2D Spectral Element Scheme for Viscous Burgers' Equation

P. Aaron Lott
University of Maryland
Department of Applied Math
Scientific Computation

Advisor: Dr. Anil Deane

May 18, 2004

Contents

List of Figures

Abstract

In this report, we discuss the theory, design and implementation of an Adaptive Spectral Element Method using p-type refinement. We solve the 1D and 2D viscous Burgers' Equations. Finally, we compare our 1D results against analytical and numerical solutions to validate our code, show preliminary 2D results for 2D Viscous Burgers' Equation, and validation of our 2D linear advection scheme.

1 Motivation/Scientific Context

To model the dynamics of the Earth's Mantle we treat it as a highly viscous, incompressible Boussinesq fluid [4]. It is important to study the affects of flows over long time periods to better constrain the parameter space of the model. There is seismic and geochemical evidence of chemical/structural phase transition at depths of 410 km and 670 km. There are viscosity changes of several orders of magnitude. To handle these sharp interfaces one needs a refinement method to efficiently study the flow in this region.
The Governing Equations are:

$$\frac{1}{\rho}\nabla p = \nu\nabla^2 u - g\alpha\Delta T \qquad \text{(Momentum Equation)} \tag{1}$$

$$\frac{\partial T}{\partial t} + u\cdot\nabla T = \kappa\nabla^2 T + \frac{J}{\rho C_p} \qquad \text{(Thermal Energy Equation)} \tag{2}$$

$$\nabla\cdot u = 0 \qquad \text{(Incompressibility Equation)} \tag{3}$$

u	Velocity
T	Temp
p	Pressure
ν	Viscosity
κ	Thermal diffusivity
α	Thermal expansion coefficient
ρ	Density
C_p	Heat capacity at constant pressure
J	Rate of internal pressure per unit volume
g	Gravity

In the Momentum equation $\frac{du}{dt} = 0$ because we have an infinite Prandtl number, $Pr = \frac{\nu}{\kappa}$. Note, these are the Incompressible Steady Stokes Equations with the source term ΔT coming from by the unsteady, advection diffusion equation at each time step.
For this project we want to implement an p-adaptive Spectral Element scheme to solve the Advection Diffusion equations in 1D and 2D, with advection velocity $\vec{c}$ and viscosity ν. This code will provide a testbed for the refinement methods to be used to investigate mantle flows.

$$\text{1D} \quad \frac{\partial u}{\partial t} = -(\vec{c}\frac{\partial u}{\partial x}) - \nu\frac{\partial^2 u}{\partial x^2} \qquad \text{in } \Omega\in\mathbb{R} \quad t\geq 0 \tag{4}$$

$$\text{2D} \quad \frac{\partial u}{\partial t} = -\vec{c}\cdot\nabla u - \nu\Delta u \qquad \text{in } \Omega\in\mathbb{R}^2 \quad t\geq 0 \tag{5}$$

Note $\vec{c} = u$ yields the viscous Burgers' Equations.

2 Spatial Discretization-Spectral Element Method

To solve these equations efficiently while maintaining a high working accuracy over long time periods, we choose the Spectral Element Method (SEM) for our spatial discretization. We see from figure (2) that high order methods require much less work to maintain a desired working

accuracy over long time scales as opposed to lower order methods such as Finite Differencing, or Finite Element. This is due to the exponential convergence property of SEM, compared to the algebraic convergence of low order schemes. In this section we discuss the background of SEM, and its convergence properties.

