zelensky, N.P. 1994
Precision orbit determination for TOPEX/POSEIDON
Journal of Geophysical Research, Vol. 99, No. C12, pp 24,383-24,404

Tapley, B.D.; Watkins, M.M.; Ries, J.C.; Davis, G.W.; Eanes, R.J.; Poole, S.R.; Rim, H.J.; Schutz, B.E.; Shum, C.K.; Nerem, R.S.; Lerch, F.J.; Marshall, J.A.; Klosko, S.M.; Pavlis, N.K.; Williamson, R.G.; 1996
The Joint Gravity Model 3
Journal of Geophysical Research, Vol. 101, pp. 28,029 - 28,049

Todd, J 1962
Survey of numerical analysis
New York, McGraw-Hill

Le Traon, P.Y.; Gaspar, P.; Bouyssel, F.; Makhmara, H.; 1993
Reducing ERS-1 orbit error using TOPEX / Poseidon data
Proceedings Second ERS-1 Symposium, Hamburg, October 1993, Vol. 2, pp 759-763

Wahr, J.M.; 1981
The forced nutations of an elliptical, rotating, elastic and oceanless Earth'
Geophysical Journal, Royal Astronomical Society, no. 64, pp 705 - 727

Time and space — Appendix A

The equations of motion (2.51) require the choice of one or more inertial axis frames in which the position of the satellites are defined, as well as a system of time coordinates with respect to which the velocity and acceleration can be described as first and second derivatives of the position. This Appendix will summarise the main spatial and temporal reference systems in Faust, as well as the way in which these systems are related to each other.

A.1 Geodynamics

For the analysis of satellite orbits around the Earth the most convenient choice for the origin of a coordinate frame is the centre of mass of the planet, and to comply with general vector algebra we obviously adopt rectangular cartesian axis frames. These assumptions already limit the freedom of choice to a mere selection of two perpendicular directions, together defining the orientation of a geocentric axis frame in inertial space (by which we mean the distant stars that do not appear to move on any relevant time scale). In practice various directions are used, all based on the rotation of the Earth and its orbit around the Sun. The differences between these systems are related to variations in the Earth's rotational and orbital motion, so a practical way to describe reference systems is to discuss the dynamical behavior of the Earth. In this respect, it is useful to distinguish between translational movements, rotational movements, and effects caused by the Earth's non-homogeneous mass distribution or departures from rigidity.

A.1.1 Translations of the origin

The translational acceleration of the Earth within the solar system is determined by the gravity of the Sun, the Moon and the planets. The largest component is obviously the solar gravitation which continuously bends the velocity vector of the Earth into an orbit around the Sun, within a plane called the ecliptic plane. However, the other masses in the solar system also cause changes of both the direction and the size of the Earth's velocity vector, causing variations of mainly the inclination and nodal angle of the Earth's orbit, but also of its period, perigee angle and eccentricity. The influences of the masses in the solar system - other than

Sun and Moon - are collectively called *planetary precession*, referring to precession of the Earth's orbit rather than of its rotation axis. Effects of the gravitational forces of masses outside the solar system are irrelevant to the orbital motion of satellites around the Earth.

A.1.2 Earth rotation

The rotational acceleration of the Earth is dominated by interactions of the gravitational attraction of the Sun and the Moon with the non-homogeneous mass distribution of the Earth. The resulting nett torques influence both the direction and the size of the Earth's angular momentum vector, and therefore the orientation of the momentary rotation axis of the planet as well as its rotational rate. The combined effects of these torques upon the rotation vector of the Earth are called *luni-solar precession*, this time referring to precession of the rotation axis. A non-zero constant component of these torques causes the rotation axis of the planet to describe a conical shape through space with a period of about 26000 years. Superimposed upon this regular motion are many short-term accelerations of the rotation axis - called *nutation* (being the non-uniform part of precession) -, related to the periodic motion of the Sun and the Moon relative to the Earth. The influence of gravity of other planets upon the rotational movement of the Earth is negligible.

A.1.3 The structure of the planet

The Earth is not a perfectly rigid body with a homogeneous mass-distribution, which has consequences both for the shape of the planet and for the rate and orientation of its rotation. The gravitational forces of Sun and Moon cause complicated time-dependent deformations of the Earth's crust, oceans and atmospheres, collectively known as tides (see also Section 4.2). The Earth's nutation and tidal response depend on the internal structure and mass-distribution of the Earth. Tide-induced variations in mass distribution, as well as some seasonal variations of atmospheric and oceanic mass distribution affect the moments of inertia of the Earth and thereby add to the complexity of nutational movements. The thin outer crust of the Earth - containing less than 0.45 % of the mass of the planet - rests on liquid and viscous lower layers, and the rotational behavior of the crust does not precisely follow the principal motions of the massive core described in A.1.2. Furthermore, the main axis of figure of the planet never coincides perfectly with the momentary angular momentum vector, causing an Eulerian motion of the axis of figure around the actual rotation axis. Changes in the Earth's mass distribution and moments of inertia also imply that the momentary location of the axis of figure varies. Finally, the Earth's crust consists of tectonic plates that move relative to each other and relative to the core of the planet, which makes it difficult to define absloute spatial reference points on the Earth's surface. The movements of the point defined as the geographical North Pole, with respect to an adopted reference pole (related to the mean rotational axis of the planet), are collectively known as *polar motion*.

While the translational and rotational movements described in the previous sections can be adequately modelled on the basis of planetary dynamics, polar motion is caused by too many different interacting effects to allow a practical model of sufficient accuracy. Instead, empirical data is used in the form of time series derived from astronomical measurement techniques like VLBI.

A.2 Spatial reference frames

The above effects give rise to three pairs of directions for the definition of geocentric axis frames. The three corresponding frames are more or less standardized in space geodesy, as recommended by the International Astronomical Union.

At first, a terrestrial reference frame is used that is attached to (the outside of) the Earth, in such a way that the Z-axis *defines* the geographical North Pole, while the XY-plane *defines* the geographical equator, with the XZ-plane passing through Greenwich. This frame is referred to as the Earth-Centered-Fixed frame or ECF frame. Tectonic motions imply that it is difficult to define a terrestrial reference frame in an unambiguous way. In practice, its existence is implied by a set of station coordinates that have been solved relative to each other in a coherent way, from astronomical techniques or from satellite tracking measurements. Faust currently uses either the station coordinates published by the Intenational Earth Rotation Service ITRF94 (Boucher *et al.*, 1994) or the sets of station coordinates by Eanes and Watkins (1993).

The ECF frame experiences all regular and irregular movements of the Earth, i.e. those of the crust with respect to the core of the planet, superimposed upon the daily rotation of the planet and the precession and nutation of the rotation axis through inertial space. A satellite around the Earth will not follow these movements, which means that we would have to introduce a complicated fictitious acceleration of the satellite if its equations of motion were to be described in the ECF frame. Better suited for that purpose is a geocentric frame that does not follow the rotations of the ECF frame, but derives its orientation from two fixed directions in inertial space. For these directions we use the *mean* rotation axis of the Earth as the Z-axis, and the *mean* equinox line (the intersection between the mean equatorial plane and the mean ecliptic plane) as the X-axis, positive towards the vernal equinox. The mean directions are those that do not follow the nutational movements of the planet.

Still, planetary precession and luni-solar precession cause these directions to vary in time. We therefore distinguish between a frame called the true frame of date (the TOD frame), that uses the *momentary* mean rotation axis and equinox line, and a second frame that is defined by the orientation of the TOD frame at a certain fundamental epoch. Sadly, even the latter is only a *semi*-inertial frame : although its rotational inertia is perfect, it still follows the translational accelerations of the Earth within the solar system. This effect is taken into account in the modelling of the tidal potential and the third-body gravitational attractions.

Some aspects of force models or tracking observations are defined in the ECF frame, while the mathematical equations of motion are defined in the semi-inertial J2000 frame. It is therefore necessary to be able to rotate coordinates from one axis frame into another. The way in which the rotation matrices between the three frames are obtained is closely linked to Earth rotation and the concept of *time*, which will therefore be defined first.

A.3 Time

Modern space geodesy knowns four independent time systems, called sidereal time, solar time, ephemeris time and atomic time respectively. In addition, several versions of Universal Time have been introduced over the years, based upon one or more of the four fundamental time measures. All systems count time in units of seconds, minutes, hours, days, etc. but the durations of units from different systems are not identical, nor are they even constant with respect to each other. A numerical integration process in a programme like Faust inevitably uses a strictly regular mathematical time unit, but observations in the real world are inevitably labelled in terms of any of the four *physical* time systems. In order to connect physical time to mathematical time, the various time systems will be briefly described here. A more comprehensive discussion can be found in literature, for instance in the extensive treatment by Taff (1985).

A sidereal day is defined as the period between two successive passes of the vernal equinox across a given longitude in the ECF frame. Because the equinox line experiences the combined fluctuations of the equatorial plane and the ecliptic plane, we distinguish between the *apparent* equinox line and the *mean* equinox line, the latter being the average direction of the first after elimination of the nutations of the Earth's rotation axis. The corresponding difference between apparent and mean sidereal time is called the *equation of equinox*, expressed either as a rotation angle around the TOD Z-axis, or as a time correction. Note that if Earth rotation is used as a time standard, angles can be assumed equivalent to time.

A solar day is defined as the period between two successive passes of the centre of the Sun's disk across a given longitude in the ECF frame. Obviously, the Earth's orbit around the Sun lasts one solar day less than it contains sidereal days, as the annual revolution itself eliminates one relative rotation of the Sun around the Earth. The solar day exceeds the sidereal day by the amount of time that is required to rotate the planet over an angle equal to the Earth's change in orbital anomaly during that day. To compensate the eccentricity of the Earth's orbit and some of the fluctuations in its rotational rate, the concept of *mean* solar time was introduced, corresponding to a fictitious sun that moves across the celestial sphere with a constant - average - rate, rather than with a rate that varies substantially throughout the year.

The first type of Univeral Time (nowadays referred to as UT0) was defined as the local mean solar time in the positive XZ-plane of the ECF frame (i.e. Greenwich Mean Solar Time). Polar motion influences the actual length of the mean solar day, which resulted in the introduction of UT1, being UT0 corrected for polar motion. A final refinement followed by correcting UT1 for seasonal variations in the rotational rate of the Earth, leading to a time standard called UT2. Because all these versions of Universal Time follow complicated conventions and compromises, neither of them has found wide application in practice.

The phase lag between the main lunar tidal bulge and the direction to the Moon causes a nett torque that gradually decelerates the rotational rate of the planet. To retaliate, the Earth has not only done exactly the same to the Moon - so that the Moon is now tethered in its orbit, with its heaviest part permanently turned towards the Earth - but also accelerates the Moon's orbital motion, pushing it into a higher orbit and thereby forcing the Moon to gradually decrease its gravitational pull. Although this contest will inevitably end in a draw, with the Moon in an orbit that is *then* geosynchronous, it implies that for the foreseeable future the length of a mean solar day increases and that each form of solar time or Universal Time will slow down. A new uniform time scale was therefore introduced by the astronomical community, in which the duration of a second was *defined* to be strictly constant, and chosen to be equal to the mean solar time second of the year 1900. This constant time scale is called ephemeris time, because it ensured that predictions for the positions of celestial bodies would actually match with later observations within the ephemeris time system.

An obvious disadvantage of ephemeris time is that it has been defined by the abstract concept of the duration of a particular second in the past, which can hardly be used to calibrate clocks or to perform laboratory measurements. However, an accurately reproducable unit of time can be found by counting the highly regular oscillations of certain atomic resonances. To this purpose, the atomic second was introduced as the period in which a resonating cesium atom performs 9,192,631,770 cycles. This value was determined by counting cesium oscillations over the period from 1956 to 1965, and dividing the total count by the number of ephemeris time seconds within that period. The atomic time second is therefore in excellent agreement with the abstract ephemeris time second, hence with the GMST second of the year 1900.

Based upon atomic time, a final - hybrid - form of Universal Time was introduced, called UTC (C for coordinated). In UTC, the *duration* of each second is defined by the reproduceable atomic time second, but epochs are still labelled in the more practical terms of solar time, i.e. in terms of the normal Julian calendar (a Julian century contains 36525 mean solar days of 86400 seconds each). To make sure that the difference between the strictly constant atomic scale and the decellerating solar calendar stays within reasonable limits, UTC is occasionally adjusted via the introduction of leap seconds.

In agreement with general practice in modern space geodesy, Faust uses UTC as its concept of time. The strictly regular mathematical time count within the programme therefore corresponds to the atomic time scale, while individual *epochs* are named in terms of Julian dates that may occasionally contain a leap second. Julian dates represent a continuous count of mean solar days, starting at Greenwich noon, the first of January 4713 B.C. on the normal Julian calendar. Because in modern times the Julian day count has reached very large numbers - the year 2000 starts at Julian date 2,451,545.0 - the Modified Julian Date was introduced as the Julian day number minus 2,400,000.5 (i.e. removing the first two digits and making the day start at midnight rather than at noon). The fundamental epoch for the semi-inertial frame as recommended by the IAU is J2000.0, i.e. the start of 1 January, 2000 in

Greenwich. This is also the fundamental epoch that is used by Faust.

A.4 Computation of rotation matrices

Luni-solar precession and planetary precession only depend on the ephemerides of the planets in the solar system and of the Moon around Earth, which are all known with great accuracy. For any time of interest, the rotation matrix from the J2000 frame to the true-of-date frame can therefore be computed by interpolating the required matrix coefficients in tables from astronomical almanacs. The procedure used by Faust follows the MERIT standards by Melbourne *et al.* (1983).

Computing the rotation matrix from the TOD frame to the ECF frame requires modelling of luni-solar nutation and polar motion. Again, Faust follows the recommendations of the IAU as documented in the MERIT standards. For the nutation, the series by Wahr (1981) is used. The modelling of nutation will be briefly discussed here, because it also plays a major role in the analysis of tides (see Chapter 4).

