EPCC-SS97-02

3D Modelling of Heat Flow in Sedimentary Basins

Dirk Carstens

Abstract

For the successful exploration for hydrocarbon deposits it is crucial to determine the thermal history of a given area. A three-dimensional model for the heat flow has therefore been developed in this project. Because of its computational expensive nature the code has been parallelised and run on the T3D. This report documents the implementation of the model and problems encountered. The model is tested for several geological scenarios and the results obtained are discussed.

Contents

1 Introduction

The report describes a project on the development of a 3-dimensional simulation of heat transport in sedimentary basins. The project was part of the Summer Scholarship Programme (SSP) 1997 hosted by the Edinburgh Parallel Computer Centre (EPCC).

Oil industry drilling as well as scientific drilling projects have allowed people to gain insights into the geological structure of the earth. In regions unaffected by volcanic intrusions and tectonic deformations one is likely to find a formation made up out of well-defined, sedimentary layers. The 'core' extracted by a test drilling can be studied to obtain information of material properties and explicitly thermal properties [3] like conductivity of the different types of rocks being present.

Taking these empirical results into account it is possible to set up a model for the heat flow through these sedimentary layers and the earth crust. Determining the thermal history of individual formations, and especially layers containing a mixture of sedimentary rock and organic material, is of great importance for the oil industry. The above mentioned type of rock is often referred to as *source rock* and it is the main originator of hydrocarbons. If the source rock is heated to temperatures ranging from 80 C up to approx. 160 C for a long enough period of time a conversion of the primary organic material to hydrocarbons, oil and gas, can take place. For a correct assessment of the status of this so-called maturation process an accurate simulation of the heating process is required. Subsequently predictions of possible locations of hydrocarbon deposits can then be made.

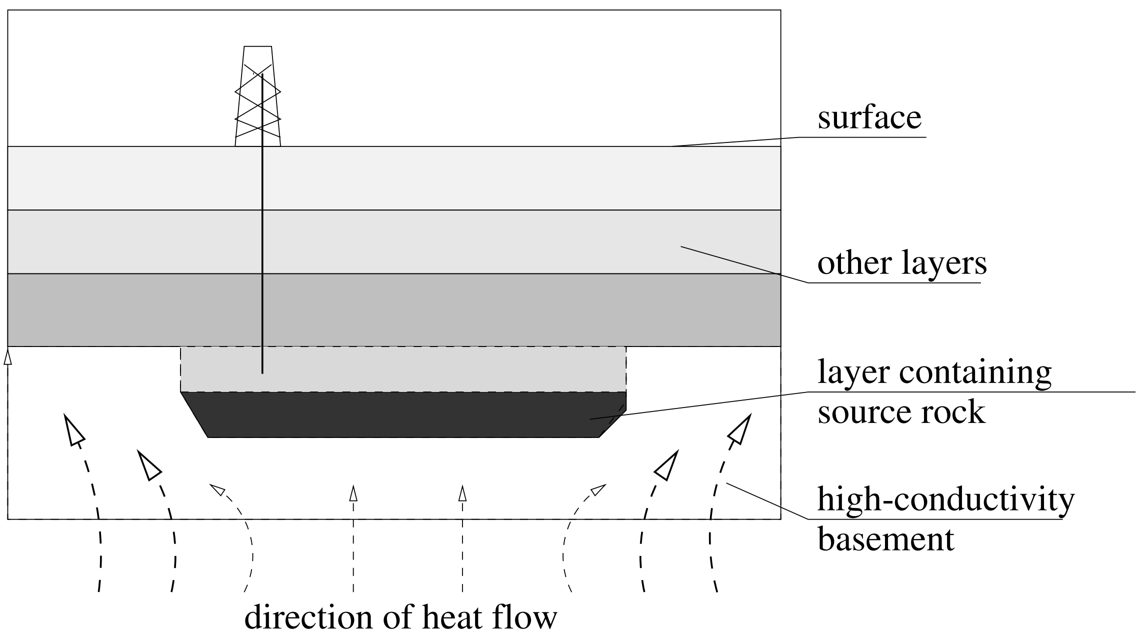

Figure 1: A geological scenario - trench shape

A number of one-dimensional simulations, like Hedera from PGS Tigress already exist. During the development of this model Hedera was also used to compare the initial results. In many circumstances, like the one sketched in figure 1, the thermal history of a source rock cannot be simulated correctly by a one-dimensional simulation. In the scenario illustrated in figure 1 a layer containing source rock is embedded in a high conductivity basement. A constant heat flow enters the model from the bottom and leaves it at the surface. The heat flow, as indicated by arrows in the sketch, is due to the shape of the basement, redirected around the source rock layer. This causes the temperature in the central 'valley' to be lower then the one that would have been expected at the given depth if a 1D model was used.

In the following section the physical background used for this model will be outlined and any approximations made will be discussed. Section three describes how the model was implemented on a serial machine and what kind of solving algorithms were used. Starting from the serial code, the fourth part deals with the parallelisation. One aim of the project was to develop visualisation tools to illustrate the gained results. For this purpose the package 'Advanced Visual System' (AVS) was used. The report concludes with a number of case studies that illustrate the possible applications of the model.

The serial code was mainly written in Fortran 90 and the parallelisation was done using High Performance Fortran (HPF). The serial code was run on Sun-workstations and on the Meiko CS-2 'Jurassic' (18 Sparc nodes). The parallel code was run on the DEC-Alpha workstation cluster and on the Cray T3D. All computers were situated at EPCC.

2 The physical background

The model is concerned with the lithosphere of the earth, which ranges to a depth up to 100km and contains the earth's crust and the uppermost part of the mantle. This implies that we will deal with matter having solid and rigid properties and the simulation will only take heat conduction into account.

The situation would be different in the athenosphere (100 - 350km depth). Here the pressure and temperature have increased to such an extent that matter has become soft and plastic [1]. Under these circumstances it would be unavoidable to include heat convection, i.e. heat transfer via material currents, into the model. The transport of heat may also be governed by radiation [2]. But this process may be neglected for the temperature range one expects.

