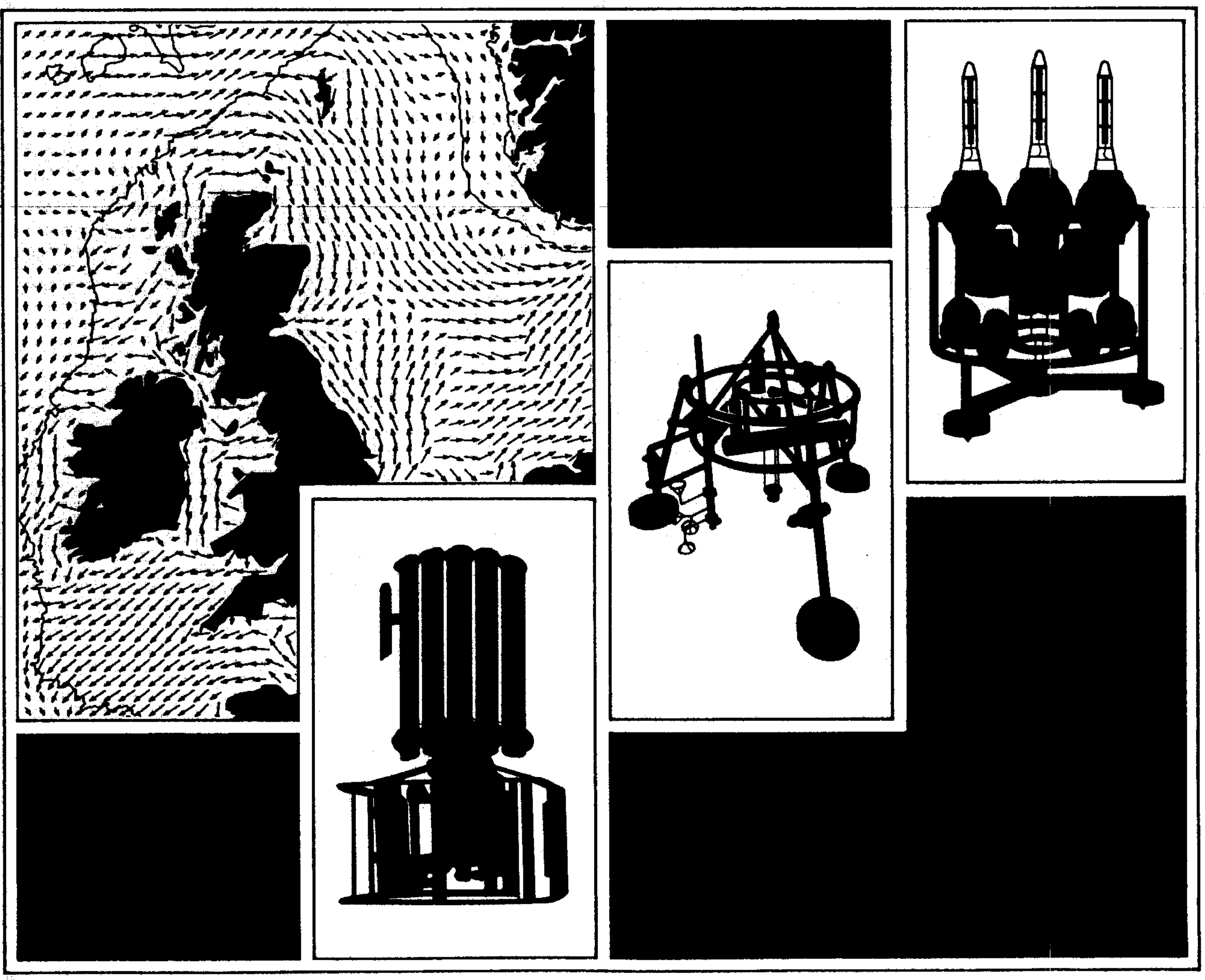
Proudman Oceanographic Laboratory

Global Ocean Tide Inversion: Preliminary Results and Parallel Implementation

OB Anderson, M Ashworth & S Wilkes

Report No. 41 1995



Natural Environment Research Council

PROUDMAN OCEANOGRAPHIC LABORATORY

Bidston Observatory,
Birkenhead, Merseyside, L43 7RA, U.K.

Telephone: 0151 653 8633
Telex 628591 OCEANB G
Telefax 0151 653 6269

Director: Dr. B.S. McCartney

Natural Environment Research Council

PROUDMAN OCEANOGRAPHIC LABORATORY

REPORT No. 41

Global Ocean Tide Inversion:

Preliminary Results and Parallel Implementation.

O.B. Andersen, M. Ashworth and S. Wilkes

1995

DOCUMENT DATA SHEET

AUTHOR ANDERSEN, O.B., ASHWORTH, M. AND WILKES, S.	*PUBLICATION DATE* 1995

TITLE

Global ocean tide inversion: preliminary results and parallel implementation

REFERENCE

Proudman Oceanographic Laboratory, Report No. 41, 52pp.

ABSTRACT

This project was initiated in 1994 to investigate and use the sophisticated methods for ocean tide determination derived by A. Bennett [1992] using global inversion. As distinct from other empirical approaches, this approach uses the fact that the tides are constrained by two different types of information. The first is that obtained from the laws of physics, and the second is the empirical data collected from tide gauges and satellite altimetry. The aim of data assimilation in the tidal model is to derive a solution for the ocean tides that in a mathematical sense best fits the hydrodynamic equations as well as the observational data. The code for running the model is highly parallel and, therefore, extremely well suited for massively parallel computers. It has been run on the CM-200 Connection Machine at the Edinburgh Parallel Computing Centre (EPCC).

For further progress with the method, it is essential to transport the scheme to a more powerful machine. Use of the Cray T3D will enable the model to run with grid sizes of higher resolution, thereby obtaining better spatial distribution of 'representers'. Also the facilities available on the CM-200 at Edinburgh EPCC will soon end. Hence a large part of the project has been the conversion of the code into Fortran 77 and PVM (Parallel Virtual Machine) message passing on the new Cray T3D machine at Edinburgh.

The major difference to the approach taken by Egbert et al. [1994] is that we attempt to derive a hydrodynamic ocean tide model using harmonic constants derived from tide gauge readings AND from observations from TOPEX/POSEIDON, whereas Egbert et al. only used direct altimetric observations from TOPEX/POSEIDON. Egbert et al. [1994] used the direct observations from the TOPEX/POSEIDON satellite at crossover locations and entered these directly into the formulations, whereas we derive the harmonic constituents from the altimetric observations first, and subsequently use these constants to derive the overall ocean tide signal. Thereby, we were able to run one constituent at a time, simply estimating the coefficients for a single constituent each time, which lowered the number of unknowns substantially.

We also used a different set of parameters for the hydrodynamic equations that better matches the local conditions in the North Atlantic Ocean, and particularly matches the conditions in the Northwest European shelf region, as this was our main area of interest. This also entailed the inclusion of enhanced bathymetry, different dissipation parameterisation and the use of a different ocean tide model for the northern boundary of the model grid.

Finally, we attempted to use different resolutions of the model grids: one run having 512 x 256 global elements corresponding to a spatial resolution of 0.7 x 0.6 degrees for the calculation of the representers and also of the final model. Another run, constrained to observations from tide gauges and from TOPEX/POSEIDON altimetry in the Baltic Sea, used a global regular grid with a size of 1024 x 512 elements. This corresponds to 0.35 x 0.3 degrees or approximately 17 x 30 km. For the latter run, we only calculated 25 representers, whereas we for the coarser model we calculated around 400 representers. The approach that we have applied here seems to provide the basis for further improving the accuracy of ocean tide models, especially in the Northwest European shelf region, and a number of scientific communities will be able to benefit from this improvement, especially those that use ocean tide models as a standard correction to be applied before studying other signals related to the ocean.

ISSUING ORGANISATION Proudman Oceanographic Laboratory Bidston Observatory Birkenhead, Merseyside L43 7RA UK Director: Dr B S McCartney	*TELEPHONE* 051 653 8633 *TELEX* 628591 OCEAN BG *TELEFAX* 051 653 6269
KEYWORDS TIDE MODELS DATA ASSIMILATION SCHEMES	*CONTRACT* *PROJECT* MLL-14-2 *PRICE* £15.00

Copies of this report are available from:
The Library, Proudman Oceanographic Laboratory.

Table of contents.

1. Introduction

The ocean tides have a long and extensive history as a subject of scientific research [*Cartwright*, 1977; *Hendershott*, 1981]. With respect to altimetry [*Ray*, 1993] found from an analysis of GEOSAT data that the ocean tidal corrections represented 82% of the total reduction in variance of co-linear differences, resulting from all standard corrections. So it is clear that these large tidal corrections place heavy demands upon the relative accuracy of global tidal models.

Altimetric data from the TOPEX/POSEIDON mission launched in 1992 will be used for studies of global ocean circulation and marine geophysics, but before the data can be used it is necessary to remove the ocean tides which are present in the raw data. *Ray* [1993] concluded that no sufficiently accurate global tidal models exists to satisfy the aims of the mission. So even though we have a good understanding of the physics of the tides, and numerical models have been able for some time to reproduce the majority of the principle features, there is still room for improvement in the modelling of global ocean tides [*Egbert et al.* 1994].

One of the tremendous achievements of the T/P satellite mission has been the release of several ocean tide models based extensively on T/P altimetry [i.e. *Schrama and Ray*, 1994; *Ray et al.*, 1994; *Desai and Wahr*, 1994; *Knudsen*, 1994; *Sanchez and Pavlis*, 1994; *Egbert et al.*, 1994; *Ma et al.*, 1994; *Eanes*, 1994; *Andersen*, 1995]. All of these models provide significant improvements for all constituents, when compared to older models by *Schwiderski* [1980] and *Cartwright and Ray* [1991]. An intercomparison of these recent global ocean tide models has been presented by *Andersen et al.* [1995].

This project was initiated in 1994 by P. L. Woodworth to investigate and use the sophisticated methods for ocean tide determination derived by A. *Bennett* [1992] for ocean tide modelling. As distinct from other empirical approaches this approach uses the fact that the tides are constrained by two different types of information. The first is that obtained from the laws of physics, and the second is the empirical data collected from tide gauges and satellite altimetry. The aim of data assimilation in the tidal model is to derive a solution for the ocean tides that in a mathematical sense best fits the hydrodynamic equations as well as the observational data.

Ole Baltazar Andersen became associated with the project during a seven month during stay at POL (spring, summer 1994), and as initial results showed to be very promising he has continued working on the project after leaving POL. Recently the first preliminary results were presented at the European Geophysical Society, XX General Assembly, Hamburg, 3-7 April, 1995 in the session on remote sensing and modelling of ocean phenomena. The title of the presentation was: "Integrating hydrodynamics and direct observations in global ocean tide inversion." Chapter 2 summarizes the theory involved in global ocean tide inversion as well as development and experience using the code for global inversion for ocean tide modelling with main attention on the North Atlantic Ocean.

The code was kindly provided to P. L. Woodworth by Andrew Bennett, Gary Egbert and Rodney James from the Oregon State University (OSU). We are extremely grateful to all of these authors, as without their help and support this project would not have been possible. We adopted their code, and made numerous changes to code as we used a slightly different approach. However the structure of the code is still identical to the original version provided by OSU.

The code for running the model is highly parallel and therefore extremely well suited for massively parallel computers. It has been run on the Edinburgh CM-200 connection machine at Edinburgh Parallel Computing Centre (EPCC). This version is data parallel code written in Fortran 90, the parallelism is implemented by the use of compiler directives and array syntax.

The code provided from OSU was tuned to run on a CM-5 Connection Machine in USA. As a supercomputer expert Mike Ashworth was involved in the project as to guide, tune and implement the code for running on the Edinburgh CM-200 connection machine at Edinburgh Parallel Computing Centre (EPCC). For further progress with this Global Tidal Model it is essential to transport the scheme to a more powerful machine. Use of the Cray T3D will enable the model to run with grid sizes of higher resolution, thereby obtaining better spatial distribution of representers. Also the facilities available on the CM-200 at Edinburgh EPIC will soon end. Hence a large part of the project has been the conversion of the code into Fortran 77 and PVM (Parallel Virtual Machine) message passing on the new Cray T3D machine at Edinburgh.

This part of the project has been undertaken by S. Wilkes (during autumn 1994) as partial fulfillment of his degree of Master of Science in advanced scientific computation in the faculty of science at the University of Liverpool. Chapter 3 report a summary of the results from his masters thesis on the parallel implementation of the representer code on the Cray T3D

The initial aim was to derive ocean tide models constrained to the North Atlantic ocean using updated code on the facilities on the Cray T3D. However, during the final stage of the project, it was revealed that the Cray T3D was very heavily booked. It was therefore decided to run the North Atlantic model using the CM-200 connection machine.

The status of the project is presently that all authors have left Bidston to work elsewhere. However, this does not mean that the project will not be continued, as at least Ole B. Andersen hopes to continue to work on ocean tides within the frame of this project. The results presented are convincing, and demonstrate the high potential of the method. New ocean tide models with different resolution and for different areas of the North Atlantic are presented. These models are comparable in accuracy to the widely used local model for the area. However, these results clearly demonstrate that we can make much more progress using the method, but that it is essential to use higher resolution in the ocean tide models in order to achieve that goal. Several suggestions to the practical implementation of this are presented, so we hope to be able to improve both the technique as well as the resulting ocean tide models using the facilities available on the Cray T3D machine at Edinburgh.

The approach that we have applied here seems to provide the basis for further improving the accuracy of ocean tide models especially in the Northwest European shelf region, and a number of scientific communities will be able to benefit from this improvement, especially those, that use ocean tide models as a standard correction to be applied before studying other signals related to the ocean. I.e. altimetrists, geodesist and oceanographers. Furthermore, inaccuracies in the applied ocean tide models have recently been seen [*Baker et al.*, 1995] to influence loading computations, which again can be seen to influence the determination of very accurate gravity measurements as well as GPS measurements.

Copenhagen, 16 June 1995

Ole Baltazar Andersen
Author for correspondence

Affiliation:

Ole Baltazar Andersen is presently at:
National Survey and Cadastre (Kort- og Matrikelstyrelsen)
Rentemestervej 8, 2400 Copenhagen NV, Denmark
Phone: +45 3587 5255, Fax: +45 3587 5052,
E-mail: ole@kms.min.dk, WWW URL: http://www.kms.min.dk

Mike Ashworth is presently at:
Deutsches Klimarechenzentrum (DKRZ) (German Climate Research Computing Centre)
Bundesstrasse 55, D-20146 Hamburg, Germany
Tel: +49 40 41173 395, Fax: +49 40 41173 270,
E-mail: ashworth@dkrz.d400.de, WWW URL: http://www.dkrz.de

Stuart Wilkes adress:
Springfield, Lyonshall, Kington
HR5 3LT Herefordshire

2. Global ocean tide inversion using Bennetts method (O. Andersen)

The tides are constrained by two different types of information. The first is that obtained from the laws of physics, as the second is data collected from tide gauges and satellite altimetry. Prior to the launch of the TOPEX/POSEIDON satellite, data was limited to sea surface height measurements from coastal tide gauges as well as a small number of pelagic tide gauge readings. By themselves the coastal tide gauges are of limited use for global ocean tide modelling due to the complexity of the tides in coastal regions. However, *Schwiderski* [1980a, 1980b] developed a hydrodynamic interpolation scheme in which the estimated harmonic constants were used as a boundary condition for the numerical model derived using the dynamic equations. Imposing these extra boundary conditions required local adjusting of the bottom-friction coefficient and relaxation of the no-flow boundary conditions at the coast.

Due to the nature and origin of the code and method, parts of this description follow closely the method of Egbert et al. and hence to what they reported in *Egbert et al.* [1994]. However, the theory will be briefly reviewed in the context of the present analysis with main attention to the differences in the two approaches.

Hydrodynamic equations

The ocean tides arise from the influence on the ocean from the sun and the moon. The ocean tide field at a position (ϕ,λ) is in the following denoted $u(\phi,\lambda)$. The ocean tide field is a complex vector field consisting of a number of constituents $u_i(\phi,\lambda)$ like

$$u(\phi,\lambda) = |u_1(\phi,\lambda),\dots\dots,u_n(\phi,\lambda)| \quad where \quad u_i(\phi,\lambda) = \begin{vmatrix} u_x(\phi,\lambda) \\ u_y(\phi,\lambda) \\ h(\phi,\lambda) \end{vmatrix} \tag{1}$$

where $u_x(\phi,\lambda)$ and $u_y(\phi,\lambda)$ are the transport components in the north and east directions respectively and $h(\phi,\lambda)$ is the elevation. It must be noted that all individual constituent fields that make up the ocean tide field $u(\phi,\lambda)$ are complex vector fields - this way 3n complex values are required to completely describe the ocean tide field at each location, with the total ocean tide field arising from n constituents. Solving for the major four constituents (M_2, S_2, K_1, O_1) would then give a total of 12 complex parameters to be described at each location (ϕ,λ).

The code provided to P. L. Woodworth by Andrew Bennett, Gary Egbert and Rodney James from the Oregon State University (OSU) allows a combination of all observational and hydrodynamic information into global tidal fields best-fitting both the data and the dynamics in a least squares sense [*Bennett,* 1992].

The linearized shallow water equations are typically assumed for tidal modelling with parametrized corrections for dissipation, tidal loading and ocean self attraction. To a good approximation the tidal fields $u(\phi,\lambda)$ at a frequency ω satisfy the linearized shallow water equations, for the effects of open ocean self attraction and tidal loading [*Hendershott,* 1981] subject to boundary condition on ∂O

$$Su = f_o \ \wedge \ \begin{vmatrix} u.n = 0 \ on \ \partial O_{coast} \\ h = h_o \ on \ \partial O_{open} \end{vmatrix} \tag{2}$$

where O is the full domain represented by the global oceans, f_o is the astronomical forcing and where the indices for position (ϕ,λ) have been omitted for convenience.

The model has two types of boundaries ∂O. The rigid boundaries along coastlines ∂O_{coast} and the open boundaries along the (northern) edge of the model domain ∂O_{open}. The coastline boundary condition is no flow normal to the coast, as $n = (n_x, n_y, 0)$ is the direction normal to the rigid boundary. Along the open boundary the conditions are specified through the elevation at each grid cell using an existing ocean tide model (e.g. the model of Schwiderski).

The Laplace tidal operator S for linearized shallow water equations is given by

$$S = \begin{vmatrix} i\omega + K & -f & gHG*\nabla_x \\ f & i\omega + K & gHG*\nabla_y \\ \nabla_x\cdot & \nabla_y\cdot & i\omega \end{vmatrix} \tag{3}$$

where H is ocean depth, f is the Coriolis acceleration, ω is the constituent frequency, $\nabla.$ and ∇ represent the two dimensional divergence and gradient operators on the spherical earth, K represent dissipation and $G*$ represents convolution with the Green's function for tidal loading and ocean shelf attraction [*Hendershott*, 1972].

Selected model parameters for the hydrodynamic equations

The forcing parameter f_0 for the astronomical potential given by *Cartwright* [1993] was used. This parameter includes a correction for earth tide potentials. Convolution with the Green's function for tidal loading and ocean shelf attraction $G*$ was treated using the simple scalar correction factor $\sigma = 0.9$ as also used in a number of other hydrodynamic models (e.g Schwiderski). It must be noted that this scalar approximation is not the same as the loading correction used to correct the altimetry - for altimetry the loading correction is simply the elastic deformation due to the height of the water masses.

The bathymetry is obtained from the DBDB5 data base, which is notoriously inaccurate on the shelf and near the coast. However the data base was supplemented by the bathymetric data from the northwest European shelf bathymetry data base with the kind help of R. Flather. This way a much more reliable bathymetric model for the area of main interest was used.

Dissipation parameters for the hydrodynamic equations.

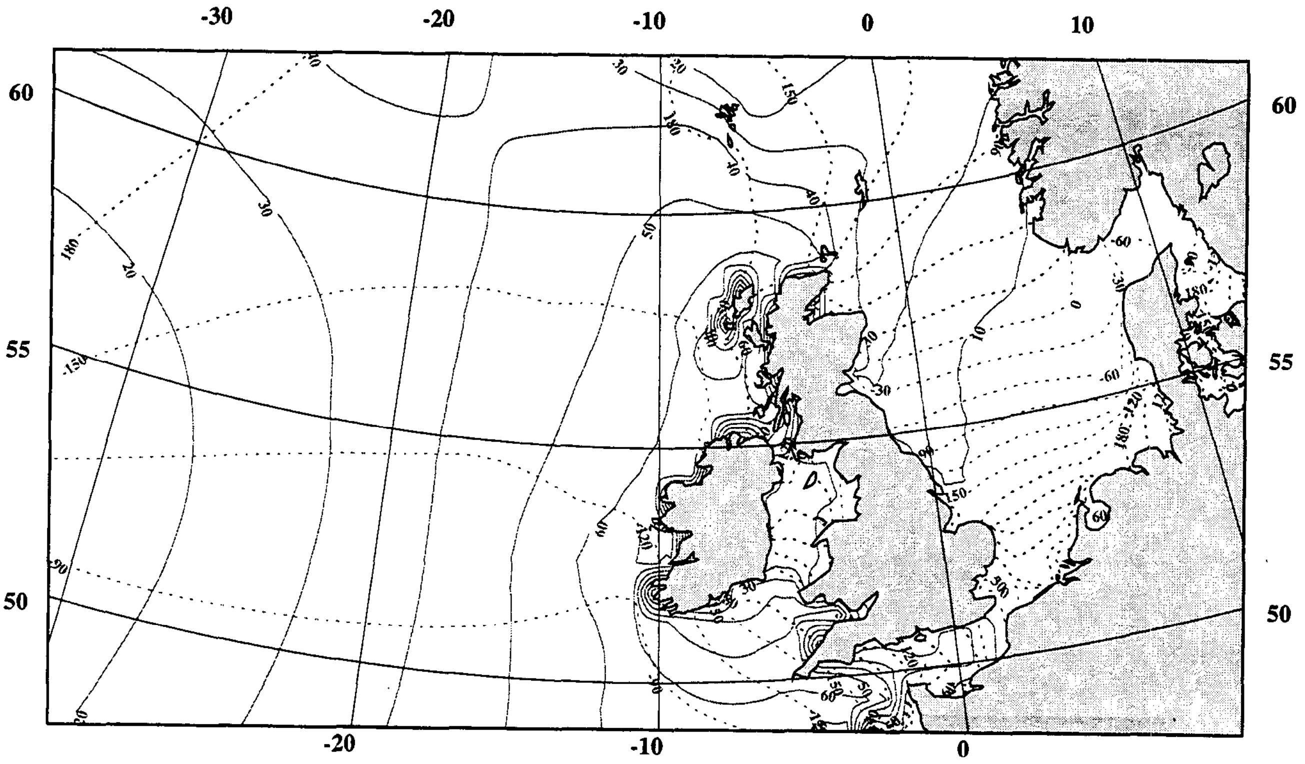
Dissipation is a crucial parameter, and a wide range of formulations have been suggested in the literature. Here we have investigated the use of two simple parametrizations. The first of these uses a linear relation between bottom dissipation and bathymetry, using a linear decrease in dissipation with depth like

$$K = K_0 / \max(H, H_0) \tag{4}$$

This parametrization is defined by the drag coefficient K_0 and a minimum depth H_0 above which the dissipation is defined to be constant. The parameters chosen by *Egbert et al* [1994] was (K_0 = 0.3 and H_0 = 300 meters), where Schwiderski used quite a different set of parameters (K_0 = 0.1 and H_0 = 20 meters). *Egbert et al* [1994] chose their parameter set because they obtained the overall best result when compared to a global set of 80 tide gauges. However, all of the gauges that they compared to were located in the deep ocean. Hence, their choice of parameters might not be advisable for the shallow waters on the northwest European shelf. In order to investigate which set of parameters that best fits the local conditions it was decided to test several set of parameters. Two forward models for the M_2 constituent were derived using the two different dissipation parameter set above. These solutions were computed to satisfy the Laplace tidal equations (Equation. 3), which were solved by time stepping forward in a an Arakawa C grid with the mentioned parameters. The different solutions are shown in Figure 1. In both Figure 1 and Figure 2 solid lines represent amplitudes and dashed lines phase lag with respect to Greenwich. The contour interval for the amplitude is 10 cm. For the phase the contour interval is 30 degrees.

