EPCC-SSP-1997

Parallelisation Option for the Core-Mantle Boundary Flow Code ITERTRAN

Elena Degoli

Abstract

ITERTRAN is a geophysical code which implements one of the models existing to calculate the flow of electrically conducting fluids at the Core-Mantle Boundary using measurements of variations in the Earth's magnetic field at the surface.
This project concerned the parallelisation of the serial code through a task-farmed approach over independent timeslices.
This parallelisation is realized at a high level in the code. Through the storage in memory of some input data we have the possibility of reducing total memory requirement for each processor in the parallel code with respect to the relatively large requirement for only one processor in the serial code. An alternative to the task-farmed approach is a domain decomposition model for the parallel code and a combination of these two approaches will be interesting in order to improve the performance of the code.

Contents

List of Figures

List of Tables

1 Introduction

This document, after a description of the geophysical problem and an overview of the sequential geophysics code ITERTRAN, describes a high-level task-parallel implementation of that code. As part of the task-farm design there is also an analysis of I/O patterns of the original program and a description of the parallel reduced one.
This work continues with a discussion of the implementation of parallel features, describing also problems encountered in its achievement.
The last part of this report contains results from the parallel code and a speedup diagram of it with respect to the number of processors in order to understand better the improvement in performances obtained with the parallel version.

2 Background

The background of this project is composed of 2 sections:

- the description of the geophysical problem and its mathematical background
- the description of the serial ITERTRAN code, to understand how the problem is studied and solved.

2.1 Description of the geophysical problem

Maps of the fluid at the core surface are important to understand the workings of the geodynamo, to predict geomagnetic time variation, to understand changes in length of day, and to derive temperature variations at the core-mantle boundary (CMB). Maps of the fluid together with CMB topography help us to investigate influence of topography on the flow.
The structure of Earth's interior (fig.1) can be described with a inner solid iron core immersed in a fluid iron outer core, and surrounded by a region called the mantle. Unlike mantle convection, only very small variations in core density are required to produce the flow; these density variations are too small to be imaged seismically, so instead geomagnetic time variation is used to infer the fluid flow of the outer core.
Of course ideally we should like to be able to determine fluid flow throughout the core; since mapping the flow requires knowledge of the magnetic field, and our knowledge of the magnetic field does not extend into the core, we are restricted to mapping the flow just beneath the CMB. Assuming that the mantle is an electrical insulator, we can take surface observations of the magnetic field and infer the magnetic field at the CMB.
The starting point is the magnetic induction equation

$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{u} \times \mathbf{B}) + \eta \nabla^2 \mathbf{B}, \tag{1}$$

where $\mathbf{B}$ is the magnetic field, $\mathbf{u}$ the fluid velocity, $\frac{\partial \mathbf{B}}{\partial t}$ is the time rate of change of magnetic field the so called secular variation (SV) and η the magnetic diffusivity.
This equation is derived from the complete set of Maxwell equation in which

$$\nabla \times \mathbf{H} = \mathbf{J} + \frac{\partial \mathbf{D}}{\partial t}, \tag{2}$$

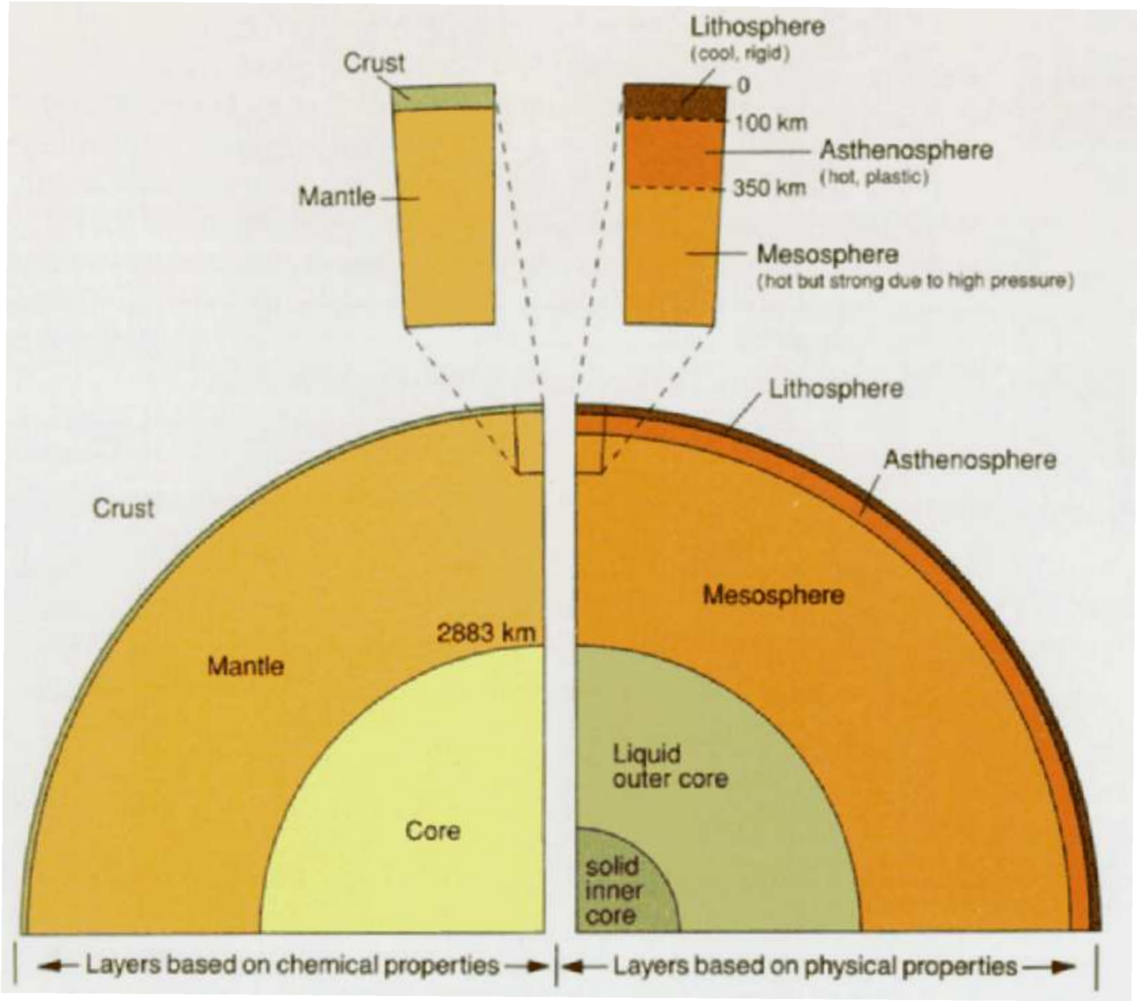

Figure 1: Structure of Earth's interior.