The ephemerides of the Moon and the Sun - relative to the Earth - are expressed in five fundamental arguments : the mean ecliptical longitudes of the Moon and the Sun, of their perigees, and of the mean node of the Moon. These are obtained from short time series in time T, expressed in Julian centuries since the reference epoch J2000 :

$$\begin{aligned}
L &= 485866.733 + 1717915922.633\,T + 31.310\,T^2 + 0.064\,T^3 \\
L' &= 1287099.804 + 129596581.224\,T - 0.557\,T^2 - 0.012\,T^3 \\
F &= 335778.877 + 1739527263.137\,T - 13.257\,T^2 + 0.011\,T^3 \\
D &= 1072261.307 + 1602961601.328\,T - 6.891\,T^2 + 0.019\,T^3 \\
N &= 450160.280 - 6962890.539\,T + 7.455\,T^2 + 0.008\,T^3
\end{aligned} \tag{A.1}$$

Linear combinations of sines and cosines of these angles form the arguments in the Wahr series expansions that take into account the total nutational movement of the planet between the time of interest and the reference epoch J2000. The time correction resulting from this series is added to the (atomic) time of interest. From this new time tag, the Greenwich Mean Sidereal Time is computed by means of yet another time series that provides the momentary rotation angle of the planet :

$$GMST = t_{JD} + \frac{1}{86400}\left(24110.54841 + 8640184.812866\,T + 0.093104\,T^2 - 0.0000062\,T^3\right) \tag{A.2}$$

From this, the Greenwich Apparent Sidereal Time (GAST) is obtained by adding the equation of equinox (determined by interpolation in analogy to the luni-planetary precession data). This angle GAST is the required total rotation angle around the TOD Z-axis, between the TOD X-axis and the ECF X-axis. The effect of polar motion is expressed in small rotation angles α_x and α_y around the TOD X-axis and Y-axis, found from interpolation for the time of interest in tables published by the IERS. The total rotation matrix from TOD to ECF, involving the three rotations over GAST, α_x and α_y can then be computed from basic trigonometry.

The rotation matrices from J2000 to TOD, from TOD to ECF, and their product that rotates from J2000 to ECF directly, only depend on time. They are therefore evaluated only once in each integration step, even if several arcs are processed in parallel. In addition, these matrices may have to be computed for the time of an observation or another processing event.

Software structure — Appendix B

This Appendix describes the structure of the code of Faust. To start with, the overall functional design of the programme is outlined and the main modules within the code are briefly introduced. In a second Section some design conventions and programming tools are listed that aim at keeping the code managable and accessible for future programmers. The third Section consists of an alphabetical index of all subroutines, in order to assist analysis of (parts of) the code by briefly explaining the task of each subroutine, and referring to an appropriate Section of the thesis for further information. A final Section lists groups of interacting subroutines, which may have to be studied in combination rather than separately before changing any part of the related code.

B.1 Functional structure

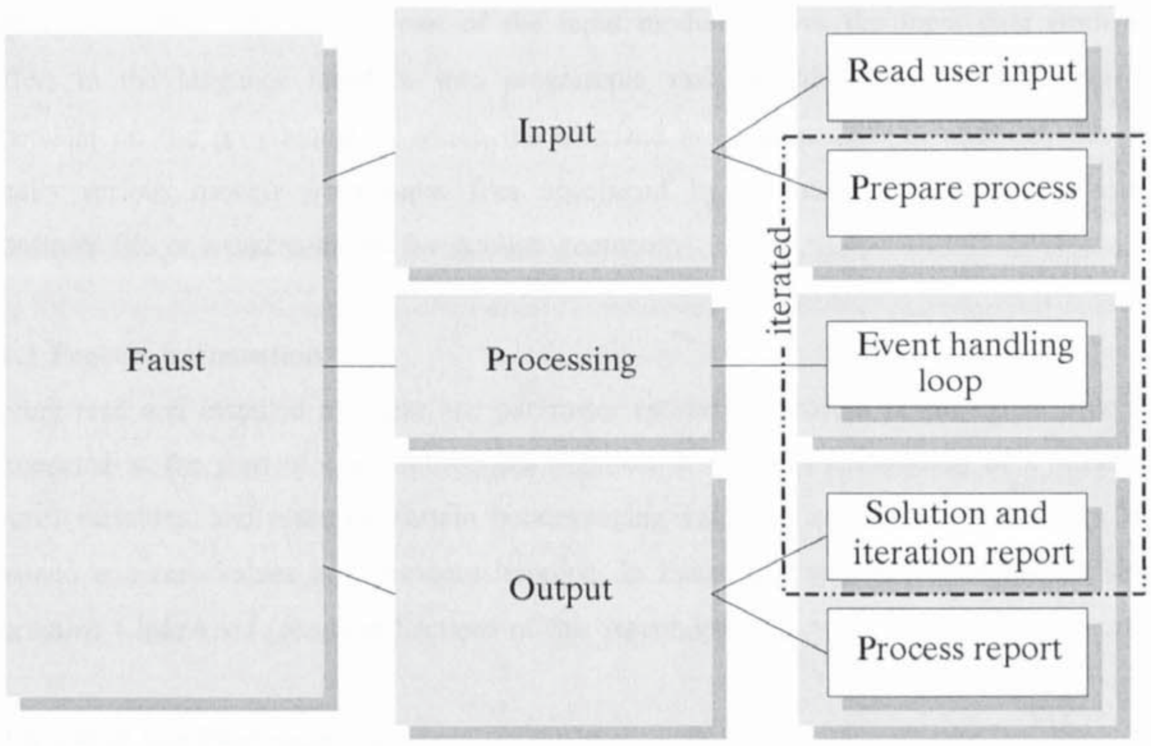


Figure B.1 High level structure of Faust

B.1.1 Main programme

Like practically every data processing programme, Faust can be modelled at its highest level as a three-part structure formed by a data input module, a data processing module and a data output module. The central processing module is formed by the event handling loop discussed in Chapter 2, and may be iterated several times. Parts of the input module and output module are also repeated at the start and end of each iteration, for instance to reset relevant memory variables or to output statistics for an individual iteration. The overall structure therefore contains five parts rather than the three described above : the input and output module each split into a repeatable and a non-repeatable part, as illustrated in Figure B.1. Each of the five blocks will be described below in some detail.

B.1.2 User input

This module is formed by the natural language interface that is explained in detail in Appendix C. The interface itself consists of a series of consecutive subroutines. The first part of the language interface deals with the process of reading and interpreting the user input file, according to definitions in an external language database. This part is independent of the programme in which it is implemented, and other programmes - like Wagner (see Section 5.5) - can use exactly the same code, although they will employ their own external language database. The second part of the input module moves the input data from data buffers in the language interface into programme variables and arrays, and is therefore dependent on the programme in which the interface is implemented. In addition, this step installs various models from input files specificed by the user, for instance a station coordinate file or a macro-model for satellite geometry.

B.1.3 Process preparation

Having read and installed all input, the parameter estimation process is initialised. This step is repeated at the start of each subsequent iteration. It involves installation of a variety of control variables, and resetting certain houdekeeping variables and matrices that may have obtained non-zero values in a previous iteration. In Faust, this step is formed by the single subroutine **ClearArc.f** (see later Sections of this Appendix).

B.1.4 Event handling sequence

This step forms the central process in Faust. It consists of three major elements. The first element is formed by the control routine **GetEvent.f**, that will determine which event is the next to be processed by comparing the epochs of scheduled events and selecting the earliest of all. The second element - not always required for each event - is formed by a module that will generate the state vectors and variational partials for the epoch of the event, either via the numerical integrator and interpolator (the subroutines **GaussJackson.f** and **GetState.f** respectively), or by reading earlier generated values from file as indicated in Figure 2.6. The third and final element is formed by a series of event handling blocks, only one of which will be executed according to the type of event that takes place. After completion of this third part, the loop is closed and the routine **GetEvent** will select the next event in the series.

B.1.5 Solution block

When the last event has been processed, the loop is abandonned. The normal matrix will have been built up gradually during the event handling loop by way of observation-events, and must now be inverted in order to compute the parameter corrections. The corrections and new parameters will be written to the output, as will some statistics about the RMS of the residuals (per station, satellite, data type) and about the iteration process itself. At this point, it will also be decided if a next iteration is to be performed, according to a variety of convergence criteria that can be manipulated by the user.

For the multi-run solutions described in Section 2.7, the normal matrix is not constructed by an event-handling loop but is constructed by reading previously computed normal matrices from file and recombining them according to the user input. The processing of these earlier computed normal matrices is done by a subroutine called **ReadMatrix.f**, which in a multi-run process simply replaces the entire event-handling loop above. Any 'normal' process (i.e. a run that uses the event-handling loop to create a normal matrix) may output its normal matrix and parameter structure to file. This is done by a subroutine called **SaveStruc.f**, if requested by the user.

B.1.6 Process completion

If no further iteration is to be performed, some final output will be produced, all open files are closed and programme execution is terminated. In addition, a CPU-tracer system will produce statistics about the time spent within each subroutine which has appeared to be useful during programme development.

B.2 Programming conventions

Faust consists of a main programme and a large number of subroutines, set up in a systematic way. Some of the applied conventions are crucial, in a sense that if new subroutines would not follow them, their entire functionality is also lost for the existing code. This Section therefore forms compulsary reading for future programmers of Faust.

B.2.1 File structures

Each subroutine is stored in its own file, with a file name that is identical to the name of the subroutine (in *lower case*) and the extension **.f** to indicate a FORTRAN file. This name convention is exploited by a programming tool called **getstruc.f**, that creates a call tree for the programme or sub-process within the code. If, for instance, we run **getstruc** on the Gauss-Jackson subroutine, it will produce a tree diagram showing all routines called by the integrator and by any of the lower level routines, like the force models.

It is important to keep just **one** version of the source files - bar periodical backups for safety reasons - so that any modification of a subroutine will automatically be implemented in the entire programme. As soon as different sets of the source code are maintained by different people, it becomes impossible to ensure consistent behaviour of the programme and chaos will take over.

In contrast, there may be many different *executable* programme files around, which will mainly vary in terms of array sizes. Array sizes in Faust tend to be defined by means of FORTRAN **parameter** statements, collected in a single block at the start of the main programme. To allow for various programme set-ups, four or five different versions of this parameter block are included, only one of which will *not* be commented out at the time of the compilation. As an example, the gravity field inversions of Chapter 8 require the A-partition of the normal matrix to have a size in excess of 100 MB, while not needing the other normal matrix partitionings. For normal orbit determination runs it would not be practical to impose such strong loads upon the network, and an executable version of Faust with much smaller array sizes should be used. In total, the memory requirements of Faust are determined by some forty different parameters, which should ideally be set to be just large enough to encompass the intended process. At present, different executables exist for single-arc orbit determination, for multi-arc orbit determination (with and without supporting matrix partitionings for DORIS or PRARE data) and for gravity field work.

B.2.2 Data structures

As a general rule we can safely conclude that common blocks in FORTRAN form a threat to programme structure, as they allow uncontrolled intermingling of global variables with local variables. By convention Faust therefore does not use common blocks, apart from a few well-justified and carefully managed exceptions. By refusing to use common blocks, all data must pass between subroutines by means of the routine headers, and data can be easily traced back to the main programme from within each layer of the code.

B.2.3 Logical structure

In order to keep the code accessible, each subroutine only performs one clearly defined task. The general rule is : it should always be possible to represent a subroutine as a black box, with a coherent set of input variables, a concise task (which should be obvious from the subroutine name), and a coherent set of output variables. If this concept is followed, a subroutine will typically have a length that does not exceed 150 lines or so, allowing future programmers to quickly familiarize themselves with its characteristics.

B.2.4 Consistent documentation of variables

Each subroutine has a header that contains the date of implementation, the programmer responsible for the routine in question, and a list with all input and output variables passed through the header. For each variable its name, type and dimensions are listed as well as a short description of its purpose.

B.2.5 CPU tracer

During the test phase of the programme, a CPU counter system was implemented that uses two external LNAG library routines to determine the CPU time used by each subroutine, allowing optimisation of the entire process. For this purpose each subroutine has a unique **identification number** that corresponds to the index of an array called CPU, and in addition each subroutine should be listed in a file called **subroutines.f** . At the start of subroutine with number N, the current machine clock counter is subtracted from the element CPU(N), and just before returning to the calling routine the current clock value will be added. The difference between the two clock counts is thereby added to the element CPU(N). It should be noted that any time spent between entering and leaving a subroutine will be included, which includes time spent within lower level routines, or time absorbed by waiting for disk action or memory paging, which sometimes distorts the actual time used by each sub-process.

B.3 Alphabetical index of subroutines

The next pages form a look-up table to identify the task of individual subroutines. It is intended to assist in the analysis of parts of the programme that may have to be modified or extended for future applications. Listed are the internal reference number of the routine, its name, a Section of the thesis where the most relevant theory is discussed, and a description of the task of the subroutine. In addition, in Section B.4 routines are grouped according to their functional task within the overall process.