The two figures are relatively similar in the deep ocean, which they naturally should be. However, they becomes very different in the shelf area. Especially the parametrisation used by Egbert yields an ocean tide model for the M_2 constituent which is very far from what is predicted by existing local ocean tide models. The amphidrome point located in the centre of the North Sea and predicted by most ocean tide models is not present in any of the models using the linear decrease in dissipation with depth. Furthermore the amplitude of the ocean tide signal is a factor of 2 - 4 too small compared to existing models in most parts of the shallow North Sea. Especially along the east coast of the UK these models are off by more than 1.5 meters of amplitude for the M_2 constituent.

These results might be improved by still keeping the linear formulation (Equation. 3) but using a quadratic



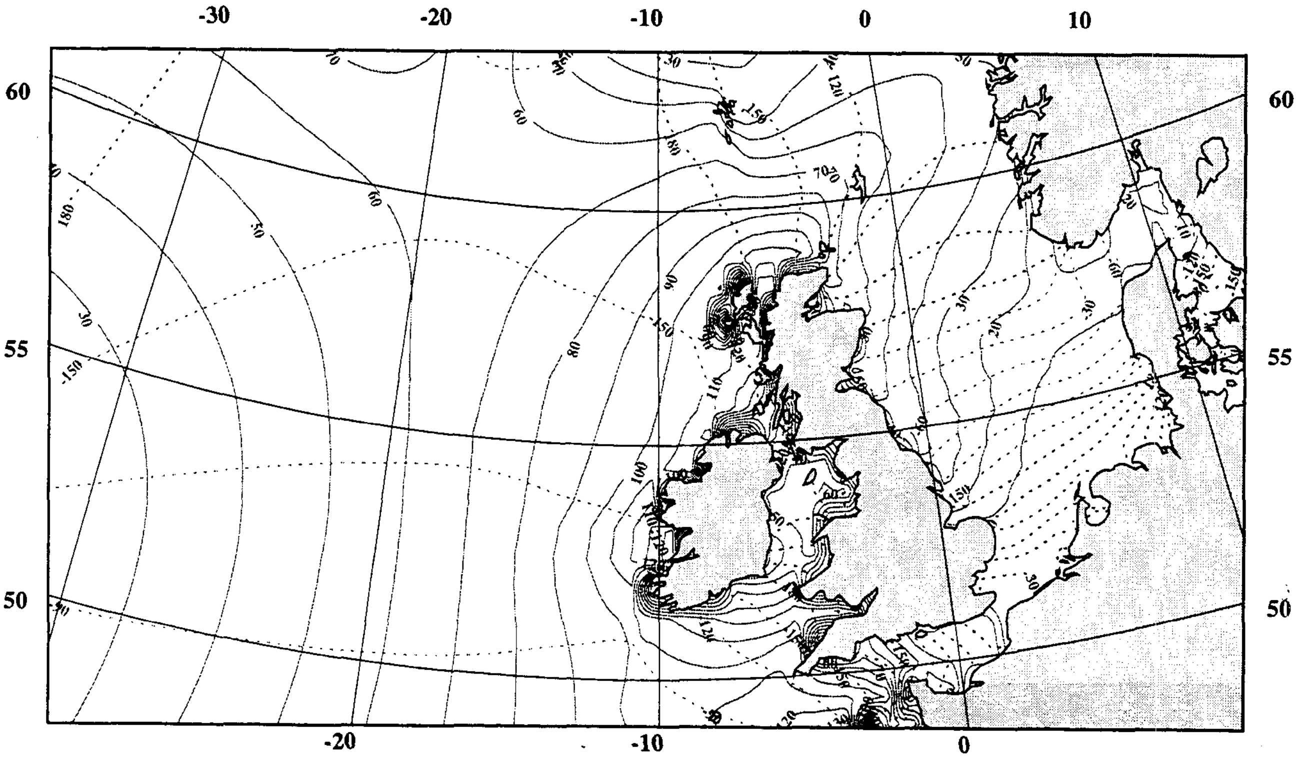


Figure 1 M_2 constituent model derived with different parameter set using linear relation between dissipation and depth. Top: K_0=0.3, H_0=300 m, bottom: K_0=0.1, H_0=20m

relation between bottom dissipation and bathymetry. This time the dissipation decreases with the square of the depth which will give a much higher change in dissipation in shallow waters than with the linear relationship. This relationship looks like

One parameter choice was tested for the quadratic approach as well namely the drag parameter K_l = 90.0

$$K = K_1 / (\max(H, H_0))^2 \qquad (5)$$

and the minimum depth of H_0 = 900 meters. This set of parameters would yield almost constant dissipation in most Northwest European shelf region, but some drastically changes in dissipation around the shelf.

The result using the quadratic relation between bottom dissipation and bathymetry is shown in Figure 2. This set of parameters yields a considerably more realistic result for especially the North Sea region. An amphidrome is now present, and relatively high tidal amplitudes are seen along the coast of the UK. However the result is still considerably different to what is predicted by local models in the coastal regions.

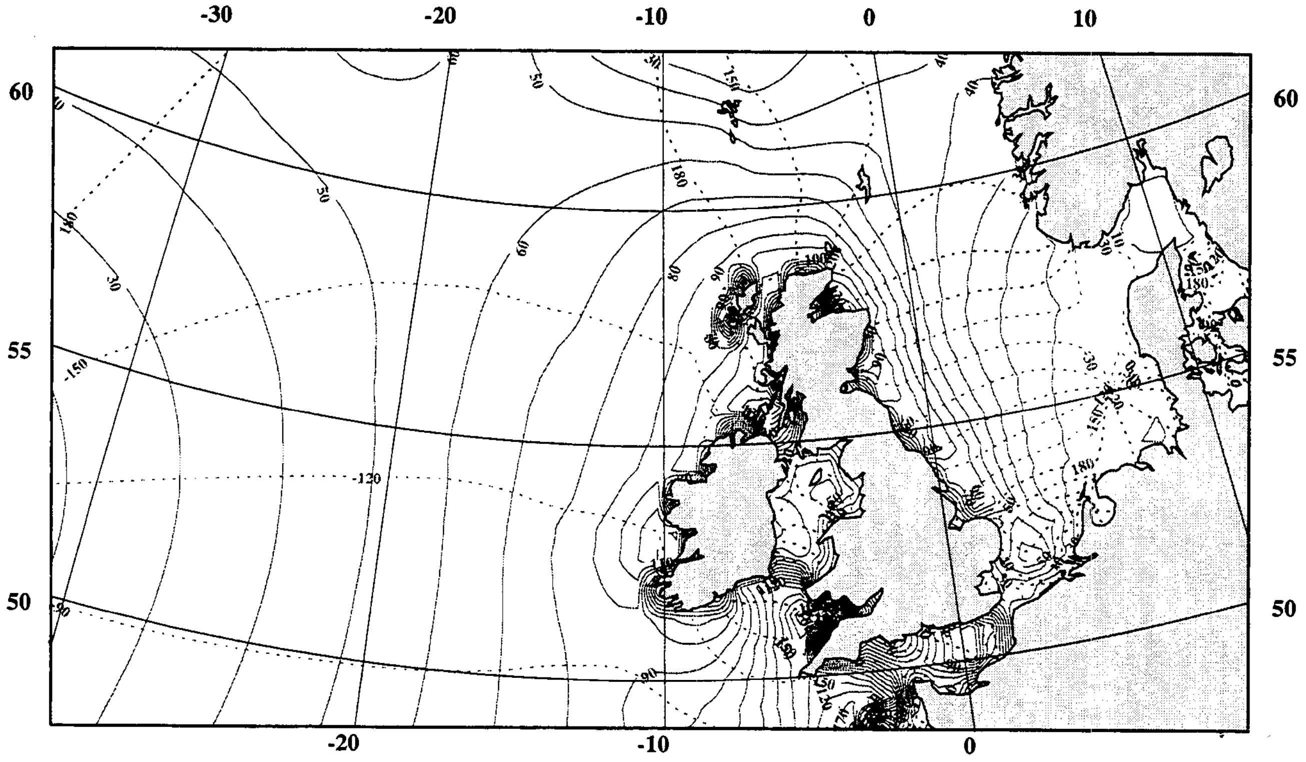


Figure 2 M_2 constituent model derived using a quadratic relationship between dissipation and depth. K_0 =90.0, H_0 = 900 m

These models differ greatly from each other, as well as from the FES94.1 model from Grenoble presented in Figure 8 [*Le Provost et al.*, 1994]. The FES94.1 version of the Grenoble model has recently been shown by *Andersen et al.*, [1995] to be a very accurate model for the Northwest European shelf region, so it is worrying that these forward models are so different. Another way to examine the forward model is by looking at the maximum amplitude reached globally.

The maximum amplitude of the M_2 constituent varies from a maximum of 1.2 meters for the model that uses linear relationship between dissipation and depth with constant dissipation above 300 meters (on the shelves) to 2 meters for the model that uses linear relation between dissipation and depth for depths greater than 20 meters up to 4.5 meters for the model that uses quadratic relationship between dissipation and depth. The latter is the most reasonable as tides having that amplitude is seen on the Patagonian shelf..

The obvious choice seems therefor to be to use the quadratic relationship between dissipation and depth. However, of some reason the linear relationship between dissipation and depth with (K_0 = 0.1 and H_0 = 20 meters) was chosen as this seemed to be the best choice for both deep and shallow waters, as it looked more realistic in the deep ocean. However, it is very worrying that the forward model using this set of dissipation and minimum depth parameters (Figure 1) has amplitudes, that are so far from the amplitudes predicted by other ocean tide models especially for the northwest European shelf region.

Observations.

The two types of information of the tidal state, that we have, can be represented like,

$$\begin{vmatrix} S \\ L \end{vmatrix} u = \begin{vmatrix} f_o \\ d \end{vmatrix} \tag{6}$$

Here our knowledge about the hydrodynamics is included through the $Su = f_o$ equation and the knowledge from observations comes from the equation $d = Lu$, where d is the data vector and L is a measurement functional on the tidal field.

While it is relatively easy to find a model, that will satisfy the hydrodynamic equation part of the equations, it is much harder to find a solution that will fit the full set of equations.

In the present analysis the only observations are height observations from tide gauge or satellite. L can include all sorts of observation of the tidal field u (ϕ,λ) like sea surface height measurements or readings from current meters. However, in the preceding analysis we have only sea surface height measurements from satellites or tide gauges. In this case the measurement functional L is equivalent to a height measurement described by

$$d = L(u)+\varepsilon \tag{7}$$

where d is the measurement vector and ε is an additive measurement noise. For the present analysis the measurements were the harmonic constituents observed from tide gauge sites or harmonic constituents derived from measurements at TOPEX/POSEIDON crossover locations.

Pelagic and Coastal Data Processing

Pelagic and Coastal data was obtained from the data banks at POL and kindly provided by D.Blackman and *Smithson* [1992] as harmonic constants for all major ocean tide constituents.

Actually we do not need the harmonic constants of the full ocean tide for the estimation of representer coefficients in Equation 13. Here we rather need the residuals to a prior model like $d' = d - Lu_0 \ (+ \varepsilon)$ For our observations this correspond to using the residual ocean tide signal rather than the full ocean tide signal like

$$\delta h_{ot} = (h_{observed} - h_{ot}) \tag{8}$$

Data were therefore reduced using an apriori ocean tide model. The choice of the apriori model is discussed in a later section. Harmonic constants for the residual ocean tide signal were subsequently used as data vector d. As standard deviation a value of 1 cm was used for all observed constituents.

TOPEX/POSEIDON Data Processing

The observations comprising the residual elastic ocean tide heights δh_{eot} used in the subsequent analysis are computed from TOPEX altimetric observations (h_{ssh}) using the following expression

$$\delta h_{eot} = h_{ssh} - (h_{sensor} + h_{eot}) - (h_{OSU91A} + h_{qssh}) - h_o \tag{9}$$

where h_{sensor} is the sum of altimetric sensor effects, h_{OSU91A} is the OSU91A geoid model [*Rapp et al.*, 1991], h_{qssh} is the quasi-stationary sea surface height, and h_o is orbit error corrections. h_{eot} is the correction for the chosen apriori elastic ocean tide model for the correction of pelagic and coastal data above. The elastic ocean tide models were derived from an initial ocean tide model using loading tide computations based on the adjusted grenoble (AG94.1) model by *Andersen* [1995] and kindly computed by R. Ray at NASA-Goddard.

Standard corrections for the altimeter sensor effects (electromagnetic bias, ionosphere, wet and dry troposphere) were applied using values provided in the GDR [*AVISO*, 1994], including a 100% inverse barometer correction with a mean pressure of 1013.3 mbar [*Callahan*, 1993].

Data were subsequently analysed using the response method [*Munk and Cartwright*, 1966] with the orthotide extension by *Groves & Reynolds* [1975] for modelling the residual tidal heights which can be represented as

$$\delta h_{eot}(\phi,\lambda,t) = \sum_{k=0}^{k=2} [u_k(\phi,\lambda)P_k(t)+v_k(\phi,\lambda)Q_k(t)] \tag{10}$$

where $P_k(t)$ and $Q_k(t)$ are the nearly orthogonal orthotide functions and u_k (ϕ,λ) and v_k (ϕ,λ) are the unknown coefficients of the orthotide functions. Full definitions of the parameters involved and orthotide coefficients as well as transformations between admittance functions and leading harmonic coefficients can be found in *Cartwright and Ray* [1990].

Finally the derived harmonic constituents for the residual elastic ocean tides signal were converted into residual ocean tide harmonic constituents using a loading value of -6 % [*Ray and Sanchez*, 1989]. This loading correction is sufficient accurate as only ocean tide residuals were to be studied. Harmonic constituents for the residual ocean tide signal were subsequently used as input in Equation 13. As standard deviation a value of 1 cm was used for all constituents similar to the accuracy of the tide gauge derived harmonic constants.

Approach

The major differences to the approach taken by *Egbert et al.* [1994] is mainly that we attempt to derive a hydrodynamic ocean tide model using harmonic constants derived from tide gauge readings AND from observations from TOPEX/POSEIDON, where Egbert et al. only used direct altimetric observations from TOPEX/POSEIDON. *Egbert et al* [1994] used the direct observations from the TOPEX/POSEIDON satellite at crossover locations and entered these directly into the formulations, whereas we derive the harmonic constituent from the altimetric observation first, derive the *residual* harmonic constants, and subsequently use these residual harmonic constants to derive the residual ocean tide signal.

The two approaches have different advantages. By using the direct observations as by *Egbert et al* [1994] they had to take inter-constituent correlation into account and solve for several constituents simultaneously. This approach is very sophisticated, and is the most correct approach, because inter-constituent correlation exists and, hence, should be modelled. However, when solving for several constituents the number of unknown increases and in order to reduce the number of unknown they had to use a linear interpolation of the admittance function in order to resolve for several constituent simultaneously. The use of linear admittance relationship is a very crude approximation, but they were Egbert et al were forced to use it in order to limit the number of parameters entering the subsequent inversion.

In our approach we did not attempt to use inter-constituent correlation, or more precisely, we assumed it to be zero which is an approximation. This could be achieved by estimating harmonic constants from the direct observations from the TOPEX/POSEIDON and entering this into the global inversion scheme along with harmonic constants from tide gauges. Thereby we were able to run one constituent at a time and just estimate the coefficients for one constituent which lowered the number of unknowns substantially to 3 complex parameters at each location.

We also used different set of parameters for the hydrodynamic equations that better matches the local conditions in the North Atlantic Ocean and particularly matches the local conditions in the northwest European shelf region as this was our main area of interest. This also entailed the inclusion of enhanced bathymetry and different dissipation parametrisation and the use of a different ocean tide model for the northern boundary of the model grid.

Finally we attempted to use different resolutions of the model grids - both a relatively coarse model with a resolution equal to the resolution of the model by Egbert at al [1994], but also testing the technique for a model with higher resolution.

Finding the improved solution.

The inverse approach attempts a minimisation of a quadratic penalty functional or equivalent the minimisation of a cost function like

$$P(d,U) = (d-L(U))^t C_e^{-1}(d-L(U)) + (\iint d\phi d\lambda (TU-f)^t C_f^{-1}(TU-f)) \qquad (11)$$

where C_e are data error covariance matrices and C_f and C_e are the covariances of the forcing error and the observations respectively.

The use of this penalty functional enables a normalisation of the two sources of errors. However, it also enables a trade-off between fitting the dynamics and fitting the data, as the observations and the dynamics can be weighed through the choices of C_f and C_e.

Following Bennett [1992] it can be shown that the weighted least-squares fit to data and dynamics found by minimising P is essentially the minimum variance best linear unbiased estimator (BLUE estimator) of the tidal field, which is also known as the Gauss-Markov smoother. If the dynamical and data errors are furthermore assumed to have normal distribution then the estimate is also the maximum-likehood estimate.

In order to solve this generalised inverse problem we minimise P. In theory minimisation of P is straightforward, and after discretisation of the model we have a large linear least squares problem. However if the ocean tides are to be represented with reasonable resolution, the number of unknown parameters becomes prohibitively large. For instance, a reasonable model with a resolution of 0.7 degrees would entail a global model with [512 x 256] grid points as shown in Figure 5. If we were to solve for one constituent this would requires 512 x 256 x 2 $\approx$ 250.000 parameters. If we furthermore want to run the high resolution model as shown in Figure 6 or Figure 7 then the number of unknown easily exceeds one million parameters.

We will therefore take the Hilbert space approach derived by *Bennett* [1992] for oceanography. This approach uses a Hilbert space approach similar to that which is used extensively in solid earth geophysics [*Parker et al*, 1987]. The derivation is based on generalised inversion using the Euler Lagrange approach and can be seen in detail in *Bennett* [1992] section 5.3 - 5.4. The following equations are also reported in *Egbert et al* [1994], however they are essential to describe the nature of the representer fields in the following section.
First of all we define the full space of possible tidal states by $\Im$. This means that any tidal field $u(\phi,\lambda)$ is in this space $\Im$. The measurement functional or "point elevation" L of the tidal field is therefore a functional acting on the elements in this state space.

The Penalty functional is a positive definite quadratic form. It can therefore be used to define an inner product on the state space $\Im$ like

$$< u_1, u_2 > = \int_o [Su_1](x)^* [C_f^{-1} Su_2](x) d^2x \qquad (12)$$

We hereby also have a norm. The subset for which $<u,u> < \infty$ (corresponding to weak assumption on C_f) forms a Hilbert space. For each functional (reasonable) in this Hilbert space there exists an element r_i in the state space $\Im$ so that

$$L_i|u| = <r_i, u> \qquad (13)$$

The element r_i is called the Riez representer of the measurement functional L (Note that in our terminology L is our evaluation functional of the ocean tide field - above for constituent i). Following *Bennett* [1992] P attains its minimum for a unique element in this state space. This element has the form

$$u = u_0 + \sum_{k=1}^{K} b_k r_k \qquad (14)$$

where u is the complete description of the tidal state, b_k are the coefficients of the representers r_k, and K is the total number of representers. u_0 is the model satisfying the Laplace tidal equation subject to rigid boundary conditions. This model is in the following often referred to as the *initial model* or the *apriori model* satisfying the Laplace tidal equations. The choice of u_0 is crucial and the subject of a section to follow.
The coefficients b for the representers satisfy the K x K system of linear equations

$$(R+C_\varepsilon)b = d - L[u_0] \tag{15}$$

where R is the representer matrix which has elements derived from interpolation in the representer fields r_k like

$$R_{ij} = <r_i,r_j> = L_i[r_j] \tag{16}$$

Hence, this approach has massively reduced the number of unknowns to its real dimension which is equivalent to one "set of parameters for the total ocean field" for each (independent) observation. But all of our observations are independent as we use harmonic constants as input. The so called "set of parameters " relates to the number of parameters, which equals three complex coefficients for each constituent solved for. This is really a massive reduction of the apparent dimension of the inverse problem to its true dimension. Where the straight forward minimization was a prohibited task then the inversion of the representer matrix is possible within the limitations of many computers. However the calculations involved are still quite formidable because we have to compute the representers and their coefficient.
These calculations can be split into three separate sections.

1) Calculation of the representer fields r_k,
2) Estimation of the coefficients of the representers (b_k)
3) Finding the improved solution u.

Calculation of representer fields r_k,

The first task is by all means the most computational intensive task. It has therefore been the aim of the implementation on the Cray T3D as it is highly parallel. The description of routines associated with this part of the code can be found in detail in chapter 3.

Representers can be computed by solving the following set of equations, which are based on equation (II.4 - II.15) in *Egbert et al.* [1994]

$$S^{+}\alpha_k = \Delta_k \tag{17}$$

$$S\bar{r}_k = C_f{*}\alpha_k \tag{18}$$

where the operator S^+ in the first equation is the adjoint operator to S for the Laplace tidal equations and is impulsive forcing at a chosen location (ϕ_k,λ_k).

The way this system of equations is solved is as follows: First, Equation 17 can be solved by stepping backward using the hydrodynamic equations using the adjoint operator S^+ with the forcing Δ_k applied at the location of the observation, for which the representer is to be calculated.
Thus, one yields the adjoint solution α_k for Equation 17. This adjoint solution α_k is then smoothed with dynamical covariance function C_f*. This yields the inhomogeneous boundary conditions for equation 18, as well as forcing parameters. Finally equation 18 is solved with respect to the representer field r_k by stepping forward from the inhomogeneous boundary conditions.

The two equations were solved using time stepping on an Arakawa C grid (Se chapter 3) by applying periodic forcing and then explicitly time stepping from homogenous or in-homogenous initial conditions. The solution was in both cases achieved in 10000 time steps using a step length of 50 seconds. This corresponds to letting the model run for 500.000 seconds which is equivalent to around 6 days, and which should be adequate for obtaining a reasonable stable accurate solution. The step length is considerably shorter than what has been reported by other authors like Egbert et al, 1994 or Gjevik and Straume, [1989]. The reason for such short time step is the fact that the northern model boundary is placed along the 82°N parallel. At this latitude the grid spacing in the longitude direction is reduces by a factor of 7 compared to the grid spacing at Equator. Hence, in order to keep the system stable, a very short time step was needed.

The solution obtained from Equation 17 is to be smoothed using the dynamic error covariance function so as to give the inhomogeneous boundary conditions. This smoothing is described like

$$\Psi * u = \int \Psi(x,x')u(x')d_2x' \tag{19}$$

Such convolution is a rather time-consuming task even on a parallel computer. This has to do with the fact that each grid point which is to be smoothed is influenced by <u>every</u> grid point in the array. This arises from the integration term which requires the contribution from all cells in the grid to be estimated, and hence, it requires communication with all points in the grid.