As a starting point we are going to use the conservation of energy to derive an expression for heat conduction in two dimensions. Let us assume a rectangular element with spatial dimensions [4] δx and δy as sketched in figure 2.

The heat flux in the x direction is denoted with q_x, that in the y- direction q_y. The rate of heat flow into the cube is given by the flux times the area. So for the heat flow into the element we obtain $q_x(x)\delta yL$ and $q_y(y)\delta xL$ for the two directions respectively. Similar for heat flow rates out of the cube we get $q_x(x+\delta y)\delta xL$ and $q_y(y+\delta y)\delta xL$. In these expressions L denotes the

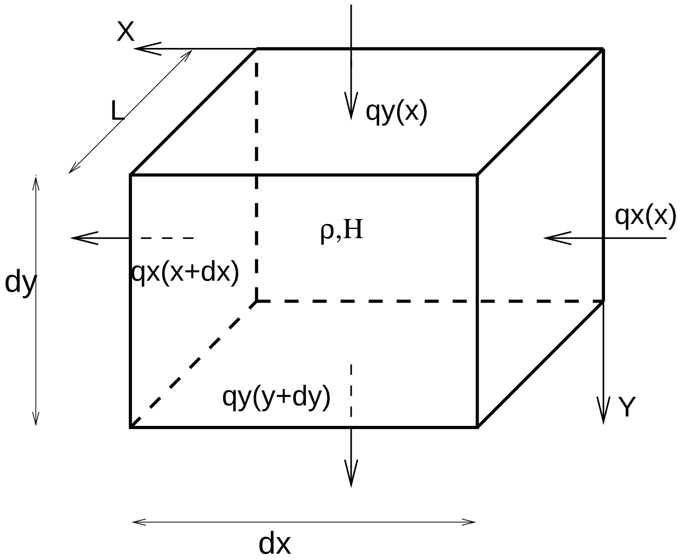

Figure 2: Heat flow in a rectangular slab of material

length of the slab in the third dimension. For simplicity the heat flow in the third dimension is assumed to be zero at this stage.

Therefore the net heat flow rate δq out of the element is given by the difference of the flux in the orthogonal directions.

$$\delta q = (q_x(x+\delta y)\delta y L - q_x(x))\delta y L + (q_y(y+\delta x)\delta x L - q_y(y))\delta x L \tag{1}$$

If one applies a Taylor expansion on (1) this approximates to

$$\begin{aligned} \delta q &= \frac{\partial q_x}{\partial x}\delta x \delta y L + \frac{\partial q_y}{\partial y}\delta x \delta y L && (2) \\ &= \left(\frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y}\right)\delta x \delta y L && (3) \end{aligned}$$

In the case of a non-zero value for δq it is implied that heat is being generated internally in the slab. The rate of heat generation in the element is $\rho H(\delta x \delta y l)$. Equating this, one obtains

$$\begin{aligned} \frac{\partial q_x}{\partial x} + \frac{\partial q_y}{\partial y} &= \rho H(x,y) && (4) \\ \nabla q(x,y) &= \rho H(x,y) && (5) \end{aligned}$$

The next thing one has to do is to introduce Fourier's law of heat conduction. It relates the directional heat flow to the temperature gradient

$$q = -k(x,y)grad(T(x,y)) \tag{6}$$

The proportionality constant is the thermal conductivity k of the rock, which cannot be treated as isotropic, i.e. it has to be dealt with as a local dependant variable. Combining the two expressions [1] for the flux gives for two dimensions

$$\frac{\partial k}{\partial x}\frac{\partial T}{\partial x}+\frac{\partial k}{\partial y}\frac{\partial T}{\partial y}+k\left(\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}\right)=-\rho H. \tag{7}$$

It is possible to generalise this expression to three-dimensional heat conduction and one obtains

$$\frac{\partial k}{\partial x}\frac{\partial T}{\partial x}+\frac{\partial k}{\partial y}\frac{\partial T}{\partial y}+\frac{\partial k}{\partial z}\frac{\partial T}{\partial z}+k\left(\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}+\frac{\partial^2 T}{\partial z^2}\right)=-\rho H. \tag{8}$$

It should be mentioned here that treating the thermal conductivity as location dependant only is another approximation. The thermal conductivity is actually a function of the governing temperature and pressure [3] . This relation is also rock-type dependant, but in general it can be said that with increasing temperature the conductivity decreases.

3 Setting up the model

3.1 Discretising the model

Equation 8 is now applied to an element of heterogeneous material. For simplicity a cube was chosen. It is now necessary to translate the problem from a continuous to a finite representation. This can be done if one restricts oneself to compute the solution only on defined points of the computational mesh. The arrangement chosen can be seen in figure 3, where a cube with $N = 4$ nodes in each dimension is sketched.

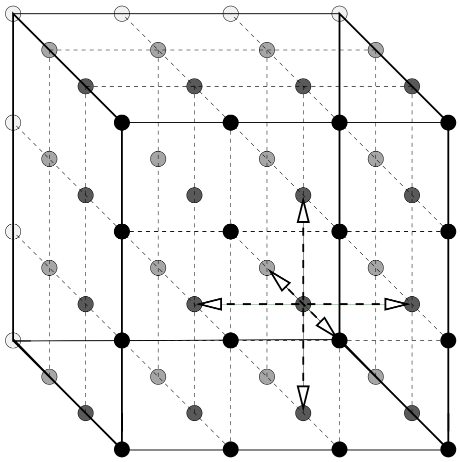

Figure 3: Meshed cube with seven-point stencil

The partial derivatives in equation 8 are approximated using finite differencing techniques. This is shown for the first order derivatives in the leading term:

[1] and taken into account that one has to use the product-rule to form the derivative

$$\frac{\partial k(x,y,z)}{\partial x}\frac{\partial T(x,y,z)}{\partial x} \simeq \frac{k(i+1,j,k)-k(i-1,j,k)}{2h}\frac{T(i+1,j,k)-T(i-1,j,k)}{2h} \quad (9)$$

Here one has introduced h as the grid spacing and expressed the spatial coordinates as multiples of h; $x = ih$, $y = jh$ and $z = kh$. This form of 'centred differenced' approximations of partial derivatives is second order accurate. The Laplacien term of equation 8 is translated in the following way:

$$\begin{aligned} \bigtriangledown^2 T(x,y,z) \quad &\simeq \quad \frac{6T(i,j,k) - T(i+1,j,k) + T(i-1,j,k) + T(i,j+1,k)}{6h^2} \\ &+ \frac{T(i,j-1,k) + T(i,j,k+1) + T(i,j,k-1)}{6h^2} \qquad (10) \end{aligned}$$

As a result of the transformation to finite space we obtain a system of N^3 simultaneous linear equations. For every node point we have a discretised equation that defines the node value as a weighted average of its neighbouring points. The term 'weighted' is used here, because it depends on the directional gradient of the conductivity as well as the temperature to determine which neighbour has a greater impact on the node value. This dependence is often referred to as a stencil. In figure 3 such a seven point-stencil has been depicted.

3.2 Boundary conditions

The definition of appropriate boundary conditions is crucial for the set up of the model. The following choices were made:

top and bottom: The nodes of the bottom and top surface of the cube were held fixed at a fixed temperature. For most of the simulations a top surface temperature of 237 K and a bottom surface temperature of 337 K was used.

sides: There is zero heat flow through the sides of the cube. This was achieved by copying the value of the node, that can be found right beneath the surface, to the surface point. This results in a zero temperature difference between these points and consequently the heat flow is also zero (compare with equation 12).

Alternatively one can use a fixed heat flux as boundary condition by directly discretizing Fourier's Law. Taking x in the direction perpendicular to the top and bottom surface we can write

$$\begin{aligned} q \quad &= \quad -k(x,y,z)\frac{\partial T(x,y,z)}{\partial x} \qquad (11) \\ &\simeq \quad -k(i,j,k)\frac{T(i+1,j,k) - T(i,j,k)}{h} \qquad (12) \end{aligned}$$

Here $i = 1\ or\ N$, to ensure that we are dealing with points on the top or bottom and q is a fixed parameter. Solving the above for T(i,j,k) one can use the expression to update the outermost nodes in accordance to the fixed heat flux conditions.

3.3 Algorithms used to solve system

There are a variety of algorithms available to solve the system of equations obtained in the last section. The ones used are presented in the following section.

3.3.1 Jacobi and Gauss-Seidel

The first method used was the *Jacobi* method. It is classified as a stationary, iterative method. This characterizes a wide range of techniques that use successive approximations to obtain more accurate solutions to a linear system with each step or iteration.

The Jacobi method is implemented by solving the discretized equation 8 for the required variable with respect to the other variables. In the present case the dependant variable is the node temperature $T(i, j, k)$ in dependence on the neighbouring temperature and conductivity values. One starts off with an initial guess for the temperature distribution. This may be a trivial guess like the one that all the interior nodes (excluding the one on the top and bottom surface) are equal to 273 K or a more advanced one like a linear gradient between the two fixed temperatures. The latter guess can result in a faster convergence.

The individual node values are now being updated according to the discretized equation. Since all the equations are being treated as independently from each other, the order in which the equations are being examined is irrelevant and a parallelisation is straightforward.
Having completed the updating of the values the boundary conditions at the side surfaces are being imposed again.

This process is repeated until the solution has converged. The convergence criterium is defined as $error \leq \epsilon$. Here ϵ is a parameter to be fixed and $error$ is defined as follows:

$$error = \sqrt{\sum_{i,j,k=1}^{n} [new\ node(i,j,k)\ value - old\ node(i,j,k)\ value]^2} \tag{13}$$

If we now assume that the equations are examined one at a time in sequence we can use previously computed values for calculating the update of a node within the same iteration cycle. This method is known as *Gauss-Seidel* and will normally converge faster then *Jacobi*.

3.3.2 Conjugate Gradient

There is also the possibility of expressing the system of equations in matrix notation. This results in an equation of the form $\mathbf{A}\vec{t} = \vec{b}$. Here $\mathbf{A}$ is the matrix representation of the discretised equations, $\vec{t}$ the solution vector comprising the temperature of the nodes and $\vec{b}$ a vector containing information about the rate of heat production for every node.

One can now make use of so called 'direct matrix methods' like Conjugate Gradient (CG) [5]. CG generates successive approximations to the solution by attempting to minimise a residual vector $\vec{r}$, which is defined as $\vec{r} = \mathbf{A}\vec{T} - \vec{b}$. The method is applicable for problems where the matrix **A** is symmetric and positive definite. It is supposed to improve the convergence speed drastically compared to Jacobi and Gauss-Seidel.

The difficulty lies in setting up the matrix **A** and especially implementing the boundary conditions in it. So far CG was only successfully applied to a homogenous model space, i.e. uniform conductivity. Further work must be done here.

4 Parallelising the code

The original code was written using *Fortran90* [6]. The parallelisation was realized using *High Performance Fortran(HPF)* [7]. HPF follows the data-parallel paradigm. The programmer does not actually intervene explicitly with the message passing task, but is restricted to give the compiler directives concerning the distribution of the data on the processors. The compiler, in our case the Portland Group Compiler: *pghpf-v21*, translates the code into a Fortran77 source with additional Message Passing function calls which will then compile to an executable file.

It is of great importance to find a distribution of the data that maximises the ratio of the local work done by the individual processor to the communication.