ID	Name	Ref	Description
098	AddDorisHold	5.2.2	Event handler routine, for storing the state vector and partials for the epoch of the start of a DORIS doppler-count interval
123	AddPrareHold	5.2.3	Event handler routine, for storing the state vector and partials for the epoch of the start of a PRARE doppler-count interval
084	AddXOhold	5.4	Event handler routine, for storing the state vector and partials at the epoch of the first crossing in a crossover observation
001	AeroInt	4.4	Routine that interpolates the GUESS area tables for evaluation of the effective drag surface area for the ERS satellites
002	Albedo	4.5	Routine that computes albedo radiation pressure for any of 13 reference surfaces in which the visible Earth is divided
003	AlbPir	4.5	Force model routine that combines total albedo and infra-red radiation pressure from each segment of the visible Earth
101	AlongTrack	4.8	Force model routine for empirical along-track accelerations
004	ArgoLat	4.7	Subroutine that computes the argument of latitude, being the orbital anomaly of the satellite with respect to the equator.
005	ATCcorr	-	Extinct routine, once used for testing the numerical integrator
006	Cal2MJD	A.3	Routine to convert a date from year-month-day-hr-min-sec to MJD
007	Centre	5.1	Routine to compute the centre-of-mass offset for the SLR and DORIS instruments
008	Cheb	A.4	Routine to evaluate Chebyshev polynomials that are used in the reconstruction of planetary ephemerides

009	**ChebCoef**	**A.4**	Used by Cheb to obtain a set of interpolation coefficients for a particular epoch
010	**Choleski**	**2.4**	Performs inversion of the **A** partitioning of the normal matrix
011	**CIRA72**	**4.5**	Density model (Cospar International Reference Atmosphere)
012	**ClearArc**	**2.8**	Major initialisation routine for each process iteration
013	**CopyField**	**4.2**	Routine that makes a backup copy of the part of the original gravity field model that will be affected by tidal corrections
014	**CopyGuess**	**4.2**	Routine that allows a GUESS-table to be shared by different arcs, in order to save memory
015	**Correct**	**2.3**	Routine that controls the inversion of the entire partitioned normal matrix and adds the corrections to the parameters
016	**DirectSRP**	**4.6**	Force model routine for direct solar radiation pressure
017	**Drag**	**4.5**	Force model routine for atmospheric drag
018	**DragGUESS**	**4.5**	Subroutine called by **drag** to evaluate GUESS surface area
019	**DTM94**	**4.5**	Subroutine for evaluating the DTM density model
020	**ECFmatrix**	**A.4**	Subroutine that computes the rotation matrix from the TOD frame to the ECF frame (inverse matrix is equal to the transposed matrix)
021	**EarthFrames**	**A.4**	Subroutine that evaluates a variety of geometrical quantities in a centralised way at the start of the force model
022	**Elements**	**A.4**	Subroutine converting a Cartesian vector to Kepler elements
023	**EllipsHt**	**5.1**	Subroutine that computes the height above the ellipsoid from a given Cartesian position vector (either ECF or TOD)
024	**ERSatt**	**4.5**	Subroutine that evaluates the attitude control algorithm for the ERS satellites, required for centre-of-mass computations or surface forces
091	**Filename**	**-**	Housekeeping subroutine that stores and saves all user filenames and can subsequently be called from any location within the code
025	**Flux**	**4.5**	Subroutine that computes the solar flux and geomagnetic data
026	**Force**	**2.5**	Central force model subroutine, called from within the numerical integrator, that manages all individual force models
027	**Frames**	**A.4**	Subroutine that computes a variety of reference frames and attitude angles as a function of time
028	**FramesGUESS**	**A.4**	Additional subroutine for evaluating GUESS-specific reference frames and rotation angles

029	**GaussJackson**	2.2	Main subroutine for numerical integration of the force model
030	**Geodetic**	5.1	Subroutine that converts a Cartesian vector to longitude, geodetic latitude and ellipsoid height
082	**GetCPU**	-	Housekeeping routine that keeps track of CPU usage
031	**GetEvent**	2.8	Major subprocess that determines which event should be handled next, thereby controlling the entire process flow
104	**GetFileDORIS**	5.2.2	Input routine that will read previously computed pass-dependent parameters for DORIS from file
124	**GetFilePRARE**	5.2.3	Input routine that will read previously computed pass-dependent parameters for PRARE from file
095	**GetMSdata**	C	Input routine that installs multi-satellite parameters and global data that can not be handled by the general input handling
032	**GetParams**	2.6	Major input routine that will determine the overall parameter structure and create the pointer array M_p from Chapter 2
100	**GetParDORIS**	5.2.2	Subroutine that will scan the observations file to determine the required amount of pass-dependent parameters for DORIS
122	**GetParPRARE**	5.2.3	As GetParDORIS, but then for PRARE data
033	**GetPointers**	C	Language interface routine that connects input lines with recognised input statements from the language database
034	**GetPoly**	A.4	Subroutine that creates an internal table with interpolation coefficients for the planetary ephemerides
035	**GetState**	3.2	Subroutine that performs a 10-point Lagrange interpolation in the ringbuffers to get state and partials at a time of interest
099	**GetStaDORIS**	5.1	Input routine for reading a separate station coordinate file for DORIS (only used for ITRF93 and older)
119	**GetStaPRARE**	5.1	Input routine for reading a separate station coordinate file for PRARE (only used for ITRF93 and older)
036	**GetStations**	5.1	Input routine for reading station coordinates in ITRF format and correcting them for station velocities
037	**GetTideF**	4.3	Subroutine that computes scaled Legendre functions for evaluation of the ocean tides
038	**GetTimeTags**	4.9	Major input subroutine that installs MJD time tags for all parameters
085	**GetXOhold**	5.5	Subroutine that retrieves crossover data that was previously written to the buffer file by AddXOhold
103	**Get_DRSut**	5.2	Input routine for reading DORIS station coordinates in UT/CSR format

102	**Get_SLRut**	5.2	Input routine for reading SLR station coordinates in UT/CSR format
039	**Gravity**	4.1	Force model subroutine that evaluates the gravity field model. This routine will typically absorb 40% of the total CPU-time
040	**InputReport**	-	Output routine that is used after the first iteration only, to produce most of the process output
105	**IntrpCoef**	A.4	Subroutine that computes interpolation coefficients in the process for reducing the step size
041	**IterReport**	-	Main output routine that will produce statistics and results after each iteration
042	**Lag12**	A.4	Subroutine that performs a 12-point Lagrange interpolation in the input planetary ephemerides data
043	**LowerCase**	C	Language interface routine to convert strings to lower case
109	**LRAtopex**	5.2.1	NASA Subroutine that evaluates the FFT optical plane correction for the retroreflector array on TOPEX/Poseidon
108	**Manoeuvre**	3.4	Major subroutine that will reduce the step size, integrate over a manoeuvre and change the step size back to nominal
045	**MapString**	C	Language interface routine that scans the contents of a long input string
046	**MariniMurray**	5.1	Tropospheric correction according to Marini and Murray
047	**MatMult1**	-	General algebraic routine to multiply a 3x3 matrix with a vector of three elements
048	**MatMult2**	-	Same as MatMult1, but using the transposed input matrix
049	**MatMult3**	-	General algebraic subroutine to multiply two 3x3 matrices with each other to form a third matrix
050	**MatMult4**	-	Same as matMult2, but using the transposed first input matrix
051	**MJD2Cal**	A.4	Subroutine that converts an MJD date to year-month-day-hr-min-sec
052	**MSIS83**	4.4	Routine that evaluates the MSIS83 density model
053	**Normalize**	-	General algebraic subroutine that computes a norm and a unit vector from a vector with three elements
054	**Nutation**	A.4	Subroutine that evaluates the Wahr nutation series
055	**OnceRev**	4.7	Force model routine to evaluate the one cycle per revolution empirical accelerations
056	**OpenFile**	-	Housekeeping routine to open an input text file with some standardised error handling

057	**Planet**	**A.4**	Subroutine that evaluates planetary ephemerides
058	**Pole**	**A.4**	Subroutine that evaluates the polar motion at a specific time and accounts for eventual leap seconds
059	**PN2000**	**A.4**	Subroutines to create a table with interpolation coefficients for the nutation and precession matrix
060	**PrepareObs**	**5.6**	Housekeeping routine that will prepare the processing of a tracking observation of any kind
093	**PrintCov**	**2.2**	Output routine that will print a post-solution covariance matrix
092	**PrintRes**	**2.1**	Subroutine that computes and prints residuals per pass
094	**ProcessRes**	**2.2**	Major observation processing routine to add the contribution from a single observation to the normal matrix
061	**ProgVar**	**C**	Major input routine that installs all input variables into programme variables
062	**ReadAero**	**4.4**	Input routine that reads GUESS aerodynamic drag tables
064	**ReadBody**	**4.2**	Input routine that reads solid Earth tide coefficients
088	**ReadBoxWing**	**4.4**	Input routine that reads the NASA Box-Wing model for TOPEX/Poseidon
065	**ReadFlux**	**4.4**	Input routine that reads the solar flux table and computes moving averages for the geomagnetic index
066	**ReadGeoid**	**4.1**	Input routine that reads the gravity field coefficients
067	**ReadGUESS**	**4.4**	Input routine to read geometry model for the GUESS tables
117	**ReadJGMmat**	**8**	Input routine that reads the JGM3 normal matrix and installs it into the triangularised normal matrix of Section 2.3
110	**ReadLRAindex**	**5.2.1**	Input routine that reads the LRA look-up table for the NASA FFT centre-of-mass correction
116	**ReadMatrix**	**2.7**	Input routine that reads previously computed normal matrices in a multi-run process
068	**ReadOcean**	**4.2**	Input routine that reads harmonic coefficients for ocean tides and converts them into an amplitude and phase coefficient
069	**ReadPole**	**A.4**	Input routine that reads the polar motion table
070	**ReadSRP**	**4.5**	Input routine that reads the GUESS tables for solar radiation pressure
115	**ReadSST**	**8**	Input routine that reads harmonic coefficients for the sea surface topography model

112	**ReadStruc**	2.7	Input routine that reads a list of previously computed matrices and establishes the parameter structure in a multi-run process
071	**ReadWords**	C	Language interface routine that reads the language database
120	**Reorganise**	4.4	Housekeeping subroutine that rearranges the parameter structure to allow for shared drag parameters etc.
063	**Reproduce**	2.4	Output routine that will write the parameter solution in a format that can be used as new input to Faust
113	**ResAltim**	5.3	Subroutine that computes residual and partials for altimetry
097	**ResDORIS**	5.2.2	Subroutine that computes residual and partials for DORIS
072	**ResLaser**	5.2.1	Subroutine that computes residual and partials for SLR
118	**ResPrange**	5.2.3	Subroutine that computes residual and partials for PRARE rng
121	**ResPrate**	5.2.3	Subroutine that computes residual and partials for PRARE dop
083	**ResXO**	5.4	Subroutine that computes residual and partials for crossovers
111	**SaveStruc**	2.7	Subroutine that writes the parameter structure and normal matrix to file, to be used later in a multi-run process
073	**ScalProd**	-	General algebraic routine that computes the scalar product of two vectors with three elements
074	**ScanInput**	C	Language interface routine that scans the user input for recognisable input statements
114	**SSTpartials**	5.3	Subroutine that computes the geometrical partials for the sea surface topography model
075	**SRPint**	4.5	Subroutine that interpolates the effective area for solar radiation pressure in the GUESS tables
076	**StartInt**	3.3	Subroutine that initialises the Gauss-Jackson integrator for a single arc at a time
077	**StaTides**	5.1	Subroutine that evaluates the tidal uplift for station coordinates
106	**StepDown**	3.4	Subroutine that reduces the integration step size
107	**StepUp**	3.4	Subroutine that increases the integration step size
044	**ThirdBody**	4.3	Force model subroutine for third-body attraction and frequency independent luni-solar tides
096	**Thruster**	4.6	Force model subroutine that evaluates the thruster force
078	**TideJGM**	4.2	Force model subroutine that corrects the harmonic coefficients of the gravity field model for ocean and Earth tides
079	**TideSch**	4.2	Extinct subroutine that evaluates the Schwiderski tides

080	**TimeString**	-	Housekeeping routine that creates a string in calendar date format from a MJD time tag
086	**TopAtt**	4.4	Subroutine that evaluates the attitude control algorithm for TOPEX/Poseidon, for centre-of-mass offset, drag and SRP
090	**TopexLRR**	5.2.1	Subroutine that evaluates the FFT correction for the laser retroreflector array on TOPEX/Poseidon
087	**TopexYaw**	4.4	Subroutine that evaluates the yaw-steering algorithm for TOPEX/Poseidon
081	**VecProd**	-	General algebraic routine for computing the vector product of two vector with three elements
089	**VectorOut**	-	Output routine that produces an output state vector at a given time in the format used in new input to Faust

B.4 Functional index of subroutines

In order to know which routines relate to a specific sub-process, this Section will list subroutines of Faust in relation to their task. In combination with the previous Section, this should enable quick navigation through the code. Because many routines are relevant to different aspects of the programme, they may appear under more than one header. A full structural breakdown of any part of the code, or of the programme as a whole, can always be obtained by running the programme **getstruc** described above.

B.4.1 Parameter estimation

Input routines :	GetParams	GetTimeTags	GetMsData
	GetFileDORIS	GetFilePRARE	
Processing routines :	ClearArc	ProcessRes	ReadMatrix
	Correct	Choleski	
Output routines :	SaveStruc	Reproduce	

B.4.2 Orbit integration

Initialisation :	StartInt		
Main integrator :	GaussJackson	Force	GetState
Changing step size :	StepUp	StepDown	

B.4.3 The force model

Overall control :	Force		
Geometry at time t :	Frames	FramesGUESS	EarthFrames
Gravity :	Gravity	ReadGeoid	TideJGM
Tides :	TideJGM	TideSch	Planet
	ReadOcean	ReadBody	GetTideF
Third-body attractions :	ThirdBody	Planet	
Atmospheric drag :	Drag	DragGUESS	ReadGUESS
	CopyGUESS	ReadAero	AeroInt
	Flux	ReadFlux	ReadBoxWing
	CIRA72	MSIS83	DTM94
Direct SRP :	ReadSRP	ReadBoxWing	DirectSRP
	SRPint	Flux	ReadFlux
Albedo and IR :	Albedo	AlbPir	
Empirical accel. :	OnceRev	AlongTrack	

B.4.4 The calculated observation

Event handling :	PrepareObs	ProcessRes	
Geometry :	EarthFrames	Centre	StaTides
	EllipsHt	SSTpartial	TopAtt
	ERSAtt	Elements	Geodetic
Buffering of partials :	AddXOhold	GetXOhold	AddDORIShold
	AddPRAREhold	GetEvent	ClearArc
Individual data types :	ResLaser	ResAltim	ResDORIS
	ResPrange	ResPrate	ResXO
	MariniMurray	TOPEXyaw	LRAtopex
	TOPEXlrr	Centre	
Editing and weighting :	PrepareObs	ProcessRes	
Output statistics :	IterReport	PrintRes	PrintCov

B.4.5 Natural language interface

Reading input :	ReadWords	ScanInput	MapString
	LowerCase		
Installation of pointers :	GetPointers	GetParams	GetMSdata
	GetTimeTags	Reorganize	

Programme variables :	ProgVar	FileName	

B.4.6 Reference frames

Central management :	Frames	FramesGUESS	EarthFrames
	Pole	ReadPole	ECFmatrix
	Planet	GetPoly	
General routines :	Geodetic	Elements	EllipsHt

B.4.7 General algebra

Matrices 3x3	MatMult1	MatMult2	MatMult3
	MatMult4		
Vector 3	ScalProd	VecProd	Normalize

The language interface Appendix C

The natural language interface for Faust forms a powerful way of communicating with the programme, and is easy to learn from a user point of view. However, the embedded software for the interpretation of natural language and the conversion of the user input to programme variables is fairly abstract. Because a basic understanding of the working of this interface will be required if additional input variables are to be added to the programme, this Appendix will describe the language interface in detail.