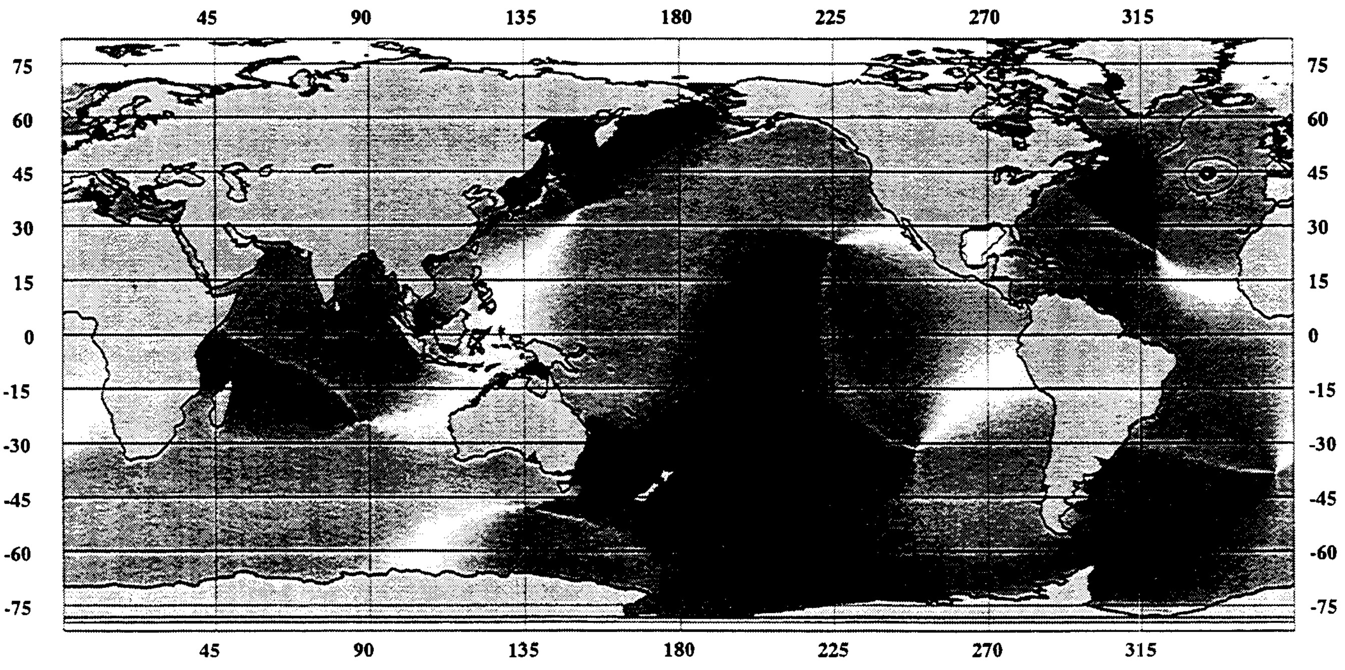
However Egbert et al [1994] suggested that an iterative approach was used to parallelise the approach. This approach substitute the summation/integration in Eq. 19 with an iterative use of a local smoother or a repeated multiplication like

$$\Psi * u = (I + \gamma D)^N u \tag{20}$$

The theory of this smoothing approximation is described in appendix B of *Egbert et al* [1994] The idea is iteratively to smooth or convolve the solution by repeating local smoothing. Local smoothing only requires communication with eight nearest points, which makes it far more suited for parallel computers and hence much faster to perform. By repeating the process, communication gradually progresses further away as also the nearest point is smoothed by the second to nearest points and to forth. A convolution with a correlation length of 500 km can be done in 250 iterations, using $\gamma = 0.01$ to maintain the stability of the process.

Representer fields r_k,

For each (independent) observation we need to calculate one representer field. An example of such an representer field is shown in Figure 3, which is a plot of the amplitude and phase of the height component of the representer field.



Representer solution for K1, Contour interval = 2 mm

0.0 ▨ **360.0 Degree**

Figure 3 is located on previous page

Figure 3 Example of a representer field (K_1 constituent) - elevation component. Forcing is applied at the dot in the North Atlantic. Contours indicate amplitudes (Interval is 2 mm). The cyclic greyscale displays phases.

Generally the representers are very broadly peaked with a maximum in the vicinity of the location of the applied forcing dependent on the local dynamical conditions. It is also evident that the representers have relatively large amplitude throughout the basin where it is located. The representer fields are actually triplets with two transport components in addition to the shown elevation component.
The representer can be understand in both a physical sense and a mathematical sense.

The physics of the representer fields are relatively hard to decipher, but in a sense the representer fields displays how the total model field reacts to an applied forcing with the chosen frequency and location. The representer field in Figure 3 thus indicate how an applied forcing at the location of the dot will result in a global field with the shown elevation and phase components.

However, the representers can also be viewed as covariance functions for the dynamical model. This can be argued as follows. As the evaluation potential is bounded *Moritz* [1989] showed, that the Hilbert space actually has a reproducing kernel [See also *Tscherning*, 1986] If the Hilbert space has a reproducing kernel *K(x,y)* [*Moritz*, 1989] then the representer for the functional *L* has a very simple form

$$r(x) = L^y K(x,y)$$

where L^y denote that *L* acts on the reproducing kernel as a function of y. This reproducing kernel might be interpreted as covariance function for the dynamical residuals. The representer will be have its maximum at or near to the location of the representer depending on the local conditions. The representer in figure 3 shows the K_1 constituent. Moving away from the location of applied forcing the phase changes and the amplitude drops, which again is influenced by the local conditions.

Similarly the representer matrix R where

$$R_{ij} = <r_i, r_j> = L_i[r_j] \tag{22}$$

may then be viewed as the covariance of $L[u-u_0] = d-L[u_0]$ ($= Rb$) in the absence of measurement errors. This means that R contains information on how the dynamic errors affect the measurement deviations from our prior model u_0.

Estimation of coefficients for the representer fields (b_k)

The estimation of the coefficients (b_k) of the representers are found from Equation 15 where the elements of the representer matrix are found using Equation 16. This inversion might also involve some quite heavy computations [*Egbert et al.*, 1994] if the number of unknowns is large.

However by assuming no inter-constituent correlation each constituent can be treated independently which limits the number of unknowns by a factor of around 10 compared to that which was used by *Egbert et al.* [1994]. The number of unknowns is then further reduced by constraining the solution to the North Atlantic. This does in practice mean that the density of representers is much higher in the North Atlantic than in any other ocean of the world which limits the number of representers drastically. These reductions lead to a number of unknowns which does not exceed 1000. The inversion could therefore be treated using standard inversion software as implemented in the GRAVSOFT software package [*Tscherning et al.*, 1994].

In order to reduce the number of unknowns it is often very informative to have a look at the singular value decomposition (SVD) of the representer matrix. performing a SVD can be viewed as a rotation of the data vectors and the measurement functionals. Subsequently the inverse solution may be expressed as where r* and d* are the rotated functionals, and λ and σ^2 are the eigenvalues of the representer matrix and the variance of the measurement errors respectively. It is obvious that if $\lambda << \sigma^2$ then the eigenvalues are much smaller than the measurement noise and this datum can safely be ignored.

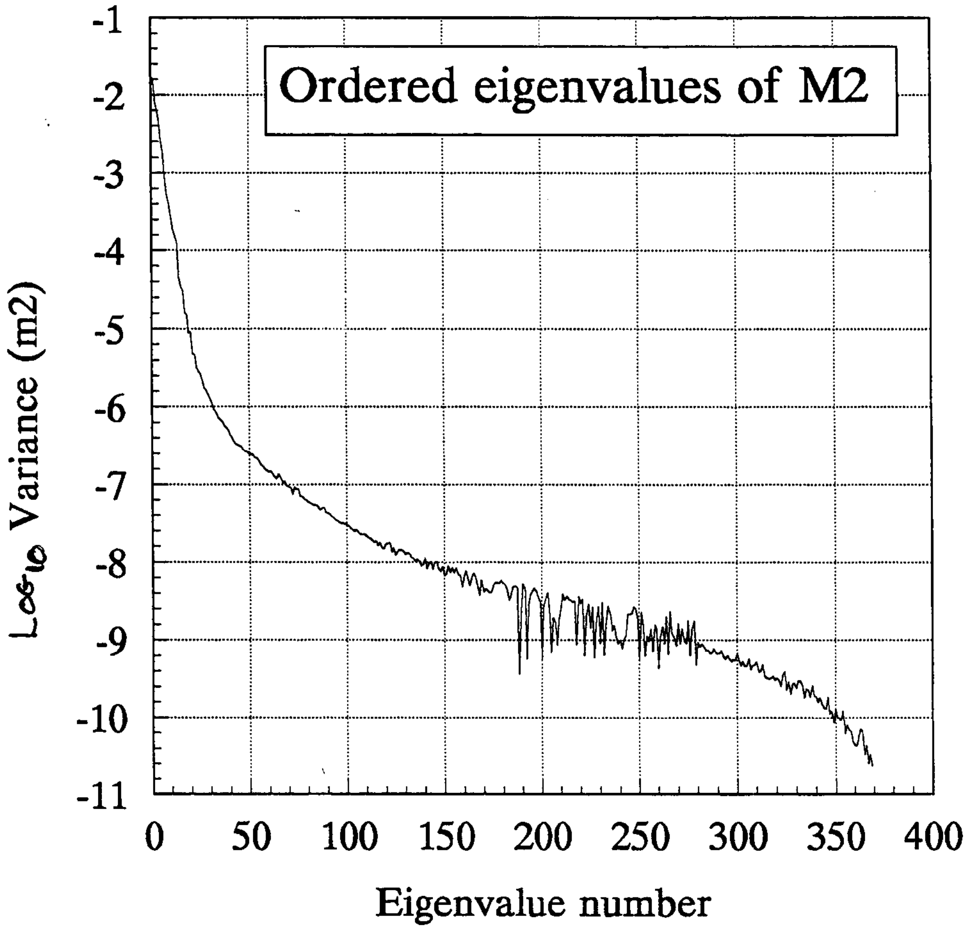


Figure 4 Ordered eigenvalues of the representer matrix for the M_2 solution from 372 representers

This practically means, that the summation can be truncated at a level *K* with out major loss of accuracy. The solution estimated from the truncated set of eigenvalues using Eq. 23 is often referred to as the Regularised least squares solution of the problem. For our solution the ordered eigenvalue spectra can be seen in Figure 4 The eigenvalues were determined from the representer matrix for the M_2 constituent. It is obvious that the summation can be truncated at a much lower number than 400 - actually a truncation level of 50 will probably be sufficiently accurate with the chosen parameters for our hydrodynamic model. Further discussion on the tradeoff problems between the number of eigenvalues and the accuracy of the final ocean tide model is presented in Egbert et al., [1994].

Finding the improved solution *u*

From the estimated coefficients the improved solution can finally be obtained in two ways.
One is through a direct summation over the representer fields using the coefficients b_k, by directly forming the summation as in Eq 14 or Eq. 23. The other approach is a little more elaborate and is to convert the coefficients b_k for the representer field into equivalent forcing and subsequently running a forward model using these forcing coefficients (*Egbert et al.*, [1994]).

We used the simple approach of just summing the representer fields whereas Egbert et al. [1994] ran a forward model for their final solution using forcing coefficients derived from the estimated coefficients b_k for the representer field. When running the final forward solution *Egbert et al.* [1994] furthermore demonstrated, that this model can be run with higher resolution than the resolution used for the calculation and running of the representer fields derived previously. This will yield a FINE resolution final model based on COARSE resolution representer fields.

Theoretically there is no guarantee that the solution on a refined grid will minimize the penalty functional in Eq. 11. However, a solution can be found, that minimizes the penalty functional by iterative use of a conjugate gradient search for a minimum solution.. Practically Egbert et al. implemented this on the CM-200 using the CMSSL library routine for Equation solving, where they used the subroutines for conjugate gradients.

We extensively tested their approach as we would have liked to use it as it seriously reduces the demand for storage capacity and CPU time. However, there were considerable problems with a number of routines in the CMSSL library on the CM-200 in Edinburgh due to different versions of the CMSSL library implemented on the CM-200 at the time of testing the code. Therefore, we finally have to abandon this approach and rather calculate a FINE resolution final model based on FINE resolution representer fields. This implied a substantially heavier use of storage facilities. But for the project the CONVEX Mass storage facilities at the Proudman Oceanographic Laboratory (POL) could be used.

Mesh sizes - Model domain.

Our primarily goal was to derive an accurate ocean tide model for the North Atlantic Ocean and especially the northwest European shelf region. The area of interest is therefore roughly limited by the 20°N and 80°N parallel and the 100°W and 40°E meridian.

However, the version of the code that we received from Egbert el al at the Oregon State University was tuned for global ocean tide modelling. One of the implications of this globalness was a clever use of shift operators in which the entire arrays could be shifted around the earth by one grid cell in the longitude direction, and which speeded up the computations.The shifting is possible as the eastern and western limits are identical in global grids. However, this was not easily ported into local models, so we therefore decided to take the simplest approach by just running global models for which we needed to alter the code as little as possible.

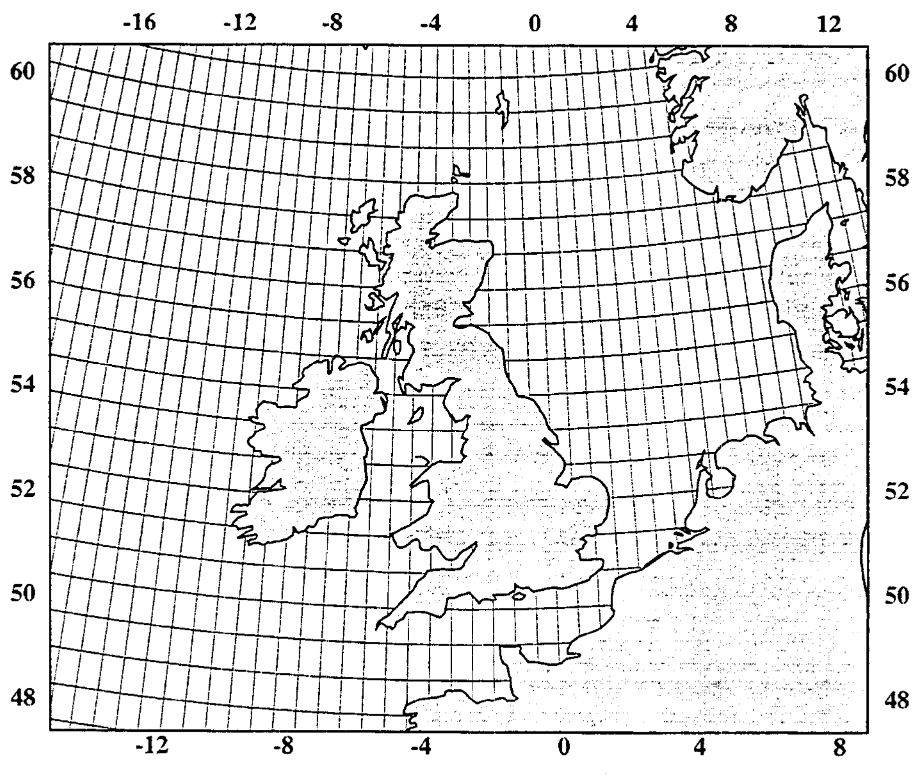


Figure 5 Mesh size for a model having 512 x256 gridcells corresponding to a spatial resolution of 0.7° x 0.6°

For the calculation of representer field Egbert et al. used a global regular grid with a longitude by latitude grid size with 256 by 128 grid points. This corresponds to a spatial resolution of 1.4° x 1.2° (longitude by latitude) of the global representer fields. The grid cells are not perfectly square as the model area is limited in the latitude direction. Egbert et al used the 70° N parallel as northern model domain boundary as they only used altimetry from TOPEX/POSEIDON. To the south Antarctica always forms a natural boundary for ocean tide models. Their final model by Egbert et al, was computed on a refined grid having a size of 512 x 256 corresponding to 0.7° x 0.6°. This resolution is shown in Figure 5 for the Northwest European shelf region.

We experimented using two different grid sizes on the CM-200. In our first approach we used a global regular grid with a size of 512 x 256 elements corresponding to a spatial resolution of 0.7° x 0.6° for the calculation of the representers and also the final model. The resolution of this model is shown in Figure 5, and in the Northwest European shelf region it corresponds to a spatial resolution of 35 km by 65 km in the longitude by latitude direction.

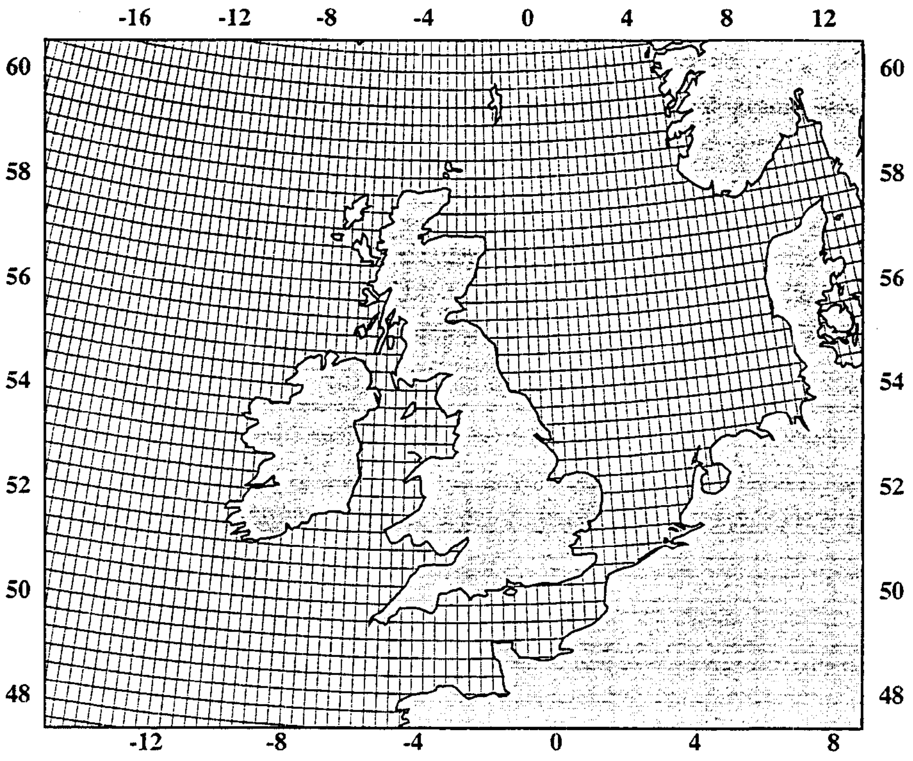


Figure 6 Mesh size for a model having 1024 x 512 gridcells corresponding to a spatial resolution of 0.35° x 0.3°

We also experimented with a global model constrained to observations from tide gauges and TOPEX/POSEIDON altimetry in the Baltic Sea using a global regular grid with a size of 1024 x 512 elements. This corresponds to 0.35° x 0.3° which in turn corresponds to (17 km x 30 km) for the calculation of the representers as well as for the final model. The resolution of this model is shown in Figure 6. For this model we only calculated 25 representers, whereas we for the coarser model above calculated around 400 representers.

For both models we used the Antarctica as southern boundary whereas the northern boundary was chosen to be the 82° N parallel. This boundary was chosen as to include all of the Greenland-Iceland-Norwegian Sea (GIN sea) as well as large parts of the Barents Sea. Furthermore this would make this model applicable to altimetry studies from the ERS-1 and ERS-2 satellites, asthese satellites have an inclination of 98°.

Finally, we hope to test a model on the Cray T3D machine in the future which should have a size of 2048 x 1024 array elements corresponding to 0.15° x 0.12° resolution for the calculation of the final model. The resolution of such a model will meet the increased demands for global models which are accurate also in coastal regions. The resolution of this global model will be similar to the resolution of the Flather Shelf model (Described below), which is often used for tidal prediction on the UK coast. And the resolution of such a model approaches the capabilities of the finite element model from Grenoble. The resolution of this model is shown in Figure 7.

However, there are considerably problems handling such a model. The basic problems will not be the limitation of resolution due to memory on the Cray T3D, as the machine is equipped with more than 16 Giga-byte of memory which is plenty for our purposes. However, handling of the global representer fields will be the most difficult and time consiming task for this model. As an example the storage of a representer field with this resolution requires (2048 x 1024 x 8 x 8 bytes (four constituents)), which is a little less that 100 Mbyte for ONE representer. A reasonable model requires between 300-10000 representers.

This storage problem can be limited in two ways. Firstly by using the approach derived by Egbert et al, which calculates representers on a coarser grid and the final solution on a fine grid. Secondly, the results so far, have shown that it is not necessary to store the entire global grids for each of the representer fields. As can be seen from Figure 3 then the representer field are broadly peaked around the location of the representer with reasonably amplitudes in most of the basin where they are located. Hence, the representers have only significant

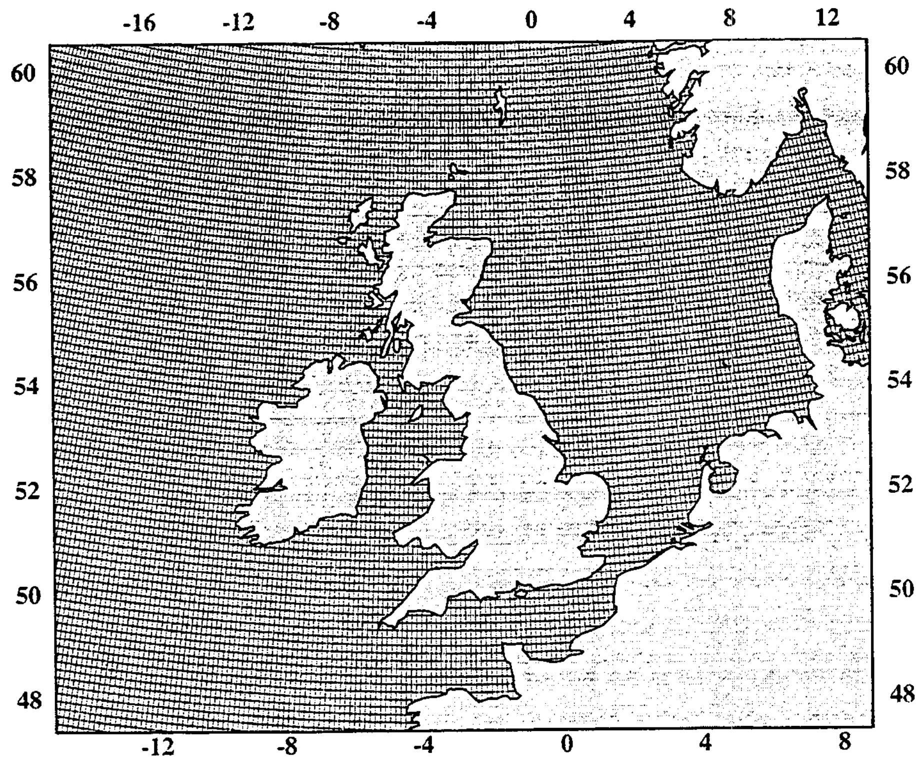


Figure 7 Mesh size for a model having 2048 x 1024 gridcells corresponding to a spatial resolution of 0.15° x 0.15° as intended to run on the Cray T3D

amplitude within around 2000-3000 km from the location of the representer. In the rest of the ocean domain the effect is merely a phase change of insignificant amplitude. This signal could therefore easily be skipped (or defined to zero amplitude) without compromising the accuracy of the method.

Practically this could be implemented by using a threshold value. If the amplitude was below this value then this value could be put to zero and ignored. This means, that one could limit the storage requirements by just storing a preselected area around each representer and defining all other ocean areas to have zero amplitude with respect to that representer. This would entail some programming, but it would be worthwhile doing when experimenting using these high resolution grids in the future.

Open boundary.

The model used in the present analysis is limited to the north by the 82 °N parallel. The 82° N parallel is therefore the only open boundary of the model, where elevation has to be specified using an existing ocean tide model. Only a few model are presently available that can be used to provide elevation along this open boundary. They are the global hydrodynamic model by *Schwiderski* [1980] and from Grenoble [*Le Provost et al.*, 1994 - see description below] A few local model can also be used. They are the models for the Arctic ocean by *Kowalik and Proshutinsky* [1993] or the models by *Gjevik and Straume* [1989] and *Gjevik et al.*, [1993].

We chose to use the local model for the Arctic ocean by *Kowalik and Proshutinsky* [1991] as this model should be very accurate in the Polar seas. However the model from Grenoble should also be very accurate, whereas the accuracy of the Schwiderski model is more questionable.