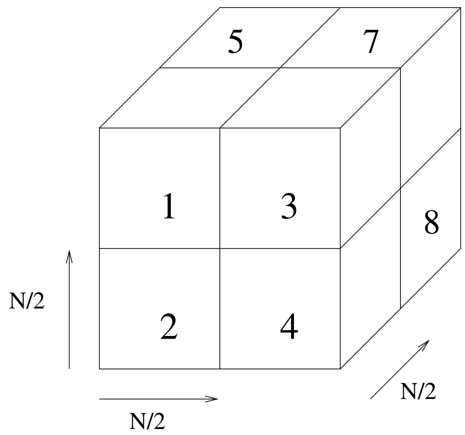

Figure 4: Virtual processor grid for data distribution

This is the case for a cube shaped processor-grid like the one sketched in figure 4. The main array containing the temperature values for every node is mapped onto this processor arrangement using the DISTRIBUTE(BLOCK,BLOCK) statement. So, referring to the sketched example this means that every one of the total of eight processor deals with a contiguous portion of the data comprising $(N/2)^3$ data entries. The other arrays containing the information concerning the conductivity and the heat source properties have to be distributed in the same way to ensure that communication between processors is not increased. This is achieved using the ALIGN directive for the latter two arrays.

Initially the parallel version of the code was run in parallel on the DEC-Alpha workstation machines, using eight of them to form a processor grid. It turned out that the interlinks between them are far too slow to deal with the amount of communication that arises during a simulation run necessary.

4.1 Performance with increasing number of Processors

The code was then imported on the Cray-T3D. In order to compare the performance of the parallel to the serial code it is necessary to calculate the so-called speed-up. The speed-up is defined as the CPU-time it takes for the executable to run using one processor divided by the CPU time taken for N processors.

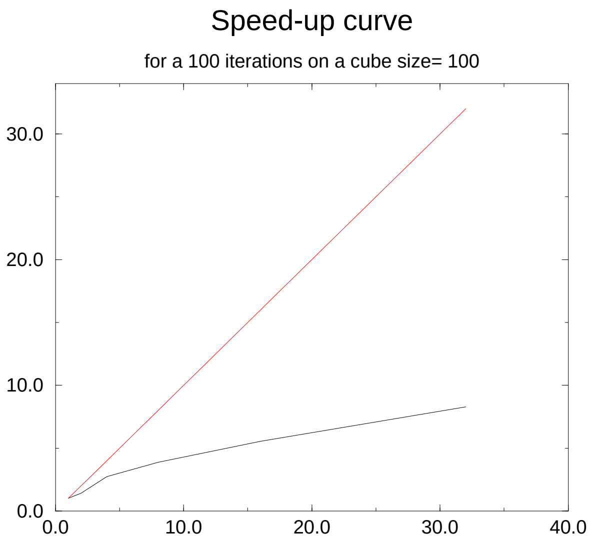

Figure 5: The ratio of CPU-Time for serial code divided by CPU Time for N processors in dependence on N

The result is shown in figure 5. Here the diagonal line represents the ideal scale up, i.e. doubling the number of processors halfs the necessary processing time. The lower curve shows the actual performance of the code.

To understand the apparent difference between the two curves it is necessary to look at the underlying communication. In the following a transcript of the message passing statistics is included. One is able to obtain this for a specific run by setting the *Mstats*-flag for the compiler.

messages	send cnt	send total	send avg	recv cnt	recv total	recv avg
0	5126785	92MB	19B	5125518	92MB	19B
1	5125518	92MB	19B	5125699	92MB	19B
2	5125518	92MB	19B	5125699	92MB	19B
3	5125518	92MB	19B	5125699	92MB	19B
4	5125518	92MB	19B	5125699	92MB	19B
5	5125518	92MB	19B	5125699	92MB	19B
6	5125518	92MB	19B	5125699	92MB	19B
7	5125518	92MB	19B	5125699	92MB	19B
tot.	41005411	738MB	19B	41005411	738MB	19B

The third column lists the average size of a message sent by a processor. At 19 bytes this is far too small. Apparently, the HPF-compiler is not sophisticated enough to detect that it would be more efficient to combine messages to larger packages and thereby decrease the number of messages exchanged. This communication overhead seems to be the main explanation for the deviation from the ideal scale up curve in figure 4. Using the *message-passing interface, MPI* it would have been possible to combine all data-points that have to be exchanged between processors into one message and perform a 'halo-swapping' at the end of every iteration.

Exploiting the HPF compiler directives it was tried to improve the performance. By setting the *-Moverlap* flag to one, it is possible to tell the compiler explicitly that the size of the overlap area of BLOCK distributed arrays is not going to be bigger than one . This improved the situation slightly because the default value for this was set to four.

4.2 The bottleneck - the data output

Another difficulty occurs when one attempts to write the data to disk at the end of the simulation run. The data has to be gathered from all the processors. This takes a rather long time compared to the calculations.

It was therefore attempted to implement a parallel output, ie every individual processor writes its results into a separate file and in a post-processing stage run on the Cray J90 the files are combined to a single file once again.

By using the external procedure *get-my-processor* each processors was given directives to save into individual files. It was ensured that during the saving process only data was accessed that according to the initial distribution directives should have been located on one processor.

Nevertheless, the compiler still prompted one with a *pghpf-get-scalar* warning for this part of the code. This means that expensive communication takes place and slows down the gathering of the data. Rereading the HPF-specification [HPF2] it seems that the actual physical processor arrangement is compiler-dependent and differs from the chosen arrangement of the abstract processors. This fact might explain the difficulties encountered.

5 Results

5.1 Visualisation issues and data format

During the initial stage of the simulation we used *Gnuplot* [9] to visualise the results. With the model advancing to three dimensions this was not sufficient.
The package called *Advanced Visualisation System AVS* was therefore deployed. It gives a variety of options to filter, alter and restructure the data. It is necessary to define the data format in a separate file, which is read in initially by AVS. Two data formats are being used:

scalar field : Every node is resembled in the data field by its cartesian coordinates (i, j, k) and scalar values for its temperature, its conductivity and its rate of heat production.

vector field : for every node the value of the heat flux in the x, y and z direction is stored.