is Ampère's law, and

$$\nabla \times \mathbf{E} = -\frac{\partial \mathbf{E}}{\partial t}, \tag{3}$$

is Faraday's law. The vectors $\mathbf{E}$ and $\mathbf{D}$ are respectively the electric field and the electric displacement, $\mathbf{H}$ is the magnetizing force, while $\mathbf{J}$ is the electric current-density vector.
The term $\partial\mathbf{D}/\partial t$ in eq.(2), Maxwell's displacement current, is smaller than all the other terms in that equation and can therefore be neglected to a very good approximation. In this approximation we can simplify eq.(2) to

$$\nabla \times \mathbf{B} = \mu_0 \mathbf{J}, \tag{4}$$

taking the value of free space permeability μ_0 in the relationship $\mathbf{H} = \mathbf{B}/\mu_0$.
Using now the solenoidal condition

$$\nabla \cdot \mathbf{B} = 0, \tag{5}$$

and eq.(4), and the vector identity

$$\nabla \times \nabla \times \mathbf{B} = \nabla(\nabla \cdot \mathbf{B}) - \nabla^2 \mathbf{B}, \tag{6}$$

we have

$$\mu_0 \nabla \times \mathbf{J} = -\nabla^2 \mathbf{B}. \tag{7}$$

Combining now eq.(7) with eq.(4), and Faraday's law with Ohm's law

$$\mathbf{J} = \sigma(\mathbf{E} + \mathbf{u} \times \mathbf{B}), \tag{8}$$

we eliminate $\mathbf{E}$ and obtain the evolution of $\mathbf{B}$ in a moving conductor (eq.(1) as we wanted to demonstrate). Equation 1 shows how the time rate of change of the magnetic field is due to the effects of MAGNETO-ADVECTION (first term on the right hand side) and MAGNETIC DIFFUSION (second term on the right hand side). For the Earth's core we can associate the time scales τ_a and τ_d with these two processes, being $\tau_a \simeq 60$ years and $\tau_d \simeq 3 \times 10^4$ years. The time scale of advection is comparable to the time scale of geomagnetic SV and, as we can see, is much shorter than the time scale of diffusion. This suggests that we should be able to neglect the diffusion term (at least over time scales short compared with τ_d) and to consider the reduced equation:

$$\frac{\partial \mathbf{B}}{\partial t} = \nabla \times (\mathbf{u} \times \mathbf{B}). \tag{9}$$

This is the FROZEN FLUX approximation.
An alternative expression of this approximation is that for the radial component of the frozen flux magnetic induction equation:

$$\int_{\partial S} \frac{\partial B_r}{\partial t} dS = 0, \tag{10}$$

where dS is an element of surface area, and ∂S is a contour of B_r=0 (a null flux curve).
If we know $\mathbf{u}$ and $\mathbf{B}$ and on solving eq.(9) we want to calculate $\frac{\partial \mathbf{B}}{\partial t}$, this is the FORWARD PROBLEM, while if the time evolution of $\mathbf{B}$ is known and we want $\mathbf{u}$, this is the INVERSE PROBLEM. Given knowledge of the SV, it is the inverse problem that must be solved to arrive at the core-surface flow.
As with all inverse problems we must consider problems of the uniqueness of the solution in addition to the practical problem of constructing solutions. In order to resolve the nonuniqueness, additional assumptions must be made about the nature of the flow. The three most widely used are that the flow is toroidal, tangentially geostrophic or steady.

Unique solutions can be obtained by assuming the flow to be steady over some finite time interval; however, if this time-scale is too short, the flow is not uniquely resolved, and if it is too long the assumtion is not valid (for example 50 years is a long time over which to assume a steady flow, and possibly too long a time over which to neglect magnetic diffusion). It has been shown that if the flow is assumed to be the same at three separate points in time, then it can be determined uniquely.
Computations on the assumption of steady flow can be justified on the grounds that the flow is driven ultimately either by deep-seated convection with a turn-over time of thousands of years, or from the CMB due to horizontal temperature gradients. Either way, these are driving forces on a much longer time-scale than the SV, and hence there may be a large-scale steady component.
In order to improve the fit to the SV data, it is better to relax the steady-flow assumption (which can only describe linear trends in SV and main field time series), and consider the flow to be steady in a frame of reference drifting with respect to the mantle.
Eq.(9) can be written as:

$$\frac{\partial B_r}{\partial t} + B_r \nabla_H \cdot \mathbf{u} + \mathbf{u} \cdot \nabla_H B_r = 0 \tag{11}$$

where B_r is the radial component of the main field and

$$\nabla_H = \nabla - \mathbf{r}(\mathbf{r} \cdot \nabla), \tag{12}$$

where $\mathbf{r}$ is a unit vector normal to the core surface.
The CMB is assumed to be spherical.
As the SV is derived from differencing the main field, the best method for finding $\mathbf{u}$ is to integrate eq.(9) with respect to time and fit the main field.
Assuming that the core fluid is incompressible, and that the radial component of the velocity vector vanishes at the surface of the core, we can estimate the velocity field $\mathbf{u}$ at the surface of the free steam, which advects the radial components of the main field (B_r), in order to obtain the radial component of SV ($\frac{\partial B_r}{\partial t}$).
Any solenoidal vector field (a vector field with zero divergence), as for example the velocity field in the core with the assumption of incompressibility, can be decomposed in its toroidal and poloidal components and then expanded in spherical harmonics.
We substitute in eq.(11) spherical harmonic expansions for main field, secular variation, and for the velocity:

$$\mathbf{u} = \nabla \times \left[\mathbf{r}c \sum_{l,m} {t_l}^m {Y_l}^m(\theta, \phi) \right] + \nabla_H \left[rc \sum_{l,m} {S_l}^m {Y_l}^m(\theta, \phi) \right], \tag{13}$$