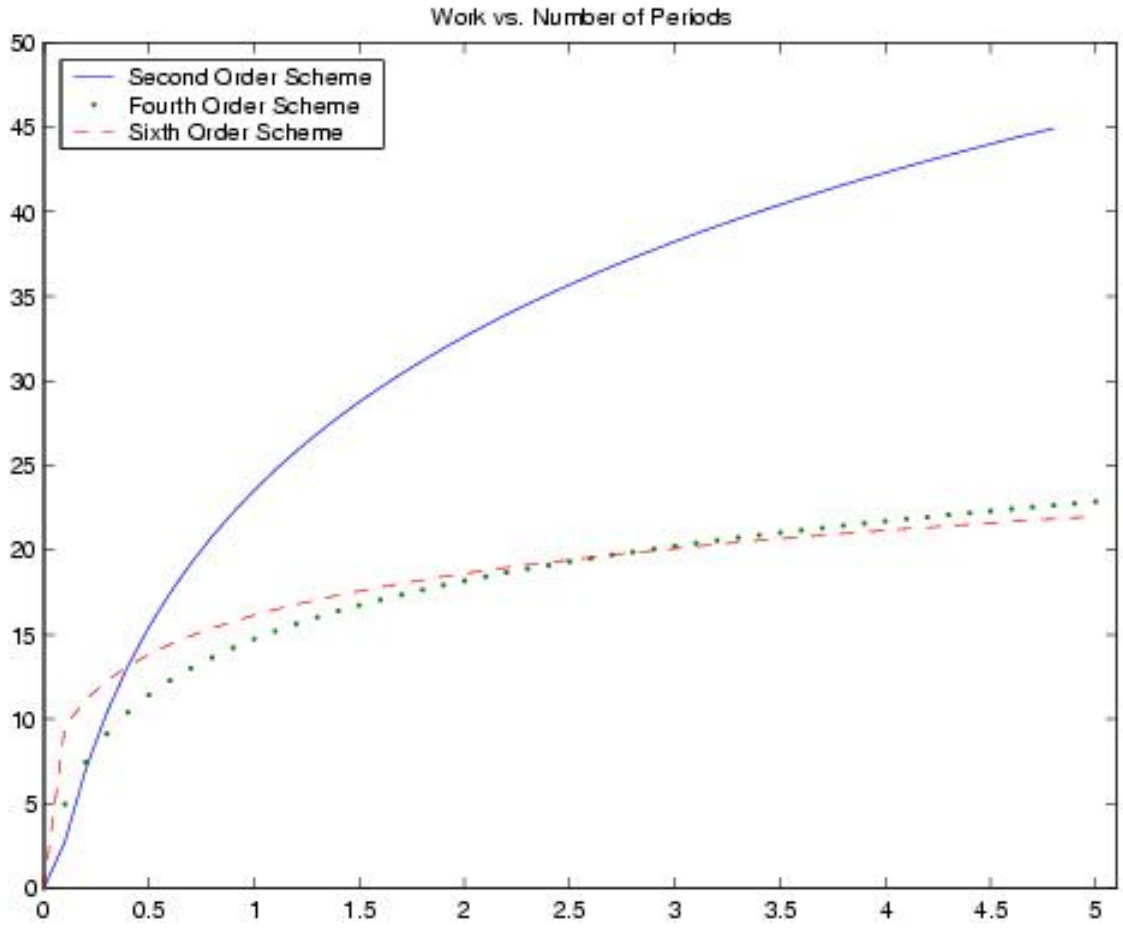

Figure 1: Computational Work (FLOPS) required to integrate a linear advection equation for 5 periods while maintaining a cumulative phase error of $\varepsilon = 10\%$.[5]

2.1 Constructing SEM - Method of Weighted Residuals

Suppose we have a linear differential equation in the domain Ω

$$L(u) = 0. \tag{6}$$

We assume $u(x,t)$ can be accurately represented as

$$u^{\delta}(x,t) = u_0(x,t) + \sum_{i=1}^{N} \tilde{u}_i(t)\phi_i(x) \tag{7}$$

where $\phi_i(x)$ are trial functions, $\tilde{u}_i(t)$ are unknowns, N refers to the number of degrees of freedom, and $u_0(x,t)$ is chosen to satisfy initial and boundary conditions.

Using the method of weighted residuals, we substitute 7 into 6 to produce the non-zero residual R such that:

$$L(u^{\delta}) = R(u^{\delta}) \tag{8}$$

In order to solve for $\tilde{u}_i(t)$ we restrict R to satisfy

$$(v_j(x), R)_{\Omega} = 0 \quad j = 1, ...Ndof \tag{9}$$

where $v_j(x)$ are test functions. The problem now reduces to a system of ODE's in $\tilde{u}_i(t)$.

By choosing the test function $v_j(x) = \phi_j(x)$ one obtains a Galerkin (modal) method, and by choosing $v_j(x) = \delta(x - x_j)$ (dirac delta) one obtains a Collocation (nodal) method. This technique is used to construct the spectral element discretization of a PDE. For SEM the trial functions $\phi_i(x)$ are to be Chebyshev polynomials $T_n(x)$, Legendre polynomials $L_n(x)$, or some member of the Jacobi polynomials $P_n^{\alpha,\beta}$. [5]

2.2 Elements

In the previous section we defined a global method for solving PDE's. However, due to having one computational domain some complexities arise. Complex geometries and Boundary conditions can be difficult to accomadate, and global transforms require a lot of communication overhead. To counter these difficulties the computational domain is partitioned into a collection of subdomains or "elements" Ω_e.