Computer resources and performance tests.

We conducted a few performance test on the Edinburgh CM-200 connection machine at Edinburgh Parallel Computing Centre (EPCC). The first test was a timing of the time stepping algorithm which is the key algorithm in the construction of representer fields. Each iteration in the time stepping algorithm required 28 floating point instructions per cell in the model grid. By running the 10000 iterations we could achieve a performance of 0.6Gflop (16K processor model) which is very satisfying and which indicates that the code is very well suited for parallel processing. The time consumed for the calculation of one representer containing the four major constituents on a 512 x 256 cell grid was estimated to be around 12 minutes on the CM-200 in the 16K processor mode (during night and weekends).

This time does not include the reading of the input files as well as writing of the output files which takes another 5-7 minutes for a 512 x 256 model and around 20 minutes for the 1024x512 model . The reading of input parameters for the calculation of representers requires knowledge/grids to 13 parameters at each cell of the grid which is required by the time stepping in an Arakawa C-grid. This is e.g. depth and mask values in the center of each cell, in the corners of each cell and on the center of each cell sides (See description in chapter 3). This is a total of 16 Mbyte of unformatted data which must be read before the calculation of representers can start. However these parameters are identical for all representers so the input reading must only be performed once for each job. One job generally corresponds to the calculation of 20 representers as the time limitations are four hours CPU time for one batch job on the CM-200 connection machine at Edinburgh. Output for each representer on a 512 x 256 grid is equivalent to 4.5 Mbyte unformatted data for a model which consists of the four major ocean tide constituents. These huge amount of output data can be treated using a sophisticated system of unformatted data and direct access files. The use of direct access files is absolutely essential for the subsequent interpolation in the representer fields when deriving the elements of the representer matrix. Here the direct access can elegantly be used so one avoid the reading of more than the required elements in the representer matrix.

Choice of apriori hydrodynamic model (u_0).

Our initial or apriori model satisfying the Laplace tidal equation subject to rigid boundary conditions u_0 is used in Eq. 14. The choice of this model is crucial in order to obtain a realistic final model. However, as indicated in the Figures 1 and 2 the pure hydrodynamic model calculated using the Laplace Tidal Equations is not very close to a realistic ocean tide model for the region like e.g. the Flather shelf model for the northwest European Shelf region [*Flather,* 1981, 1988]. Furthermore these forward models did not match tide gauge readings in the area very well. Our test using this model as apriori model and subsequently performing the ocean tide inversion and calculating a solution that fits both dynamics and data the final result was not as accurate as e.g. the Flather model. The reason for this must be found in the relatively complex tidal regime in the North Atlantic Shelf region which is extremely different to resolve with the chosen resolution for our model equal to 0.7°.
Such results is not very satisfactory, so in order to try to make a model comparable to or hopefully to make a model which is better than the present best model for the area - it was decided to make an approximation at this point by choosing an alternative apriori model u_o.

$$u \approx u_{Grenoble(FES94.1)} + \sum_{k=1}^{K} b_k r_k \tag{24}$$

Rather than using the pure hydrodynamic model calculated from the Laplace Tidal Equations as initial model, it was decided to make an approximation and try to use the model from Grenoble as apriori model and calculate residual ocean tides to that model. This has several implications, because it violates parts of the theoretical assumptions. The theory is mainly violated because the base functions (representers) have actually be calculated using a different set of hydrodynamic parameters as well as different resolution for the hydrodynamic model than the apriori hydrodynamic model (Grenoble). However, it is very important to note, that the covariance function for the dynamic error used to define the inner product (Equation 12) in our Hilbert space and hence the derivation of the least squares solution (Equation 14) was derived in a consistent way. This is due to the fact that the dynamic error covariance function is actually calculated from the Grenoble model. Hence the improved solution is actually the minimum solution in the Hilbert space except for the approximations made due to the choice and use of different model parameters.

One of the results of this approximation is, that we will not be able to get the transport components as output from our global inversion solution. We will only be able get the elevation component of the final ocean tide field, as we have not had access to the transport components on the Grenoble model. On the other hand the advantage of using the Grenoble model should be that we are able to get some of the complex content of the ocean tide signal modelled with the Grenoble model, which we can not resolve using our linear formulation. However in our assumption (Equation 24) we only assume that the residual ocean tide signal to the Grenoble model is well modelled using a linearized model. This might be even more correct, as the assumption inherent in Equation 14 (The residual ocean tide to a linear forward model is accurately modelled using a linear formulation) is questionable with the accuracy of the existing forward models.

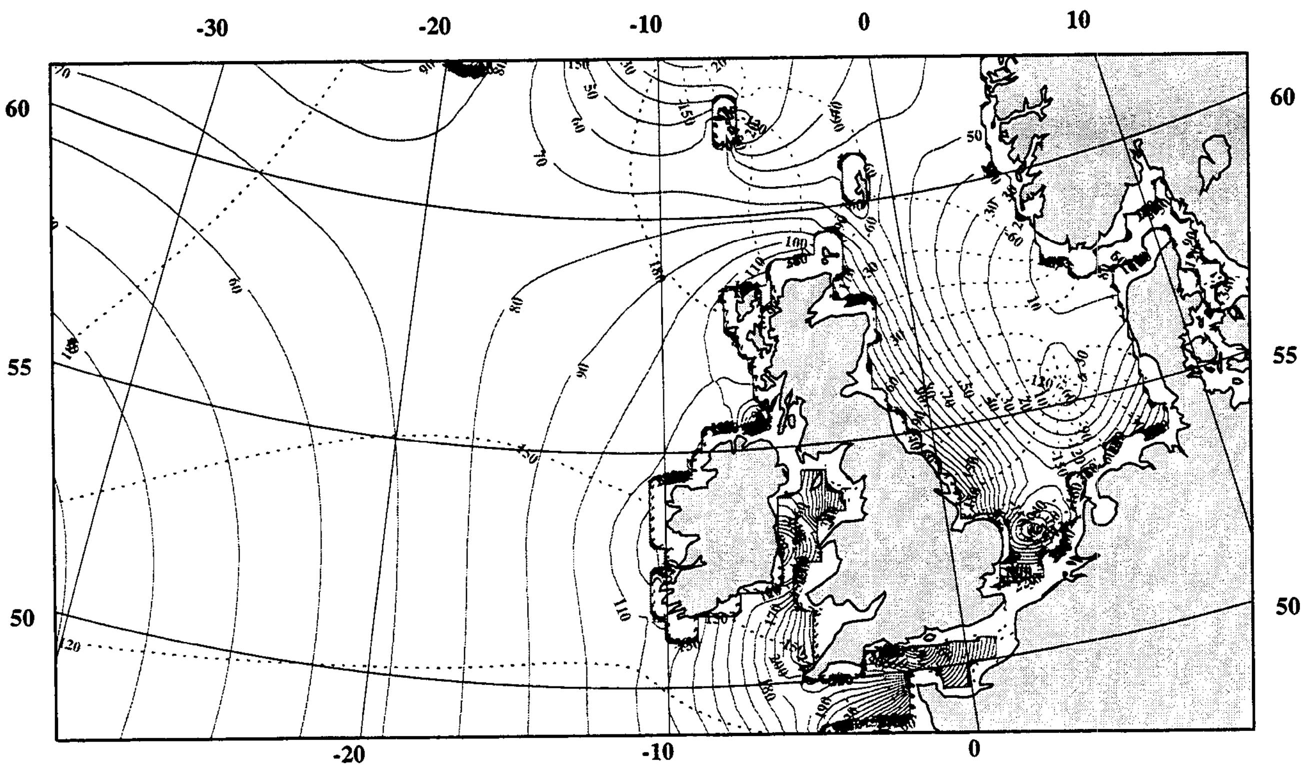


Figure 8 M_2 constituent of the Grenoble model [*Le Provost et al.*, 1994] in the Northwest European shelf region. Solid lines represent amplitudes, dashed lines phase. Contour interval is 10 cm.

The Grenoble model (FES94.1).

39 Tide Gauges	M_2	S_2	K_1	O_1	$M_2+S_2+K_1+O_1$
Grenoble (0.5° x 0.5°)	2.23	1.20	0.96	0.51	2.63
OSU, TPXO-2 (0.5° x 0.5°)	2.14	0.87	0.78	0.45	2.45

Table I Comparison with 39 tide gauges in the Atlantic Ocean (the 39 tide gauges is a subset of the 104 tide gauge set prepared by C. Le. Provost and other members of the T/P subcommittee on ocean tides.

The Grenoble model (FES94.1) is actually a very accurate hydrodynamic ocean tide model to use as starting point for modelling ocean tides. The Grenoble model has been produced by C. Le Provost, M. L. Genco, F. Lyard, P. Vincent and P. Canceil [*Le Provost et al.* 1994], and is based on a finite element hydrodynamic scheme. This model along with the model that by Schwiderski are the only two recent global models, that do not presently include satellite altimetry in their solution. The design of the model is based on a non-linear formulation of the shallow water equations, and the model covers the global ocean (including the Arctic ocean) except for some minor areas such as the Bay of Fundy. The model is provided on a global regular latitude - longitude grid of 0.5° x 0.5° resolution, with eight constituents (M_2, S_2, N_2, K_2, $2N_2$, K_1, O_1) and Q_1. Five secondary constituents (Mu_2, Nu_2, L_2, T_2 and P_1) have subsequently been deduced by admittance interpolation. The model used here is the pure hydrodynamic model version FES94.1

The M_2 constituent for the North Atlantic ocean is shown in Figure 8. In order to illustrate the accuracy of this model a comparison to 39 tide gauges in the Atlantic ocean is shown in Table 1 for this model along with the OSU TPXO-2 model by Egbert et al.

In Table 2 the comparison with a set of 65 tide gauges on the Northwest European shelf is furthermore shown The location of these tide gauges are displayed in Figure 9. These tide gauge readings consist of a selection of

65 Tide Gauges	M_2	S_2	K_1	O_1	$M_2+S_2+K_1+O_1$
Schwiderski (1° x 1°)	5.79	1.92	1.15	0.78	6.26
Cartwright & Ray (1° x 1.5°)	8.25	3.74	2.46	0.94	12.16
Grenoble (0.5° x 0.5°)	3.99	2.12	1.29	0.70	4.74
OSU, TPXO-2 (0.58° x 0.70°)	9.34	8.55	1.45	1.14	12.80
Andersen AG94.1 (0.5° x 0.5°)	3.23	2.23	1.29	0.70	4.21
Flather NEA, 1981 (0.33° x 0.5°)	5.08	2.74	1.59	1.12	6.09
Flather shelf, 1994 (0.16° x 0.11°)*	3.84	1.57	2.11	0.82	4.73

*The comparison towards the Flather shelf model was limited to 47 tide gauges located within the coverage of the model.

Table II Comparison with a set of 65 tide gauges in the Northwest European shelf region for a number of global and local models.

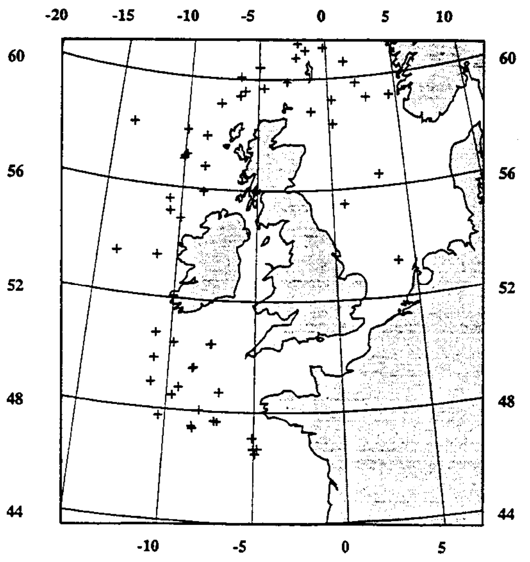


Figure 9 Location of 65 tide gauges in the North West European shelf region.

58 tide gauges published by IAPSO [*Smithson,* 1992] supplemented with 7 pelagic tide stations in the North Sea as obtained from the data banks at POL, Bidston Observatory.

The two widely used local models are added to the Table for comparison. One is the 36km Flather NEA model, a non-linear hydrodynamic finite difference model described by *Flather* [1981]. This model uses a regular (0.33° by 0.5°) latitude - longitude grid covering approximately $37° < \phi < 72°$, $-30° < \lambda < 25°$ with open boundary forcing based on Schwiderski and some tide gauges. Another local model is the 12km Flather model (0.11° by 0.16°) which covers the NW European shelf ($48° < \phi < 62°$, $-12° < \lambda < 13°$), and is used operationally for coastal flood forecasting in the UK. The model is optimized to predict water levels on UK coasts, and open boundary forcing is based on Flather, 1981 NEA model results and tide gauges around the shelf. It is important to note that a subset of the 65 tide gauge set used in this investigation has been assimilated into the Flather model as well as the Schwiderski model and the Grenoble model, which means, that the 65 tide gauges set will not give a totally independent test of these models.

It is important to note that the apparent accuracy of the OSU TPXO-2 model (an eight constituent'extended model to the model described in *Egbert et al.*, [1994]) does not provide very accurate ocean tide estimates in the Northwest European shelf region compared to the other models. This model performs very well in the deep ocean, but the model has problems in the shallow waters of the Northwest European shelf region, where it performs substantial worse than e.g. the model from Grenoble. The causes for this is probably due to a coupling between the fact that the OSU TPXO-2 model did only have two representers in the North sea, and the fact that it uses the forward model in Figure 1 as a initial model (Eq. 4) for the tides in the North sea.

Choice of covariance functions.

The observations and the dynamics can be weighted through the choices of the covariance functions C_f and C_e as mentioned in connection with the explanations following Equation 11. The penalty functional or cost function enables a normalisation of the two sources of errors, and hereby it can also be used to enable a trade-off between fitting the dynamics and fitting the data. However for the presented models we have tried to choose values for the covariance functions from a consideration, that the models should be using as realistic variance parameters as possible and still being fairly easy to implement.

Dynamic error covariance

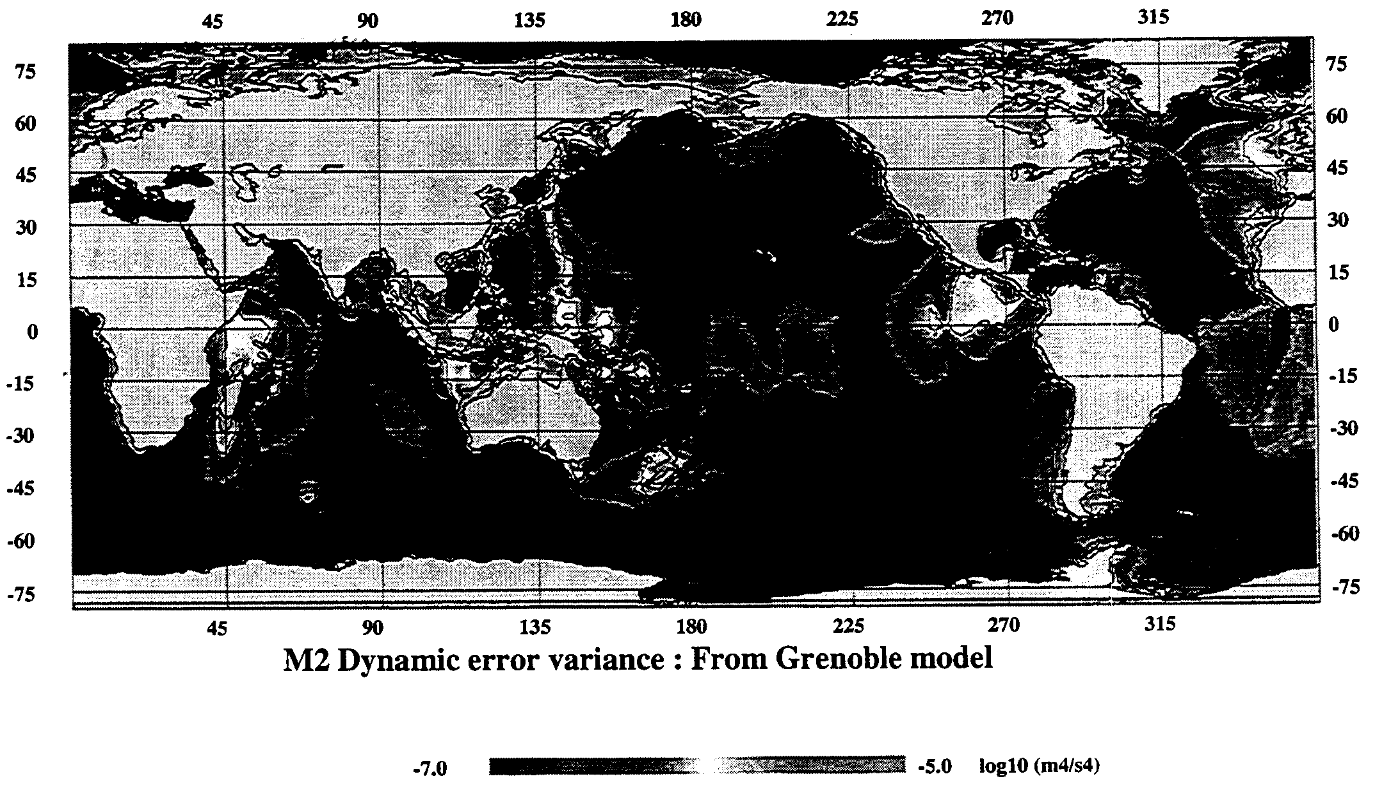


Figure 10 Dynamic error variance function for the M_2 constituent as calculated from the model of Grenoble. The greyscale is logarithmic ranging from 10^{-7} to 10^{-5} m^4/s^4.

The dynamic error covariance is used to smooth the backward solution in order to yield the inhomogeneous boundary conditions as well as forcing for the subsequent forward solution when calculating representer fields. (See the equations 15, 16). The dynamic error covariance havs 3 complex components (two from the momentum equations and one from the continuity equation). In order to keep computations as simple as possible the 3 components are assumed to be uncorrelated.

The design of the dynamic error covariance function is also largely determined by computational considerations. The most simple approach would be to use the same covariance function everywhere. However, this is not very satisfactory as the dynamic conditions change dramatically going from e.g. the open ocean into the shelf regions.

Therefore a slightly more complicated covariance function was chosen. This approach is identical to the approach used by *Egbert et al* [1994] This covariance function is described for each constituent separately using the following form

$$C[x_i,x_j] = \sigma(x_i)\,\sigma(x_j)\,\theta(\cos\psi) \tag{25}$$

where $\sigma(x)^2$ is the spatiallt varying variance and $\theta(\cos\psi)$ is the spatial correlation function . This correlation function ψ has a decorrelation length scale of around 5° which is identical to the correlation function used by *Egbert et al* [1994].

Dynamical errors covariance function will have errors due to the approximations used when describing the model. These approximations arises from the choices of:

dissipation parameters,
the bathymetric model,
the applied grid spacing,
the applied linearized shallow water equations,
the treatment of loading,

McIntosh and Bennett [1984] concluded that the crude parametrisation of dissipative effects is without doubt the largest and most dominant source of error. Especially the fact, that a number of very different parametrisation can be used to describe the dissipation parameters (Figure 1-2), implies that the error associated with these parameters must be almost of the order of the parameters themselves. So it is obvious, that one must assume very large errors in the dissipation parameters. For the zonal component the hydrodynamic equations according to Equation 3 yield

$$(i\omega + K)u - fv + \sigma g H \frac{dh}{dx} = f_{u0} \qquad \textbf{(26)}$$

Assuming 100% error in the dissipation terms, the dynamic variance or the forcing error δf_K due to errors in this term of the above equation can be described like (*Egbert et al.* [1994])

$$Var(\delta f_K) = |KU| = [\frac{K|U|}{\max(H(x),H_0)}]^2 \qquad \textbf{(27)}$$

where U is the transport scale in the northerly direction for the zonal component.

This is a very convenient formulation for computing the dynamic covariance, as it can be implemented numerically. This way the dynamic error covariance function can easily be computed from an existing ocean tide model using a geostrophic approximation for the estimation of the transport components.

As the errors in the description of dissipation is so dominant it was decided to ignore other sources of errors in the dynamic error covariance function. Hence, the dynamic error variance function was calculated from the model of Grenoble using a geostrophic approximation in order to derive the transport components. In order to avoid too large variations in the dynamical errors parameters, and in order to avoid, that possible spurious effects in the Grenoble model could propagate into the final solution, the Grenoble was smoothed prior to estimating the dynamic errors with a very simple 1 degree running mean smoothing filter.

The error variance function for the zonal component is shown in Figure 10. The dynamic error variances varies from 5.0 x 10^{-6}m^4/s^4 in the deep ocean to 1.0 x 10^{-4} in coastal regions. The figure is originally a color figure. Unfortunately the black and white version became very dark which makes it difficult to decipher. However, lightgrey in the figure corresponds to areas, where the dynamic variance is largest.

The measurement error covariance

The measurement error covariance function enters the inversion when solving for the coefficients of the representer fields. For simplicity the measurement error covariance function is described like a diagonal matrix having the form

$$C_\epsilon = \sigma^2 I \qquad \textbf{(28)}$$

Hence, the observations will be assumed to be uncorrelated. Both the standard deviation for the harmonic constants derived from tide gauges as well as from satellite observations are assumed to have similar accuracy. The value of σ will therefore constally be equal to 1 centimeter.

Location of representers and observations.

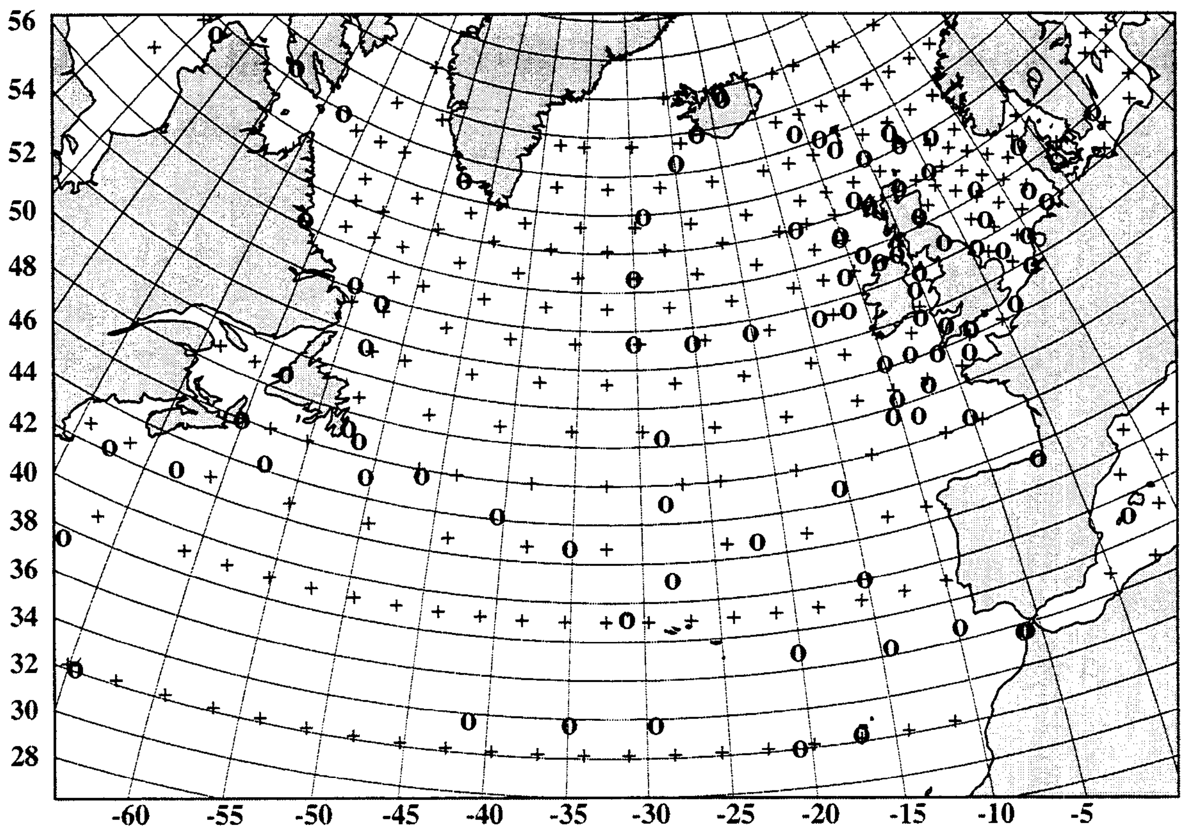


Figure 11 Location of representers in the North Atlantic ocean. (+) selected TOPEX/POSEIDON crossover locations. (o) selected tide gauge locations.