where ${t_l}^m$ and ${S_l}^m$ are the toroidal and poloidal velocity coefficients, ${Y_l}^m(\theta, \phi)$ are spherical harmonics of degree l and order m, and c is the radius of the core. The completely normalized spherical harmonics ${Y_l}^m(\theta, \phi)$ have the expression

$${Y_l}^m(\theta, \phi) = \left(\frac{2l+1}{4\pi\epsilon_m} \right)^{1/2} {P_l}^m(\theta) e^{im\phi} \qquad (-l \leq m \leq l), \tag{14}$$

where ϵ_m=2 for $m \neq 0$ and ϵ_0=1. The associated Legendre functions ${P_l}^m(\theta)$ are taken so that ${P_l}^m = {P_l}^{-m}$ and ${Y_l}^{-m} = ({Y_l}^m)^*$, where the * denotes the complex conjugate. By the choice of eq.(14), we have

$$\int {Y_l}^m(\theta, \phi) {Y_s}^{-r}(\theta, \phi) d\Omega = \delta_{ls}\delta_{mr}, \tag{15}$$

where the integral is taken over all solid angles, $d\Omega = \sin\theta d\theta d\phi$.
In an insulator, as we supposed the mantle to be, we have:

$$\mathbf{B} = -\nabla V, \qquad V = -\frac{\partial}{\partial r}(rP). \tag{16}$$

Thus, as the exterior of the core conducting region ($r \geq c$) is an insulator, toroidal magnetic fields are absent and the observed potential field

$$V = c\sum_{l=1}^{\infty}\sum_{m=1}^{l}({g_l}^m \cos m\phi + {h_l}^m \sin m\phi){P_l}^m(\theta)\left(\frac{c}{r}\right)^{l+1}, \tag{17}$$

is a poloidal field with defining scalar

$$P = c\sum_{l+1}^{\infty}\sum_{m+1}^{l}({g_l}^m \cos m\phi + {h_l}^m \sin m\phi)\frac{{P_l}^m(\theta)}{l}\left(\frac{c}{r}\right)^{l+1} \tag{18}$$

where c is the radius of the core and g_l^m and h_l^m are the so called Gauss geomagnetic coefficients which are deduced from surface measurements. Nevertheless eq.17 is also valid at r=c so we have the CMB field.
As we can see, in these expressions we have infinite spherical harmonic expansions that obviously need to be truncated. Usually the truncation level is given by l=14 but some series extend to l=28. The numerical analysis is standard; see, for example, section 6.8 of [1].
Now, after multiplying eq.(11) by the complex conjugate of the SV spherical harmonic, and integrating over the CMB, we obtain the matrix equation

$$\dot{\mathbf{g}}(t) = \mathbf{A}(t)\mathbf{u}(t). \tag{19}$$

In this relation $\dot{\mathbf{g}}(t)$ is a column vector of SV coefficients, $\mathbf{A}(t)$ is a matrix (known as condition matrix) relating the main field, SV, and velocity field; $\mathbf{u}(t)$ is a column vector of toroidal and poloidal velocity coefficients (the unknowns), and t is the time.
Now consider a steady velocity field represented by a vector of toroidal and poloidal velocity coefficients, $\mathbf{u}_s$, and a transformation of them into a drifting frame of reference. The transformation is a phase translation, so that the velocity field is spatially time-dependent but the total kinetic energy is constant with respect to time.
The physical interpretation of such a transformation might be the coupling between core and mantle. A time dependent velocity field represented by the spherical harmonic coefficients $\mathbf{u}_t(t)$ can be computed from

$$\mathbf{u}_t(t) = \mathbf{v}_p(t) + \mathbf{R}(t)\mathbf{u}_s, \tag{20}$$

where $\mathbf{v}_p(t)$ is the phase speed which has a single component ordered with the toroidal, solid-body flow component ${t_1}^0$, and $\mathbf{R}(t)$ is a square matrix which rotates the velocity field in the azimuthal direction.
Sustituting eq.(20) into eq.(19) gives

$$\dot{\mathbf{g}}(t) = \mathbf{A}(t)[\mathbf{v}_p(t) + \mathbf{R}(t)\mathbf{u}_s]. \tag{21}$$

Instead of solving this equation for the velocity coefficients $\mathbf{u}_s$ (as $\mathbf{R}$ is time dependent and non linear), we can rearrange eq.(21) as

$$\dot{\mathbf{g}}(t) - \mathbf{A}(t)\mathbf{v}_p(t) = \mathbf{A}(t)\mathbf{R}(t)\mathbf{u}_s, \tag{22}$$

where we can compute $\mathbf{u}_s$ from a prescribed $\mathbf{v}_p(t)$ with an easier numerical tecnique. As we have introduced another unknown into the velocity field, we simplify the calculation by assuming that $\mathbf{v}_p$ is time-independent. More details about the physical model can be found in [2] and [3].

2.1.1 Results of this theory

A comparison of the misfit between the steady and drifting flow is made in fig 2, where is shown an improvement in the relative fit to the SV with the introduction of the drifting frame of reference.

