EPCC-SS97-09

The World as a Cube

Ian Mulvany

Abstract

To the round earths four imagined corners; as refreshing as the poets simile this report presents a novel method for dealing with the solution of elliptic equations constrained to the surface of a sphere.

Contents

List of Figures

1 Introduction

Solving the problem of fluid flow on a sphere has importance in many areas, most notably meteorology and planetology. The equations describing fluid flow on the plane have been known in full for about 150 years, these are the Navier-Stokes equations. Solving them on a sphere amounts to mapping their solution on a plane onto a sphere. This introduces two primary difficulties:

- All mappings from the plane onto the sphere have singularities
- The intrinsic curvature requires a non unitary metric

The mapping which introduces fewest singularities in the steriographic mapping. Previous attempts to solve the fluid on a sphere problem have concentrated on combing two such maps so the both singularities are over-meshed by the complimentary map. In this report I will introduce a new approach to tackling this problem as proposed by Ronchi, Iacono and Paolucci in their paper 'The "Cubed Sphere": A New Method for the Solution of Partial Differential Equations in Spherical Geometry'

In this paper the authors outline a method for splitting the sphere into six equal contiguous areas which are in turn mapped onto the six faces of a cube. They derive the mapping equations and some basic vector equations from those mappings.
Using these I will demonstrate how to code a program for solving fluid flow on the sphere, and deal with some questions of parallel syntax along the way.

They also discuss the the suitability of their approach for implementation on massively parallel architectures.
Somewhere in here I deal with this and implement my code on a massively parallel architecture, ie the Cray T3D

This code used for this project was high performance Fortran.The motivation for writing it in HPF is to develop a non trivial algorithm that can be used to test various HPF compilers.

Results will be presented and conclusions will be made (I hope)

2 Background

A variety of different disciplines are combined in this project. A little Physics, a little mathematics and a lot of coding. Most of the material was new to me, so I will present a background on all aspects of the project.

2.1 The Physics

Computational Fluid Dynamics is concerned with solving the equations of fluid flow using numerical methods.
The general equation of motion for a fluid in a gravity field is

$$\rho[\frac{\partial \underline{v}}{\partial t} + (\underline{v} \cdot \underline{\nabla}\,\underline{v})] = -\underline{\nabla} P + -\underline{\nabla}\Omega + \eta \underline{\nabla}^2 \underline{v} + (\beta + \frac{1}{3}\eta)\underline{\nabla}(\nabla \cdot \underline{v}) \tag{1}$$

- $\rho[\frac{\partial v}{\partial t} + (\underline{v} \cdot \underline{\nabla}\, \underline{v})]$ describes how the momentum field varies over space and time
- $-\underline{\nabla}P$ is the contribution from the pressure field
- $-\underline{\nabla}\Omega$ is a term from the gravity field

The values of the pressure and the gravity vary over space and can be attributed a scaler value at each point. Their contribution to the motion is given by the gradient of their respective fields.

- $\eta\underline{\nabla}^2\underline{v}$ describes the "stickiness" of the fluid

This tells how easily the fluid runs past itself. the constant η is called the viscosity. A fluid with a high η is very sticky, for example treacle. A fluid with low η flows freely such as mercury.

- $(\beta + \frac{1}{3}\eta)\underline{\nabla}(\nabla \cdot \underline{v})$ the compressibility of the fluid

This tells us how squishy the fluid is and how well it can resist being squeezed into a smaller volume under force. β is some constant related to this internal property.
Using the following vector identity

$$(\underline{a} \cdot \nabla)\underline{a} = (\underline{\nabla} \times \underline{a}) \times \underline{a} + \tfrac{1}{2}\underline{\nabla}(\underline{a}^2)$$

equation 1 becomes

$$\rho[\frac{\partial \underline{v}}{\partial t} + (\underline{\nabla} \times \underline{v}) \times \underline{v} + \frac{1}{2}\underline{\nabla}(\underline{v}^2)] = -\underline{\nabla}P + -\underline{\nabla}\Omega + \eta\underline{\nabla}^2\underline{v} + (\beta + \frac{1}{3}\eta)\underline{\nabla}(\nabla \cdot \underline{v}) \tag{2}$$