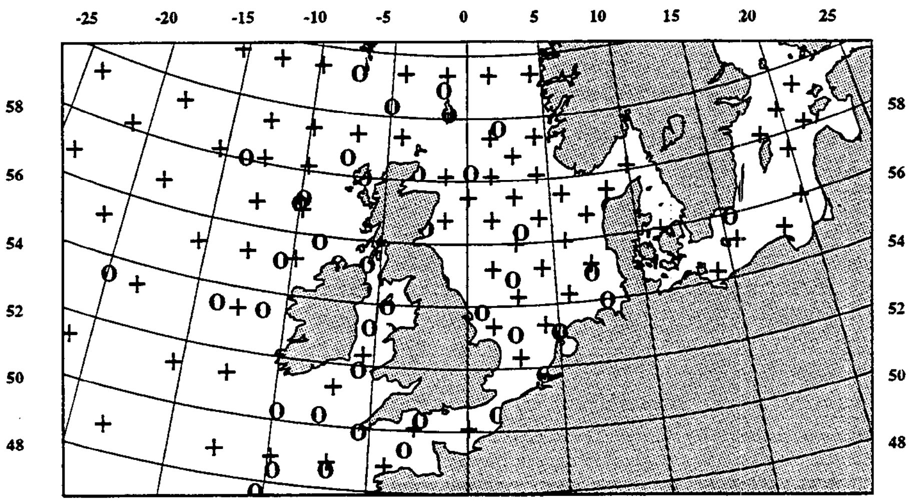


Figure 12 Location of representers in the northwest European shelf region. (+) selected TOPEX/POSEIDON crossover locations. (o) selected tide gauge locations.

There are several considerations on how the spatial distribution of the representers should be. As mentioned previously the representer field for a location (ϕ,λ) indicates, how the model reacts to forcing applied at this location. The distribution of representers should therefore have a somewhat homogenous spatial coverage, and the density of the representers should be adequate to represent the tidal features in both the open ocean as well as in the shelf regions. On the other hand, the chosen correlation length in the dynamic covariance function is five degrees. And this length is used to smooth the backward solution when calculating the representer fields.

For our COARSE model we used a global regular grid with a size of 512 x 256 corresponding to 0.7° x

0.6° for the calculation of the representers and the final model. For this solution 482 representers were calculated at the locations displayed in Figure 11. 264 out of these representers were derived at TOPEX/POSEIDON crossover locations, whereas the remaining 218 representers were computed by applying the astronomical forcing at the location of tide gauges.

An additional 80 representers are located at the tide gauge stations of the set of 104 tide gauge stations as located in Figure 14. A close up of the distribution and location of representers in the northwest European shelf region is presented in Figure 12.

The density of representer fields for observations from the TOPEX/POSEIDON satellite is relatively dense in the shallow water regions around the Northwest European shelf (indicated by +) as all available crossover locations on the shelf have been chosen for the analysis. In the deep parts of the North Atlantic Ocean only a subset of the available crossovers were selected as displayed in Figure 11.

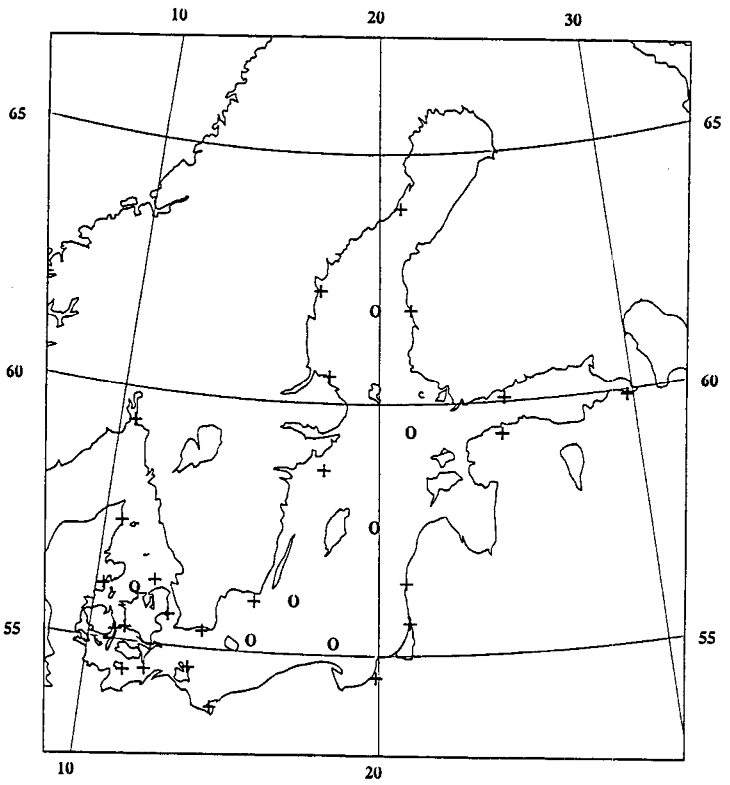


Figure 13 Location of representers in the Baltic Sea and Danish Waters.

The global model constrained to representers in the Baltic Sea used a global regular grid with a size of 1024 x 512. corresponding to 0.35° x 0.3° (17 km x 30 km) for the calculation of the representers as well as for the final model. For this model we only calculated the M_2 constituent and a total of 30 representers were used. 23 of these representers are estimated at tide gauge locations, whereas the remaining seven representers were calculated at TOPEX/POSEIDON crossover locations. The location of these representers can be seen in Figure 13.

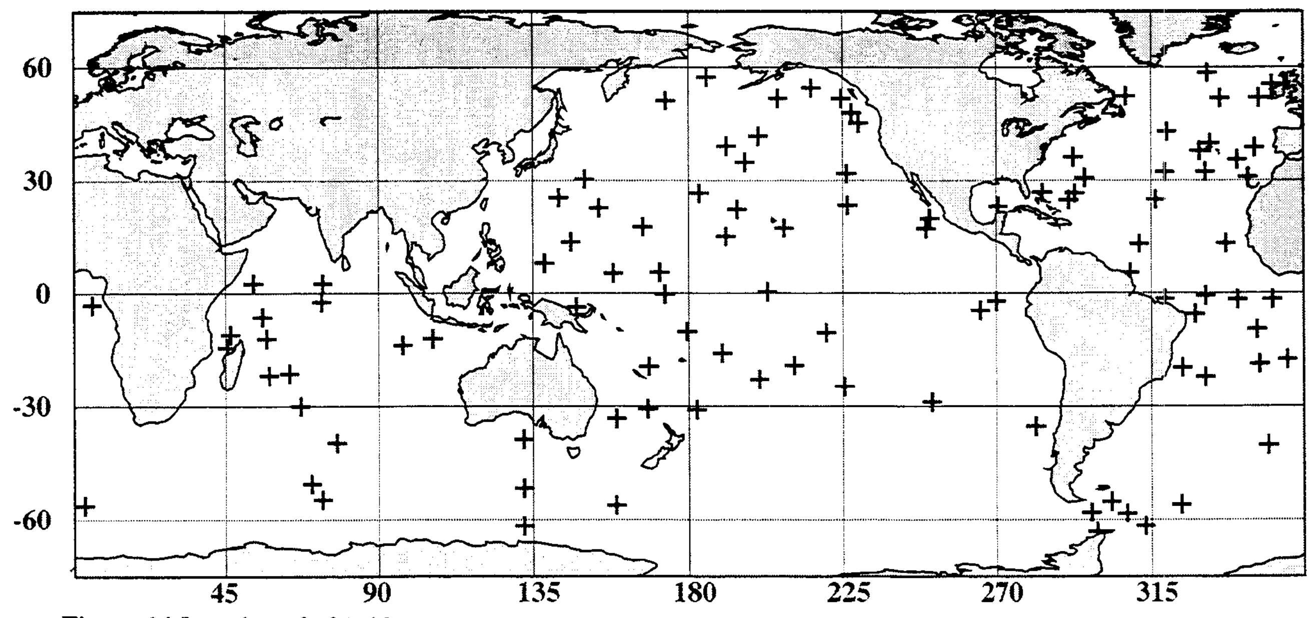


Figure 14 Location of 104 tide gauges used for the comparison of models. Outside the North Atlantic Ocean the tide gauges marks the position of representers.

Ocean tide Model Results.

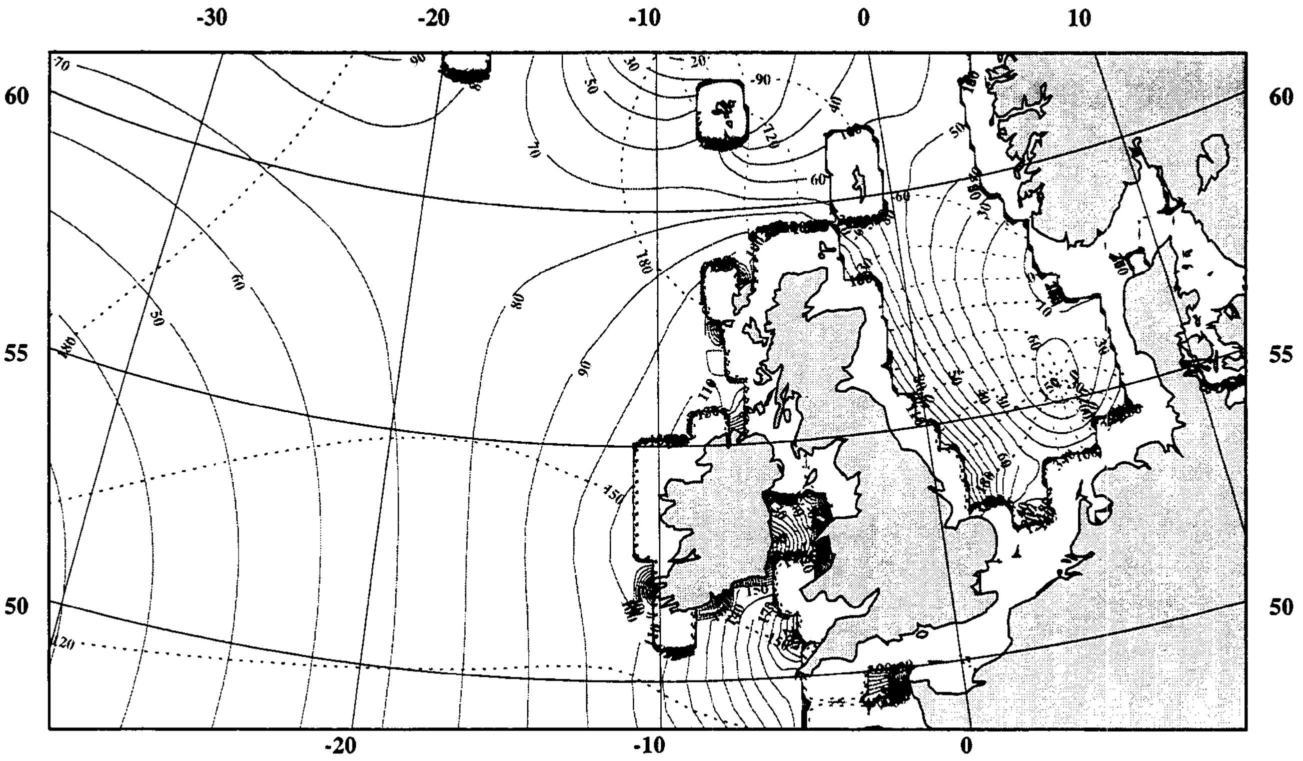


Figure 15 M_2 constituent for the Northwest European shelf region. Solid lines represent amplitudes, dashed lines phase. Contour interval is 10 cm.

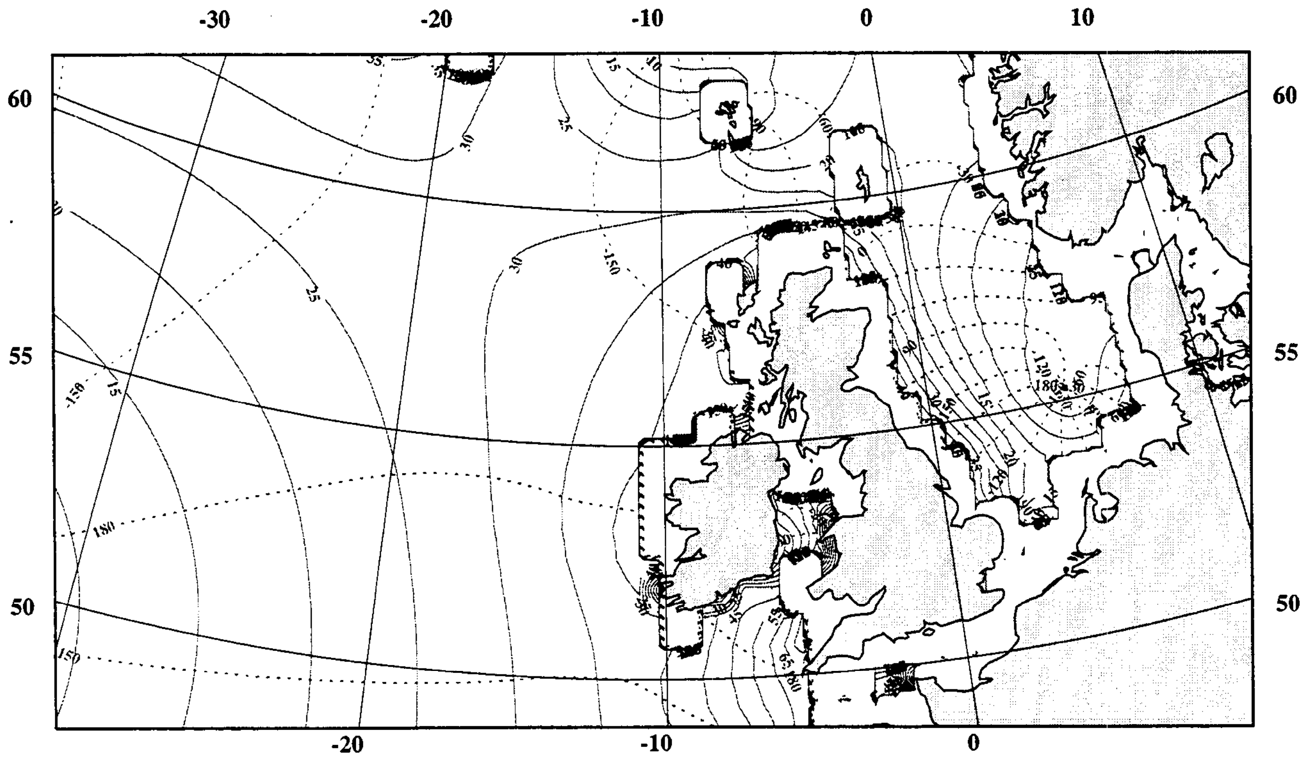


Figure 16 S_2 constituent of the model for the Northwest European shelf region. Solid lines represent amplitudes, dashed lines phase. Contour interval is 5 cm.

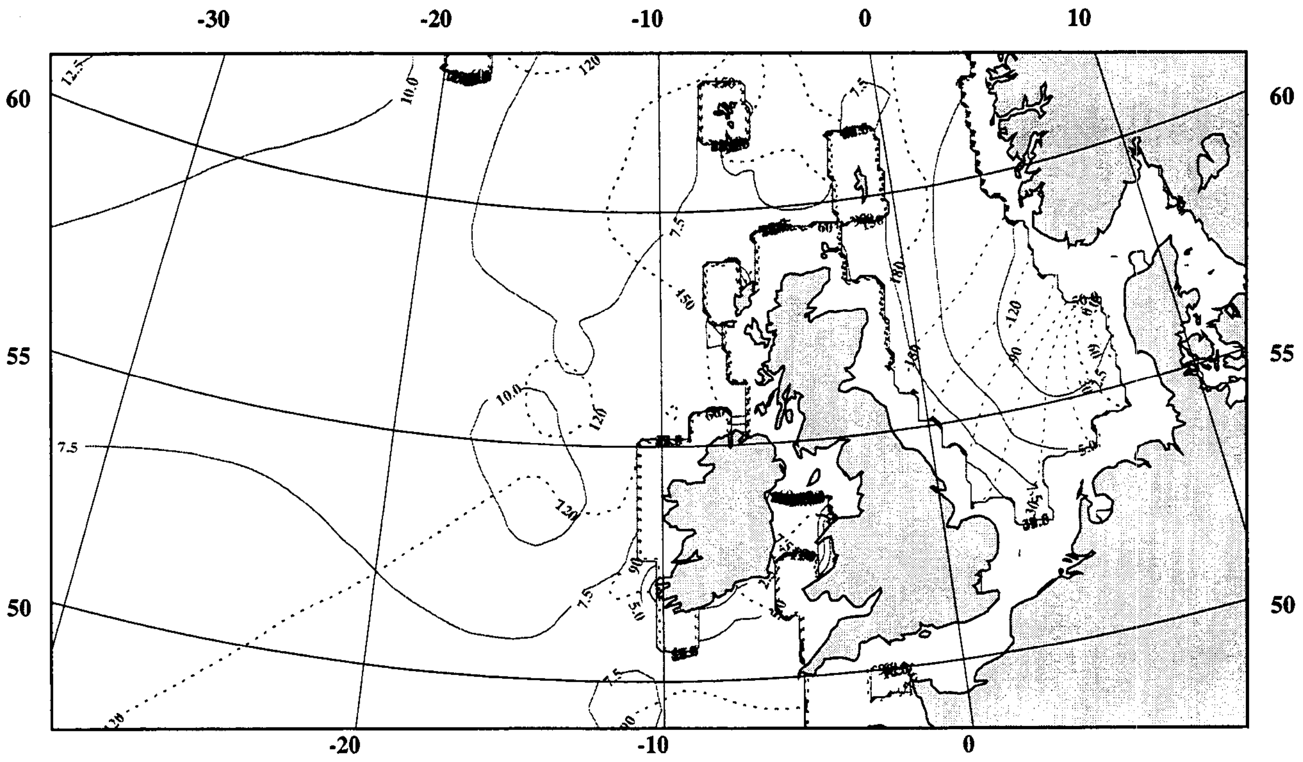


Figure 17 K_1 constituent of the model for the Northwest European shelf region. Solid lines represent amplitudes, dashed lines phase. Contour interval is 2.5 cm.

The resulting model computed from global inversion approach having a resolution of 0.7°and derived from fitting 372 representers are presented in the figures 15 - 18 for the four major constituents M_2, S_2, K_1, O_1 respectively. The models exhibit very accurate ocean tide field throughout the model domain, but it is obvious, that the major problem with the presented solution is the masking applied. The masking relates to the fact, that the resolution of the model is simply too coarse toenable proper ocean tide modelling in coastal regions. However the accuracy of these new solutions are very promising as will be seen.

In Table 3 the comparison is made with a subset of 58 tide gauges shown in Figure 8 in the Northwest European shelf region. The subset consisted of 58 stations. These were the stations from the 65 tide gauge stations located in Figure 9 which were within the coverage of the computed model (Due to the masking).

58 Tide Gauges	M_2	S_2	K_1	O_1	$M_2+S_2+K_1+O_1$
Grenoble (0.5° x 0.5°)	3.91	2.12	1.19	0.72	4.65
OSU, TPXO-2 (0.58° x 0.7°)	9.01	8.28	1.40	1.18	12.23
Global Inversion (0.62° x 0.7°)	3.75	1.67	1.42	0.61	4.38
Flather NEA, 1981 (0.33° x 0.5°)	4.89	2.61	1.52	1.01	5.83
Flather shelf, 1994 (0.16° x 0.11°)*	3.84	1.57	2.11	0.82	4.73

*The comparison towards the Flather shelf model was limited to 47 tide gauges located within the coverage of the model.

Table III Comparison to a set of 58 common tide gauges in the Northwest European shelf region for a number of global and local models..

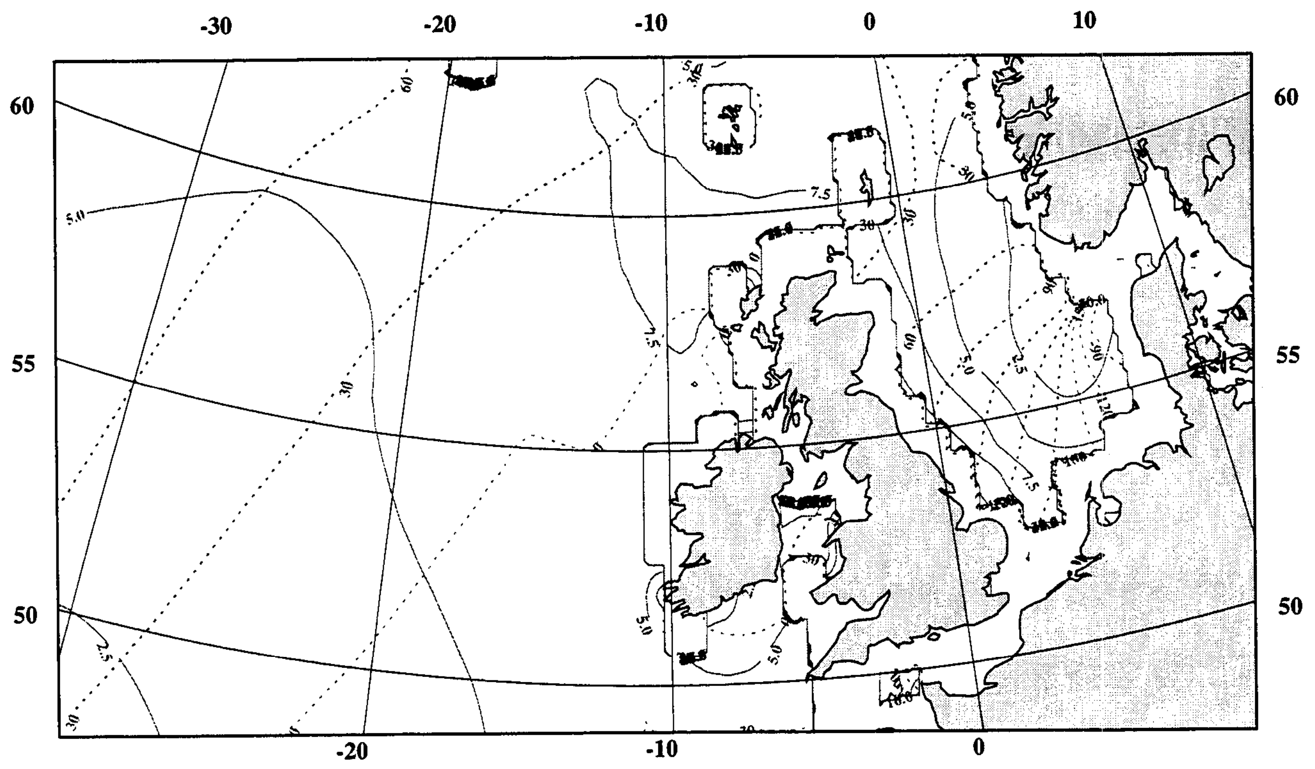


Figure 18 O_1 constituent of the model for the Northwest European shelf region. Solid lines represent amplitudes, dashed lines phase. Contour interval is 2.5 cm.