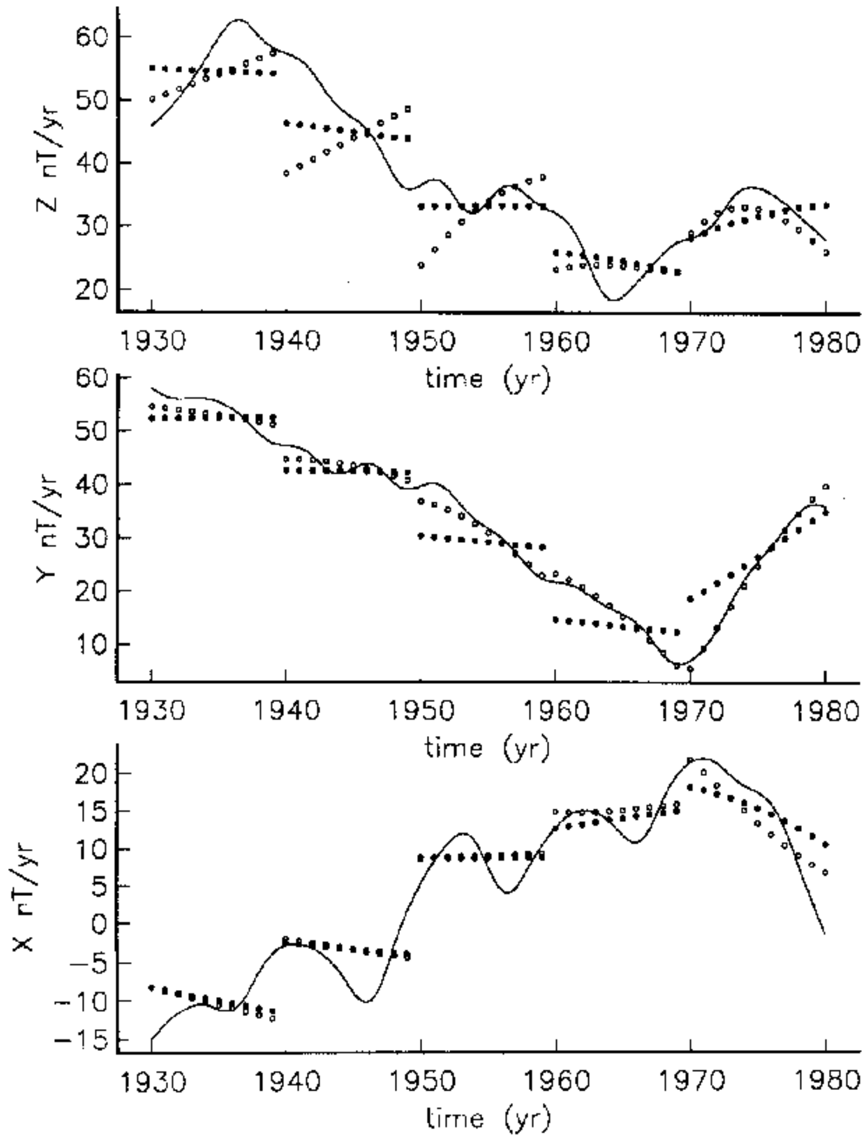

Figure 2: Times-series of the SV of the three orthogonal surface magnetic components X, Y and Z for the steady flows (solid circles) and the drifting flows (open circles) fit to the time dependent model of Bloxham and Jackson (continuous curve). The location of the site is: latitude=50.8, longitude=16.2

An example of the calculated flow pattern at the surface of the core is in fig 3 which shows a vector plot of the total flow component. The two maps are relative to years 1930-1940 (the first one) and 1950-1960 (the second one). The map for years 1940-1950 is indistinguishable from the second one. The flow in the northern hemisphere is very similar in the two flows, while the flow below the south-eastern Pacific changes direction between 1940 and 1950, and in the southern hemisphere there is strong westward flow beneath the Atlantic. The flow below the Indian Ocean is weak in 1930-40 and strengthens in 1950-60.

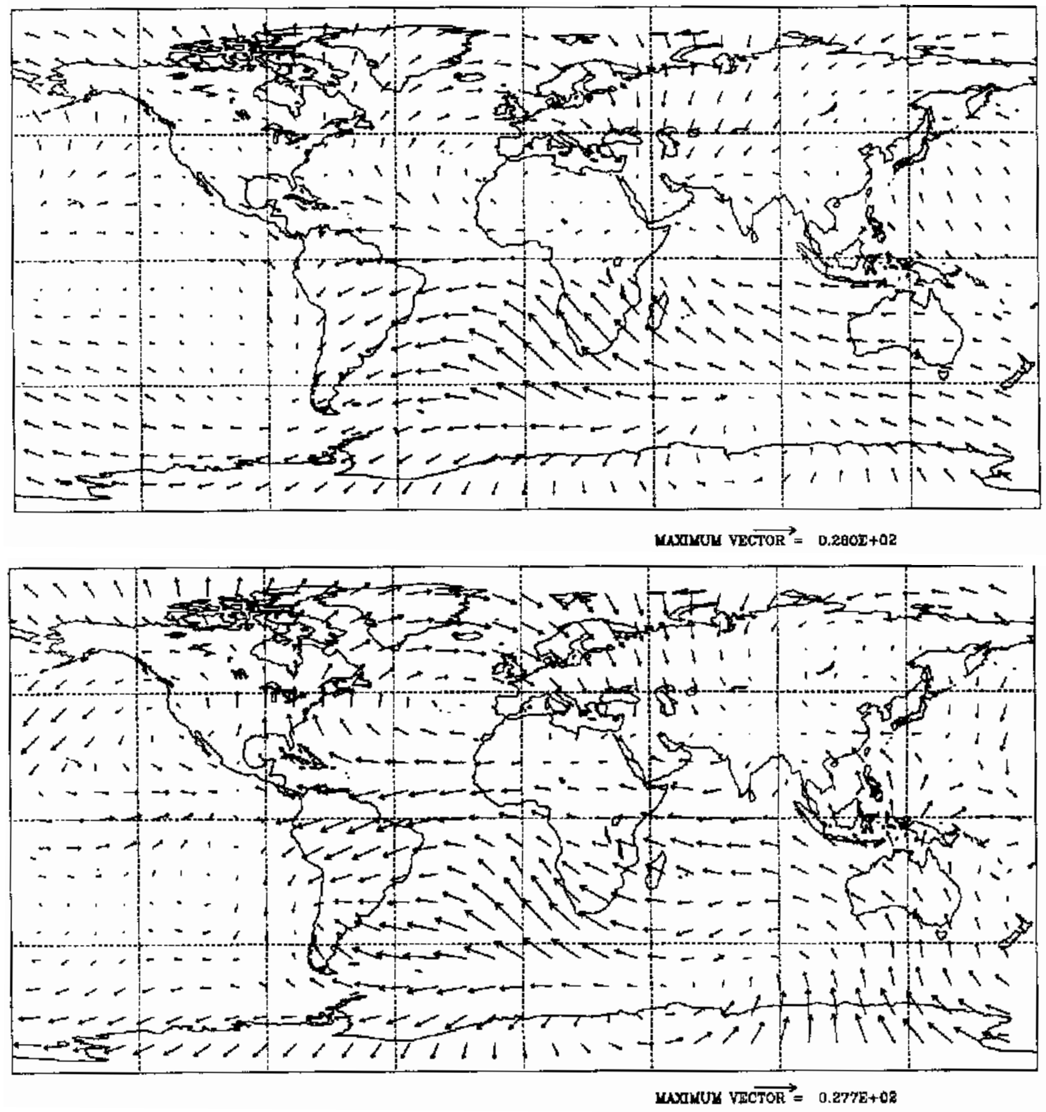

Figure 3: Vector maps of the total surface flow at the core surface with the reference vector equal to 28 KM/YEAR. The first one is for years 1930-1940 and the second one for years 1950-1960

2.2 The serial ITERTRAN code

The ITERTRAN code is designed to compute the geophysical fluid flow at the Earth's core-mantle boundary, assembling the equations of condition that relate the fluid velocity harmonic coefficients to those for the secular variation of the Earth's magnetic field, and solving these equations over a given time range and in a drifting frame of reference.