In figure 6 one can find a transcript of the network assembled to process and to visualise the scalar data. To describe the individual tasks of the modules would go into too much detail, but it turned out that one of best ways to visualise the 3D data is to cut two orthogonal slices through the cube.

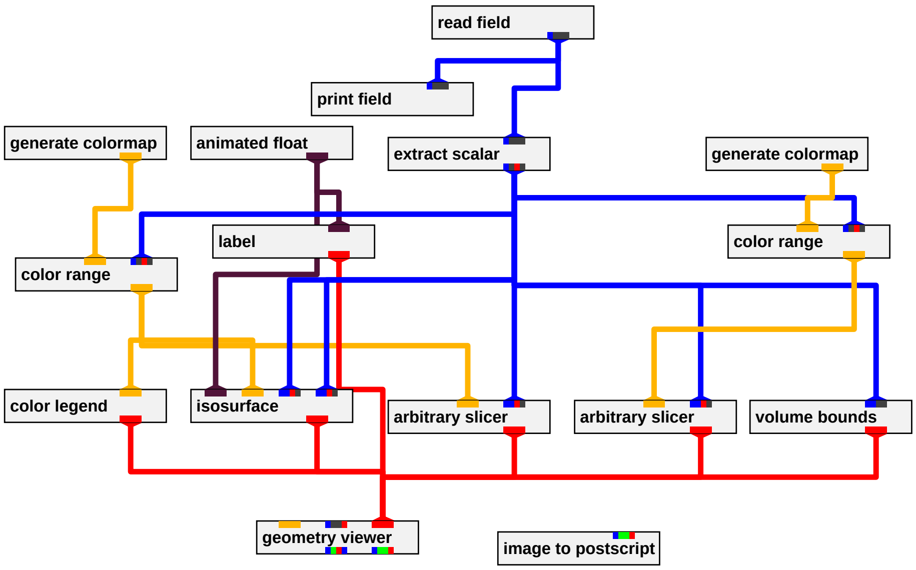

Figure 6: The visualisation network

5.2 Parameters for simulation

In the following parameters necessary for a simulation are listed and realistic values [10] mentioned.

Thermal conductivity, k: every node point of the simulation has to have a preassigned conductivity value. Realistic range: $1\ to\ 5\ W\ m^{-1}\ K^{-1}$

Heat Generation, s: every node is also assigned a value representing the heat being generated. Units are $J\ s^{-1}\ m^3$. Values can cover a wide range of values depending on the type of heat generation, i.e. natural radioactivity or volcanic intrusion.

Heat flow, q: can be used as boundary condition. Experimental data shows a realistic range from $0\ -\ 125\ mW\ m^{-2}$.A positive value stands for a heat flow into the cube.

Number of nodes, N: assigns the number of nodes per spatial dimension of the cube. Overall number of nodes is therefore equal to N^3. Values up to 50 were used.

Grid spacing, h: determines the spacing of the nodes and in combination with the number of nodes N fixes the overall spatial dimensions of the model cube. It was kept at unity, i.e. 1 m, for most of the time.

Numbers of Iteration, m: this parameter denotes the maximum number of iteration being done before the calculation is terminated, regardless whether the solution meets the convergence criterium at that stage or not.

Convergence criterium, ϵ: determines how tight the convergence criterium is set (compare section 3.3.1).

5.3 Case studies

5.3.1 A simplified trench

In the introduction (see fig. 1) an example was mentioned that could not be dealt with properly by a one-dimensional model. The model developed in this study now enables us to do so.

In figure 7 one can see two representations of the modelled volume. On the left side the arrangement of the different types of rocks are visualised. Two slices have been cut orthogonal to each other through the cube. The legend on the side represents the conductivity values. In the present situation a valley of low conductivity ($5\ W\ m^{-1}\ K^{-1}$) lies within a block of material at the bottom section of the cube that has a five times higher conductivity.

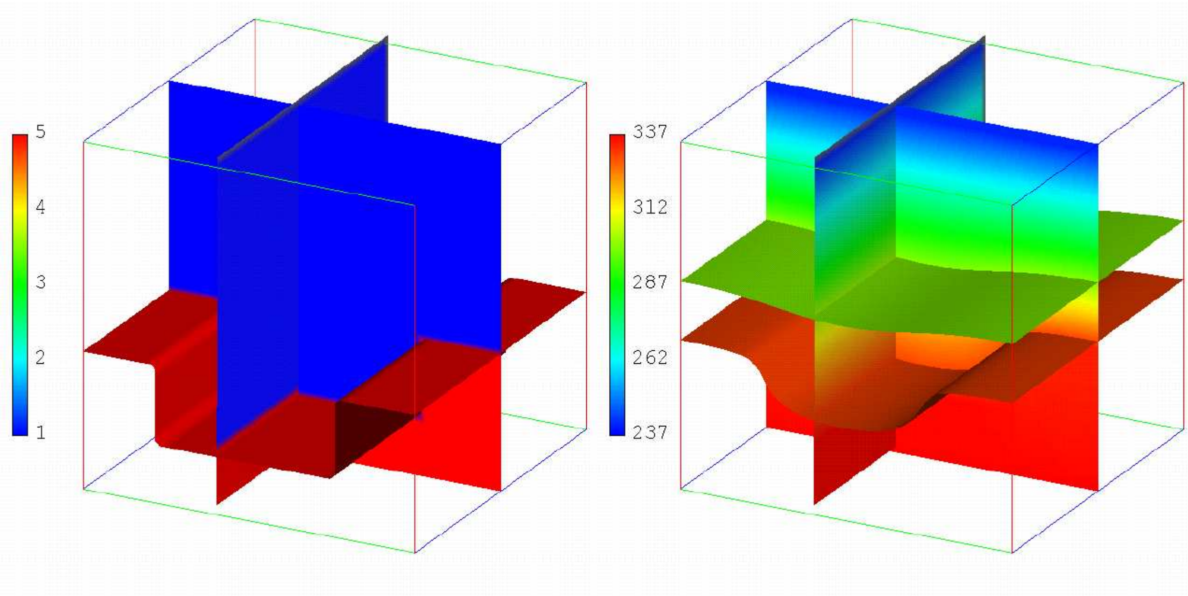

Figure 7: Simplified trench

The temperatures at the top is fixed at 237 Kelvin, whereas the bottom is held at 337 Kelvin. Number of nodes per side N is 40.