Next we take the curl of both sides of the equation. Given that the curl of a gradient is zero and letting $\xi = \underline{\nabla} \times \underline{v}$ we get

$$\rho[\frac{\partial \xi}{\partial t} + \underline{\nabla} \times (\xi \times \underline{v})] = \eta\underline{\nabla}^2\xi + \underline{\nabla} \times (\beta + \frac{1}{3}\eta)\underline{\nabla}(\nabla \cdot \underline{v}) \tag{3}$$

If the velocity of the fluid is less than the velocity of sound in the fluid we can assume incompressibility.

$$\nabla \cdot \underline{v} = 0$$

This project is only interested in solving the steady state case so

$$\frac{\partial \xi}{\partial t} = 0$$

The equation of motion of the fluid attains its final form

$$\nabla \times (\xi \times \underline{v}) = \nu\nabla^2\xi \tag{4}$$

where $\nu = \frac{\eta}{\rho}$

We introduce a streamline function ψ where the velocity field can be obtained via $u = \frac{\partial\psi}{y}$ and $v = -\frac{\partial\psi}{x}$

Hence we can see that

$$\nabla^2\psi = \xi \tag{5}$$

and

$$\nu\nabla^2\xi = \frac{\partial\psi}{\partial y}\cdot\frac{\partial\xi}{\partial x} - \frac{\partial\psi}{\partial x}\cdot\frac{\partial\xi}{\partial y} \tag{6}$$

These are the physical equations that we are concerned with solving.

2.2 Finite Difference method

Equations 5 and 6 are continuum equations. To solve them on a computer they need to be discritised. The method employed was the finite difference method. The basic finite difference forms for partial derivatives can be derived from a partial Taylor series expansions. ∇ becomes a five point stencil and $\frac{\partial}{\partial i}$ becomes a central difference where i is the direction of concern. Equation 5 becomes

$$\psi_{i,j} = \frac{1}{4}(\psi_{i-1,j} + \psi_{i+1,j} + \psi_{i,j+1} + \psi_{i,j-1} - \xi_{i,j}) \tag{7}$$

and 6 becomes

$$\begin{gathered}\xi_{i,j} = \tfrac{1}{4}(\xi_{i-1,j} + \xi_{i+1,j} + \xi_{i,j+1} + \xi_{i,j-1}) - \\ \frac{R}{16}\{(\psi_{i,j+1} - \psi_{i,j-1})(\xi_{i+1,j} - \xi_{i-1,j}) - (\psi_{i+1,j} - \psi_{i-1,j})(\xi_{i,j+1} - \xi_{i,j-1})\}\end{gathered} \tag{8}$$

R is the Reynolds number and can be constructed in a dimension-less form. $R = \frac{\rho}{\eta}V_o h$ where h is the grid spacingV_o the relative velocity of the fluid. Setting ρ, V_o and h all to 1 R becomes simply the inverse of the viscosity.

At each intercept on the grid a value is ascribed to the field. In the time independent case each value is updated to the average of it's neighbouring values. The process continues until the entire field converges. Convergence is governed by some residue parameter. This being the difference over the whole field between any two updates. variety of algorithms can be employed to do this for example Jacobi, Gauss-Seidel, Conjugate gradient. My code uses Gauss-Seidel with successive over relaxation. Gauss-Seidel uses a red-black order scheme where all red points are updated based on the black ones and then the black ones are updated based on the values of the new red ones. In this way it leap frogs to convergence.

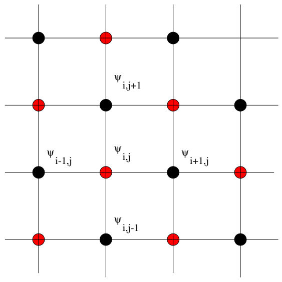

Figure 1: Red-Black ordering

2.3 Metric Transformations

When moving in a dimension we may wish to take a measurement of length. If we are constrained to one dimension we are moving along a line, two dimensions a plane, and three dimensions a space. (higher dimensions may be left for the mathematicians to contemplate)
Suppose now that we overlay our dimension with a standard grid. The nature of how length varies between grid points as we move around in our chosen dimension is a property known as the curvature of the dimension. Mathematically it is described by the metric of the dimension. The curvature of a flat plane is zero. As we move around our plane the lengths between our grid points remains unchanged. The metric of a flat plane is

$$\mathbf{g}_{i,j} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

This tensor is multiplied by our position to give a value for the variance of distances. On the plane this factor is always unity. Other surfaces have non zero curvature and hence a non identity metric. Sphere's have positive curvature and hyperbola have negative curvature.