As we said in the previous section, these equations of condition can be written as:

$$\dot{\mathbf{g}}(t) = \mathbf{A}(t)\mathbf{u}(t). \tag{23}$$

and can be solved numerically for the velocity coefficients **u**. Looking at this equation and the pattern of fig. 4 we can understand the structure of the program, which requires three include files (physical.inc, sizes.inc and velnems.inc) to fix some parameters.
The main part of the program, routine ITERTRAN, is so realized:

- subroutine INTEGRATE integrates the drift velocity over epochs
- subroutine ROTATE, which is inside a do loop over epochs, as also are MODSV and NEMS, rotates a set of spherical harmonic coefficients through an angle dependent on the drift velocity. In the first call to ROTATE, the magnetic field (MF) is rotated, while in the second and third ones SV and associated errors (ES) are rotated
- subroutine MODSV calculates the SV induced by the drift velocity
- subroutine NEMS calculates the normal equations epoch by epoch
- inside subroutine VELO, velocity coefficients **u** are calculated from normal equations through calls to subroutines SMTHTP, PREMID, PREALL, CHOLES and VPREDSV.
  SMTHTP adds a damping function to the diagonal elements of the condition matrix **A** so we have the equation:
  $$(\mathbf{A}^T\mathbf{A} + \lambda Q)\mathbf{u} = \mathbf{A}^T\dot{\mathbf{g}} \tag{24}$$
  The damping function λQ then is passed in as an argument to the subroutine ANORM that calculates the norm of a set of spherical harmonic coefficients.
  PREMID and PREALL calculate the predicted SV and misfit respectively at the midpoint epoch and at each epoch.
  To find velocity coefficients **u** from eq.(23) we must invert the "damped equation of condition" matrix $\mathbf{A}^T\mathbf{A}$ which is one of the most important problems in the numerical approach. Cholesky factorisation and solution of the linear system eq.(24) is performed by the subroutine CHOLES. Then the subroutine VPREDSV calculates the predicted SV coefficients by multiplying velocity coefficients by the condition matrix **A**.
- At this point MF, SV, predicted SV and ES are back-rotated through a quadruple call to ROTATE;
- finally subroutine RESVEC calculates the residual misfit vector for the computed velocity.
- Two other subroutines are quite important: VELNEMS and GNTELS.
  One of these subroutines is called inside MODSV, NEMS and VPREDSV through a C-preprocessor conditional construct that gives the possibility of switching between GNTELS and VELNEMS. For a given l, GNTELS calculates $2l + 1$ rows of the condition matrix **A**, at a time, using the Gaunt-Elssaser method (analytic expressions for the matrix elements), and this is made inside a do loop over l from 1 to $lmax$ which is in general equal to 14.
  VELNEMS instead calculates all $2lmax(lmax + 2)$ rows of **A** in a single call without any need of a do loop, using a spherical transform approach.

More details about the serial code can be found in [4].