C.1 Design philosophy

The importance of an adequate user interface becomes clear from inspection of various existing programmes for orbit determination or other large scientific programs. Too often the user is forced to construct input files according to a narrowly prescribed format, in which even small errors, like shifting a number within an input line or overlooking an incorrect flag setting can result in hours of troubleshooting. One of the main causes for such user-unfriendly input files can be found in the dynamic use of scientific programs. The users tend to be also the programmers of the software, and code as well as input variables are modified on a regular basis - by various people - to support new applications. Another reason can be found in the notoriously poor I/O-provisions of FORTRAN. If programs become larger or more diverse the input specified by the user will tend to become more complicated as well, and as a result will be more prone to confusion and errors. The design requirements for a systematic user interface to Faust were therefore most directly obtained by summarising some properties that it should *not* have :

(1) No rigidly prescribed input formats. Formatted FORTRAN statements date back to the days of punch cards, and although they may be convenient for implementation by the programmer, they are in general unnatural for the user.

(2) No loss of downward compatibility, or loss of systematic data order in input files, in case that additional input variables are needed for new applications.

(3) No need to specify all sorts of input parameters that are not relevant to the actual application for which the programme is run.

(4) Not difficult to learn for new users. This is in particular important for a programme that is used at a university, by students who may only have limited time to familiarize themselves with the system.

The central demand arising from the above requirements appears to be that a user interface should be designed from the point of view of the user, rather than from that of the programmer. Recalling that Faust was designed specifically as a multi-satellite / multi-arc orbit determination platform, it is also important to note that the amount of input and the character of the data will vary from case to case. On these grounds, it was decided to give Faust a natural language interface that will simply try to interprete *any* normal text file or even a combination of text files, allowing users to specify their input almost entirely according to their own preferences.

C.2 Overall description of the language interface

It is not practical to teach the entire English language to a computer programme, but fortunately one does not need a large vocabulary to adequately communicate with scientific software. The part of the language that Faust understands is stored in a language database, a file that can be modified or extended whenever required. This Section will explain how the language interface interpretes user input files on the basis of the provided language database, and how the input data is subsequently transformed into internal variables and data elements that can be understood by the executable code.

A basic layout of the language interface is shown in Figure C.1 below. The text file on the left in Figure C.1 contains the relevant input information in a format that is convenient to the user, listed in an arbitrary order chosen by the user. Internally, Faust will require this information in the form of variables of one of the four basic datatypes integers, real numbers, character strings or logical variables. The task of the language module is to interprete the user input, store the encountered data somewhere in memory, and arrange the structure of the input data into fixed patterns that are embedded in the executable code. This is done in two stages, indicated in Figure C.1 by blocks marked as 'language filter' and 'data organiser'.

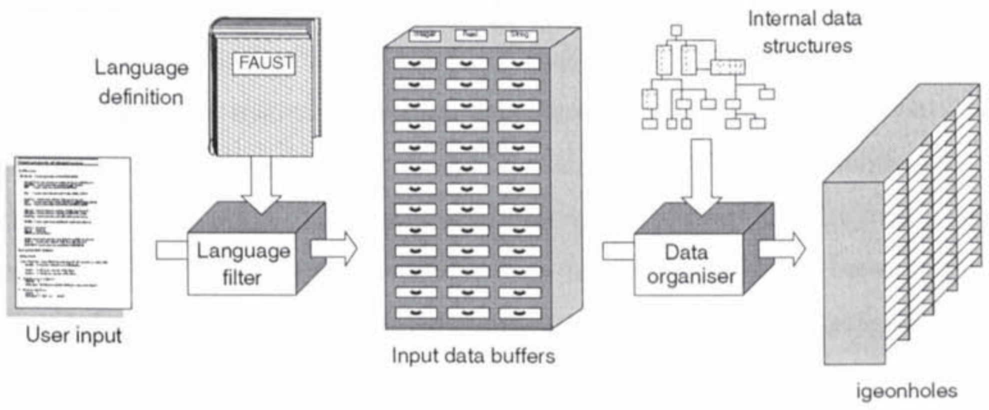


Figure C.1 Schematic view of the language interface

In the first stage the user input file is scanned line by line, and checked for recognisable language elements. The latter are defined in the language database, which is a normal text file that can easily be updated or modified in order to allow for future extensions of the programme, without changing the code of the language interface itself. Recognised input language elements, and possible data arguments that accompany this input, are stored in one (or more) of three available input data buffers. These buffers are adequately sized one-dimensional arrays of datatype integer, double precision real, and character strings. Incoming data will be collected in these input buffer arrays in order of appearance in the user input file, i.e. in the arbitrary order chosen by the user.

After completion of the first step all recognised user input data is present in the three buffers, although in arbitrary order. Each recognised input element has been marked by internal identification numbers to tell the programme where within the input buffer arrays the corresponding data elements have been stored. This language filter step is totally independent from the programme in which it is implemented : it only scans external text files (the user input) for properties and data structures that have been defined in another external text file (the language database), in a way that will be described later in this Appendix.

The second stage of the input interpretation process deals with reorganising all data that has been accumulated in the input buffers into a structure that will allow the main program to find individual variables in an unambiguous way. The output of this step is appropriately illustrated in Figure C.1 as a set of pigeonholes, each of which corresponds to a specific programme variable. The actual *meaning* of these variables is internal to the main programme, and therefore unknown to the language module. However, from the calling programme the data organiser step receives a 'map' to the programme data structure. This map tells the language module in which pigeonhole(s) the data from a particular input language element should be stored. Unlike the language definition database, this map is *hardcoded* in the program in the form of statement identification numbers that correspond to elements of the pigeonholes. This hardcoded map forms the only link between the abstract language elements defined in the external database, and the associated programme variables that are internal to the code.

The user does not need to know the abstract internal statement identification numbers, but only the natural language elements that will define certain input. On the other hand, the programme has no perception of the natural language elements, but processes the data via the identification numbers. The defined language elements are therefore completely decoupled from the internal interpretation in the code : the language database file could be replaced by a Russian or Chinese version without affecting the working of the program and even without having to recompile the executable. The actual language interpretation step is performed by the input filter in the first stage described above, when recognised natural language elements in the user input are replaced by their internal identification numbers.

C.3 Definition of the user language

In analogy to computer languages the user language of Faust is defined by specific keywords that are recognised by the interpreter, possibly accompanied by arguments in the form of data numbers or character strings. Contrary to most computer languages, the user language of Faust does not prescribe any particular syntax for its statements. Furthermore, the implemented language interface is a learning system that contains knowledge in the form of the external language database. It can be taught to understand new words simply by extending or modifying the language database. To maintain maximum flexibility, the entire concept of language is defined in Faust by the following basic rules :

(1) User input consists of lines of text, formed by series of words and numbers which are separated by spaces or commas.

(2) A line of text is intelligible if it contains a pair of keywords in combination with at least the amount of data arguments required for that pair. If a line of text is not intelligible, it is automatically taken for a coment line.

(3) Keywords and data arguments may be placed in arbitrary order within the line, although multiple data arguments on a particular input line will be used in order of appearance, from left to right.

Rules (1) and (2) state the concept that any line of text - like this one, for instance - will pass the input scanner without causing error messages, while only those lines that contain recognisable input will be interpreted as such. The language interpreter acts as a text filter that will try to match each line of the user input file with any of the input statements that have been defined in the language database. The expression **'at least'** in rule (2) is used to state that apart from the words and numbers that form the actual input statement there may be other words and numbers on that line, without causing errors (i.e. without interfering with the filter). Rule (3) reduces the restrictions even further, by not imposing a particular order for the words and numbers within the input line. The only restriction to the order of data on the input line relates to input statements that involve, for instance, two or more integer numbers as data arguments. In such cases the two left most integer numbers on the line will be interpreted as the required input data. If less than two integer numbers are found the statement is incomplete, and will therefore be ignored as a comment line.

The three types of data arguments that can be offered as input to the program are integer numbers, real numbers (internally always treated as double precision reals) and character strings. Logical data can be entered into the program through *implication* by keywords, without requiring data arguments. As an example, we could have complementary statements 'USE ALTIMETRY' and 'IGNORE ALTIMETRY', that might be defined to set a variable somewhere in the code for using altimetery data or not.

The classification of data arguments as integers, real numbers or character strings is not as strict in Faust as in most computer languages. The general rule is : what makes sense to the user will make sense to the language interface. This means that weak datatypes may always replace stronger datatypes, with integers being the weakest and character strings the strongest datatype. If, for instance, a statement requires two real numbers but only one real number and an integer are found on the input line, it will be assumed that the integer is in fact a real number. If a character string is expected - for instance a station identifier like 'YARAG' - but the user has chosen to identify a station by a number - like '7090' - then this number will be taken for a character string rather than an integer number.

A statement *definition* in the language database consists of four parts. First, it has a unique identifier number that is used for internal processing, and is therefore mainly of interest to programmers. Then, the definition contains two keywords that together form the basic recognisable statement. In order to indicate how many data arguments should *at least* accompany this statement, three numbers are specified, namely the amount of required integer numbers, real numbers and character strings. The fourth part of the statement definition consists of three processing flags that will be introduced in the next Section. The statement definition therefore follows the template

ID_number Keyword_1 Keyword_2 nI nR nC flag_1 flag_2 flag_3

(C.1)

in which nI, nR and nC are the expected amounts of integer arguments, real arguments and character strings respectively. To the user of Faust only the two keywords and the required arguments are of importance; the other elements of the statement definition are internal to the program and used to assist the correct interpretation of the statement.

The practical application of the defined user language is best illustrated with an example.

<u>EXAMPLE C.1</u> *In the input we want to specify a time interval during which all tracking data should be ignored. This could be useful in case of an unmodelled attitude control manoeuvre that temporarily causes errors in the centre-of-mass correction for calculated observations.*

In the language database we might now add a definition like

$$\underbrace{234}_{\text{Unique identifier number}}\ \ \underbrace{\text{DATA}\ \ \text{IGNORE}}_{\text{Pair of keywords}}\ \ \underbrace{0\ 2\ 0}_{\text{Required integers, reals, strings}}\ \ \underbrace{0\ 0\ 0}_{\text{Processing flags}} \tag{C.2}$$

This means that if anywhere within a single input line the words 'data' and 'ignore' are detected, as well as two numbers, the line will be recognised by the language interpreter as the statement with internal number 234. In the actual input file for Faust the user could now include - among numerous other possibilities - any of the following text lines :

(a) DATA IGNORE 4.988600D+05 4.988625D+05

(b) 49886 49886.25 Ignore data

(c) IGNORE all DATA from 49886.00 to 49886.25

(d) If there is any tracking data between MJD 49886 and MJD 49886.25, ignore it, because of the unmodelled attitude manoeuvre in that interval.

Example (a) is a literal implementation of the statement definition, namely the two keywords followed by the two required (real) numbers. This basic format closely resembles normal computer code. Although it is perfectly valid to use elementary input lines like this, such practice does not exploit the full capabilities of the language interface. Example (b) shows that the order of the keywords is irrelevant, that the input scanner is case-insensitive, and also that an integer may be used even if a real number is expected by the statement.

Example (c) shows that the additional character arguments 'all', 'from', and 'to' will have no effect at all upon the interpretation of the statement. The addition of such natural language is in fact strongly recommended in order to make input files understandable for other users, or to ensure that six months after running a certain process the input file can still be understood. This principle of adding extra information is illustrated in a rather more imaginative way in example (d), which shows that the input scanner of Faust can indeed be considered as a natural language interpreter. The first line of (d) contains the actual input, namely the two keywords 'data' and 'ignore', as well as the two expected numbers. The second line is only of interest to the user, and the language module will discard it as a comment line without causing errors of any kind.

Some final remarks should be added here with regard to the language interpretation by Faust.

#1 The language scanner routine will always produce a report file - called **input.scan** - in the active working directory from which Faust was started. This file contains a copy of the entire user input, while for each line is indicated how it has been interpreted by the program. The latter is done by means of the unique statement identifier number (if a recognisable statement was found on that line) or by short self-explanatory comments, like 'missing data', 'comment line', 'no keywords', etcetera. In case of doubt about how the scanner has interpreted the input, this report file should always be the first subject of investigation : Faust does not prescribe a particular syntax, which implies that it is also unfamiliar with the concept of syntax error.

#2 Note that within an input line, words may be separated by (any number of) spaces or commas. The comma after the second timetag in example (d) above will therefore not cause problems : it is not considered part of the character sequence **49886.25** that represents the actual data argument. If only spaces could be used as separators, the sequence **49886.25,** is a character string rather than a real number, because of the comma at the end. In that case the statement would be discarded (with a 'missing data' message in the file **input.scan**), which is in disagreement with the most natural interpretation of the statement.

#3 It is possible to explicitly mark an input line as a comment line by starting the line with an asterisk '*'. This is useful if a comment line happens to contain a pair of keywords that may be recognised by the language processor, and then might lead to unexpected behavior of the program. It is most often used to exclude certain statements from an input file for later runs, using slightly different settings of certain flags or numbers. Only lines that do not start with an asterisk are actually scanned by the input processor routine.