3 The Problem on The Sphere

For the Finite difference method to work there must be uniform grid so that the five point stencil and the forward differences of equations 7 and 8 can be obtained. The sphere has non zero curvature and all of the complications of CFD on the sphere arise because of this. The problem is that no one regular grid can cover the sphere without the prescence of a singularity. Some previous attempts to get around this concentrate on having two skewed londitude lattitude grids and interpolating between them on the areas of overlap.

3.1 Cubing the sphere

This project deals with the problem by decomposing the sphere into six regular meshes each derived from two sets of angularly equidistant great circles. running between $-\frac{\pi}{4}$ and $\frac{\pi}{4}$

each panel corresponds to a face on a cube.

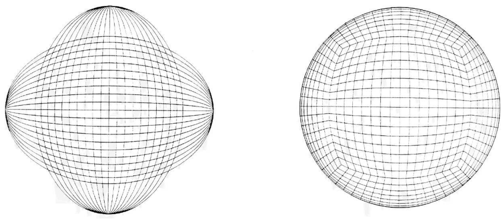

Figure 2: New Decomposition of Sphere

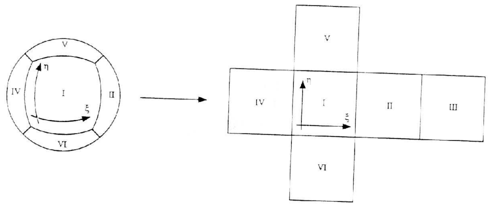

Figure 3: Cubing the sphere

3.2 Region Borders

If each ordinate of each face is divided into N pieces then all border points allign along the border. Moreover two neighbouring faces share the same verticies along the direction of adjoinment. This is because the grat circles generating thse vertices share a common origin at a point $\frac{\pi}{4}$ above or along from the centre of these verticies. The points of overlap not immediatly on the border then only coincide in one vertex. A one dimensional interpolation is required to fix these points.

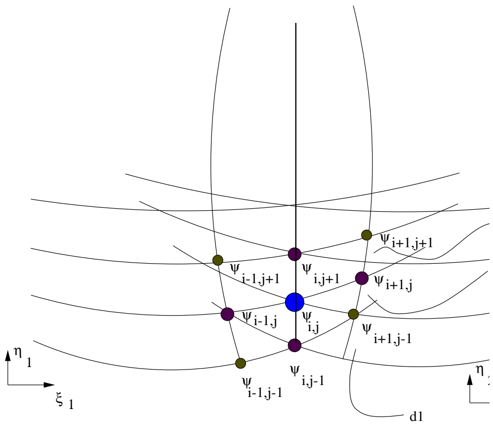

Figure 4: A border between regions

It will be shown later that the point shown above is updated from values of the field on the eight neighbouring points. In order to do this it is nessacary to evaluate *d1, d2* and *d3*

3.3 Metric on the Sphere

The symmetry of this method is reflected in the fact that the metric tensor is the same on all regions

$$\mathbf{g}_{i,j} = \begin{pmatrix} 1 & \frac{-XY}{CD} & 0 \\ \frac{-XY}{CD} & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{9}$$

where $X = tan(\zeta), Y = tan(\eta)$,$C = (1 + X^2)^{\frac{1}{2}}$, and $D = (1 + Y^2)^{\frac{1}{2}}$

The equation for in-viscid flow is

$$\nabla^2\psi = 0$$

which is

$$\frac{\partial\psi^2}{\partial x^2} + \frac{\partial\psi^2}{\partial y^2} = 0 \tag{10}$$

The finite difference scheme for this is

$$\psi_{i,j} = \tfrac{1}{4}(\psi_{i-1,j} + \psi_{i+1,j} + \psi_{i,j+1} + \psi_{i,j-1})$$

Under the new metric on each face **??** becomes

$$\delta[C^{-2}\frac{\partial}{\partial\zeta}(\frac{\partial\psi}{\partial\zeta}) + D^{-2}\frac{\partial}{\partial\eta}(\frac{\partial\psi}{\partial\eta}) + \frac{2XY}{C^2D^2}\frac{\partial^2\psi}{\partial\zeta\partial\eta}] = 0 \tag{11}$$