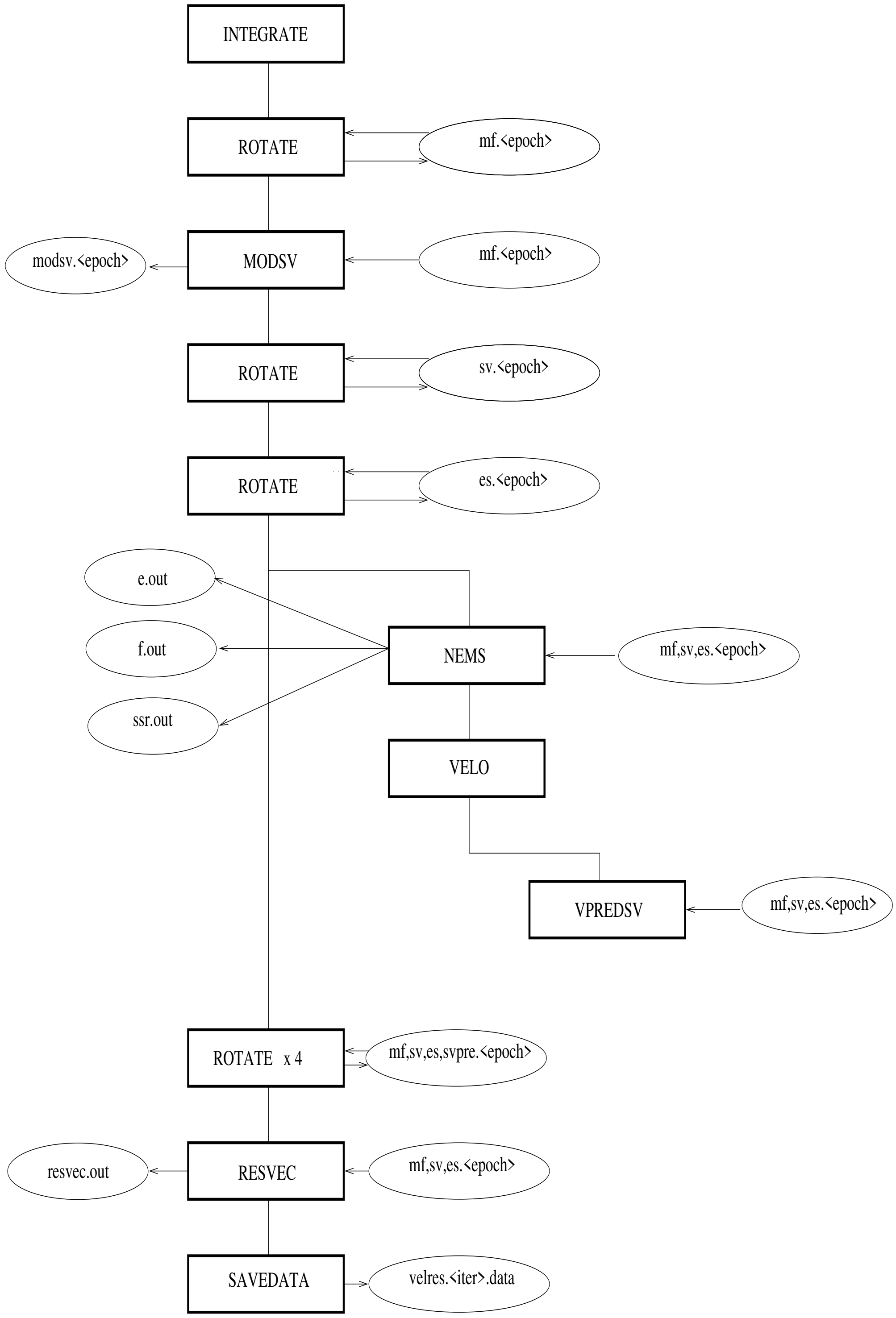

Figure 4: Schematic diagram of I/O patterns in the serial ITERTRAN code. Boxes indicate subroutine, ovals represent files or group of files. The arrows indicate whether a file is read or written

3 The parallel code

In this section is the description of the parallel approach choosen for the ITERTRAN code.
The first subsection shows, after the explanation of motivations that induced us to choose a task-farm model, the design of the task-parallel code through a diagram that can be compared with previous serial design (diagram in fig. 4).
In the second and third subsection then there is the description of the new I/O patterns and some details about the implementation of the code.

3.1 Task-farm design

The choice of a task-farm design for the parallel code comes from the fact that:

- ITERTRAN is structured as a number of closely related but loosely coupled tasks operating on the same data
- the code usually runs with between 10 and 150 epochs, each with their own dataset, each making their private contribution to the condition matrix $\mathbf{A}$
- the calculations for each epoch are independent of any other epoch, except in the final assembly of $\mathbf{A}^T\mathbf{A}$
- only before the final stage it is necessary to synchronize the normal equations matrix $\mathbf{A}^T\mathbf{A}$

For all these reasons it is possible to perform calculations on each epoch in parallel by distributing work on different processors. In that way, each processor makes calculations only on a limited number of epochs and only in the final stage is communication between processors necessary to assemble matrix $\mathbf{A}^T\mathbf{A}$.

3.2 Implementation details and problems encoutered

The implementation of the parallel code has been realized using the Message Passing Interface (MPI).
As I said in the previous section, the work in the parallel version of the code is organised so that each processor works on a limited number of epochs that obviously depend on the global number of processors and epochs.
In order to explain how the work is now organized, we have to look at the key loops over epochs that occur inside various subroutines.
In table 1, we can see how this do loop was organized in the serial code and how it is organized now in the parallel one.
In the serial code, we have one of these do loops in the main routine. It starts immediately before the call to subroutine ROTATE and continues until subroutine NEMS, working between 1 and the total number of epochs (NEPS). This means that each one of these subroutines makes calculations on a single epoch at a time.

Serial Code	**Parallel Code**
time = TIMEST DO iep = 1, NEPS .... time = time + TIMESTEP END DO	time = TIMEST + rank*locep*TIMESTEP DO iep = 1, locep igep = iep + rank*locep ..... time = time + TIMESTEP END DO

Table 1: Comparison of serial and parallel version of the do loop on epochs

In the parallel code instead all calculations are done serially before this do loop, so all processors do the same work on the same data. Then, inside the do loop and also in some other parts of the code, calculations are done in parallel, so that all processors do the same work but on different sets of data.
In the parallel code we have some new parameters:

- LOCEP that is equal to the total number of epochs over the total number of processors ($locep = NEPS/SIZE$)
- IEP which is a local index on epochs
- IGEP which is a global index on epochs

In order to understand how the parallel code works through table 1 we can fix TIMEST=1960, TIMESTEP=1, the total number of epochs = 20 and the total number of processors = 4. In this case locep = 5 so processor 0 starts its do loop at 1960 and works for 5 epochs until 1964, processor 1 starts at 1965 and works until 1969 and so on. Only at the end of this do loop is communication between processors necessary.
The main problem to overcome in this implementation was due to the fact that there are a lot of these do loops distributed inside subroutines. Almost all of them have been parallelized in this way but this choice sometimes created problems because of the continuous alternation between serial and parallel parts of the code. With this alternation we require global arrays for serial calculations and local arrays for distributed calculations.