#4 To make life even easier for the user, the language scanner has a 'memory' feature that allows abbreviation of input statements. If the scanner only detects one existing keyword within an input line, it will attempt to construct valid input by combining this keyword with either of the two words of the most recently accepted statement. This means that if two similar statements follow each other (for instance to define a list of many different station weights) one of the two keywords may be left out to save some typing. Examples of this principle can be found in the final Section of this Appendix.

#5 It should be noted that the language interpreter module will comfortably accept any input that agrees with a statement definition, regardless of the inherent logic in the data arguments, and unaware of the fact that the given data may not be realistic. It would for instance accept the statement in Example C.1 even if the first number is greater than the second, or if an impossible MJD value of -3.1415 was given. The language scanner has no further tasks than to filter the user input for statements that match any of the definitions in the datatbase, and offer it to the main program in the correct pigeonholes.. Further in this Appendix it will be explained how input that is accepted by the language interpretation module is processed internally, and tested for coherence and validity.

C.4 The processing flags

Before moving to the programmatic aspects of the language filtering process, some remarks will be made in relation to the data contents of an input file.

For a multi-satellite / multi-arc program, three main categories of input variables can be distinguished. Some input will be related to a single arc within the solution, like the epochs for start and end of that arc. Other data will be related to a specific satellite, while more than one arc for this satellite may be present in the solution process. Finally there will be global data that affects all arcs and satellites, like an upper limit to the amount of iterations in the least-squares process. The total amount of different satellites will usually be small, and therefore only a distinction between two categories is used in Faust, namely between 'global' and 'local' data (local meaning that it is related to a single arc only). The category of satellite dependent data is in most cases treated as global data, as the solution process may involve many arcs for the same satellite.

An important aspect of this classification, in relation to the organisation of the user input, is that global input statements may be placed anywhere within the input, but local statements will somehow have to be attached to a specific arc. To do this, local statements are grouped in text blocks that relate to a single arc only, so that at any time the language processor knows to which block the data belongs. Within the language database the first processing flag of (C.1) is used to indicate if the statement, when found in the input, is attached to the active local block (if the flag is 0) or if it is a global statement (if the flag is set to 1). In addition, a special input statement is used, aimed at the language filter routine rather than at the main program. This special statement will tell the language scanner to start a new block of local statements. Any local statement found from that point will be related to the most recently started local block (how such special statements are implemented in the language scanner will be described further below).

Some input statements will be required for every run of Faust, while other input is only needed for specific applications. Already at the level of the language scanner, the user input can be checked for the required presence of all compulsary input. If one or more of these statements are missing there is no point in continuing the current process, and the language scanner can stop programme execution with one or more error messages. Because the only action of the language processor is filtering of the user input according to the statement definitions in the database, it will be necessary to indicate in the database if a statement is compulsary or optional. This is done by means of the second processing flag in (C.1), which will normally be **0** (for optional statements) but may be set to **1** to indicate a compulsary statement. Each statement for which this flag is set to **1** should be present at least once in the user input.

A further data classification exists between input statements that specify a single data item (like a value for the speed of light, or the name of the report file for the programme of which there will be just one within a given process) and input statements that may be repeated many times within the input (like statements to specify the a priori standard deviations for different ground stations). The data of the first type will be called scalar data, that of the second type will be referred to as list data. In analogy, corresponding input statements will be called scalar statements and list statements respectively. It will be obvious that a list statement requires in general at least two data arguments, namely the actual data value that we want to enter into the programme, and in addition an index number or other form of identification in order to specify which element of the list is intended. Although the classification in scalar data and list data relates to a different internal treatment of the input, at the level of language recognition no special action is needed. In the language database are therefore no flags to specify 'scalar' or 'list', and the way in which this distinction is implemented in Faust will be explained in following Sections.

The third processing flag in the statement definition (C.1) is used to indicate special statements that require other than the standard response from the language filter. Several special statements (i.e. different values for the third processing flag) are implemented in the language scanner. The values that are supported at present will be shortly described here; in the future, additional values may be implemented. It is suggested to document such changes in the header of the language database file, where also several other comments have been included. Note that this flag should only be used to control the language filtering process from Figure C.1; any other input (aimed at the main programme) should be treated as a normal input data statement.

flag_3 = 1 This statement forms the group separator statement described above, used to start a new block for local statements. At present only one group separator statement is defined in the language database of Faust, to start the block with input statements for a particular arc within the solution.

flag_3 = 2 It has been mentioned that more than one input file can be offered to the program, but it was not explained how this is done. The language scanner will have to know that the current input file is followed by another, so a special statement is used to define a filename for continuation of the user input. If a statement with a processing flag **2** is defined, the language scanner will interprete the first character string argument in the corresponding input line as the name of the chained input file. In this second file, the same statement can of course be used again, so that many different input files may be chained together. They will be processed as if all files were just one consecutive text file.

flag_3 = 3 Some statements may require input character strings that actually contain spaces or commas (for instance a process title), i.e. characters that would normally be used to separate two different elements within the input line. If the third processing flag is set to **3**, the statement will be treated by the language processor in a special way, namely as if all words and numbers present on the input line (after the second keyword) form one consecutive character string.

flag_3 = 4 A program like Faust is modified on a regular basis, and not only is it possible that new statements are added to the program, but occasionally it may also happen that an existing statement becomes superfluous and will no longer be supported by the main programme. In following Sections it will become clear that it is ill-advised to remove existing statement definitions from the language database, though it is always possible to add new language elements. To indicate that certain statements from the language database are no longer used, the third processing flag can be set to **4**, which will cause the language scanner to treat this statement as a comment line and ignore it. Old input files that contain this statement will still be usable, although the statement has lost its meaning.

C.5 Input buffers and pointer structures

This section will move deeper into the numerical structures of the language processor by explaining how the data is actually read from a valid input statement and then stored internally. The central concept of the user language is that the input file is not forced into a particular format and that input statements may be placed in arbitrary order, with the weak exception of local statements that are always related to the most recently declared local block. In general, neither the precise amount of data nor the order in which it is specified in the user input is known in advance. Within the code however, the many variables and arrays that may be influenced by user input have a fixed internal organisation. The task of the language interpreter is to transform the free format input data into strictly predetermined data structures. Because it would be inefficient to scan the user input for each input variable explicitly, the transfer from data from the user input file into program variables in working memory is done in the two stages illustrated in Figure C.1. These stages will now be explained in more detail than was done above.

<u>STAGE 1 : LANGUAGE FILTERING</u>

The first action of the language interface is to read the entire language database into RAM memory, by means of a subroutine called **readwords.f**. The language scanner routine **scaninput.f** will then read each line of the user input, and try to match this line with any of the defined input statements. For each recognised line of input, the language scanner determines the following numbers :

(1) The unique *identification number* for the statement on that line.

(2) The *block number* of local statements to which this statement is related. The block separator statement described in the previous Section does nothing else than increasing the block number counter variable by one. For global statements, this block identification number has no meaning at all.

(3) Each encountered input data argument is stored in the first available position of the relevant input data buffer array (i.e. in one of three available arrays, for integers, real numbers or character strings). The language scanner will attach three numbers to the statement identification that provide the *location* of any encountered data arguments within the three buffer arrays.

(4) Three numbers that provide the *amount* of integers, real numbers and character string arguments found on the input line, hence the amount of places occupied in the input buffers. Together with the indices from (3) all input data arguments can now be connected with a particular statement, even if more than one integer number, real number or character string was found in the input.

The combination of (3) and (4) forms a pointer structure similar to the mapping arrays introduced for the parameters and partials in Section 2.6.

The output produced by the language scanning stage consists therefore of

(a) the three input buffer arrays
(b) eight identification numbers for each line of user input

Note that these buffers are still organised in the order of the user input file. However, the natural language elements have now been entirely eliminated from the interpretation process.

STAGE 2 : DATA ORGANISATION

The data that is stored in the input buffers will now be organised into a structure that can be used by the main program. This corresponds to filling the 'pigeonholes' of Figure C.1, after which the main program will be able to find individual input variables by looking in the appropriate pigeonhole. *Scalar* data will be organised in pigeonholes for scalar data, while *list* data is put in 'three-dimensional pigeonholes' that allow for entire arrays to be stored.

Hardcoded within Faust are two arrays with statement numbers. One array contains the statement numbers for all scalar statements, the other provides the statement numbers for all list statements. For each recognised input statement, the programme will search both these hardcoded arrays until it has found the corresponding statement identification number. A statement number can only appear in one of the two arrays, so by locating the appropriate identification number in one of the two internal arrays the programme also determines if the statement in question is a scalar statement or a list statement. In addition, the index n_{ID} of the hardcoded array at which the statement number is found is established. This index corresponds to the 'number of the pigeonhole' in which the data should be stored. The two hardcoded arrays are therefore nothing else than maps of the two pigeonholes, passed to the language module by the main program. For list data, the 'pigeonhole' is in fact an array of arrays. The total length of an input list is limited by the constant size of each of these buffer arrays, but can be modified by means of a FORTRAN parameter statement in the main programme (by means of a parameter called **MAXLIST**).

This second stage is encorporated in a subroutines named **'GetPointers.f'**. As the name suggests, the input data is not actually copied from the three input buffer arrays into another part of memory, but the 'pigeonholes' are filled with the location of the relevant data within any of the three input buffers. The 'pigeonholes' form a collection of pointer variables in an unambiguous order, and the routine **GetPointers.f** assigns the appropriate locations to these pointers.

As soon as all user input has been organised into the 'pigeonhole' structure, the task of the language interpretation module ends. However, these pigeonholes are still only a collection of arrays with meaningless internal names, like **iBuffer, dBuffer** and **cBuffer**. Within the programme, it would be most awkward to employ variables like iBuffer(24), dBuffer(162) or cBuffer(724) instead of more sensible names like **'nStation'**, **'tStart'** or **'ProcessTitle'**. The language interpretation routines are therefore followed by a routine called **'progvar.f'** that will transfer individual input variables from their position in the abstract input buffers into programme variables that have more meaningful names. Although this may seem slightly inefficient from a memory management point of view, it is a small price to pay for the flexibility that is offered by the natural language interface itself. The overall memory requirements of the programme will usually be dominated by the size of the normal matrix, in comparison to which the input buffers are very small.

This discussion of the input buffer structure is inevitably rather abstract. To make it easier to understand the interaction of the various parts of the language interpretation system, Appendix E includes a step-by-step description of how to add a new input statement to the program. It can also be useful to inspect the actual code of the routines ReadWords.f, ScanInput.f, GetPointers.f and ProgVar.f, for instance by following the path of an individual input line through the entire langauge interface.

A user guide to Faust — Appendix D

Throughout the period of development and operational application of Faust, a substantial amount of practical experience has been accumulated. In order not to lose this experience for future users of the code, this Appendix will summarize a variety of practical instructions for new users of Faust. In particular the following applications of the program will be described :

D.1 Creating observation files with Wagner

Wagner is used to create an observations file that can be used by Faust. To be able to run the program, we need the following :

(1) The executable file **wagner**

It is recommended to use a symbolic link to a central executable file, in order to ensure that upgrades or extensions of the program are adopted automatically.

(2) The language file **wagner.statements**

Again, it is best to use a link rather than to copy the actual file.

(3) The file **wagner.inp**

This file contains the user input and will therefore be a physical file in the directory from where **wagner** will be run.

(4) Various tracking data files

The input statements that can be used in **wagner.inp** are documented by means of comments in the language file **wagner.statements** (which is a normal text file). New users are referred to this file in order to find out what wagner can and can not do. An example input file is given below. For normal cases, we define in **wagner.inp** :

(a) The name for the output observations file, and for the associated report file that contains a survey of the data in the output.

(b) An input 'block' for each input data file from which observations have to be included. Usually, each block only consists of the 'data block' statement and a statements that defines name and datatype of the input file.

(c) Start and end epoch of the period for which an observation file is wanted. If these are not defined, the entire raw data input file will be used.

Once that the input file has been set up for the application of interest, Wagner can be run from the command prompt. It will print several messages to the screen, and in the unlikely case of an unrecoverable error it will stop with an error message. If, for any reason, the normal execution of Wagner is interrupted, it is best to delete all produced temporary files before a next run. Temporary files created by Wagner are files of the form **INTERIM.nnn** or **scratch.n,** in which n are digits 0 - 9.

D.2 Orbit computation from scratch : ERS

To start a precise orbit determination process with Faust, we need

(1) The executable **faust**

As with **wagner,** it is best to use a symbolic link to a central executable rather than an actual copy of the file.

(2) The language file **faust.statements**

A link to the centrally maintained language database file.

(3) The input file **faust.inp**

By convention, this file only contains those input statements that will hardly ever change, like file names for satellite models, the gravity field, etc. In addition, **faust.inp** contains a '**next file**' statements, that will connect a job-specific input file to the input of faust. By having several of these '**next file**' statements, we can switch between different processes by un-commenting a particular input file.

(4) A set of a priori parameters

The only parameters that are really required to start an orbit determination process are the initial position and velocity of the satellite at the time of epoch. In general, these are obtained by producing an output state vector at the end of a previous arc (see **faust.statements** for the input command that will do this). All orbit parameters are defined in the input files by means of statements like 'estimate ...', 'fix ...' etcetera. See the example input file below.

It is assumed that we start a new ERS arc from scratch, i.e. apart from approximate values for the initial position and velocity at epoch we do not have any further a priori parameter values. Depending on the available amounts of tracking data, we will want to solve for several drag scale parameters - typically 4 per day -, empirical once-per-revolution accelerations - typically one set of four parameters for a five-day arc - and possibly a scale factor for solar radiation pressure. As no accurate values for these parameters are available, we can only choose nominal standard values (i.e. all scale factors are nominally 1, all once-per-revolution accelerations are nominally 0).