$\delta = \zeta$ and η are the coordinates local to each face. and the finite difference scheme becomes

$$\begin{aligned}\psi_{\zeta,\eta} = &\frac{fac1}{2\cdot fac1\cdot fac2}(\psi_{\zeta-1,\eta} + \psi_{\zeta+1,\eta}) + \frac{fac2}{2\cdot fac1\cdot fac2}(\psi_{\zeta,\eta-1} + \psi_{\zeta,\eta+1}) \\ &+ \frac{fac3}{8\cdot fac1\cdot fac2}(\psi_{\zeta+1,\eta+1} - \psi_{\zeta+1,\eta-1} - \psi_{\zeta-1,\eta+1} - \psi_{\zeta-1,\eta-1})\end{aligned} \tag{12}$$

where $fac1 = \delta C^{-2}$, $fac2 = \delta D^{-2}$ and $fac3 = \delta\frac{2XY}{C^2D^2}$

with the diagonal contributions in the third part of the left hand side coming from the fact that ζ and η are non orthogonal.This is why, as mentioned earlier each update requires the eight nearest neighbours. This has implications for using the Gauss-Seidel algorithm. The red-black ordering pattern is coupled and the red and black are no longer independent from each other. This indicated that the use of some other algorithm such as conjugate gradient might be more appropriate. This difficulty was noticed late in the project and no attempt was made to correct for it. The Gauss-Seidel algorithm still seemed to give intuitively reasonable results and this seemed justification enough not to panic.

3.4 Interpolation

3.4.1 edges

A liner interpolation function was assumed so the value of the field between points is α at a, β at b and

$$\alpha + val(\beta - \alpha)$$

at a point between a and b where val runs from 0 to 1. To calculate the vale of the field at the intersection point shown in the diagram below we need to find the ration of the distance *d1* to the distance it $\dot{2}$.

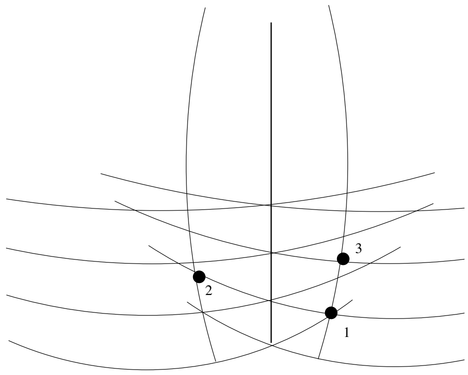

Figure 5: Points needed for interpolation

Since we can assume that the Sphere is a unit sphere and given that all of the angles are in radians, the ratio of distances is simply the ratio of angles that these points make with the origin of the sphere.

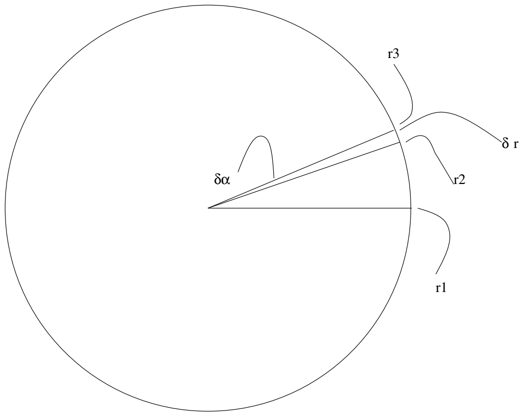

Figure 6: Distances on the sphere

d1 is the difference between point 2 and 1 *d2* is the difference between points 3 and 1. The value of val is then $\frac{|13|}{|12|}$. To calculate the distances the points 1,2 and 3 have to be converted into global coordinates. The relationship between the local coordinate system of ζ and η and a global spherical coordinate system using r, θ, φ is given by

$$\begin{pmatrix} A_\zeta \\ A_\eta \end{pmatrix} = \begin{pmatrix} 0 & \frac{CD}{\delta^{\frac{1}{2}}} \\ -1 & \frac{XY}{\delta^{\frac{1}{2}}} \end{pmatrix} \cdot \begin{pmatrix} A_\theta \\ A_\varphi \end{pmatrix} \tag{13}$$

This means that for some given point $A_{\zeta,\eta}$

$$A_\theta = \frac{tan(\zeta)tan(\eta)}{(1+tan(\zeta)^2)^{\frac{1}{2}}(1+tan(\eta)^2)^{\frac{1}{2}}} - A_\zeta$$

For every point along the edges at least two of these interpolations will have to be preformed

3.4.2 Corners

On the corners things are not so clear. The Field on the corner is shared by 3 different grid points. Each corner point has only six nearest neighbours due to an overlapping of points.