These elements are similar to the elements of a Finite Element grid. However, instead of having low order basis functions on these subdomains, we retain the high order approximation of spectral methods by using spectral basis functions $\phi_i(x)$ on each element.The Spectral Element Method combines the flexibility of the Finite Element Method with the accuracy/convergence properties of Spectral Methods. For this project, we use Gauss-Legendre-Lobatto polynomials as our local basis functions. We can now write our solution in terms of local and global modes,

$$u_N^e(x) = \sum_{i=0}^{N} \tilde{u}_i(t)\phi_i(x) = \sum_{e=1}^{Nel} \sum_{p=0}^{P} \pi_p^e(\xi)\tilde{u}_p^e. \tag{10}$$

$\phi_i(x)$ are global modes, whereas $\pi_i^e(\xi)$ denotes the i^{th} degree GLL polynomial scaled to the size of the local element Ω_e, and zero on all other elements. Thus this is a "local" calculation, except on element boundaries.

2.3 Properties of SEM

2.3.1 Convergence

At the beginning of the section we mentioned that SEM has an exponential convergence property which yields an advantage over low order schemes for computing flows over long time periods.

Definition 1 (Algebraic Convergence) *For fixed polynomial degree and increasing number of elements, $u_n(x,t)$ will algebraically approach $u(x,t)$, that is, as we double the number of elements, we get roughly $\frac{1}{2}$ the error.*