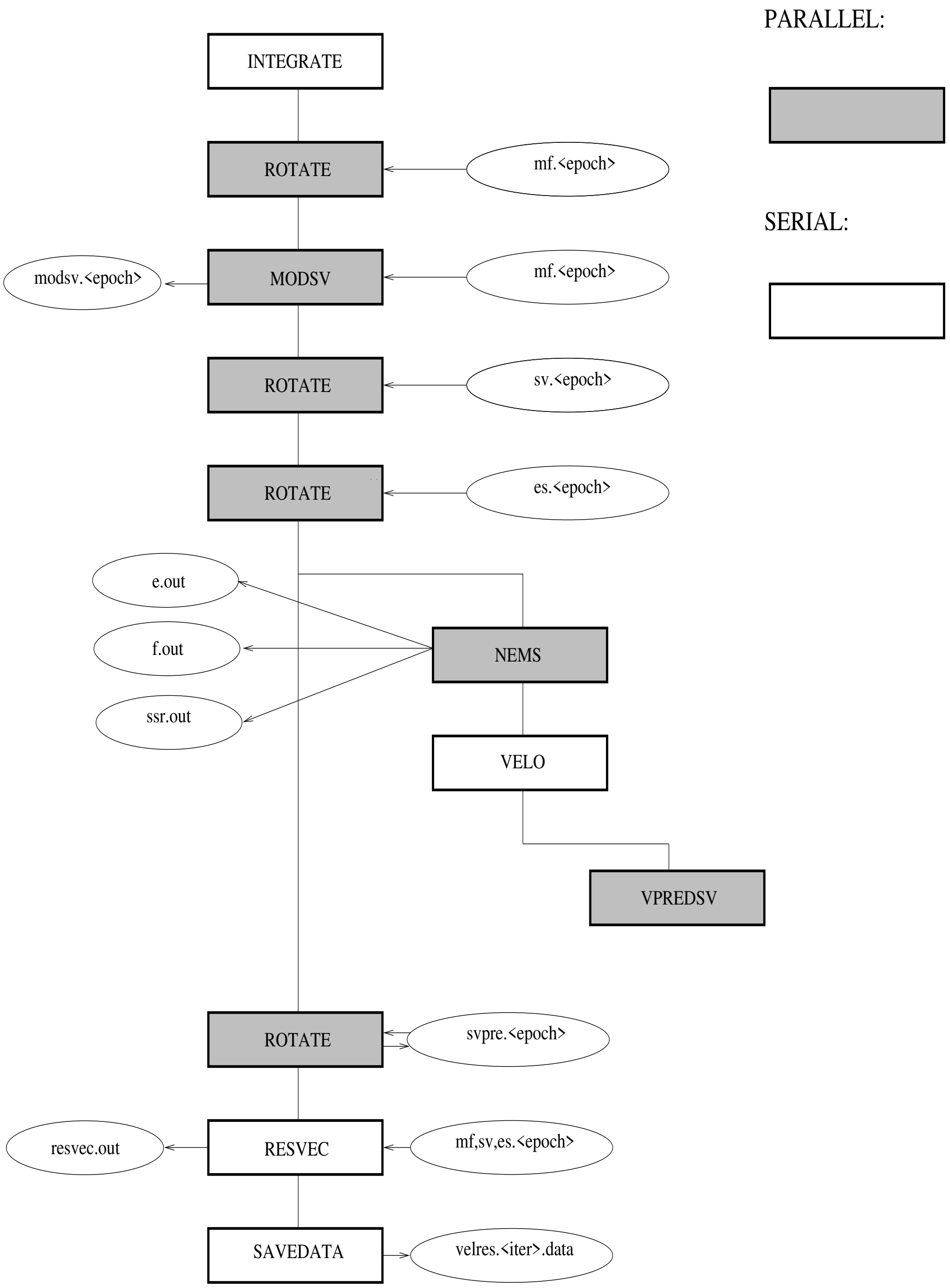

Figure 5: Schematic diagram of I/O patterns in the parallel ITERTRAN code. White boxes indicate serial subroutine, shaded boxes indicate parallel subroutine, ovals represent files or group of files. The arrows indicate whether a file is read or written

3.3 New I/O pattern

Making a comparison between the diagrams in figs. 4 and 5, we can observe some important changes in the input/output patterns.
From fig. 4 we can see that the subroutine ROTATE reads the files of spherical harmonic coefficients of MF, SV and ES rotates these values and finally overwrites the files. These overwritten files are then read by subroutines NEMS and VPREDSV. In the last part of the main routine there is a quadruple call to subroutine ROTATE; this multiple call is necessary in order to backrotate MF, SV and ES spherical harmonic coefficients and restore input files as they were originally. After this a new iteration can start.
In the parallel code (fig. 5), the subroutine ROTATE reads the input files of MF, SV and ES spherical harmonic coefficients, rotates their values and stores them in memory in arrays gmfstore, svstore and esstore.
These arrays are passed as additional arguments to successive subroutines so that we have only input in the first part of the code. Subroutines NEMS and VPREDSV have no need to read anything because they receive these values directly from ROTATE.
Finally, at the end of the program, only one of the four calls to ROTATE is necessary (the one in which modsv.epoch is rotated) because the files of MF, SV and ES spherical harmonic coefficients are not changed so it is not necessary to backrotate them.
This strong reduction of the input/output with the storage in memory of rotated values of MF, SV and ES spherical harmonic coefficients was not possible in the serial code because of the big requirement of memory that it involves. For the parallel code instead this requirement is not so big because, as I said before, each processor works only on a limited number of epochs.
To make a comparison between the requirement of memory in the serial and parallel code we have that:

- usually the number of epochs is not more than 150
- each array for each epoch is of size $l(l+2)$ where l is the maximum velocity truncation degree. l is usually equal to 14 but in the worst case it can be equal to 28
- we have three arrays (gmfstore, svstore and esstore)
- so the requirement of memory for the serial code is:

$$3 * 150 * 28 * 30 * 8bytes \sim 3Mbytes \tag{25}$$

This cost was too large for a single processor as it is in the serial code.
For the parallel code, we divide this memory by the total number of processors, so that for each processor the amount of memory required is less.
Realizing the storage in memory of these arrays I encountered one problem due to the presence inside routine VPREDSV and NEMS of the subroutines LP1 and DOWN.
The routine LP1 scales main field coefficients by factors of l+1 while the routine DOWN scales main field coefficients by factors of a/c (Earth radius to core radius).
These calls were necessary in the serial program where VPREDSV and NEMS routines read datas from MF, SV and ES files in which data is not scaled. They are instead errors inside the parallel program where this input data is stored in memory after a first call to LP1 and DOWN and is then passed to VPREDSV and NEMS. I didn't realized immediately this fact so my initial results were wrong because input data was scaled incorrectly.

Concerning the reorganization of input/output in the parallel code, we applied these simple rules:

- INPUT
  - in general input files are read by all processors
  - input data not necessary for the work of all processors is read only by processor 0
- OUTPUT
  - each processor creates a different file for output of private data and each one of these output files is distinguished through the number of the processor that created it (for example *itertran.output.001* for processor 1)
  - common information is written only by processor 0
  - in any case in which an array whose different parts are calculated by different processors and the global array has to be written to a file, only processor 0 will open the file and write to it after communication from the other processors.

4 Results obtained

The results obtained realizing this work can be so summarized:

- the ITERTRAN code has been parallelised succeessfully using the task-parallel client-server model
- Serial and parallel codes give comparable results so the new code runs correctly
- the time scaling from serial to parallel code is quite good

This last result is of particular note, as the speedup with respect to the number of processors observed is almost linear (fig. 6).
In this picture we have the number of processors equal to the number of epochs involved. This figure has been realized for 1, 3, 6, 8 and 10 processors and the efficiency we obtained is equal to 87% for 1, 3 and 6 processors and then becomes 67% for 8 and 10 processors.
Maybe this reduction in the efficiency with increasing number of processors is due to the increased number of communications implied.