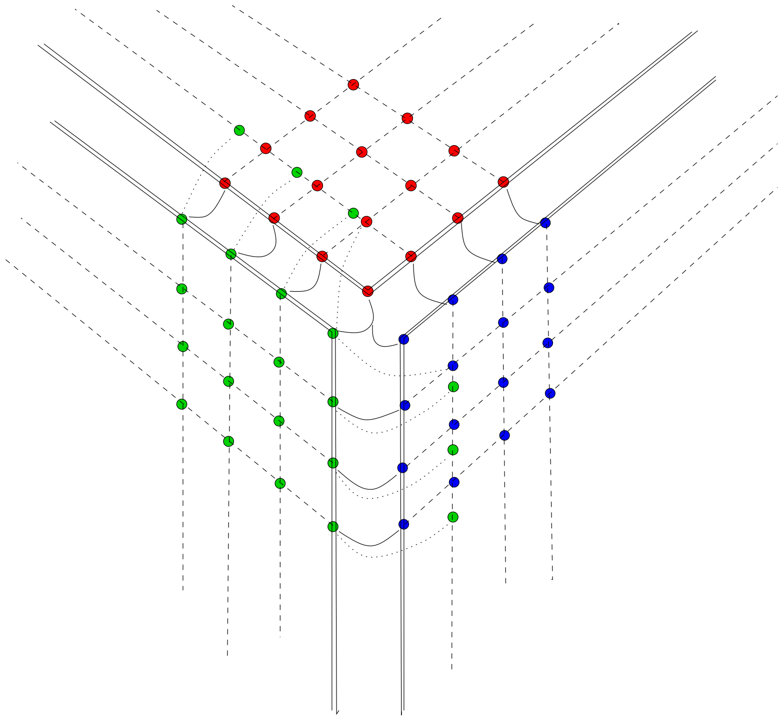

Figure 7: Corner of virtual cube

4 The Code

Values for the grid spacing and the relaxation parameter are inputted into the code manually before compilation. From the grid spacing the program sets the sizes of the arrays. Subroutines are called which calculate the factor array. and the interpolation array. The factor array is a set of values, three for each location on the grid, which take account of the effect of curvature

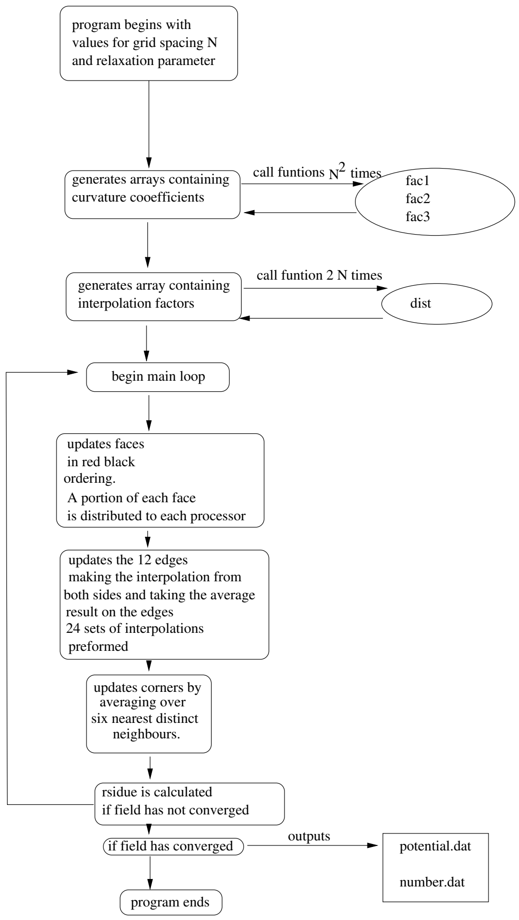

Figure 8: Plan of code

on the sphere. The interpolation array calculates the values of the intersection distances *d1...N* along an edge. So long as each face is divided by the same number of grid lines the intersection distances need only be calculated once.