The only observations available at this point are the land-based tracking data types SLR and possibly PRARE for ERS-2 : radar altimetry is not used for the orbit determination process, and crossover locations can only be computed *after* an initial orbit has been converged. In general only SLR is used for the initial orbit determination, while PRARE (for which we have to solve a large amount of arc-dependent parameters) can be added once that reasonably accurate orbital parameters have been found. The remainder of this Section is concerned with the computation of an SLR-only orbit for ERS-1.

If all SLR data would be accurate and reliable the orbit determination process would be fully automatic. Unfortunately, practically each orbit arc will contain one or more passes - or sometimes just an individual observation - that are hopelessly inaccurate. The least squares process will be indiscriminate to such incorrect measurements, and even a single bad measurement may prevent convergence to centimetre level, especially for ERS-1 for which global coverage by SLR tracking is poor. The problem of ERS orbit determination is therefore usually equivalent to removing the bad SLR observations from the available dataset., after which the orbit solution will converge very quickly.

A reasonably fool-proof method of doing this is described in the following steps. If these steps do not have the desired success, Section D.4 will contain a list of trouble-shooting tips.

(a) Run a single process iteration with **faust**, creating an output ephemeris file as described in Chapter 3. In this first iteration, all SLR data is included and Bayesian constraints are weak (or are not used at all), in order to allow the parameters to move freely. Do not apply any rejection mechanism, and set the level of 'absurd residuals' to a very high value (1.D6 or larger).

(b) Inspect the produced residuals file to detect observations that are obviously wrong. In general, this is not easy because in this first iteration all residuals will be large. If bad observations are found, these can be excluded from successive runs by means of input statements like 'ignore pass ...', 'ignore station ....', etc.

(c) Inspect the computed parameter corrections and new parameter values. Bad data will produce unrealistic parameter corrections, like negative drag scale factors or large empirical accelerations. In combination with (b), it is usually possible to identify the pass that is responsible for such behavior.

(d) Rerun **faust** with the same a priori parameters as before but now using the previously generated ephemeris file as *input* ephemeris. Set the upper limit to the amount of iterations to 2 or more, so that after the first - fast - iteration the process will continue by integrating through a second iteration.

The first iteration will now be very fast, and as soon as the second process iteration starts it is usually obvious whether the process is working correctly or not. If the residuals in the second iteration are still high (several metres), it may be necessary to repeat (b) and (c). The advantage of using the input ephemeris file is that we can repeat the steps (b) to (d) several times without the need to re-integrate the orbit, which would take much more time. Nominally, the iteration process will immediately converge to residuals at centimeter level in the second iteration. If this is not the case, and no obviously incorrect observations can be detected, it is recommended to try one or more of the options from Appendix D.5.

Once that the parameter estimation process is working satisfactory, we can run two or three automatic iterations, applying a window rejection level of about 20 cm. Convergence will be reached when the parameter corrections become very small or when the RMS of residuals hardly decreases from one iteration to the next. Faust can detect these convergence criteria automatically, but in general we will define a fixed number of iterations in the input.

D.3 Orbit determination from scratch : TOPEX/Poseidon

For TOPEX/Poseidon, the global coverage by SLR data is much better than for ERS-1, while its sensitivity to atmospheric drag variability is much less important. This implies that in most cases the orbit determination process for TOPEX/Poseidon is more straightforward than for ERS, especially as we can use the additional large amounts of DORIS doppler data.

The quickest way to determine a precise orbit for TOPEX/Poseidon is to compute an SLR-only orbit (in the way described in Section D.2 for ERS-1) and not include the DORIS data before the SLR residuals have been reduced to a level below 10 centimetres. This is done to avoid substantial absorption of orbit errors in the pairs of pass-dependent parameters for the DORIS data, which would slow down the convergence process.

Once that reasonably accurate orbital parameters are available from SLR only (note that for TOPEX it is not unusual to have negative drag scale factors, as these parameters tend to behave as empirical along track accelerations), a next iteration can be run using the 'autoDORIS' option from Faust (see **faust.statements** for a description of the related input statements). This feature will scan the input observations file to count the amount of DORIS passes, then create the pairs of pass-dependent parameters and initialise them with default zero values. After matrix inversion, a file with DORIS parameters will be produced (there are too many of these parameters to include them in normal parameter input statements) that can be used in successive iterations by using the 'fileDORIS' options of Faust. In general, we only need two iterations to fully converge the pass-dependent parameters for DORIS data and simultaneously adjust the orbital parameters to fit to the doppler data. In later runs of Faust, we can then use a 'freeze DORISparam' statement, which tells the program not to solve for the pass-dependent parameters anymore but keep them fixed at the input values.

D.4 Trouble shooting

With unfortunate regularity, processes appear that do not follow the nominal straightforward convergence scheme. Symptoms of bad behavior may be : insufficient reduction of residuals after one or more iterations; unrealistic parameter values; large correlations between parameters; parameters that do not converge but keep sliding from one iteration to another.

The list below forms a collection of tricks that may help to analyse and solve the problem.

(1) Use a minimum dataset

If it is suspected that a particular arc contains one or more bad passes, but it is not clear which stations are the culprits, it is usually possible to converge the orbit in a certain 'safety mode'. This method implies that we exclude all observations, apart from data from a handful of strategically located stations that hardly ever produce bad measurements. This is done by using statements like '**ignore station 1953**' etc, for *all* stations apart from the trusted ones. Note that it may be necessary to reduce the amount of parameters solved for, accordingly. The following classification of SLR stations has risen from substantial practical experience :

- Very reliable SLR subset :	7840, 7090, 7210, 7403
- Notoriously unreliable SLR stations :	7843, 8834, 1953, 7530

In addition, certain stations (e.g. 7080, 7109) regularly produce a perfect pass that ends with a single incredibly wrong measurement, which may be hard to detect in a first iteration. In 'safety mode' we only use the four very reliable stations, which should at least provide a useful initial state vector, and daily drag parameters. Once that this has worked we can add the other SLR data and solve for more parameters (once-per-revolution accelerations, etc.). Bad passes will now show up almost immediately, after which they can be excluded from further iteration by means of statements like '**ignore pass 14**', or '**ignore station 7530**'.

(2) Change the parametrization

Poor convergence may be a result of poor tracking data coverage, i.e. we try to solve for too many parameters (in particular too many drag scale factors or empirical accelerations). Because the empirical accelerations only serve to fine-tune the orbit model, it is often a good idea to solve just for the initial state vector and a modest amount of drag scale factors (daily or two-daily). Once that these parameters have converged to an acceptable level, we may add more parameters, avoiding divergence by means of adequate constraints (i.e. small sigmas). In extreme cases, we could solve for the initial position only in a first process, add two drag parameters in a successive process, then gradually increase the amount of drag parameters in further runs. This approach is fairly time-consuming but is almost guaranteed to work.

(3) Shift the starting epoch

It sometimes happens that the initial position and velocity do not converge very well because of awkward data distribution, or poor data coverage, at the start of the arc. By using '**data vector**' statements in the same arc, we can choose a later epoch as the starting time of the arc (accepting a gap with respect to the previous arc). Alternatively, we can obtain an earlier epoch by using the same 'data vector' statement in a solution process for the preceding arc.

(4) Vary the constraints

In the case of a well-defined orbit solution (with plenty of reliable data) we do not need strong constraints on the parameters, i.e. we define no sigmas at all, or sigmas of at least twenty percent or so of the parameter value itself. If bad behavior of the solution process is observed from unrealistic parameter solutions (e.g. a negative drag scale factors for ERS-1), the pass responsible for this problem can often be identified if the drag parameters around the suspected problem are severely constrained, by means of small sigmas. This will still allow most of the remaining orbit arc to converge normally. Once that the bad data has been identified and eliminated, the strong constraints can be removed to allow for normal convergence. Note that this method does not work very well if the trouble area is close to the start of the arc, as the occurring errors will propagate through the integration process.

(5) Use altimetry data

This is a more drastic measure that should only be applied as a last resort. Unexpected behavior of an SLR-only orbit is usually due to poor data coverage. Even if the orbit does not converge to RMS levels below one or two metres, we can still use it to compute the crossover locations. Adding these crossovers may then help to obtain a better orbit fit and to identify the bad data. In extreme cases, we could even use the altimetry data itself for orbit determination. Such practice should preferably be followed by a re-run *without* the altimetry, once that fairly accurate orbital parameters are available. This re-run - with adequately constrained parameters - will help to decorrelate the orbital solution from possible geophysical signals in the altimetry data.

D.5 Combining single-arc solutions into a multi-arc run

In theory, it should not make any difference if a set of orbital parameters for a certain arc is solved from a single arc solution or from a multi-arc solution. In practice, however, two problems occur that will result in slight offsets between the two cases. The first problem is related to finite machine precision and in particular to the internal clock of the numerical integration process. In principle the internal time can be obtained in two ways, namely by addition and by multiplication :

- addition : $$t_k = t_0 + \sum_{i=1}^{k} h \qquad \text{(D.1)}$$

- multiplication : $$t_k = t_0 + k.h \qquad \text{(D.2)}$$

The integration step h is typically 30 seconds ($3.47 * 10^{-4}$days) while the time t_k is needed in UTC, i.e. as a count of Mean Julian Days that involves numbers of the order 10^4. The large differences in order of magnitude implies that even in 8-byte arythmethic the effect of accumulative round-off errors in the internal clock t_k becomes observable after an integration time of several days. In this respect, it should be noted that it hardly matters if we use the additive time (D.1) or the multiplied time (D.2), although the latter construction appeared to perform slightly better. This error is an error in t_k, but it will internally appear as an error in all observation timetags UTC because the clock time t_k is assumed to be the standard.

The effects of internal clock inaccuracies are normally absorbed by means of a minute along-track shift of the initial state vector. However, this very small offset (in the order of several centimetres or less) will depend on the initial epoch t_0 of the internal clock, which is changed if we combine several consecutive arcs into a new multi-arc parameter estimation process. The earliest arc within the multi-arc process will provide the multi-arc clock epoch t_0, and this earliest arc will therefore not notice any difference in its internal clock errors. Any later arc, however, will experience slightly different internal clock round-off errors than in its single-arc solution, which appears as a small along-track 'error' in the initial state vector of the multi-arc run. The integration process will gradually amplify this error in the initial values, causing the orbit to diverge from its previous good fit to the data.

The obvious solution to this problem is to run a first multi-arc process in which we only solve for the initial state vectors of the second and any later arc, while keeping all other parameters fixed. The time-tag bias of the internal clock is then easily absorbed by means of small along-track adjustments of the initial positions, of all arcs but the first.

The second problem, also related to the internal clock, is that Faust inevitably phases the integration steps of arcs that are integrated in parallel (see Chapter 3). If we compute a 35-day multi-arc solution for ERS-1 and then add four parallel TOPEX/Poseidon arcs to the solution process, the step boundaries of the parallel arcs of ERS and TOPEX will be determined by the same internal clock t_k. In a multi-arc process for a single satellite (e.g. ERS-1) it may happen that there is a small gap between consecutive arcs, for instance if a manoeuvre occurs at that point. Such a gap will normally result in an interruption of the internal clock t_k, because the event handling sequence of Faust will simply jump to the start of the next arc and begin a new integration process for that arc. However, as soon as we introduce a parallel arc (e.g. for TOPEX) the internal clock is not interrupted at the gap between consecutive arcs because the integration of the parallel orbit will have to continue. The result is once again a difference in the internal clock round-off errors, appearing as an along-track offset of the initial position for the second ERS-1 arc, relative to the earlier solution (see Figure D.1).

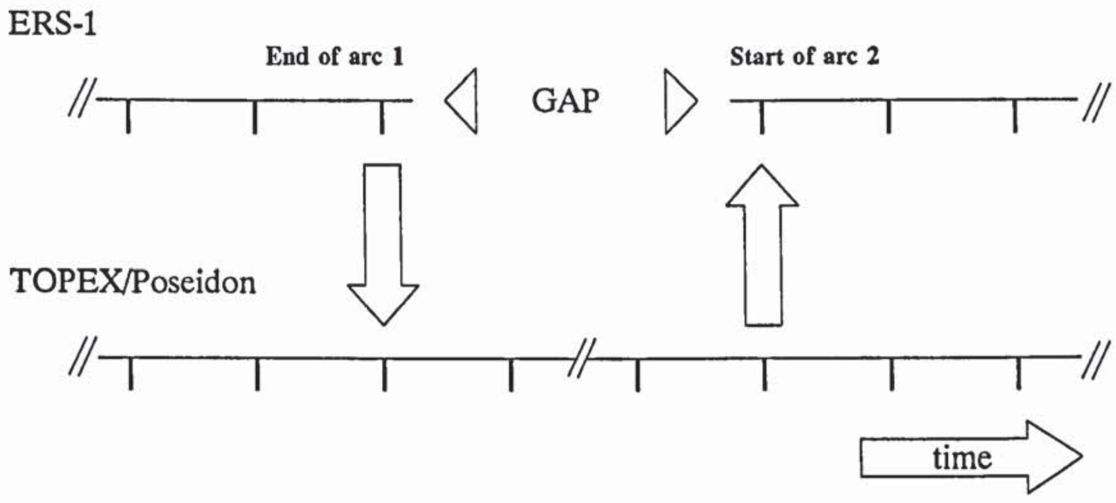


Figure D.1 Adoption of the internal clock error via parallel arcs. The TOPEX arc will have the same internal clock error as arc 1 of ERS-1, with which it is in phase. This error is subsequently adopted by arc 2, when it is phased with the step boundaries of the continuing TOPEX arc. If no parallel TOPEX arc would be included in the solution, the internal clock error of arc 2 is independent of that of arc 1 and the two will therefore be different.

To tackle these problems, Faust allows for the possibility to define a **reference timetag** in the user input, which corresponds to the clock epoch t_0 in (D.1) and (D.2). We can then define the same reference timetag for all separate single-arc solutions that will later be used in a multi-arc run, and their internal clocks will always use the same starting epoch. The obtained parameters can subsequently be inserted directly in a multi-arc process.