5 Other possible parallelisation routes

In this final section, we examine two other possible parallel developments routes for the ITERTRAN code.
In the code, calculations are made on a (ϕ,θ) grid and that suggests the possibility of a geometric decomposition along one of these axes (θ axis).
Actually, in terms of dependencies between adjacent gridpoints there are restrictions only due to the presence of Fourier transformations in ϕ, in contrast there are no data dependencies in the θ grid direction.
Decomposition along the ϕ axis is essentially precluded by the Fourier transform (one dimensional parallel FFT's are notorious in performing much more poorly than serial equivalents).

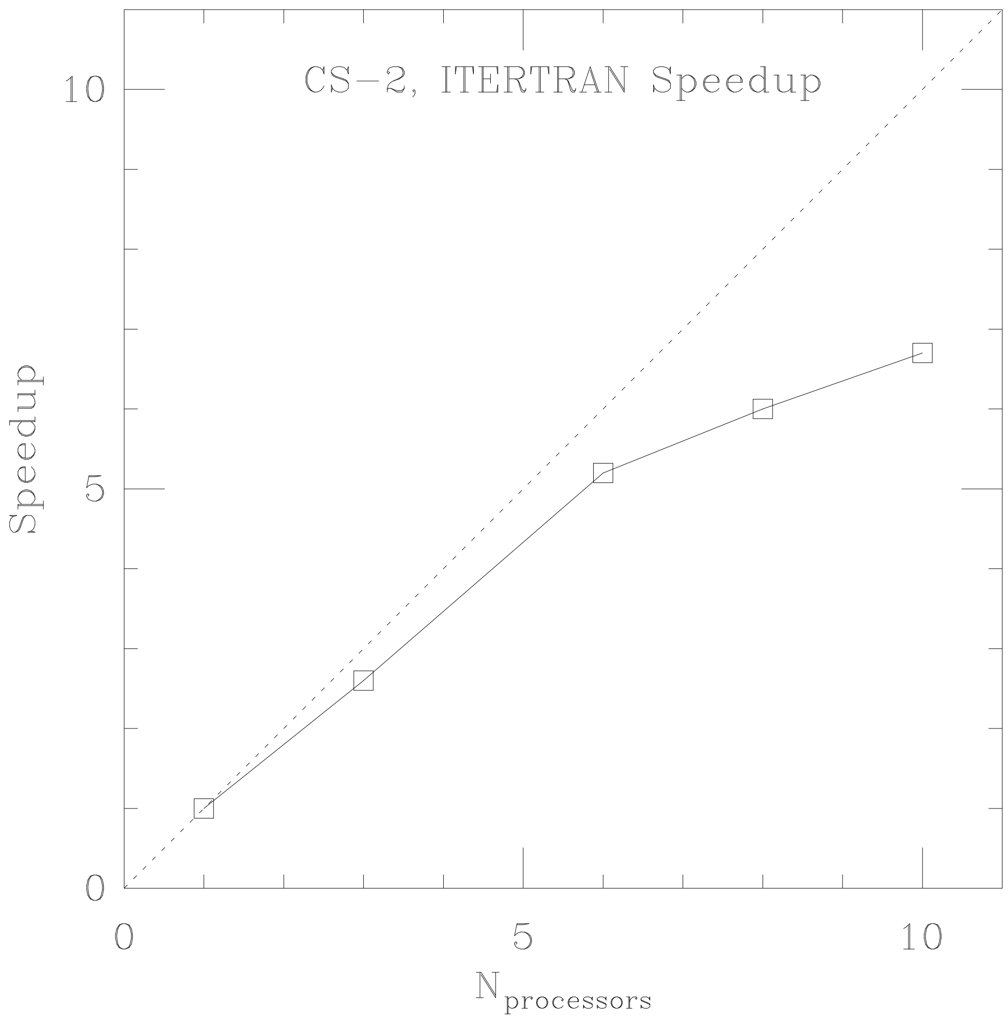

Figure 6: Speedup of the parallel code with respect the total number of processors. The number of processor is, in this case, equal to the number of epochs.

This suggests that only a geometric decomposition over processors in the θ direction would yield reasonable scaling behaviour.
The only point at which processors would have to communicate would be during the initial and final I/O stages.
A θ decomposition would reduce runtime almost linearly but unfortunately it would have essentially no impact on the more important question of memory scalability.
After this new parallel implementation it would be interesting to create a further version in which these two approaches are coupled.

6 Summary

In this report we have described the geophysical ITERTRAN code and in particular its parallelisation features.
In summary we have:

- realized the parallel implementation of the code through a task-farm approach over independent timeslices. The features used for this goal are:
  - all do loops on epochs have been reorganised so that each processors works on a limited number of epochs
  - input files are read in general from each processor
  - output files are created from each processor in order to distinguish private results
  - common information and global arrays are sent to processor 0 from all other processors and then written only by processor 0
- restructured the I/O pattern through the storage in memory of MF, SV and ES rotated spherical harmonic coefficients
- reduced the number of calls to subroutine ROTATE

To conclude, we have obtained:

- a good memory scaling
- an almost linear parallel speedup

References

[1] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P.Flannery
Numerical Recipes: The Art of Scientific Computing CUP, 1992.

[2] Jeremy Bloxham and Andrew Jackson
Fluid flow near the surface of Earth's outer core
Reviews of Geophysics (1991), **29(1)**, 97-120.

[3] R. G. Davis and K. A. Whaler.
Determination of a steady velocity field in a rotating frame of reference at the surface of the Earth's core.
Geophys. J. Int. (1996), **126**, 92-100.

[4] R. M. Baxter.
ITERTRAN: Analysis and Optimisation Report
Technical Report EPCC-PG-GLG-2, EPCC, 1996.

Project of:

- Elena Degoli,
  born in Modena the 12th of April 1972.
  Graduate in Physics at the University of Modena (Italy) the 7th of March 1997.
  After the secondary scientific school where I discovered my passion for science, I decided to attend the faculty of Physics to satisfy my curiosity about all the world around me. That led me, in the last period of my university training, to study in depth the properties of matter that means the properties of everything around us: my thesis actually concerned the *"Optical Properties of Ultrathin Layers of Silicon"*.
  These theoretical researches involve a considerable use of computers which induced me, in these last two years, to learn how to use them to the best effect. This experience at EPCC helped me to appreciate the power of these machines and all the possibilities that they give us in every field.

Supervisors of the project:

- Dr. Rob Baxter of the EPCC
- Prof. Kathy Whaler of the Departement of Geology and Geophysics of the University of Edinburgh