Compared with the apriori or initial model from Grenoble the major improvement in RMS differences with the selection of tide gauges is seen for the S_2 constituent. A minor improvement is also seen for the M_2 and O_1 constituents. It must be borne in mind, that in such a comparison both inaccuracies in the model as well as in the derived harmonic constituents will limit the obtainable accuracy.

For the M_2 constituent a RMS difference of 3.75 cm with tide gauge readings is actually VERY accurate, as the averaged RMS amplitude of the M_2 constituent is close to 1 meter in the shelf region. A puzzling result is the fact that the RMS differences do not seem to become better for the K_1 constituent - on the contrary it becomes worse. One reason for this might be that we still have problems with the software. However, it is VERY puzzling, that we have problems for the EXACT same constituent as the Flather shelf model. This indicates, that the problems might arise from the harmonic constants for the tide gauges - and that they might not be sufficiently accurate The recording length of a number of these tide gauges (which also entered the Flather shelf model) are less than one month, which is too short to perform a reliable and stable extraction of the K_1 constituent See eg. [Baker, 1990] for a discussion on this problem.

Finally Figure 19 presents the M_2constituent for the Baltic Sea. However, we are not able to claim anything about the accuracy of this model, since there were problems in stabilising the solution and since we did not have access to any independent model that we could test the model against - actually we know of no other ocean tide model for the baltic Sea - except for the unpublished model by Lakshmi Kantha from the university of Colorado.

Conclusions - Future work.

The final model for the North Atlantic Ocean presented in the Figures 15 through 18 for the four major constituents M_2, S_2, K_1, O_1 respectively presents very "accurate"prediction of the tides in the North Atlantic and on the Northwest European shelf region. This demonstrates the high potential of the method applied for ocean tide modelling. Also Figure 19 presents a new M_2 ocean tide model for the Baltic Sea - even though it was not possible to compare this model to other existing model, as we did not have access to any other ocean tide model for the Baltic Sea.

The reason for writing "accurate" above arises from the fact that the derived model is constrained to tide gauge data and TOPEX/POSEIDON data. This introduces a problem of finding an independent data set in order to perform an independendent test of the accuracy of the model. This means, that a comparison to tide gauges

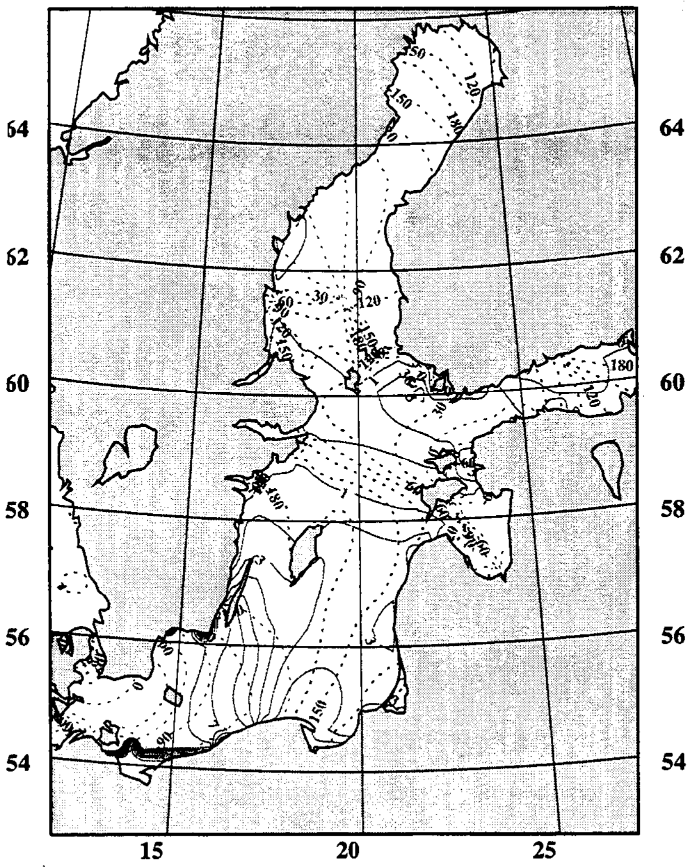


Figure 19 M_2 constituent of the high resolution test model constrained to representers in the Baltic Sea. Solid lines represent amplitudes, dashed lines phase. Contour interval is 2.5 cm.

data and TOPEX/POSEIDON data will never give an independent evaluation of the model as the data enters the model in its derivation. It is therefore relatively hard to make an independent test of the accuracy of the model. However it can be done by selecting a independent set of tide gauge data, and e.g. using data from other satellites than the TOPEX/POSEIDON satellite. This can be data from the ERS-1 or ERS-2 satellites or data from the older GEOSAT satellite.

The most striking result is probably that the resolution of the presented models in Figure 15-18 is clearly not good enough in order to model the tides on the shelf. We simply need to consider running a model on the Cray T3D machine in the future which should have a size of 2048 x 1024 array elements corresponding to 0.15° x 0.12° resolution for the calculation of the final model. The resolution of such a model will meet the increased demands for global models which are accurate also in coastal regions. The resolution of this global model will be close to the resolution of the Flather Shelf model, which is frequently used for coastal tidal prediction around the UK. And the resolution of such a model approaches the capabilities of the present finite element model from Grenoble. There are severe problems handling such a model. However, the present investigation has given us inspiration and experience on how to use the method, and also on how we can solve the practical problems. Hence, we hope to be able to improve the method further in the future.

Our approximation of using the model from Grenoble as apriori model, rather that using the pure hydrodynamic model calculated from the Laplace Tidal Equations as initial model has been shown to result in very accurate ocean tide estimations. However this approach violates the theory and there are several implications when taking this approach. On the other hand the advantage is that we are able to get some of the non-linear content of the ocean tide signal modelled using the Grenoble model, so that we just have to assume that the residual ocean tide signal is well modelled using a linearized model.

In our approach we avoided the use of constituent correlation, or more precisely we assumed it to be zero which is an approximation. This could be achieved by estimating harmonic constants from the direct observations from the TOPEX/POSEIDON and entering this into the global inversion scheme along with harmonic constants from tide gauges. Thereby we were able to run one constituent at a time and just estimate the coefficients for one constituent which substantially lowered the number of unknowns. The advantages of this simplified approach were so large, large that it must be recommended for further test with the model.

We also attempted to use a set of parameters for the hydrodynamic equations that best matched the local conditions in the North Atlantic Ocean and particularly matches the local conditions in the northwest European shelf region rather than using parameters that best fits global conditions even though the model was global. This entailed the inclusion of enhanced bathymetry and different dissipation parametrisation and the use of a different ocean tide model for the northern boundary of the model grid.

The crude parametrisation of dissipation was shown to be crucial to the accuracy of the final model. We chose a linear relationship between depth and bottom friction. However, we recommend that test is performed to show if one obtains better results using the quadratic relationship between depth and bottom friction, and what results will come out if one used the relative accurate forward model as initial or apriori model which could be derived using these parameters (Se Figure 2).

The status of the project is unfortunately that all authors have left Bidston to work elsewhere. However, this does not mean that the project will not be continued. At least Ole Andersen hopes to continue to work on ocean tides using this approach within some framework in the future.

The results presented in this report are very convincing, and demonstrate the high potential of the method, and the models derived in this analysis are comparable in accuracy to the widely used local models for the area. However, our experience with the method convinced us that we can still make more progress in modelling the ocean tides in the North Atlantic ocean and especially on the Northwest European shelf, but that we must use higher resolution in the ocean tide models in order to achieve that goal. However, it is important to realize that the work performed by Stuart Wilkes and Mike Ashworth on adopting the code to run on the Cray T3D actually means that most of the practical preparations has already been accomplished.

This approach as developed by A. Bennett combines hydrodynamic and data from tide gauges and satellite observations seems to provide the basis for further improving the accuracy of the ocean tide models especially in the Northwest European shelf region. A number of scientific communities will be able to benefit from this improvement. This will especially benefit communities that use ocean tide models as a standard correction to be applied before studying other signals related to the ocean. This will be altimetrists, geodesist and oceanographers and other scientist involved with measuring i.e. dynamical changes of the earth, as the ocean tides indirectly affect loading computations, high precision gravity measurements and even GPS measurements.

3. Parallilisation of representer code (S. Wilkes, M. Ashworth)

The project will involve working solely on parallelising the first section of code that forms the representer matrix, this first program "repx" contains no calls to external library routines, and hence is totally self contained. The calculation of the representer matrix executed by the first section of code is split up into several main subroutines. This code is data parallel code written in Fortran 90, the parallelism is implemented by compiler directives and array syntax. The compiler directives tell the CM-200 how to map out the data over its array of processes. The array syntax shows the compiler specifically where operations may be executed in parallel.

Description of Code and Method

Description Of Repx

istep:

This is the main program, its task is to read in from unformatted files the main data arrays that will be used in the calculation of the representers. Both two and three dimensional arrays are handled at this stage of type integer, real and complex. Also other important types of information are read into the algorithm including data for setting up the gridsize, locations of open boundary nodes, frequencies of the constituents, parameters to determine the harmonic analysis sampling rate, and the total number of sampling locations. Next, the main program enters a loop which completes one cycle for each representer, inside this loop all of the main subroutines are used in forming the final result. Finally when the main loop has been completed the results are written to a file.

chkpar:

This is the first routine that is called by the main program, it is called immediately after information about the gridsize has been read. Its task is to check that the parameters used in this particular run of the algorithm match those set for the main program in file 'gridsize.inc'

mkwts:

This is next to be called. It is a routine that computes latitude-dependent weights for the discrete approximation of integrals on the finite difference staggered grid and boundary. In this routine Fortran 90 'cshift' statements are used upon arrays to calculate updates for the finite difference scheme used

avgker:

This is the first routine to be called inside the main loop (avgker is called prior to this but as a co-ordinate checker only) it takes an individual point and constructs a number of complex arrays, which are estimates of the forcing amplitudes integrated over the timestep used. These arrays also provide details of fixed boundary conditions at the open boundary nodes

bstep,tstep:

This is a backward time stepping routine that uses the forcing arrays calculated in avgker. Initially various sign changes are performed upon some of the arrays used in this routine, but these are undone later. The routine then enters a time stepping loop, values are updated using 'cshift' statements again corresponding to the finite difference scheme. After the time stepping loop is complete, a set of equations are solved using Cholesky decomposition . tstep is identical in structure to bstep, except it solves the forward tidal equations by time stepping

chdec, ltslv, utslv:

The routine chdec is called from inside bstep and performs the Cholesky decomposition. Routines ltslv, and utslv are linear equation solvers for lower and upper triangular matrices respectively

fwdfrcx, fwdfrc:

After bstep is called, either fwdfrcx or fwdfrc uses the results from the backward stepping problem to calculate the forcing arrays and boundary conditions for the forward stepping routine. The calculation of the closed boundary nodes are completed first using cshift statements upon whole arrays. The next stage is the forming of boundary conditions for the open boundary nodes, these are calculated by taking a location of a point at a time to build the final array. Next these boundary conditions are smoothed using the Fortran 90 intrinsic 'sum' statement. After all calculations have been completed upon the boundary nodes, forcing arrays are calculated for the interior locations. The difference between the two routines is that fwdfrcx computes interconstituent forcings using cross correlation, and fwdfrc does not.

Finite Difference Scheme Used

The scheme is based upon the solution of the linear hydrodynamic equations below:-

$$\frac{\partial z}{\partial t}+\frac{\partial}{\partial x}(h\int_0^1 v d\sigma)=0$$

$$\frac{\partial u}{\partial t}-\gamma v=-g\frac{\partial z}{\partial x}+\frac{1}{h^2}\frac{\partial}{\partial\sigma}(\mu\frac{\partial u}{\partial\sigma})$$

$$\frac{\partial v}{\partial t}-\gamma u=-g\frac{\partial z}{\partial y}+\frac{1}{h^2}\frac{\partial}{\partial\sigma}(\mu\frac{\partial v}{\partial\sigma})$$

where t denotes time, u and v are x and y components of velocity respectively, g is acceleration due to gravity, γ is the geostrophic coefficient taken as constant, u is the vertical eddy viscosity, z is the free surface elevation, and h is the water depth. Which leads to the following difference equations:-

$$z_{i,j}^{t+\delta t}=z_{i,j}^{t}-c_k^0(u_{i,j,k}^{t}-u_{i-1,j,k}^{t})-c_k^1(v_{i,j+1,k}^{t}-v_{i-1,j,k}^{t})$$

$$u_{i,j,k}^{t+\delta}=c_k^2(u_{i,j,k}^{t}+{}_k^3(v_{i,j,k}^{t}+v_{i+1,j,k}^{t}+v_{i,j+1,k}^{t}+v_{i+1,j+1,k}^{t})-c_k^4(z_{i+1,j}^{t}-z_{i,j}^{t})+c_k^5$$

$$v_{i,j,k}^{t+\delta}=c_k^6(v_{i,j,k}^{t}+{}_k^7(u_{i,j,k}^{t}+u_{i,j-1,k}^{t}+u_{i-1,j-1,k}^{t}+u_{i-1,j,k}^{t})-c_k^8(z_{i,j}^{t}-z_{i,j-1}^{t})+c_k^9$$

where i and j indices are in the x and y directions on the finite difference grid in Figure 21. The k index is the vertical mode, and c_k^m are certain model parameters. At each time step the field is updated explicitly, and the u and v fields are updated semi implicitly to improve stability for all i, j, and k [*Harding and Wait*, 1994].

The type of grid used during the main time stepping loops of routines bstep, and tstep is an "Arakawa C-grid" as shown below:-

z_{i-n-1}	u_{i-n-1}	z_{i-n}	u_{i-n}	z_{i-n+1}	u_{i-n+1}	↑
v_{i-n-1}		v_{i-n}		v_{i-n+1}		Δy
z_{i-1}	u_{i-1}	z_i	u_i	z_{i+1}	u_{i+1}	↓
v_{i-1}		v_i		v_{i+1}		
z_{i+n-1}	u_{i+n-1}	z_{i+n}	u_{i+n}	z_{i+n+1}	u_{i+n+1}	
v_{i+n-1}		v_{i+n}		v_{i+n+1}		
←	Δx	→				

Figure 21

The serial implementation of the finite difference scheme described above contains the following computational kernel, represented in pseudo code (for one constituent):-

```
do for all timesteps
  do i,j for all gridpoints
      update z(i) from u(i,j), v(i,j), u(i-1,j), and v(i,j-1)
    enddo; enddo
  do i,j for all gridpoints
      update u(i,j) from z(i,j), z(i+1,j), v(i,j), v(i+1,j), v(i,j-1), and v(i+1,j-1)
    enddo; enddo
  do i,j for all grid points
      update v(i,j) from z(i,j), z(i+1,j), u(i,j), u(i-1,j), u(i+1,j), and u(i+1,j-1)
    enddo; enddo; enddo
```

So it can be seen with reference to figure 21 that the update to each z value depends on its neighbouring u's and v's, with updates to the u and v velocity fields requiring four v and u points respectively [*Ashworth and Davies*, 1991].

Identifying Communications

A parallel code can be perceived as a number of processes all of which are functioning individually. Hence during the execution of a parallel program there is need for communications to take place between the separate sequential processes. Initially a decomposition will take place of the full data domain into subdomains. After this stage the majority of messages to be passed between processes will involve the use of the finite difference scheme described above (identified in the code by 'cshift' statements). There also exists lines of code where single locations in arrays are is operated on individually, therefore only the processes that holds the corresponding values must be used. Intrinsic sum operations on arrays require data to be passed between the processes in such a way that after the operation all processes must hold a value of the sum performed upon the full domain. Finally after all the necessary calculations are complete each process must return its subdomain to a master process for writing of the final result.

Environment For Code Development

Workstations running a UNIX operating system were used to develop the code on the CRAY EL98 at Proudman Oceanographic Laboratory (POL). The message passing software used to develop the parallel code is Parallel Virtual Machine (PVM). PVM is a software system that enables a collection of heterogeneous computers to be used as a coherent and flexible concurrent computational resource [*Beguelin et al*, 1994]. PVM version 3.3.3 was obtained by sending electronic mail to *netlib@orln.gov*, and the received files PVM were installed on the CRAY EL98 (See Appendix Section 2). When installed on workstations a number of identical executable processes are spawned by the master process, then a daemon process handles all the necessary communications that need to take place between separate processes. This particular system is architecture independent, hence enabling the code to be ported easily between machines if necessary. PVM can be used with programs written in either C or Fortran, access to PVM is made by calls to PVM library routines for functions such as process initiation, message transmission and reception, and synchronisation.

Code development was in Fortran 90, and compilation of the program was performed by use of a makefile which kept track of those routines which were not up to date (See Appendix Section 3). Owing to limits placed upon interactive use upon the CRAY EL98, the code was executed as a batch job (See Appendix Section 3). Presently there is no Fortran 90 compiler expected on the T3D until mid 1995 (though Fortran 90 array syntax is supported), so after a working version was developed, a translation of the source code into Fortran 77 will take place.

The Cray T3D presents the user with a Single Program Multiple Data (SPMD) model for programming the machine. This means that for every processor in a set of processing elements on the T3D, holds a copy of the same executable, one per processor, and that each of these processes is started at the same time [*Booth et al*, 1994]. This contrasts with workstation PVM as no spawn command is needed or is available. Therefore for easier porting of the final code the parallel program will be written in a SPMD style. Each process in an SPMD program runs the same code, though they are not constrained to follow the same path through the program. Advantages of SPMD are the programs are of a simple structure, easy synchronisation, and that processes can break synchronisation if necessary. One draw back of programming in an SPMD style is that if the code contains tasks performing completely different roles, then each processor has to allocate instruction and data space it will never use (*Simpson*, 1994)

The T3D contains 256 DEC Alpha 21064 processors, and has a total of 16Gb of memory, the front end attached is a Y-MP with two processors. The nodes are configured as a 3D torus, with each node containing two processing elements. Transfer rates between nodes have a peak of 300Mbs^{-1} in all six directions.

Preliminary System Design

Decomposition Of Data

As the model is a "Global" tidal inverse problem, then the data used can be perceived as belonging to a longitude/latitude grid. Hence we must assume cyclic connectivity ie. 180°W is equal to 180°E. At the start of the code a decomposition of the full data domain into subdomains needs to be carried out. On the CM-200 the grid was mapped onto a rectangular array of processes (by compiler directives), therefore cshift statements were used to enforce the concept of cyclic connectivity. The T3D has fewer processors so the domain is to be partitioned onto a rectangular array of processes each possessing a subdomain.

The input and output of data fields covering the whole domain is not trivial when the domain is split up over a number of processes. There are two main options for approaching this problem. The first is the master process reads in the whole data set, then sends out the appropriate portions of data to each slave. When the work is complete each slave sends back its result to the master, which then writes the final results to disk. The second option is to arrange the format of data files so that the data for each subdomain are on different records of a direct access file. Each process can then read and write its own data in parallel. Option two was rejected as much shuffling of the data files would be required, and also this method is not scalable owing to different data files being required for different numbers and configurations of processes. Although the first idea is more complex to program, it is easily scalable, and has the advantage of keeping the data in the same order, this is a geometric decomposition of the data set. It would be desirable for the master process to divide up the whole domain amongst the slaves, ensuring that an equal part of the full domain is kept for itself. This is because if the master lay idle, doing no work except that of input and output of data this would require on odd number of processes (assuming the full domain is split into a rectangular array of processes) which is not allowed upon the T3D. Therefore this idea would allow a more simple porting of the final code to the CRAY T3D.

This distributed domain decomposition affects a few a operations, these are the shifting of arrays (cshift statements), any operations upon a single point in the domain, and global summations upon arrays (sum statements).

Shifting Of Arrays

The cshift statements used for the implementation of the finite difference scheme correspond to a communication between all the processes (See Figure 22). Here the result of a shift east on data partitioned into rectangular subdomains is shown.

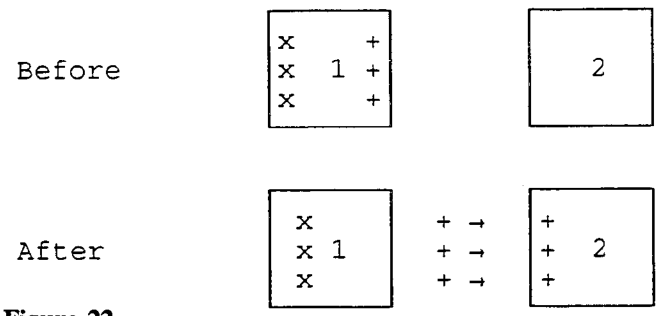


Figure 22

The 'x' symbol on process 1 will just be shifted east one place as, this is occurring within the interior of the subdomain, no communication between processes are required. However the '+' symbol initially present on process 1 is situated in the east column of the subdomain and is required by process 2 for the next value update. In this case a communication is required that will involve sending all the values in the right hand column of process 1 to the left column of process 2. Hence whenever a cshift statement occurs, a one dimensional slice of each subdomain will need to be sent to neighbouring processes which requires it. Therefore communications will occur only at the boundaries of each process. An efficient implementation of this strategy can be achieved by ensuring that each process holds copies of certain sections of the neighbouring process's data. Each process maintains a halo or guard band (See Figure 23).

The symbols show the values that would exist in the halos on each process. Using this concept of halo regions for each process implies that the data required for a cshift operation is immediately available without any communication needing to taking place. Though, after the guard band has been used for an update of values within the subdomain, the used guard band must then be updated itself with the value from the neighbouring process.

Figure 24 represents a shift of data in a westerly direction. Process 1 needs the data stored in its east halo in order to complete the operation, when the figures have been used process 2 will then update the eastern halo of process 1. It is not necessary to update the other three halo regions as shifting data in an west direction within

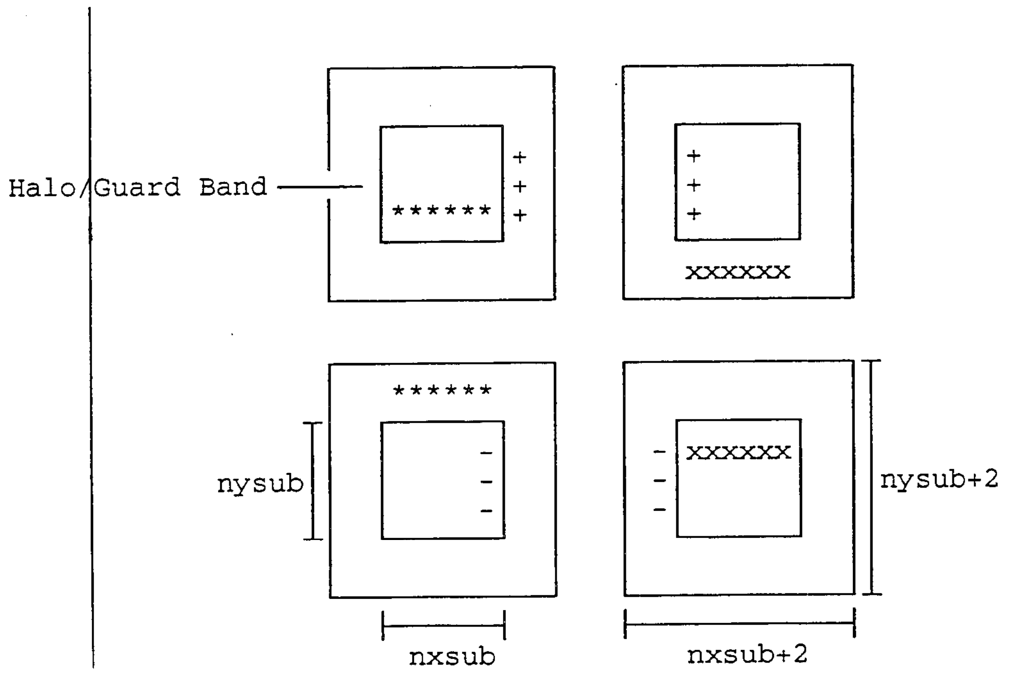


Figure 23

that subdomain only leaves the east halo out of date. Hence if during a calculation four cshift statements were used in north, south, east and west directions then it would be necessary to update all four guard bands of the subdomain.