The main loop of the programme updates the faces followed by the edges followed by the corners. A residue is calculated and the loop proceeds to convergence.

At the edges interpolating from different directions will reference different points because the panels only coincide along the edges. I decided to preform the interpolation twice, once from each direction, and then to take average value between these two interpolations as the value of the edge points. To date the corners are just taken to be the average of their six nearest distinct neighbours.

Finally two files are written as output.

- number.dat which contains the number of grid spaces.
- potential.dat which contains the potential field ψ

A second programme was written to read in these files and output files contain-g the velocity fields in a form readable by AVS.

5 Parallelisation

The code was written in HPF. The only parallelisation issue considered was the way in which the main arrays would be distributed. At present the parallelisation being used is

```
!HPF$ distribute (block,block,*)   :: Psi
!HPF$ align with psi :: Oldpsi,mask
```

and the code doesn't work on the Cray so there are no results to indicate which type of distribution is the better.

6 Results

6.1 syntax

Different types of syntax were tested on the Cray. The intrinsic functions eoshift and cshift were tested against using array triplet specifier assignments. The array triplet was found to be fastest but not by any great order of magnitude. For the same code with only the manner of array assignment changed the following results were obtained

Syntax used	time taken (s)
eoshift	3.1098258839121806
cshift	2.4365346986378773
array triplets	2.1603929661698307

6.2 One Panel

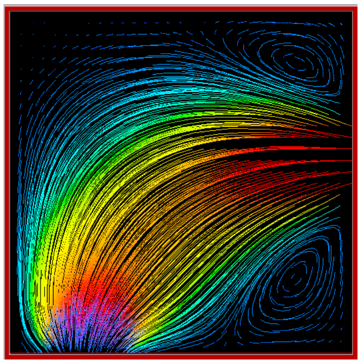

Figure 9: Fluid flow on plane

This is a picture of fluid flowing into a closed cavity on the plane. The program calculates viscosity and vortices can be seen at the corners. The Reynolds number for this flow was 1.7

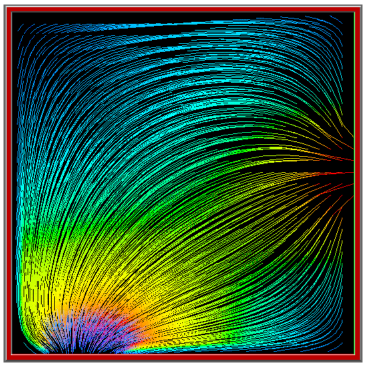

Figure 10: Fluid flow on sphere

This is a picture of the same flow, but without viscosity and on the surface of a sphere. The flow looks similar to that on the plane and a picture of flow on the plane looks identical however if you take a difference field of this flow between the plane and on a section of sphere the following figure is obtained.

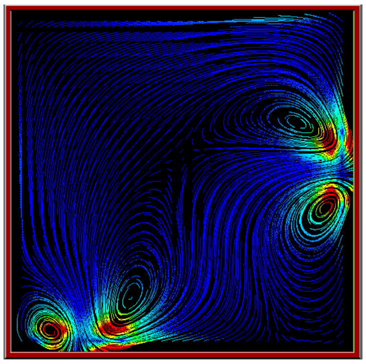

Figure 11: A difference plot

6.3 The linked cube

The boundary conditions for the above cases all involved solid walls. Clearly for the sphere-d cube this will not be the case. Initial conditions that were tested were predominantly permutations of having point sources of waves or beam sources of waves.

This is a picture of a beam souce that starts at the centre of face one and continues horizontally through to the centre of face four. At the time the top and bottom of the cube (faces V and VI) are not connected.