In the right hand part of figure 7 the temperature distribution obtained when the solution has converged is shown. As suggested in the introduction, the temperature within the valley is lower than in the surroundings. To visualise this two isotherms, surfaces of equal temperatures, have been drawn for approx. 330K and 295K into the image. Both show a clear depression for the region of low conductivity.

For further processing of the temperature data, gained from the simulation, a small Fortran code was written that reads in the field data and subsequently computes the gradient of the temperature in all three direction for every node point. Multiplied by minus the conductivity of the node this gives the directional heat flow. The vectors are then stored in a separate file and may be visualised using an altered AVS network.

This has been done in figure 8. For a two dimensional slice the vector magnitude has been calculated and is represented by orientation and length of *streamlines*. A change of colour occurs where streamlines overlap. It can be seen that the heat (entering at the left edge of the picture) flows partially around the shape of the 'trench' before leaving the area of high conductivity. The focusing of the streamlines at the upper edges of the valley is of particular interest.

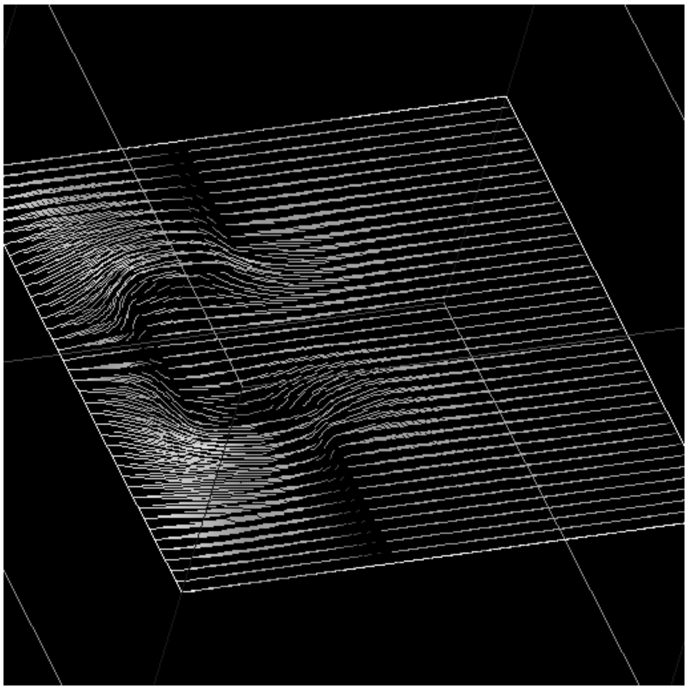

Figure 8: Vector representation of heat flux

5.3.2 Additional scenarios

Other figures for a variety of scenarios are listed in Appendix A. A brief list of their contents is given here:

Figure 9 : the example consists of 5 layers that have conductivity values ranging from $1\ to\ 5\ W\ m^{-1}\ K^{-1}$. Since all layers have the same spatial extents the problem is effectively just one dimensional. The resulting temperature distribution can be found on the right.

Figure 10 : a pillar of high conductivity is embedded in layers of varying conductivity. This could, for example, represent a salt deposit in sedimentary layers. The temperature distribution on the right shows how the heat flow is directed through the pillar into the centre of the volume.

Figure 11 : a columnar intrusion into a homogeneous area. The rate of heat generation of this dyke is shown on the left, the resulting temperature distribution on the left. There is also an isothermal surface enveloping the heat source and indicating that the heat spreads out in all directions from the dyke.

Figure 12: (LEFT) two identical, slanted ridges are placed on the bottom of the cube.(RIGHT) two slices where cut perpendicular to the extension of the ridges, to visualise the heat

flowing upwards through the ridges and being focused at the top.

6 Future work and Conclusion

The model for the three dimensional heat flow was successfully developed from scratch in *Fortran90* and parallelised using *HPF*. In a couple of scenarios it was shown that given sufficient data concerning the composition of the material the model could be used to study and determine the exact temperature distribution.

The number of possible geological scenarios that may be tested is virtually unlimited and hopefully work on this project will be continued on one point.A few of these possible directions are outlined below.

It would be desirable to make it easier to set up different scenarios. This could be done by writing procedures which enable the user to place geometrical objects within the model space. A first step in this direction was done by writing two procedures, that given the appropriate coordinates and parameters, assign nodes located within a block-shaped or pyramid shaped volume, a certain common value, e.g. thermal conductivity.Using these procedures it was possible to set up more complex scenarios containing for example fault blocks.

Within the model itself there is space for improvement. The spatial dimensions of the modelled space, i.e. the value of N, depends basically on availability of computing resources. The fact that most of the here described runs were done on a $40 \times 40 \times 40$ cube was due to the fact that AVS would have become to slow on the workstations for larger values.

Nevertheless, it would be worthwhile to decrease the time it takes for the solution to converge. Therefore it would be necessary to succeed in implementing a more sophisticated algorithm, like Conjugate Gradient mentioned earlier on, to solve the system of linear equations.

Last but not least, the performance of the parallelised code could be improved by making use of MPI instead of HPF.