Location Of A Single Point

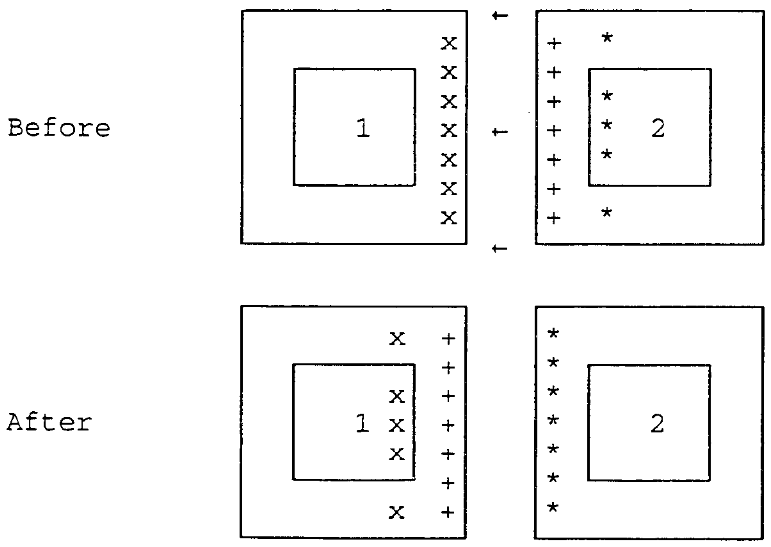


Figure 24

Contained in routines avgker, fwdfrcx, and fwdfrc are sections of code that operate open one single point in the full domain at a time. In the routine avgker a location of a point elevation easurement is considered, information is also given as to whether the point is to be treated as a boundary elevation or not, from this the forcing arrays for the back time stepping problem are written. A similar method is used in fwdfrcx and fwdfrc for constructing the forcing arrays for the forward time stepping problem. This will imply that just one process will do the relevant section of work whilst those remaining continue with other tasks. Hence it can be seen that a method of mapping a point in the full domain to the same point in the sub domain is required (See Figure 25).

If the work being carried out by the process owning the data point is of interest to the other processes then they will all wait for that particular slave to finish its task. When that process has finished the calculations, communications will need to take place in order to update those processes not aware of the new values. However the work may be of importance in a local sense only, if this is the case then the other processes can continue. It would be possible therefore, for each process to be synchronously carrying out a piece of useful local work on several co-ordinate points in the full domain.

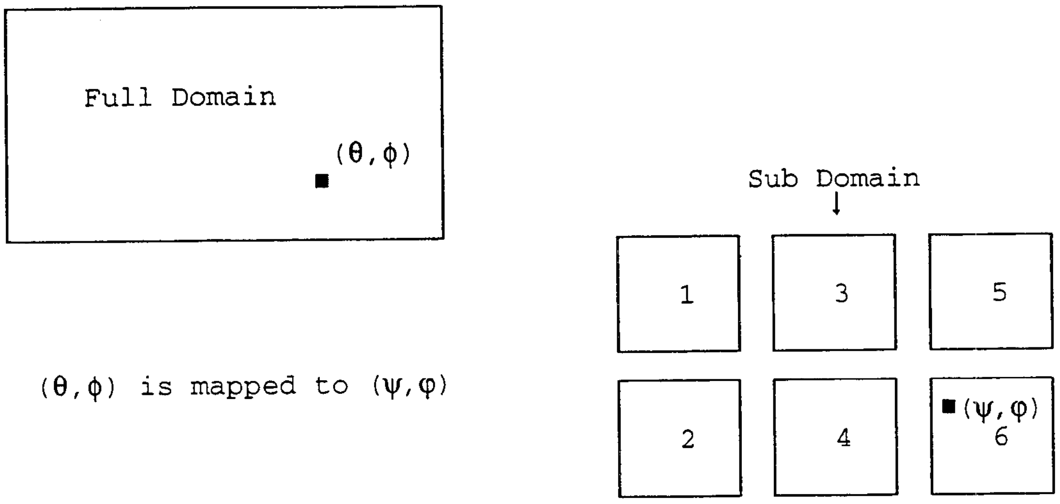


Figure 25

Global Summations

Places in the code where the Fortran intrinsic 'sum' is used upon an array indicate that a communication will have to take place between all the processes. Instead of operating in the whole subdomain each process should perform the intrinsic operation on just the interior area. This is because the same values should not be used in the global sum twice ie. those values stored in the guard bands are copies of other stored values elsewhere in the whole domain. As soon as each process obtains its local sum communications can begin.

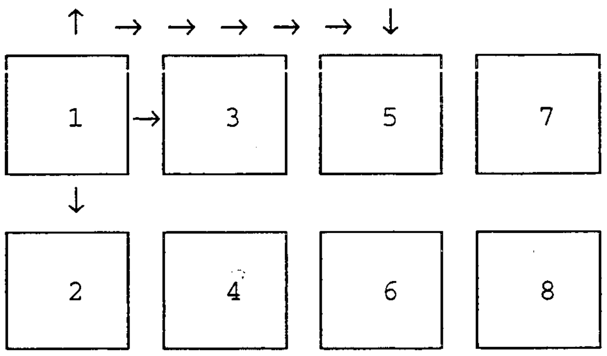


Figure 26

There are a number of ways in which a set of local sums can be accumulated into one global sum. One method is to send all the local sums to the master process, the summing of each result to attain the global sum is carried out just by the master. Then depending upon whether each slave needs a copy of this value the master may inform all other processes. Another method is for each process to send its local sum value to the next process in a circular fashion, when the neighbour receives the value it adds it to its running total then passes the value its just received on again. This continues until all processes have the same running total. One of the methods requiring less communications than the two mentioned previously is shown in Figure 26.

If in general there exist 2^n processes (where n=1,2,3,...) then a global sum can be calculated as follows: firstly each process sends its local sum one step right, after this the original value each process held is added to that which has just been received. Next each process sends its own newly calculated local total two steps right and repeats the summation with the current total and the value just received. For example in Figure 26 process 1 sends to processes 2, 3, and 5. In general for each cycle, t, process 'i' sends to process '2^{t-1}', where the first cycle is t=1. This then continues with the steps rising as a power of two until all process hold the global sum value needed.

The first method is easy to code and performs well for low numbers of processes. Though for large numbers of processes this is not a good idea especially if the global sum is needed for subsequent calculations. The second method is preferable as the sum is performed in a smaller number of steps, therefore less communications are needed. The third method uses the lowest number of steps, in the general case (for 2^n processes) a global sum could be held by all processes in nstep communications given by :-

where nproc is the number of processes.

$$nstep = \frac{\ln\ (nproc)}{\ln\ (2)}$$

Returning Of Results For Output

After all calculations are complete, each process must send back its result to the master for writing to disk. Effectively this will be the reverse process of the data decomposition. The difference being that just the interiors of each subdomain will be sent back, and the halo regions will be discarded. The master receives all the data necessary and stores it into arrays equal in size to that of the full domain.

In all the above designs for paralellisation of these operations the PVM communications details is separated from the computational kernels where possible, except in the case of operations upon a single location in the full domain for the routine fwdfrcx.

Implementation Strategy

Initialisation Of PVM

Within the parallel code the first task is to initialise PVM, this takes place at the beginning of the main program istep (Appendix Section 4). An immediate call is made to pvmfmytid This routine enrols that particular process into PVM, creating a unique task identifier (tid) for the calling process. The next step is to spawn a number of copies of the executable file. The number spawned is determined by how many processes are required, if 'nproc' processes are required, then 'nproc-1' will be spawned because the master process is included. Each of the new processes has its own tid which is used for identification. Task identifiers are essential in ensuring that each process takes the correct path through the code, and also are used when messages need sending between processes.

Setting Up Subdomains

Before the whole domain can be farmed out to each process it is necessary to set up the sub domain in the required shape. Given that the total number of processes used is one of the following integers;

$$nproc = n^x$$

where nproc is the number of processes, x = 1, 2, 3, 4, . . . , and n is given in figure 27 as

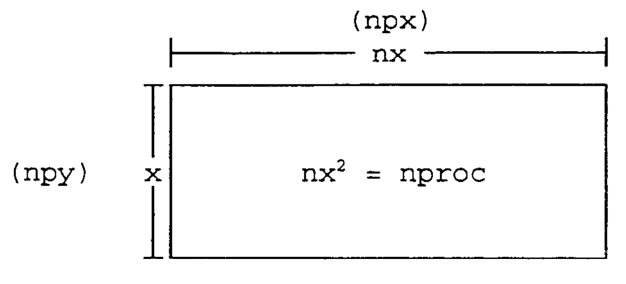


Figure 27

From the equation the number of processes needed in the horizontal, and vertical (npx and npy) can be calculated. The gridsize being used as a test case for calculation of the representers is of dimension 256x128 (ratio 2:1), hence nproc can be chosen to be 2, 4, 8, 16, etc., the processor array can therefore be formed to be in a ratio of 2:1. For development upon the CRAY EL98 eight processors were used giving a 4x2 array of processes.

Now that the shape of the process array has been established it is necessary to ensure that each process knows the identity of the processes handling the adjacent sub domains. Calculation of the neighbours is dependent upon the identification number of each process. From an array of tids (that holds each processes task identifier) each process was identified stating from zero (for the master) upwards to 'nproc-1'. A formula is then used to calculate the correct north, east, south and west neighbours using modulo arithmetic. (Note for this code, neighbours in the ne, nw, se, and sw directions are not needed as shifting of data is only in a horizontal and vertical sense.) The calculation also depends upon the number of processes in the vertical 'npy', and the total number 'nproc'. For an array numbered in the same way as figure 25, (vertically), the north, east, south, and west neighbours are calculated as follows (where me is 0, 1, 2, . . . nproc-1):-

```
north = me-1:  if(mod(me,npy)=0) then north = me+(npy+1)
south = me+1; if(mod(me,npy)=1) then          south = me-(npy-1)
```

```
east  = me+npy;      if(east > npy-1) then    east = east - nproc
west  = me-npy;      if(west < 0) then        west =west + nproc
```

The routines initialising the process array are in pvm_setup.f

Farming Out Of Full Domain

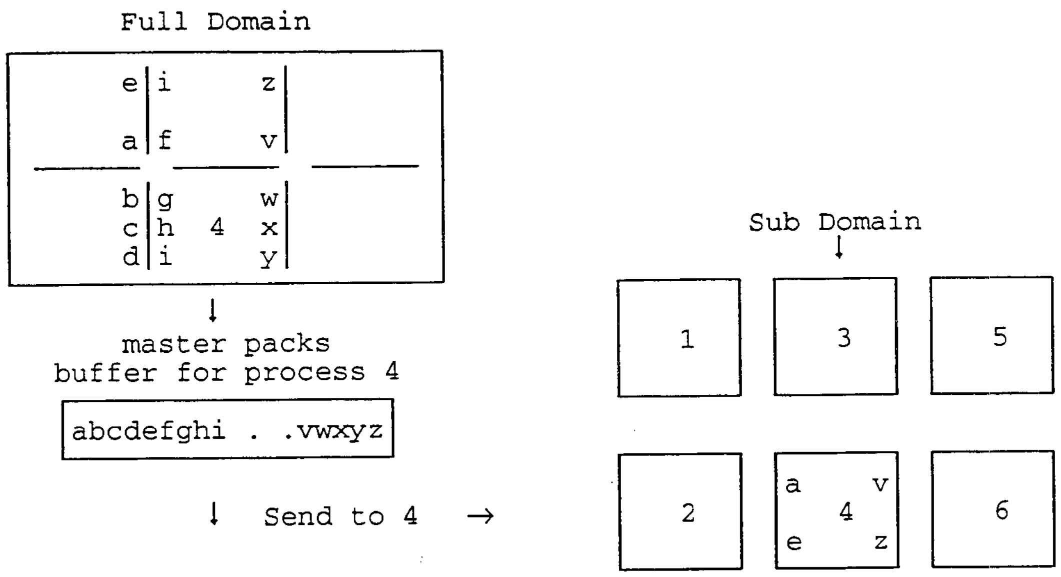


Figure 28

The process array has now been initialised into the correct dimensions, ensuring that each process knows its adjacent neighbours. The task of farming out the initial arrays can now start. Initially all processes must know how large their subdomains are, nxsub, and nysub horizontally and vertically. If the whole domain is nxm, then nysub=npy/n, and nxsub=npx/m, though two must be added to each of these to include the guard bands (see figure 23). Then knowing the size of each subdomain the master process locates the correct array section and sends it to the appropriate slave. The master does this by copying all of the correct elements for a particular process into a temporary buffer. This is a vector of length equal to the number of elements in the subdomain ie. (nxsub+2)*(nysub+2). The buffer is then sent to the corresponding process, which then unpacks the contents by copying the elements into its own sub array. This procedure is repeated for each process and for each array (See Figure 28).

Figure 28 illustrates that the master process copies to the buffer all the values for the guard band, as well as those for the interior region in the subdomain. As this is a global model it is correct to implement the subdomain such that the north and east edges of the whole domain are neighbouring to the south and west edges respectively. Hence if the data for a process is situated on the edge of the domain, then the master must copy halo values from the correct location on the opposite side of the array. For example in figure 28, elements 'e, i, z' from the north edge are used to make up the south part of the halo region for process number 4. A piece of pseudo code to perform this is written below:

```
if(master)then
  do iproc for all processes:
          do i for nxsub+2; do j for nysub+2
          ii=(i-1)*nysub+2+j
          buffer(ii) = element in full domain
          enddo; enddo
  send buffer to iproc
  enddo;
elseif(iproc)then
  receive buffer from master
  do i for nxsub+2; do j for nysub+2
          ii=(i-1)*nysub+2+j
          element in subdomain = buffer(ii)
  enddo;enddo
```

```
        endif
```

The section of code performing the farming out of the arrays is in pvm_farm.f (Appendix Section 10). The routine is called by all the processes after the main arrays have been read from the disk in program istep.f. The routine is called every time a new array is to be split up over the subdomain. Assuming that initially the best way to port a section of code to another machine is with the minimum of change, pvm_farm.f is used in the following way. As integer, real and complex arrays are used throughout the code three separate routines were formed, the following code fragment illustrates use of the routine:

```
if(iam.eq.0)then
        do j=1,mm
          do i=1,nn
                read(1,rec=irec)lmv(i,j),lfu(i,j)
                do ic=1,nc
                        read(2,'(2e10.4)')lcau(ic,i,j)
        enddo, enddo,enddo
endif
call farm_i(lmv,mv);
call farm_r(lfu,fu);
do i=1,nc
        call farm_c(lcau(i,:,:),cau(i,:,:))
enddo
```

In the above example, the first two calls to the farming routine are for two dimensional integer and real arrays respectively. The final call represents how the routine would be used upon a three dimensional complex array, slices of the array are sent out to the slaves one at a time. The first argument in the call is the array name for the full domain, and the second is the name used for the sub domain. The name of the arrays containing the full data set were changed (an 'l' was added to the front of the name to indicate large array (first argument)). But the original names of arrays used throughout the code remained unchanged and were used for the sub arrays (second argument). This routine is also used for farming out data in the subroutine mkwts.

Communications Required For Cshift Statements

Implementations of the cshift statement (in both the Fortran 77 and 90 message passing codes) depends upon each process knowing its four adjacent neighbours in the north, south, east, and west directions, also the direction and dimension of the shift must be known. The first use of a cshift statement is inside the routine mkwts (for making the integration weights). From the previous section it is clear that a shift of data in a particular direction requires the opposite guard band to be updated (shift to north implies update southern halo). Before any implementation can take place it is important to realise that all Fortran compilers access elements of an array columnwise. This has the following implications See Figure 29

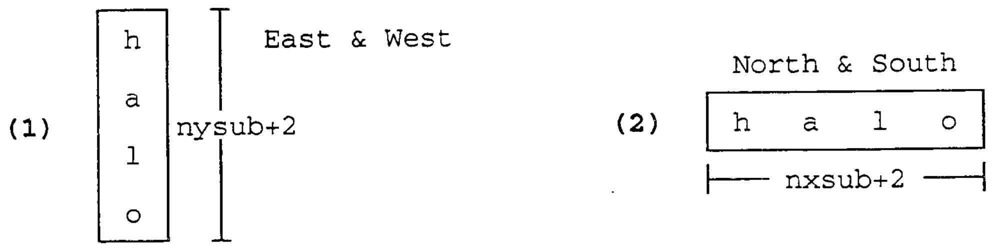


Figure 29

Any shifts in an East/West direction will involve sending a one dimensional matrix of dimension (nysub+2,1), illustrated above by (1). However a North/South shift of data requires a (1,nxsub+2) dimensional matrix to be sent. PVM allows data to be sent with a stride defined, in the first case this is not necessary so a stride of one is used, but in the second example it is. For a North/South halo update the stride will be equal to the dimension of the subdomain in the vertical, ie. nysub+2. So to ensure that the correct values of an array are sent differing strides were used in the west/east and north/south updates:-

usage:	call pvmfpack(type,name,length.stride,info)
eg:	call pvmfpack(REAL4,array(2,1),nysub+2,1,info) (west)

```
call pvmfpack(REAL4,array(2,1),nxsub+2,nysub+2,info)    (north)
```

The above code tells the process to pack a particular piece of data in order to send it to another process; REAL4 is the type of data we are wishing to send, array(2,1) is the start address of the data, the next argument is the number of items to be packed (the length of the halo), next is the entry which declares the stride, and finally info which returns a negative value if an error has occurred. After the data has been packed all the processes send it in the desired direction. Next each process receives the data being sent and copies it to the correct guard band in its subdomain. All the routines for the halo updates can be found in pvm_comms.f (Appendix Section 11), as with the routine used for division of the full domain a separate piece of code was written for handling real, complex and integer arrays. Implementation into the code has the following structure:

```
hu(:,:) = (hz(:,:) + cshift(hz.dim=1,shift=-1))/2.
call one_minus_one_r(hu)
```

Here the real array hu is updated by use of a shift south, so immediately afterwards (before hu is used in any further calculations) the halo is updated by use of the appropriate subroutine. The cshift statement is used extensively throughout the code. In particular the back and forward time stepping routines (Appendix Sections 6, and 8), for the test case being looked at the main time stepping loop was 2000 iterations. Therefore calls to the routines stored in pvm_comms.f account for the majority of communications during the calculation of the representers.

Location Of A Single Point

Before a position in the whole domain can be mapped to a point in the subdomain the process holding the entries of the arrays of interest must first be located. This can be performed by using the co-ordinate values in the full domain in conjunction with the dimensions of the subdomains. One implementation is to check the first column of processes, if the global co-ordinate is not located here move to the second column and so on. As soon as the correct process has been located the mapping will be identical for all processes, again dependent upon the size of the subdomain. The mapping used is as follows (adopting the notation from the previous section; (θ,ϕ) is mapped to (ψ,φ)).

$$\psi : \psi \mapsto \text{mod}\ (\theta,\mathit{nysub})+1$$
$$\varphi : \varphi \mapsto \text{mod}\ (\phi,\mathit{nxsub})+1$$

The only values of θ and ϕ for which both of the above mapping gives an incorrect result are θ = i*nysub, and ϕ = j*nxsub (where i,j=1,2,3 ...). If this situation arose then this would cause a mapping of θ and ϕ to a value of one in each case. In physical terms (θ,ϕ) equalling (i*nysub,j*nxsub) implies that the point in the full domain actually lies in the last row, and or last column of the interior of the subdomain. So for a sub domain of dimension (nysub+2, nxsub+2), the correct mapping would be $(\theta,\phi) \mapsto$ (nysub+1,nxsub+1). This can be easily implemented using 'if' statements.

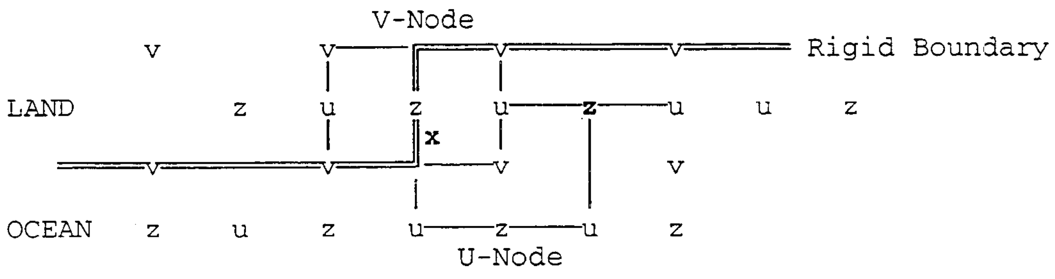


Figure 30

The updating of single points at a time is used in three routines, avgker, fwdfrcx, and fwdfrc. The next consideration is whether any results from calculations being performed by the sole process are needed in by other processes. In both fwdfrcx, and fwdfrc all the calculations made effect only the arrays held upon the working process. However in the avgker routine (that outputs the forcing and boundary conditions for the backward problem) this is not the case. For a point on the coastal boundary, two u and v node squares are considered (See

Figure 30).

The point being considered is marked as x above (this applies to any point on the boundary that lies between two u and v node squares). The boundary node x is interpolated from an average of any and all of the eight u and v nodes lying on the rigid boundary, shown above by the two squares. The possibility could arise that the u-node values and the v-node values lie on different processes. If this is the case, when the weights are calculated for bilinear spline interpolation the two separate processes must share the calculated weights to reach the final answer. This would take the following form in pseudo code wtot_u and wtot_v are respective weights totals for the processes dealing with node u and v:-

```
if(slave_u is not equal to slave_v)then
  if(slave_u)then
          calculate wtot_u total and send to slave_v
          receive wtot_v from slave_v
          update my forcing array with wtot_u+wtot_v
  endif
  if(slave_v)then
          calculate wtot_v total and send to slave_u
          receive wtot_u from slave_v
          update my forcing array with wtot_v+wtot_u
endif; endif
```

Global Sums

Intrinsic sum functions are used inside the fwdfrcx and fwdfrc routines to temporarily calculate the contribution of the penalty functional to the boundary errors. As described above implementation of this is by ensuring that each process performs the intrinsic sum of the interior of its subdomain. After, a call will be made to a routine which performs the operation described in figure 26. It is used in the following way which each process executes:

```
pbdry = sum(bu(1,2:n-1,2:m-1)*conj(bu(1,2:n-1,2:m-1))*wbu*usc(1,2:n-1,2:m-1) *usc(1,2:n-1,2:m-1)
call sum_r(pbdry)
```

the argument passed to the routine is each processes local sum, see pvm_sum.f (Appendix Section 12).

Sending Back Arrays To The Master

Implementation of this is very similar to the routine that divides up the full domain, the major difference being that at this stage the halo regions are discarded, just leaving the interior regions where the important results needed are stored. Initially each process copies its interior region into a buffer (now containing (nxsub*nysub) elements). Each slave then sends its work buffer back to the master. The master process receives the message from the slaves in order, so that it can correctly unpack the information to construct a result equal in dimension that of the full domain. This has the following structure:-

```
do i = 2,nxsub+1
        do j=2,nysub+1
        buffer((i-2)*nysub+j-1) = subarray(j,i)
enddo; enddo;
if(iam not the master)then
        send buffer to master
        wait for reply from master
elseif(iam the master)
        copy my buffer to array
        do iproc=1,nproc-1
                receive buffer from iproc
                copy iproc's buffer to array
        enddo
        send message to slaves
endif
```

An important addition to this routine is that all the slaves wait to receive an acknowledge from the master process after their work has been sent. The master sends the acknowledge after it has received all the necessary data. If this extra code was omitted then some of the worker process could send the information, and exit the code before the master had received it. Without this acknowledge it was found that the code would sometimes 'hang', as the master process would be trying to receive data from one of its slaves that no longer existed, in this situation the message would never arrive, and the master would wait for ever. The routines for sending back arrays to the master are contained in the file pvm_return.f (See Appendix Section 13) and are used in the reverse sense compared to the farming out of data.

```
do ii=1,nc
        call send_back_c(caz(ii,:,:),lcaz(ii,:,:))
enddo
```

The first argument is the name of the subdomain, and the second is the name of the full domain which just the master process will hold.