In addition to the clock round-off errors, other - less relevant - differences occur between single-arc runs and multi-arc processes that are constructed from earlier single arc jobs :

(1) The reference epoch for tectonic plate motion will normally be the centre of the solution period. Because the solution period of a single arc run differs from that of a multi-arc run, minute differences between station coordinates may occur for long multi-arc processes or for 'fast' stations.

(2) To efficiently evaluate planetary ephemerides and precession and nutation of the Earth, time series are used for each of the coefficients of the various rotation matrices. The coefficients of these internal time series are obtained via Tchebychev polynomials through tabulated coefficients of other time series. Because the reference epochs for single arc runs and multi-arc runs differ, very small differences may occur in the coefficients of these time series.

(3) Tabulated values for the $F10.7$ solar radiation intensity are evaluated by means of a moving average over two solar cycles at the start of the program. The internal table that is the result of this evaluation may differ marginally for single-arc solutions and later multi-arc solutions, in case of extreem solar activity just outside the time frame covered in the single arc solution, but inside that of the multi-arc case.

None of these effects have ever seemed to cause problems, although their presence has been identified. The resulting differences are usually too small to be observed within the a priori noise bands of the tracking data sets.

D.6 Constructing multi-run solutions

The gravity field solution discussed in Chapter 8 is an example of a least squares process that is constructed by recombining normal matrices that have been produced by earlier smaller processes, as described in Chapter 2. In general, this is necessary if a process either requires a very large normal matrix and / or involves a very large amount of observations. In both cases the load on the computer network would seriously interfere with normal daily activities, for which reason the process will have to be split into more manageble jobs that can be run overnight or during the weekend.

A multi-run process consists of a number of preparatory jobs that each generate parts of the ultimate normal matrix, and one final job that will read back all previously generated matrices, invert the overall matrix, and compute the parameter corrections. As mentioned in Chapter 2, each submatrix will be stored in its separate file, and these (very many) files form the individual building blocks for any multi-run process that we might want to construct later on. The names of these - automatically generated - files follow the format

[projectID].[ArcID1]-[ParamID1].[ArcID2]-[ParamID2].[Version ID]

The two ArcID numbers identify the two arcs to which this submatrix is related, the two ParamID numbers specify the two involved parameter types. Global parameters (i.e. parameters that are shared by all arcs) employ the generic arcID number 9999, which should therefore be avoided as a user-specified arcID number.

Apart from these normal matrix files, the preparatory jobs will create smaller files that contain the a priori parameter values that were used in the single runs, as well as various other internal parameter description variables.

To convert a normal job of Faust into a preparatory job for a multi-run porocess, all that needs to be added to the input file is the statement '**data ProjectID [MyProject]**', and an identifier number for each included arc by means of a statement '**data ArcID [MyArcNumber]**' within the data block for that arc. These numbers can be anything between 0 and 9998, and need not be consecutive as long as they are unique. As soon as a project ID is defined Faust will automatically produce the normal matrix files on disk, rather than inverting the normal matrix in the usual way. Because these files are written to the working directory from which Faust was started, it makes sense to run each separate preparatory job in its own subdirectory in order to avoid excessive amounts of files in one single directory.

The actual multi-run process to combine and invert the matrices uses exactly the same input as the preparatory jobs (these files may therefore be chained by means of '**next file [MyNextFile]**' statements). In addition, the multirun flag needs to be toggled by means of the statement '**multirun on**'. All parameter definition files - *not* the matrix files - should be present in the working directory from where the multirun process is started.

Because it would be impractical to copy all matrix files to a single directory for the multi-run job, Faust uses an index file of the normal matrices in the form of a text file that is declared in the input via the statement '**multirun file [MyFilelist]**'. It is the responsibility of the user to create this list correctly, which is easily done by piping directory listings into text files (like : **ls -l > MyList**) for each of the working directories used, and then combine & edit these files into a single text file. The reason for using this method is that it offers an extra way of controlling which of the earlier generated matrices will be included in the multi-run process : any line of the index file starting with an asterisk will be taken for a comment line and ignored. Note that ONLY the matrix files that are present in the index file will be included, regardless of the parameters that are defined in the user input.

If mismatches are detected between the user input, the parameter definition files, and / or the normal matrix files, a corresponding error message will be written to the report file and the related parameters will not be included in the solution process. It is however very difficult to make the multi-run process crash, as it has been written in a very robust way. Few things are more frustrating than not being able to invert a normal matrix that has been accumulated during many months of large overnight processes.

A programmers' guide to Faust — Appendix E

Only after sufficiently long practical experience with the code will a new programmer be able to know immediately which subroutines, arrays and variables are affected by - for instance - the implementation of a new tracking data type or parameter type. In order not to lose existing experience with the program, this Appendix contains a variety of 'How To ...' receipes that provide step by step instructions to the programming tasks that are most likely to occur in practice. The discussed subjects are :

E.1	Adding a new subroutine to Faust
E.2	Adding a new input statement to the language database
E.3	Implementation of a new parameter type in Faust
E.4	Implementation of a new tracking data type
E.5	Implementation of a new satellite
E.6	Implementation of a new event in the event handler loop
E.7	Tuning array sizes for specific tasks

It is recommended to read and understand each Section before attempting to actually modify the code. Please refer to Appendix B for navigating through different parts of the program. In addition to the instructions of this Appendix, most parts of the code contain a variety of comment lines to assist in future modifications.

E.1 Adding a new subroutine to Faust

Obviously, this description only covers the programming technicalities for adding a new subroutine to the existing code. The contents and functionality of a new subroutine are entirely dependent on the design of the programmer.

STEP 1	Study Appendix B.2 to understand the main programming conventions used.
STEP 2	Create a basic subroutine file in the source code directory of Faust : decide upon a descriptive subroutine name, then copy an existing subroutine (a short one, like

thruster.f) to a file under the new name, with the extension **.f** of course. DO NOT overwrite existing subroutines. Remove all code from your new file, apart from the statements that relate to the CPU tracer (at the very beginning and the very end of the code - see existing subroutine files).

STEP 3 Determine the internal index number of the subroutine. This is done by inspection of the file **subroutines.f**. In principle you can select the first available number, as these numbers have no other purpose than uniquely identifying the elements of the array CPU, and of certain arrays in program **getstruc** (see Appendix B). Add you new subroutine to the file **subroutines.f**, please maintain alphabetical order.

STEP 4 Edit both CPU tracer statements in your new subroutine, in order to correspond to the array index that you have just determined in **subroutines.f**.

STEP 5 Add your subroutine to the **makefile**. This is done by adding its name (without the extension **.f**) to the existing list of routines in **makefile**. Again, please maintain alphabetical order for practical reasons.

STEP 6 Write your subroutine in the new shell file, making sure that any variables that are passed through the header have corresponding variables in the calling statement. Insert a calling statement in the existing code where necessary.

STEP 7 Run **make** in the source code directory and fix possible compilation errors.

E.2 Adding a new input statement to the language database

Although this process may look complicated at first, it is in fact fairly straightforward and, with some experience, can be done within five minutes. Note that the process is in principle identical for **faust** and **wagner**, although names of individual arrays may differ slightly in the two programs. The description below relates to **faust**.

STEP 1 Run program **kwindex**. If it asks for a statement file, type **faust.statements**, or **wagner.statements** accordingly. The program **kwindex** will give you the index number that should be used for your new input statement.

STEP 2 Create a new statement in the language database file, using the new index number provided by **kwindex**. Please read Appendix C for understanding the various elements of a statement definition. Try to maintain existing conventions (in particular for **estimate / consider / fix / sigma** statements which should always have **four consecutive statement identification numbers** - see existing statements for examples).

STEP 3 Open the main program file **faust.f** in your text editor of choice.

STEP 4 In **faust.f**, find the table that documents the structure of the interface data (the names of the 'pigeonholes' from Appendix C). This 4-column table is located somewhere towards the end of the long block of comment lines at the start of **faust.f**, it starts with a line that describes '**Mapdat(i,j,k)** ....'.

The changes that you have to make to **faust.f** are different for scalar statements and list statements. List data is repeatable input data, i.e. coming from input statements that may be present many times in the input file. Make sure to understand the differences between scalar / list statement and between local / global statements (see Appendix C). A statement is either local or global, and at the same time either scalar or a list statement. Note that it IS possible to ruin the existing interface structure by inappropriate manipulations of the arrays **inx** and **listinx** (if you don't trust yourself to do everything right the first time, be sure to make a safety copy of **faust.f** before changing anything).

SCALAR STATEMENTS

STEP 5 In step 2 you have defined how many integers, reals or character strings will be passed to the program by your new input statement. To reserve 'pigeonholes' for these new input variables, you have to reserve elements of the appropriate column(s) of the table, so that future programmers also know which elements of the buffer arrays correspond to these input variables. If the columns are too short, follow STEP 6 first. If there is still sufficient room for your new variable(s), go to STEP 7 directly.

[STEP 6] To increase the amount of 'pigeonholes', increase parameter **MaxInx** with 5. In addition, you will have to add 5 'empty' slots to the table in Faust. This is done by adding 5 extra zeroes to the end of **each** of the four sub-arrays of array **inx**. (the four sub-arrays correspond to data of type integer, double precision, character string and logical). The array **inx** is defined in a **data** statement in **faust.f**, and commented as '... *pointers for correct interpretation of input* *statements* ...'. Please identify and understand the four-part structure before enlarging <u>each</u> of the four subarrays (by adding five zeroes : '**,0 ,0 ,0 ,0 ,0**'). **Mistakes here will seriously affect the working of the language interface.**

STEP 7 Update the elements of array **inx** that correspond to your new input data, by <u>replacing</u> the zero values in the array with the identification number of your new statement (as obtained in STEP 1). Array **inx** forms a 'map' to the pigenoholes, and contains the identification numbers of input statements from the file **faust.statements** that fill a particular place of the pigeonhole. If you don't understand the involved principles, try to trace one of the existing input statements by using the table for MapDat (the table that you have changed in step 5). You can now proceed to STEP 8 below.

<u>LIST STATEMENTS</u>

STEP 5 A list statements will normally have one integer argument and one real argument. The integer is used to identify an element of the list (i.e. a station number, a simple index number 1 ... N, a number to identify a measurement type, etc.). The real number contains the actual list data. All list input is stored into two arrays, **ListID** and **dList** respectively. The amount of different lists is given by the parameter **MaxList**. The first thing that you have to do is therefore to increase the number **MaxList** in the block(s) with **parameter** statements. Note that if your new list input relates to a new parameter type you will have to add **four** new lists, namely one for **estimate**, one for **consider**, one for **fix** and one for **sigma**.

STEP 6 Make sure that future programmers will know that the newly created input lists relate to your new statement. This is done by **documenting** your new list variables in the header of **faust.f**, just below the table for MapDat(i,j,k) (see STEP 5 of the SCALAR STATEMENTS above). From inspection of the existing comment lines it will probably be clear how to document the new list statement(s).

STEP 7 Add the identification number of your new list-statement(s) to the **data** array **listinx**. This array is located in a **data** statement, close to the array **inx** that is used for SCALAR STATEMENTS (see above).

The following steps are again identical for list input and scalar input :

STEP 8 Recompile **faust** by running **make** on the standard makefile for faust. In principle, the language interface should now be able to correctly identify and accept your new input statement. Test this, by putting your new statement somewhere in the input and running the program (you can cancel it as soon as it starts printing to the screen). Now, inspect the file **input.scan** to check if you new statement has been marked with the correct statement identification number.

STEP 9 Even though the language interface already accepts your new input data, the program itself does not do anything with your data yet (*unless you are adding a new parameter statement - see Section E.3)*. However, in subroutine **progvar.f** you can extract your new scalar data from the 'pigeonholes', and new list data from the arrays **ListID** and dList. The easiest way of learning how to do this, is by inspection of the existing code of **progvar.f** : look for code like

$$\mathbf{MyVar = iBuffer\ (\ MapDat\ (\ i, j, k\ )\ )} \qquad \text{(E.1)}$$

in which **i** = 1 for iBuffer, **i** = 2 for dBuffer, **i** = 3 for cBuffer, **i** = 4 for lBuffer. The **j** indicates the group to which this input variable relates (note that global data will always use group 1). The **k** corresponds to the number of the pigeonhole, i.e. the index of an element of array **inx** that contains the statement number for this data element. If all of this sounds obscure, please try to follow existing index numbers of **MapDat** and **inx** and try to match these with a particular input statement of **faust.statements**.

The above discussion relates to normal, straightforward input variables and lists, which includes practically everything from statements like 'gravity file [...].', 'station sigma [...]", and so on. However, some specific statements can not be dealt with in this standard way, for which reason additional manipulations of the input buffers take place in subroutine **GetMSdata**. An Example of such 'complicated' statements is the defintion of a relative crossover bias, which takes as its arguments a parameter index, two satellite identification numbers, a value for the bias and optionally a character string like 'topex' or 'poseidon' to distinguish between the two altimeters on one satellite. Please study the file **getmsdata.f** if such more complicated statements might be required.

E.3 Implementation of a new parameter type

In principle, Faust contains a generalised parameter infrastructure that deals with the entire least-squares process, independent of the specific types of parameters (state vector, drag scale factor, etc.). Adding a new parameter type to the program is therefore fairly straightforward, and mainly involves the implementation of a few new input statements (as described in Section E.2) and modification of a few data arrays in **faust.f**. After that, the new parameters can be used throughout the code via the arrays **param** and **mappar** (the latter corresponds to the pointer array M_P of Section 2.6).

STEP 1 Add four new input statements to the language database, in the way described in Section E.2. The four required statements are list statements for **estimate, consider, fix**, and **sigma**, in which the second keyword is the parameter name (see examples in **faust.statements**). It is very important that these four new statements have four consecutive statement ID numbers (in the order estimate/consider/fix/sigma). Note that the flags for 'global' or 'local' should be set correctly, i.e. 0 for arc-dependent parameters and 1 for other parameters.

Some parameter types will require special measures to pass them to the input, in particular those parameters of which there are too many to fit into the normal input lists (e.g. DORIS and PRARE parameters, gravity field / tides / sea surface topography parameters, etcetera). **Even if parameters are not passed to the program through normal user input, the four parameter statements for 'estimate', 'consider', 'fix' and 'sigma' MUST be defined in faust.statements** This is because several generalised parameter processing facilities in Faust exploit the presence of these statements in the language interface.