References

[1] W. K. Hamblin and E. H. Christiansen: *Earth Dynamic Systems*. Prentice Hall, 1995

[2] H. U. Fuchs: *The Dynamics of Heat*, 301pp. Springer Verlag, 1996

[3] C. C. Clauser and Ernst Huenges: *Thermal Conductivity of Rocks and Minerals*, 105pp. American Geophysical Union, Reference Shelf 3 - Rock Physics and Phase Relations, 1995

[4] D. L. Turcotte and G. Schubert: *Geodynamics*, 134pp. John Wiley and Sons, 1982

[5] J. Demmel: *Application of Parallel Cmputers*, lecture 15/16, online: www.cs.berkeley.edu/ demmel/cs267, March 1996

[6] I. M. Smith: *Programming in Fortran 90*, John Wiley and Sons, 1995

[7] *Writing Data Parallel Programmes with High Performance Fortran*, EPCC internal course notes, Version 2.0

[8] *High Performance Fortran Language Specification*, Version 1.1 , 1994

[9] Thomas Williams and Colin Kelley: *Gnuplot manual, version 3.6*, 1995

[10] C. A. Stein: *Heat Flow of the Earth*, 144pp. American Geophysical Union, Reference Shelf 1 - Global Earth Physics, 1995

Biography for Dirk Carstens:

I am originally from Munich, Bavaria (Germany). Two years ago I moved to Scotland and joined a five years MPhys degree in Physics at the University of Edinburgh, which I will hope to finish in June 1999.

Acknowledgements:
Many thanks for all the support and help received from my supervisors, as well as from other people of EPCC like the two TRACS guys Jens Hoerner and Manfred Oevers.

The supervisors for this project are listed below:

Jon Turner, Dept of Geology and Geophysics, University of Edinburgh
James Iliffe, PGS Tigress Ltd
Jean-Christophe Desplat, EPCC