Conversion To Fortran 77

The major change to the code that is required in order to enable porting of code to the CRAY T3D is the replacement of the Fortran 90 intrinsic cshift (discussed above). No PVM syntax needs to be used in this routine as each time a cshift statement is reached all the necessary process control has been set up previously. The routine simply copies the correct slices of the original array to a temporary array to form the shifted version. Implementation of this routine was also achieved by introducing some temporary arrays (all temporary arrays were named after the original array followed by a 't'), and is used as follows.

```
ujm(:.:) = cshift(mu(:,:),dim=2,shift=-1)*ujm(:,:)
call cshift_i(mu(:,:),mut(:,:),2,-1)
ujm(:,:)=mut(:,:)*ujm(:,:)
```

The line of code on the left becomes the two lines of code on the right when the cshift statement is replaced. As with previous routines the '_i' represents an integer array. The first argument is the array to be shifted, the routine returns the shifted array 'mut' to be used in the calculation. The final two arguments correspond to the dimension and shift direction to be used. The cshift replacements are in file 'cshift.f' (Appendix Section 14).

The next task is the replacement of any variable declarations using the double colon (::) syntax illustrated below,

```
complex, dimension (nc,n,m) :: cau, cav, caz
```

is replaced with,

```
complex cau, cav, caz
dimension cau(nc,n,m), cav(nc,n,m), caz(nc,n,m)
```

Also parameter declarations must be replaced with Fortran 77 syntax. Finally at the start of back and forward time stepping routines, a double precession complex declaration is made upon a vector. This is also not supported on the T3D, therefore a single precision declaration must be used instead.

Using Different Numbers Of Processes

All of the above implementations are designed for in general any number of processes (provided the full domain can be divided up equally amongst all processes). There are two files that must be changed to vary the number of processes, these are parallel.inc, and gridsize.inc (See Appendix Section 1). In parallel.inc there are some parameter statements, NPROC is the number of processes to be used, nx, and ny which were the sizes of the full domain. Changing NPROC this tells the main program to spawn more or less processes, nx, and ny are altered for different sized data sets. The setting up the dimensions of the process array is explicitly carried out in the parallel.inc by setting npx and npy, on porting to the T3D the factorisation routine will be used in place of this. The file gridsize.inc contains the dimensions of the sub domains to be used these are 'n' and 'm' (the original dimension of the full domain in the serial version) where m=nxsub+2 and n=nysub+2. The size of the full domain is represented by nn and mm. These parameters need to be considered when a different data set of number of processes are going to be used.

Debugging and Testing

Debugging and Testing Techniques

All the debugging and testing carried out upon the code was with the use of 'write' statements. In the final code just the master will handle Input and Output (I/O), but for development the slaves were also used for writing information to files.
Using the above code, if slave 3 (iam=3) writes to unit 50 this has the effect of producing an output file called 'debug.3out' which contains only data which slave number 3 has written. Updates on the arrays were tested by using routines in 'pvm_return.f' ie. sending the appropriate array to the master for writing to a file. This is coded as follows:

```
do ii=1,nc
        call send_back_c(caz(ii,:,:),lcaz(ii,:,:))
enddo
if(iam.eq.0)then
        open(77,file='caz.par')
        write(77,*)lcaz(1,:,:)
endif
```

The above section of code was used extensively throughout the testing process. For comparison with the original code the same code fragment appeared in the serial version at the same point, the two output files were then compared. Finally the change from double precision complex to single precision complex, (needed in the parallel code) was also carried out for the serial version to ensure that the same precision was used throughout both pieces of code, this is important when comparing results between the two versions.

Debugging & Testing Results

Initially all development of the PVM version upon the CRAY EL98 was carried out with eight processors. Using four vertically and two horizontally, the dimension of the subdomains on each process was 66 by 66. This number of processes was considered a 'safe' maximum number allowed upon the CRAY EL98, as for numbers greater than eight (sixteen was tested) some of the processes were not able to spawn (due to limits imposed on the CRAY EL98 at Bidston). This left messages not being sent or received. Implementation was carried out in the order described above, though initially the Fortran 90 cshift statements were not replaced until after the first full run through of the parallel version (in Fortran 90) was complete. The first changes were made to the main program istep, and then to the routines in the same order they are called by the main program.

Initial conclusions were that differences in the ninth significant figure had accumulated over the 2000 iterations of the time stepping loop to the extent of that observed in the outputs from this routine (differences in the first significant figure). This seemed a reasonable conclusion as the forcing arrays also contain the fixed boundary conditions at the open boundary nodes, so a change in the entries to these arrays also has the effect of slightly changing the boundary conditions for the problem. To verify this theory a number of tests were carried out which involved inter changing data sets between the original code and the parallel version.

The first test was to use forcing arrays from the original code as inputs to the forward time stepping routine in the parallel version. It was found that the results obtained were extremely close compared to the original results, differences in the twelfth significant figure were observed. This confirmed that the parallel implementation of this particular routine was correct. The next test involved using the outputs of the complex forcing arrays from the parallel routine as inputs for the tstep routine in the serial version. This test produced results similar to the original data also, where differences were being found in the fifth significant figure, not the first as expected. The results of this test disprove initial conclusions.

As results of a reasonable accuracy (compared to the original result) had been obtained from both the parallel and serial codes by reading the forcing arrays from the parallel code into the serial version and vice versa. The next logical test was to run the parallel code again, but instead writing gu, gv, and gz to a file to be read back in before the first reference to them. Immediately after the last update of the forcing arrays (for tstep) gu, gv, and gz were written to a separate files. Each of these files were read back into the code by the master process and farmed out to the slaves, each process then continues with the rest of the routine as previously. For a comparison of the output of the results obtained from the serial code (using the forcing arrays output from the parallel version), differences were observed in the sixth significant figure. A close match was also observed for the original code using the original forcing arrays, differences were only in the fifth significant figure. The table below summarizes the results found from this series of tests:

Code Type & Results Used	Serial Code Inputs from Parallel Code	Parallel Code Inputs from Serial Code	Parallel Code Inputs from Parallel Code	Parallel Code No Inputs (Straight Run)
Serial Code (Original)	Difference in 5th Sig. Fig.	Difference in 12th Sig. Fig.	Difference in 5th Sig. Fig.	Difference in 1st Sig. Fig.
Serial Code Inputs from Parallel Code	N/A	Difference in 5th Sig. Fig.	Difference in 6th Sig. Fig.	Difference in 1st Sig. Fig.

The above testing has revealed that the parallel code gives the correct answer only when the forcing arrays gu, gv, and gz are output to a file and read back in again before they are referenced. This suggests that the arrays gu, gv, and gz are being overwritten somewhere between the two following code sections:

1. repx_par.f (Appendix Sec. 4)

```
if(lcrosscov)then
        call fwdfrcx( )
else
call fwdfrc( )
endif
```

2. tstep.f (Appendix Sec. 8)

```
    do istep = istep1,istep2
       if(mod(istep,2).eq.1)then
             force(:,:) = 0
    do 350 l = 1,nc
350 force(:,:) = force(:,:) + real(gu(l,:,:))
```

There are no references to gu, gv, or gz between the two code fragments shown above. Fragment number one represents the last update of gu, gv, and gz. The second fragment indicates where the next reference to any of these three arrays is made(in forward time stepping loop for odd steps). Comparison of the forcing arrays used as input to reach the correct answer, and those giving the incorrect answer reveal no difference between the two data sets.

All of the above results obtained were for eight processes, the outputs are identically reproducible using four processes on the Cray EL98. This was found in testing that the code runs for nproc number of processes where nproc=2^n and n=1,2,3,... .

Conclusion on parallel implementation using PVM

Development of the parallel code was achieved by use of PVM and Fortran 90 upon the CRAY EL98 at POL. The full domain of the model was divided by the master process into sub domains containing halos over an array of slave processes. The master also held a subdomain of its own so each process performed some "useful" work during the execution of the program. Necessary communications were then implemented within the code for operations requiring data to be sent between processes. These operations were cshift statements, global summations and the update of a single location in the domain. After all necessary calculations were complete, each slave sends its result back to the master process for writing to a file.

This method resulted in a parallel code that obtained the same results to the original serial code within an accuracy of five significant figures (for in general 2^n processes; 4 and 8 processes were tested). For large programs, the order in which instructions are accessed by a compiler can sometimes lead to small changes in results for the same operations. After insertion of code for process and communications control small changes in the values of the forcing arrays used in the forward time stepping routine were observed. This first piece of code where the parallel version's results deviate from those of the serial code is probably due to the compiler operating on instructions in a different order. After implementation of the communications, replacement of the cshift statement for conversion to Fortran 77 was carried out. This lead to deviations being observed at an earlier stage in time after completion of the back time stepping routine. Therefore using a subroutine for replacement of the cshift statement seems to lead to more implications of accuracy at the level of the compiler.

The code currently reaches an answer to within five significant figures only when the forcing arrays used in the forward time stepping routine are; written to a file after the last update, and read back into the code just before the main time stepping loop. Otherwise the answers obtained differ from the original results in the first significant figure. Tests carried out were inconclusive, as no differences were found between those forcing arrays output to files, to those used in an execution of the code giving incorrect answers. This problem may be caused by an overwriting issue, but due to time considerations the bug was not located.

The Fortran 90 code was converted into Fortran 77 ready for compilation and running upon the CRAY T3D in Edinburgh. Due to the Fortran 77 compiler not allowing array slices to be passed to subroutines as arguments, there was insufficient time to change the code to enable a measure of performance of "repx" upon the T3D. This is the first recommendation for further work that needs to be completed by assigning temporary two dimensional arrays to each slice of the three dimensional arrays to allow compilation.

The next consideration is for more testing of this code to be carried out. The bug causing problems with the forcing arrays is an important issue as the input and output of such massive files (gu, and gv are 3.5 Mb in size and gz is 27.5Kb) limits performance. There also remains a section of code that has possibilities for parallelisation. After the time stepping loops in the routines tstep, and bstep have been completed a Cholesky decomposition and solve takes place. Currently this section of the code is handled by the master process, but improvements in performance could be gained by parallelisation, one approach would be try and locate a message passing routine (all ready written) to "plug-in" or adapt for the Cholesky decomposition and linear solver.

Once successful porting of the code to the T3D has taken place, some time spent on optimisation the code would be beneficial in producing significant speed up. Work also needs to be started upon the two remaining codes (still coded for the CM-200 connection machine). These are for estimating the coefficients of the representers, and finding an improved solution. The second program contains a CMSSL library call for a matrix inversion. This library call will have to be ported to an equivalent message passing parallel operation upon the Cray T3D.

Finally perhaps a more natural way for this Fortran 90 Global Tidal Model to be ported to the Cray T3D is to use Cray Research Adaptive Fortran (CRAFT). CRAFT uses Cray Fortran 77 and compiler directives, it also has some Fortran 90 extension such as array syntax and intrinsics, However this compiler on the Cray T3D is not yet available.

References

Alcock, G.A. and D. E. Cartwright, Some experiments with orthotides, *Geophys. J. R. Soc., 54*, 681 - 696, 1978.

Andersen, O. B., Ocean tides in the northern North Atlantic Ocean and adjacent seas from ERS 1 altimetry, *J. Geophys. Res., 99*(C11), 22557-22573, 1994.

Andersen , O. B, Global Ocean Tide Inversion Using the CM-200 Parallel Computer. Seminar, POL-Bidston, June 10, 1994.

Andersen, O. B., Global ocean tides from ERS 1 and TOPEX/POSEIDON altimetry, *J. Geophys Res.*, In Press, 1995a

Andersen, O. B., Adjusting ocean tide models from TOPEX/POSEIDON altimetry, *Geophys. Res. Lett.,* Submitted, 1995b.

Andersen, O. B., Modelling tides, In: New views of the Earth, Scientific achievements of ERS-1 *European Space Agency Sepc. Publ., ESA SP1176*, 23, 1995c.

Andersen, O. B., P. L. Woodworth and R. A. Flather, Intercomparison of recent global ocean tide models, *J. Geophys. Res.,* submitted, 1995.

Andersen, O. B., M. Ashworth and S. Wilkes, Integrating hydrodynamics and direct observations in global ocean tide inversion (abstract), *Annales Geophys., 13*, C252, 1995.

Archivage, Validation, Interpretation des donnees des Satellites Oceanographiques (AVISO), AVISO User Handbook: Merged TOPEX/POSEIDON Products, *Publ. AVI-NT-02-101-CN,* Touluse Cedex, France, ed. 2.1, 1992.

Ashworth, M. and A. M. Davies, Restructuring Three Dimensional Hydrodynamic Models For Computers With Low and High Degrees of Parallelism. *Parallel Computing*, 553-560, 1991

Baker, T. F., D. J. Curtis and A. H. Doodson, Ocean tide loading and GPS., *GPS world,* 54-59, March issue, 1995

Bettadpur, S. and R. Eanes, Geographical representation of radial orbit perturbations due to ocean tides: Implications for satellite altimetry, *J. Geophys. Res, 99(C12)*, 24883-24894, 1994.

Beguelin, A., J. J. Dongarra, G. A. Geist, W. Jiang, R. Manchek, K. Moore, and V. S. Sunderam, *The PVM Project*, 1994

Bennett, A. F., *Inverse methods in Physical Oceanography*. Monographs on mechanics and applied mathematics, Cambridge University press, Cambridge, 346pp., 1992.

Bennett, A. F., and P. C. McIntosh, Open ocean modelling as an inverse problem, tidal theory, *J. Phys. Oceanogr.*, 12, 1004-1018, 1982.

Booth, S., A. Simpson, C. Brough, *Notes on Porting PVM Codes To The T3D*, EPCC, August 19, 1994

Callahan, P., *TOPEX/POSEIDON Project GDR Users Handbook,* JPL D. 8944, rev. A, Distributed Active Archive Center, JPL, Passadena, 1993.

Canceil, P., R. Agelou and P. Vincent, Barotropic tides in the Mediterranean sea from a finite element numerical model, *J. Geophys Res.*, in press, 1994.

Cartwright, D. E., Oceanic Tides, *Rep. Prog. Phys. 40*, 666-708, 1977.

Cartwright, D. E., Detection of tides from artificial satellites (review), in *Tidal Hydrodynamics*, edited by B. B. Baker, pp. 547-568, John Wiley, New York, 1991.

Cartwright, D. E., Theory of ocean tides with application to altimetry, in *Satellite Altimetry in Geodesy and Oceanography.* Lecture Notes Earth Sci. Ser., vol. 50, edited by R. Rummel and F. Sanso, pp. 100-143, Springer-Verlag, New York, 1993.

Cartwright, D. E. and A. C. Edden, Corrected tables of tidal harmonics. *Geophys J. R. Soc.*, 23, 253-264, 1973.

Cartwright, D. E. and E. Taylor, Tables of tidal harmonics. *Geophys J. R. Soc.*, 23, 100-105, 1973.

Cartwright, D. E. and R. D. Ray, Oceanic tides from Geosat Altimetry. *J. Geophys. Res.* 95 (C3), 3069-3090, 1990.

Cartwright, D. E. and R. D. Ray, Energetics of global ocean tides from Geosat altimetry, *J. Geophys. Res.*, 96, 16897-16912, 1991.

Cartwright, D. E. and R. D. Ray, On the radiational anomaly in the global ocean tide with reference to satellite altimetry, *Oceanologica Acta*, in press, 1994.

Cray Research, Cray T3D, Technical Summary, *Cray Research,* Incorporated, 1993.

Desai, S. D. and J. M. Wahr, Another Ocean tide model derived from TOPEX/POSEIDON altimetry, (abstract), *EOS, 75(46)*, 57, 1994.

Eanes, R. J., Diurnal and semidiurnal tides from TOPEX/POSEIDON altimetry, (abstract), *EOS, 75(16)*, 108, 1994.

Egbert, G. D., A. F. Bennett, and M. G. G. Foreman, TOPEX/POSEIDON tides estimated using a global inverse model, *J. Geophys. Res. 99(C12)*, 24821-24852, 1994.

Fisher, J., The Cray T3D Hardware and Software, *Course Notes at EPCC*, 1994.

Flather, R. A., Results from a model of the northeast Atlantic relating to the Norwegian Coastal Current. In *The Norwegian Coastal Current*, edited by R. Saetra and M. Moerk, Vol 2, pp. 427-458, Bergen, Norway, 1981.

Flather, R. A., A numerical model investigation of tides and diurnal-period continental shelf wave along Vancouver Island. *J. Phys. Oceanog., 18*, 115-139, 1988.

Gjevik, B. and T. Straume: Model simulations of the M_2 and K_1 tide in the Nordic Seas and the Arctic Ocean. *Tellus, 41A*, 73-96, 1989.

Gjevik, B., E. Nøst and T. Straume: Model simulations of the tides in the Barents sea. *J. Geophys. Res. 99*, (C2), 3337-3350, 1994.

Glorioso, P. D. & R. A. Flather, A barotropic model of the currents off southeastern South America. *J. Geophys Res.*, in press, 1994.

Groves, G. W. and R. W. Reynolds, An orthogonalised convolution method of tide prediction. *J. Geophys. Res, 80*, 4131 - 4138, 1975.

Hendershott , M., The Effects of Solid Earth Deformation on Global Ocean Tides. *Geophys. J. R. Astram. Soc. 29*, 389-403 1972

Hendeshott, M., *Long Waves and Ocean Tides. Evolution of Physical Oceanography.* MIT Press , Cambridge. 1988

Harding, T. J. and R. Wait, A Shallow Sea Model on Transputer Arrays. *Technical Report., University of Liverpool*, December 12, 1994.

Jackson, D. D., The use of a priori data to resolve non-uniqueness in linear inversion. *Geophys.. J. R. astr. Soc., 57*, 137-157, 1973.

Jourdin, O. F., F. P. Vincent and P. Mazzega, Some Results of Heterogenous Data Inversions For Ocean Tides. *J. Geophys. Res. 96*, 20267 - 20288, 1991.

Knudsen, P., Altimetry for geodesy and oceanography. In *Geodesy and Geophysics. Lecture notes for NGK autumn school 1992*, edited by J, Kakkuri, Pub. Finnish Geod. Inst., Helsinki, 1993.

Kowalik, Z. and A. Y. Proshutinsky: Diurnal tides in the Arctic Ocean. *J. Geophys. Res., 98* (C9), 16449-16468, 1993.

Le Provost, C., M. L. Genco, F. Lyard, P. Vincent, and P. Canceil, Spectroscopy of the world ocean tides from a finite-element hydrodynamic model. *J. Geophys Res. 99*(C12), 24777-24797, 1994.

Ma, X. C., C. K. Shum, R. J. Eanes and B. D. Tapley, Determination of Ocean Tides from the First Year of TOPEX/POSEIDON altimeter measurements, *J. Geophys. Res., 99*(C12), 24809-24820, 1994.

Mazzega, P: M2 model of the Global Ocean tide derived from SEASAT altimetry, *Marine Geodesy, 9* (3), 335 - 363, 1985.

McIntosh, P. C., and A. Bennett, Open ocean modelling as an inverse problem: The M2 tides in Bass strait, *J. Phys. Oceanography*, 14, 601-614, 1984.

Molines, J. M., C. Le Provost, F. Lyard, R. D. Ray, C. K. Shum and R. Eanes, Tidal corrections in the TOPEX/POSEIDON geophysical records, *J. Geophys. Res., 99*(C12), 24809-24760, 1994.

Moritz, H. *Advanced Physical Geodesy*, Wichmann, Karlsruhe, Germany, 1989.

Munk, W. H. and D. E. Cartwright, Tidal spectroscopy and prediction. *Philos. Trans. R. Soc. London*, Ser. A, 259, 533 - 583, 1966.

Parke, M. E., R. H. Steward, D. L. Farless and D. E. Cartwright, On the choice of orbits for an altimetric satellite to study ocean circulation and tides. *J. Geophys Res., 92*, 11693-11707, 1987.

Parker, R l., L. Shure, and J. A. Hilderbrand , The Application of Inverse Theory to Seamount Magnetism., *Rev. Geophys 25*, 17-40, 1987.

Rapp, R. H., Y. M. Wang and N. K. Pavlis: The Ohio State 1991 geopotential and sea surface topography harmonic coefficient models. *Report 410, Dept. Geod. Sci. Surveying, Ohio State University*, Columbus, 1991

Ray, R. D. Global ocean tide models on the eve of TOPEX/POSEIDON, *IEEE Trans. Geosc. & Rem. Sensing, 31*(2), 355-364, 1993.

Ray, R. D. and B. Sanchez: Radial deformation of the Earth by oceanic tidal loading, *NASA tech. Memo,* *100743*, Goddard Space Flight center, Greenbelt, Maryland, 1989.

Ray, R. D., B. Sanchez, and D. E. Cartwright, Some extensions to the response method of tidal analysis applied to TOPEX/POSEIDON. (abstract), *EOS, 75(16)* 108, 1994.

Sanchez B., and N. Pavlis, The estimation of main tidal constituents from TOPEX/POSEIDON altimetry data using a Proudman function expansion, *J. Geophys. Res.*, submitted, 1995.

Schrama, E. J. O., and R. D. Ray, A preliminary tidal analysis of TOPEX/POSEIDON altimetry, *J. Geophys. Res., 99*(C12), 24799-24808, 1994.

Schlax, M. G. and D. B. Chelton, Aliased tidal errors in TOPEX/POSEIDON sea surface height data, *J. Geophys. Res., 99*(C12), 247´61-24776, 1994.

Schwiderski, E. W., Ocean tides, Part 1, Global ocean tidal equations. *Mar. Geodesy., 3*, 161 - 217, 1980a.

Schwiderski, E. W., Ocean tides, Part 2, A hydrodynamical interpolations model. *Mar. Geod., 3*, 218 - 257, 1980b.

Simpson , A. D., Parallel Programming on The Cray T3D, *Course Notes at EPCC.*, 1994.

Smithson, M. J.: Pelagic tidal constants. Vol. 3. IAPSO, *Publication scientifique*, No. 35, 1992.

Thomas, J. P. and P. L. Woodworth: The influence of Ocean Tide Model Corrections on Geosat Mesoscale Variability maps of the North East Atlantic. *Geophys. Res. Lett.*, 17(13), 2389-2392, 1990.

Tscherning, C. C., *Functional methods for gravity field approximation*, In Mathematical and numerical techniques in physical geodesy. Lecture Notes Earth Sci. Ser., vol. 7, edited by Hans Sunkel pp. 3-49, Springer-Verlag, New York, 1986.

Tscherning, C.C., R. Forsberg, P. Knudsen, The GRAVSOFT package for geoid determination, Proc. *First workshop on the European Geoid*, Prague, may, 1992

Tsimplis, M. N., and P. L. Woodworth, The global distribution of the seasonal sea level cycle calculated from coastal tide gauge data. *J. Geophys. Res.*, in press, 1994.

Wagner, C. A.: How well do we know deep ocean tides ? - An intercomparison of altimetric, hydrodynamic and gage data. *Manus Geod., 16*, 118-140, 1991.

Wahr, J. M., Deformation of the Earth induced by polar motion, *J. Geophys. Res., 90*, 9363-9368, 1985.

Wilkes, S., *Parallelisation of a global tidal model for modern high performance computing systems*, Masters dissertation of scienc in advanced scientific computations in the faculty of science, at the university of Liverpool, 1995.

Woodworth, P. L., and J. P. Thomas, Determination of the major semidiurnal tides of the Northwest European continental shelf from Geosat altimetry. *J. Geophys. Res., 95*(C3), 3061-3068, 1990.