It is NOT necessary to add anything to **progvar.f** : parameters are processed in subroutine **getparams.f** which will immediately work with new parameter types if their statements for 'estimate' etc. are defined in **faust.statements**.

STEP 2 In **faust.f**, the parameter **MaxInxP** should be increased with one. The new value of **MaxInxP** will also become the internal parameter identification number for the new parameter type (like '1' means state vector, '2' means GM, '3' means drag, etc.). To make sure that future programmers will also be aware of this, the comment lines in the header of **faust.f** contain a list of parameter types : please add your new parameter type to this documentation, as a reference for others.

STEP 3 Several **data** statements in **faust.f** must be extended for the new parameter type :

- Array **inxP** should be extended. This array contains the statement identification number for the **estimate** statement of all parameter types. The other three statements (consider, fix, sigma) are then also known to the program, as long as consecutive statement numbers are used.
- Array **VarPrt** should be extended with .TRUE. if the partials for this parameter type are variational partials (see Chapter 2), or .FALSE. for 'scalar' partials.
- Array **GlobalPar** should be extended with .TRUE. for global parameters, .FALSE. for local parameters. Note that these flags should match the corresponding flags for 'local' or 'global' in the file **faust.statements**
- Array **iParCat** should be extended with '1', '2' or '3', according to the parameter category to which the new parameter belongs (see Section 2.3). In principle, category 2 is only used for the DORIS parameters and category 3 for the PRARE tropospheric scale factors, all other parameters are of category 1. However, future tracking systems may also exploit the matrix partitionings of Section 2.3, which will then require **iParCat** to indicate the correct category.

The way in which parameters are accessed via **MapPar** and **Param** can be analysed in the various force model subroutines (see Appendix B) and by studying Section 2.6, in which the mapper arrays are explained.

Parameters like the gravity field coefficients etc. which are not entered to the program through the normal input are usually treated individually. The parameters for gravity field, tides and sea surface topography are copied from the various model files for these quantities, while the pass dependent parameters for DORIS and PRARE (see Chapter 5) are stored in separate files. Please study the code for such individual cases.

E.4 Implementation of a new tracking data type

As with parameters, most of the internal processing of observations is generalised for all tracking types so that the implementation of a new data type is fairly straightforward. In **wagner,** we have to include a subroutine to read a data record from the raw tracking data file while in **faust** we have to implement a subroutine to compute a calculated observation C_i from (2.1) and the partials $\frac{\partial C_i}{\partial x}$ for x, y and z.

STEP 1 Determine the record format ofr the new data type : the amount of satellite identification numbers (usually 1), station identification numbers (usually 1), observation components (usually 1), real data arguments and integer data arguments. The raw data file will probably contain many data fields per record that are not necessary in **faust,** and it makes sens to keep the amount of data arguments in the observations files for **faust** as small as possible. See Section 5.5 for a description of the record format in **faust** and **wagner.**

The following steps concern modifications to **wagner.f** :

STEP 2 Increase the parameter **MaxGrpID** which is in fact the total amount of different tracking data types recognised by the program.

STEP 3 Add the five record size descriptors for your new data type, as determined in STEP 1 above. To do this, you add the five integer numbers to the **data** statement for array **iRecSize** in **wagner.f.**

STEP 4 Add a **file** statement to the language database **wagner.statements.** See Section E.2 for a description of how to do this; the same array names **MapDat** and **inx** are used in **wagner** and in **faust.** See statements like 'file MERIT1' etc. for examples.

STEP 5 Add the statement identification number of your new **file** statement to the array **iGrpInx** (which is located in one of the **data** statements).

STEP 6 Add the code to **wagner.f** for opening your input file. Most raw data files will be ASCII text, in which case you don't have to add anything to the existing code. Some files may be unformatted, or contain a header that has to be read after opening the file. See the existing code for examples; if necessary, add a statement ' ... else if (j.eq. N) then .....' to the block that opens the files, in hwich N is your new data type number.

STEP 7 Write a subroutine '**getMyData.f**' like the existing routines for MERIT, crossovers, etcetera. This subroutine reads **a single data record** from the raw data file. Add a statement '... else if (j.eq.N) then ....' to the block that calls these routines; see the existing code for examples. See Section E.1 to find out how to add a subroutine to the existing program. Recompile **wagner**.

At this point, **wagner** should be capable of processing your new tracking data type, also in combination with any other data type. The remaining steps relate to **faust.f**.

STEP 8 Modify the array **Tstr** (located in one of the **data** statements of **faust.f**). Make sure that the element that you modify corresponds to the observation type number for your new data type (i.e. the 'N' in STEP 6 and 7 above).

STEP 9 Define an integer variable **iMyDatatype** (like **iDORIS, iPRARE, iMerit, etc**). This variable acts as a symbolic constant to identify your new data type, and you should therefore initialise it in the data statement just below that for **Tstr**.

STEP 10 Update array **nSatMs**, just below the data statement of STEP 9. This array indicates for each observation type how many satellite arcs are involved in the observation, i.e. usually it will be 1. For crossovers, or possible future satellite-to-satellite tracking types it may be 2 or more.

STEP 11 Update array **ObsUnits** in the data statement just below the one from STEP 10. This array contains unit conversion factors from megametres or megametres per day to centimetres and centimetres per second. See existing code for examples.

STEP 12 Write a subroutine like **resaltim.f**, **resprange.f** etc. that will compute the residual and partials for your new data type. See Section E.2 for how to add subroutines to **faust**. Add its calling statement to the 'if ... else if ..' construction in the event handling loop (part 6A of the main program file **faust.f**).

STEP 13 Update the output routine **IterReport.f** to make it work with your new data type. This subroutine should in fact automatically produce a new output table for your new data type, but you may have to add some numbers to data statements (like unit conversions, a name to your tracking type, etc.).

STEP 14 Recompile **faust** with the **makefile**.

The above should suffice for 'simple' tracking types similar to SLR, Prare range, altimetry, etcetera. More elaborate data types, in particular doppler data types that involve two different UTC time tags may require the implementation of an entirely new event in the event handling loop. See Section E.7 for a description of how to do this.

E.5 Implementation of a new satellite

Although the code of **faust** was kept as general as possible, certain parts of the code are inevitably satellite dependent, in particular the geometrical models for the satellite as used to evaluate surface forces, attitude control algorithms, etcetera. Because a new satellite will probably require the development of a new macromodel for the satellite geometry and its related code, it is difficult to discuss a 'standard' way of teaching **faust** to support a new satellite. This Section will summarize the parts of the code that will probably be affected, and it is advised to study the related sections of **faust** to see how things are done for ERS and TOPEX/Poseidon.

STEP 1 The satellite geometry will have to be read in, which happens at the start of the main program just before the start of the iteration process. The GUESS tables used for ERS-1 and ERS-2 are 'generalised' in a sense that they can be used for more than one satellite. The only practical implication of this generalisation is that if ERS-1 and ERS-2 are involved in a simultaneous solution, **faust** will only have to read one table, and all ERS arcs will use the same model data (by means of 'GUESSalias' statements - see **faust.statements**).

STEP 2 The force model routines **drag.f, directsrp.f** and **albedo.f** will have to be modified, to compute the effective surface area for atmospheric drag and solar radiation pressure. This will probably also involve calling a new attitude control algorithm somewhere within these routines, like **ersatt.f** and **topatt.f**.

STEP 3 It may be necessary to extend the routine **frames.f** which evaluates geometrical quantities in each integration step, in particular rotation matrices and angles of incidence for particle fluxes. Again, this may involve a call to the attitude control subroutine for the new satellite.

STEP 4 It may be necessary to update a number of subroutines that employ satellite dependent constructions of the form **if (sat = sat1) then .... else if (sat = sat2) then ....** To find out which subroutines contain satellite dependent aspects, run **'grep '9105001' *.f'** (or other satellite ID number) in the source code directory for **faust**. This should provide a list of references to 9105001, or the satellite ID of your choice.

E.6 Implementation of a new event in the event handler loop

The time-organised event handling loop of **faust** is organised as an '**if ... then .... else if ...**' construction, that has a separate entry for each event possible. The selector variable that is tested is **iEvent**, which, although being an integer variable, is compared with symbolic constants for each event that may occur. These symbolic constants are defined somewhere among the data statements in **faust.f** and collected in common block **Events**, which is only used by the main program **faust.f** and the priority selector routine **getevent.f**. The steps to add a new event are as follows :

STEP 1 Determine a name for your new event, and define an integer variable (preferably starting with **i** like **iObservation**, **iAddXOhold**, etc.) that will serve as a symbolic constant to handle your event internally. See existing code for examples. The value is assigned to the symbolic constant at the very beginning of the code of **faust.f**; make sure event numbers are consecutive, i.e. choose the first unused number.

STEP 2 Increase the size of array **StateNeeded** and add a flag to this array in its **data** statement. The flag **StateNeeded** indicates if the event requires the numerical integrator to output the state and partials at the time of the event, i.e. it triggers the interpolation routine. More importantly, only the 'needed' state and partials will be written to the ephemeris files. Examples of events that do NOT need the state and partials are for instance the start and end of an orbit arc (these events are of course already defined in the code).

STEP 3 The UTC timetags for your events must be handed to faust one way or another. Observation-related events will usually know their UTC timetag from the observation data files, while other events (like the **iVector** event that creates an state vector for future runs of **faust**) will be defined in normal user input. If your new events are user-defined, you may have to add one or more input statements to the language database, and collect a list of event time tags in a new array in **faust.f.** See Section E.2 for details of how to implement new input data in **faust**.

STEP 4 Modify the all-important subroutine **getevent.f** to make it test for your new event. This may not be straightforward. The order in which the subroutine tests for different events actually affects the behavior of the program, as it automatically implies a certain system of priorities for events that occur at exactly the same time. In most cases, however, it will only be necessary to add a variable of time **tMyEvent**, and make sure that in each call to **getevent** the variable will point to the *earliest* occurrence of **MyEvent** after the time that is reached by the event handling loop (and indicated by **tEvent**). Please inspect the existing code of **getevent.f** to see how this works, it is mainly a successive system of logical tests. If your event is earlier than any other event in the remaining process, the variables **iEvent** and **tEvent** should be set to indicate your event and the time (UTC) at which it occurs.

STEP 5 Write an event handling block in the main program **faust.f**. See the existing code for examples. After completing your event, the program will simply close the loop and make another call to **getevent**.

E.7 Tuning array sizes for specific tasks

Because in FORTRAN the possibilities for dynamic memory allocation are rather limited, the memory requirements for a particular application have to be controlled by a variety of **parameter** statements at the start of the code. The purpose of each of these parameters is documented with comment lines at the very start of file **faust.f.** In addition, the program contains a variety of array size tests throughout its subroutines, and will in most cases stop with an explicit error message if an array is underdimensioned (e.g. '**Array size MyLimit exceeded in subroutine MySub**'). This Section summarizes some of the most relevant array settings that may have to be modified according to the application of faust.

The settings of these arrays will severely affect the particular capabilities of the program (room for DORIS parameters or not, compiled for single arc solutions or for large multi-arc solutions, etcetera). For this reason, the executable files are usually not simply called **faust** but have an extension to indicate their particular application. The past has seen executables called **faustS, faustSD, faustSP** for single arc solutions of ERS-1, for TOPEX (with the **D** extension to indicate partitionings for DORIS parameters) and for ERS-2 (with a **P** for PRARE); **faustM, faustMD, faustMP** for multi-arc solutions for ERS-1, TOPEX and ERS-2 respectively; **faustG** for gravity field normal matrix generation; **faustGI** for gravity field inversion, and a variety of **faustT** executables for testing. The name of the executable is defined in the **makefile** for faust. The sizes for these various executables are outlined below.

The array sizes that are particularly important are :

MaxType1	The size of normal matrix partitioning A (Chapter 2), i.e. the maximum amount of paramaters of Category 1 that can be solved for in a single run.
MaxType2	The maximum amount of (*pairs of*) parameters of Category 2.
MaxType3	The maximum amount of parameters of Category 3.
MaxArc	The maximum amount of satellite arcs in a single run.
MaxExpl	The size of the integration buffers for variational partials. As each arc will reserve its own room for partials, this number should roughly be equal to the product of **MaxType1** and **MaxArc**, assuming that most Category 1 parameters require variational partials.
MaxLinPrt	The size of the memory buffers for non-variational partials. As these are not involved in the numerical integration process, the total memory requirements will not be very strongly dependent upon this parameter.
MaxXObuf	The size of the crossover buffer arrays in memory (see Section 5.6). This number will mainly depend upon the limit to the crossover interval, as discussed in Chapter 7.
MaxPass	The size of the tables for some pass-dependent statistics.
MaxUserInp	The maximum amount of lines of user input.
MaxBCmat	The size of the actual normal matrix partitionings B and C. This is either identical to **MaxType2**, or it is minimised to 1 in which case there **must** be a statement '**freeze dorisparam**' in the user input. This allows for a reduction in memory requirements if the doris parameters are not actually solved for, but only read from file and kept fixed.
MaxFmat	Size of the normal matrix partitioning F. This may also be minimised to 1 if the PRARE parameters are kept fixed.

For each of the executables mentioned above, typical values for these dimensions are given in the table below.

faust :	S	SD	SP	M	MD	MP	G
MaxType1	100	100	150	750	750	900	6500
MaxType2	1	1500	1	1	10000	1	1
MaxType3	1	1	750	1	1	7500	1
MaxArc	1	1	1	10	10	10	10
MaxExpl	100	100	150	1000	1000	1000	60000
MaxLinPrt	100	3000	1000	1000	10000	10000	60000
MaxXObuf	750	1000	750	5000	6500	5000	7500
MaxPass	200	1750	1500	2000	15000	20000	5000
MaxUserInp	200	200	250	1000	1000	1000	2000

Table E.1 Typical array sizes for various executables of Faust