A List of figures

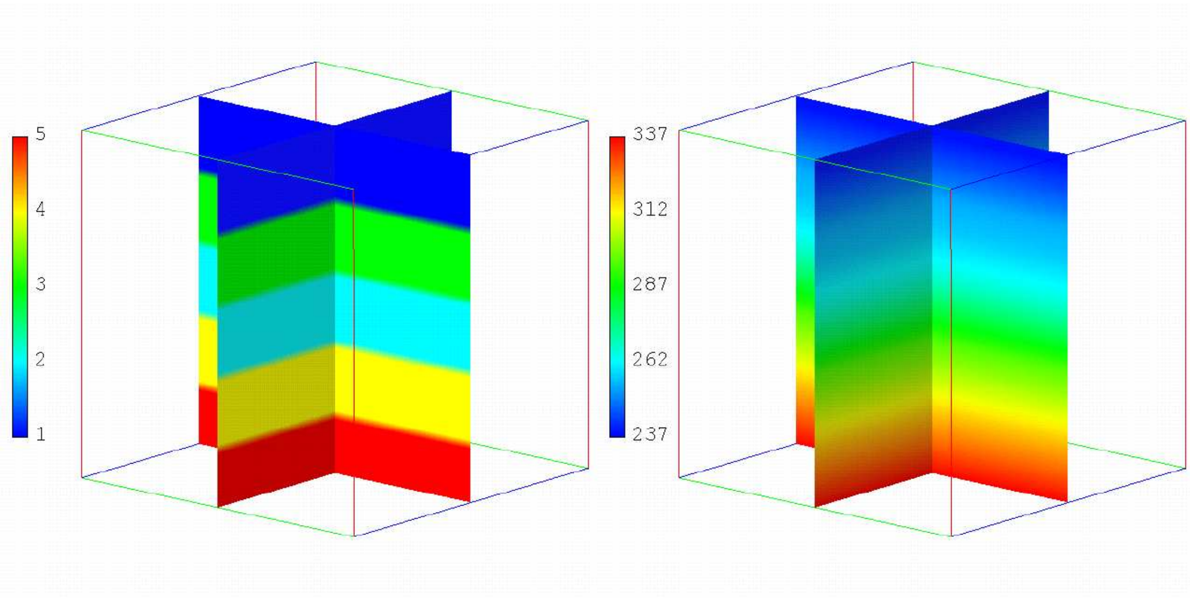

Figure 9: A layered structure

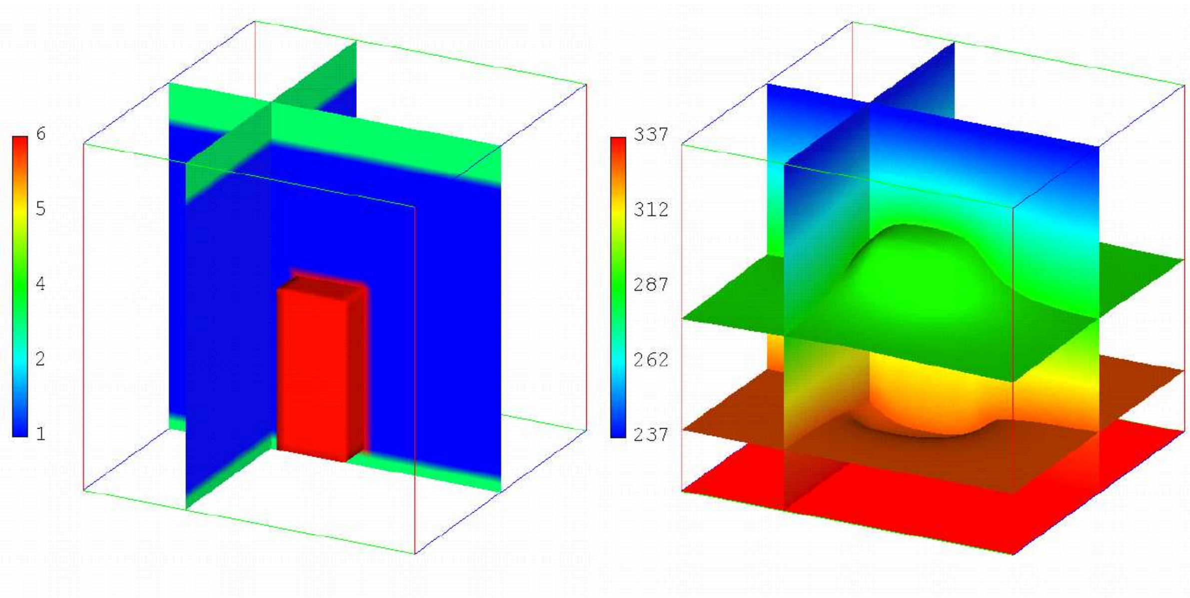

Figure 10: A salt pillar

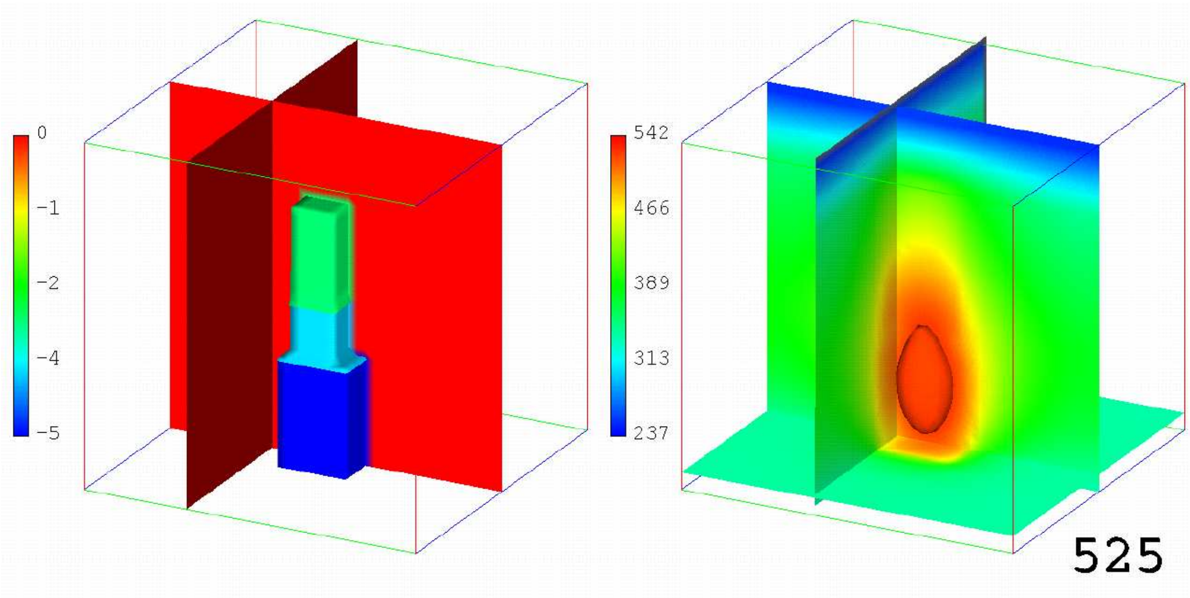

Figure 11: A volcanic intrusion

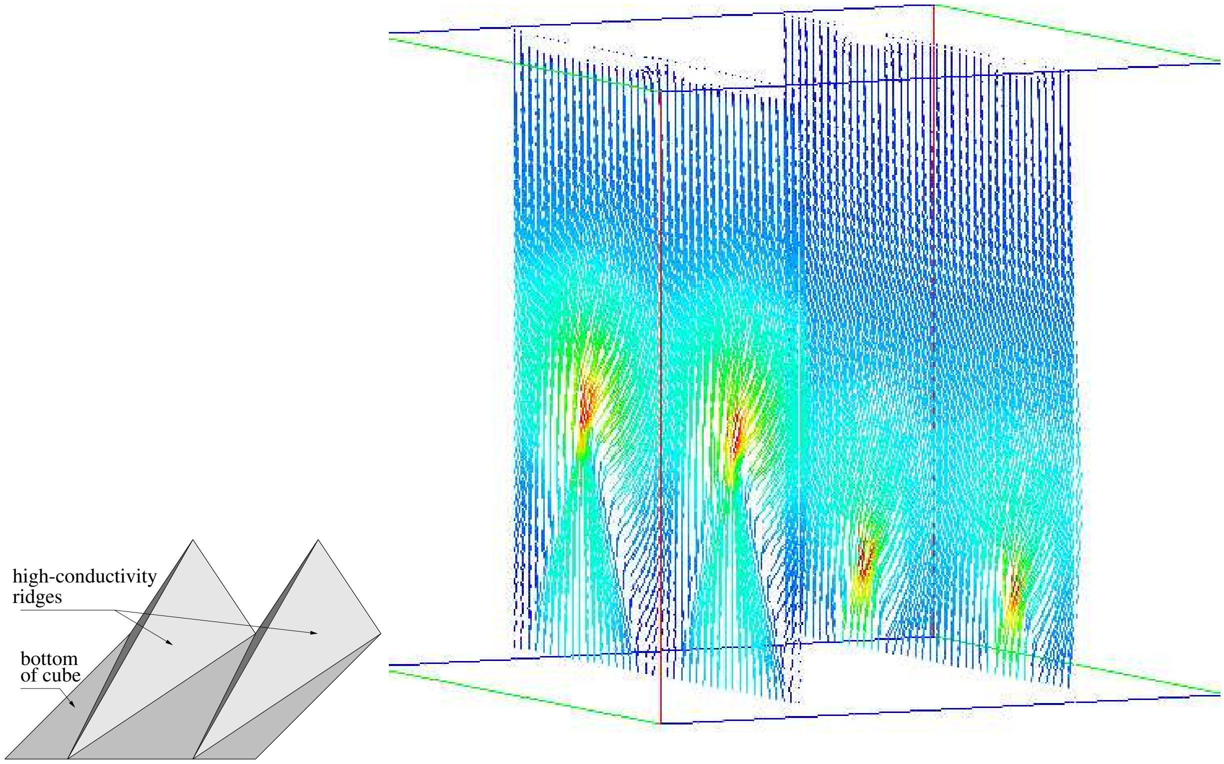

Figure 12: Two identical slanted ridges