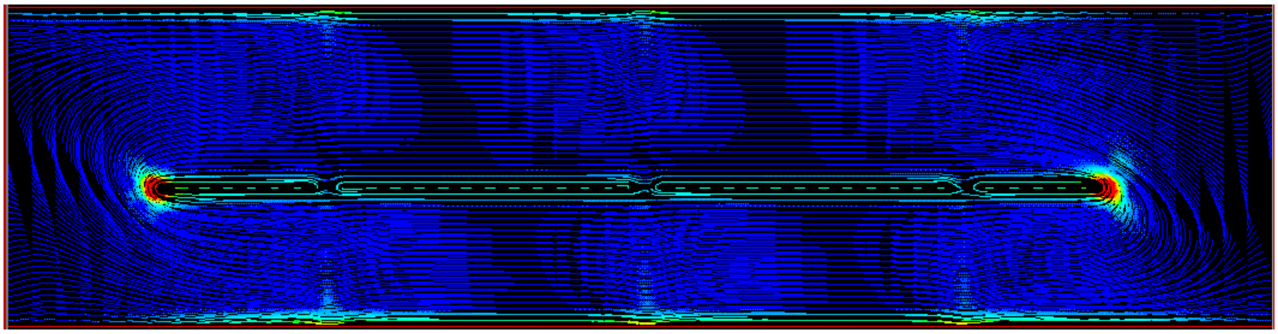

Figure 12: Faces I - IV joined

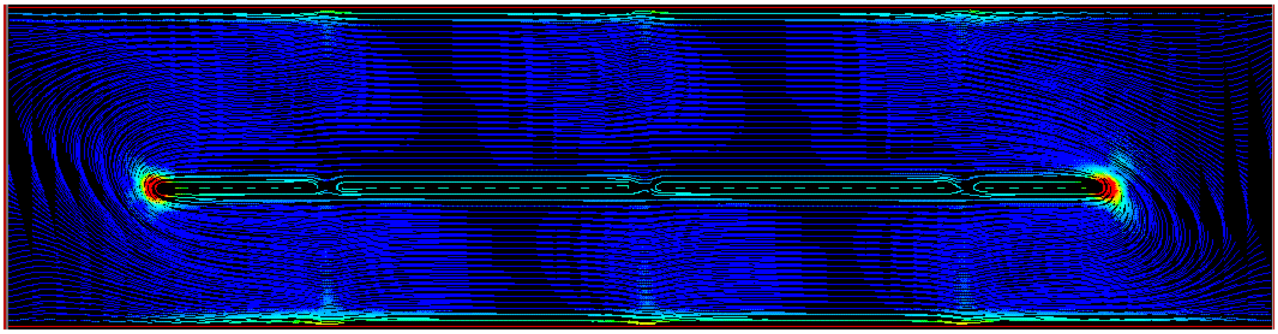

Figure 13: A variety of sources

6.4 The linked cube

7 Errors

The main sources of error in the program are bound to lie in the interpolation sections and the sections dealing with the calculation of the corners. The code which returns values for the overlap distance d_N should intuitively give a value of 1 for its first value. In fact it gives a value of 0.631 . My intuition might be wrong.

It might not be nesacary to do the interpolation along each edge twice as the extra calculation might not improve the second order accuracy already present with just one interpolation.

As mentioned above the corners are only dealt with in an approximate fashion. Instead of the field actually being calculated for the corners they are just smeared out to the average of the value of their nearest neighbours. However those neighbours have good values for the field and so the corners them selves do not tend towards critically erroneous values as would happen with a longitude latitude grid at the poles.

8 Further Work

Work that could be conducted leading on from this project

- Further visualisation

Including mapping the results onto a sphere so that an undistorted view of the flow could be seen and creating a movie of the flow heading to convergence. All though this movie would have no physical significance it would give an indication of how the code could be optimised to speed u convergence

- Inclusion of viscosity

This should be relatively straightforward. It simply need the inclusion of " " terms into the calculation of the field at every point.

- Solving for the time dependant case

This would be hard but by no means impossible

- Improving parallelisation

Perhaps converting the code to MPI to test the relative merits of the two predominant methods of parallelisation.

9 Conclusion

My main failing with this project has been to overlook the order of accuracy to which the various approximations are valid. As a result of this I have no clear idea of how well the code relates to the real world. Certainly this is difficult when dealing with 'dry water'.

However with this in mind the results obtained look intuitively good.

The author of this project is a student of Astrophysics at the university of Edinburgh. He is never happier than when dangling precariously over some precipice, or programming in parallel with panache. He is Irish but has never had it held against him

Supervisor: David Henty
A great humanist and veteran of the insect liberation cause. He has never spurned the call of nature. David speaks many languages and is one of a rare breed, a proponent of HPF programming

Acknowledgements:
I wish to thank primarily
My supervisor : David Henty
the EPCC
the SSP (they will kill me if I don't)