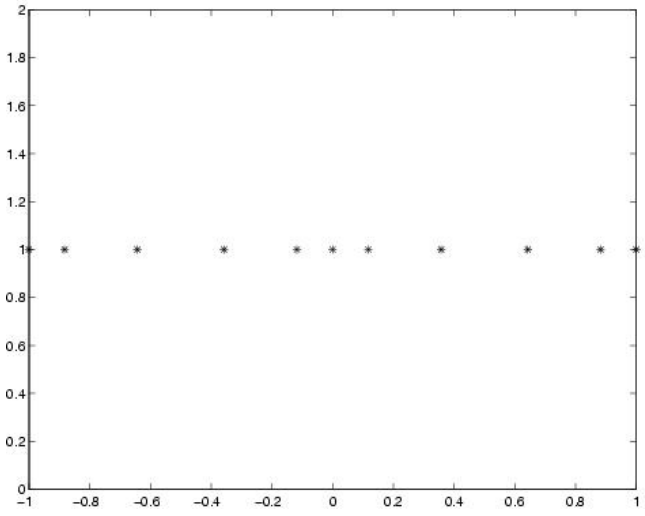

Figure 2: 1D GLL grid NEl=2 P=5

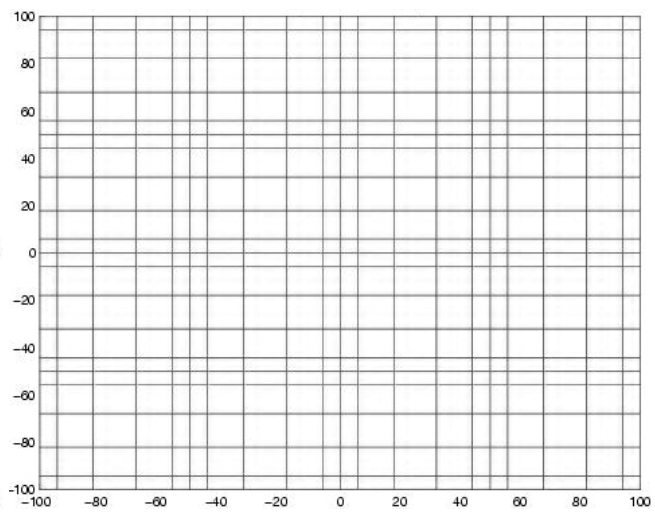

Figure 3: 2D GLL grid. NEl=16, Px=5, Py=5

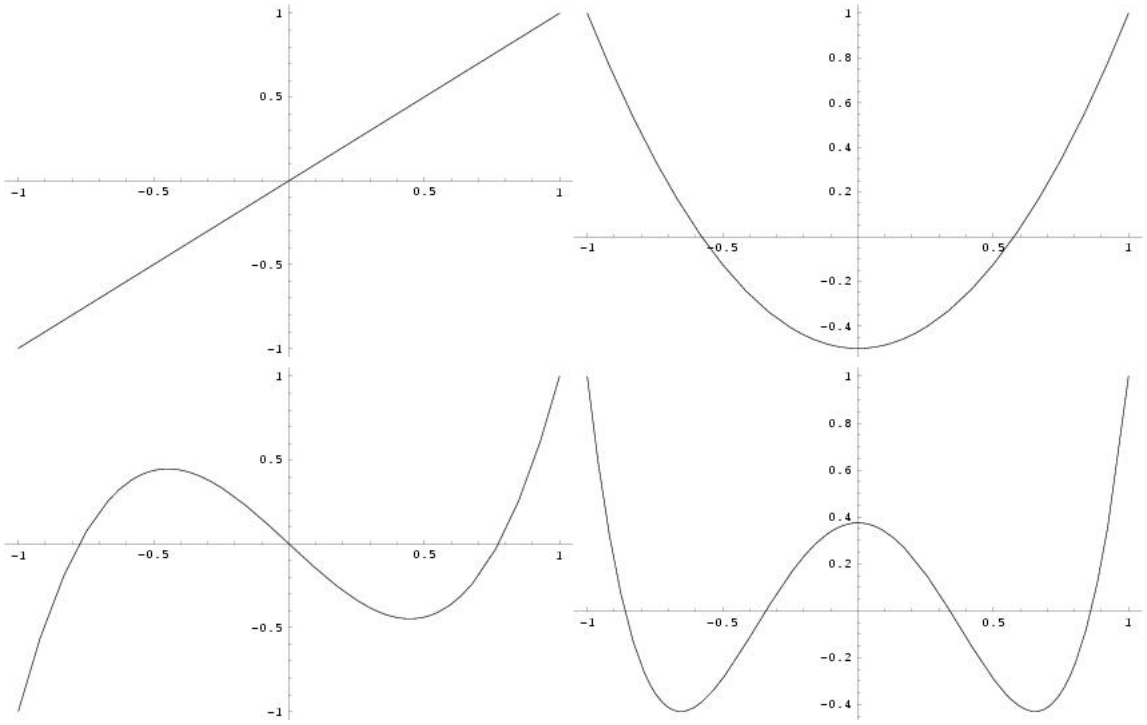

Figure 4: GLL Polynomials of degree 1 through 4

Definition 2 (Exponential Convergence) *For fixed number of elements and increasing polynomial degree, $u_n(x,t)$ will exponentially approach $u(x,t)$, that is, as we double Polynomial degree on each element we get roughly 2 orders of magnitude error reduction.*

2.3.2 Tensor Products

The Spectral Element Method scales well to higher dimensions. One of the key reasons for this, besides locality, is the tensor product formulation of spectral elements in higher dimensions. [8]

In 2D we can write our solution

$$u_{M,N}(x,y) = \sum_{i=0}^{M}\sum_{j=0}^{N} u_{ij}\pi_{M,i}(x)\pi_{N,j}(y). \tag{11}$$

Definition 3 (Kronecker Tensor Product) *If $A_{k\times l}$ and $B_{m\times n}$ the Kronecker Tensor Product $C_{km\times ln} = A\otimes B$ is given by*

$$C := \begin{pmatrix} a_{11}B & a_{12}B & \dots & a_{1l}B \\ a_{21}B & a_{22}B & \dots & a_{2l}B \\ \vdots & \vdots & & \vdots \\ a_{k1}B & a_{k2}B & \dots & a_{kl}B \end{pmatrix}. \tag{12}$$

Thus, if we want to perform a linear operation on u, for example a derivative, it can be written as a tensor product of two matrices times u. Suppose w_{pq} represents

$$\frac{\partial u}{\partial x}(\xi_{M,p},\xi_{N,q}) \;=\; \sum_{i=0}^{M}\sum_{j=0}^{N} u_{ij}\pi'_{M,i}(\xi_{M,p})\pi_{N,j}(\xi_{N,q}) = \sum_{i=0}^{M} u_{iq}\pi'_{M,i}(\xi_{M,p})$$

In matrix-vector format this is written as

$$w = D_x u := \begin{bmatrix} \hat{D}_x & & & \\ & \hat{D}_x & & \\ & & \ddots & \\ & & & \hat{D}_x \end{bmatrix} \begin{pmatrix} u_{00} \\ u_{10} \\ \vdots \\ u_{MN} \end{pmatrix} \tag{13}$$

where $\hat{D}_x$ is the one dimensional derivative matrix. D_x and D_y can be expressed as $D_x = I\otimes\hat{D}_x$ and $D_y = \hat{D}_y\otimes I$.

Now we see how tensor products turn up in spectral element methods, but the reward of using them is in this property.

$$(A\otimes B)\vec{u} = BUA^T \tag{14}$$

where U is the properly reshaped version of $\vec{u}$. Thus, calculations of the form

$$(A\otimes I)(I\otimes B)\vec{u} \tag{15}$$

which would be $O(n^4)$ for square A and B using straight forward matrix vector multiplication, is reduced to a $O(n^3)$ calculation. In general, matrix vector operations involving a discretization with n mesh points per spatial dimension, with d spatial dimensions results in $O(n^{d+1})$ operations.

2.4 1D Viscous Burgers' Equation Discretization

Starting with the PDE

$$u_t + uu_x = \nu u_{xx}, \qquad \text{in } \Omega = [a,b]$$

we assume the solution

$$u(x,t) \approx u_0(x,t) + \sum_{i=1}^{N} \tilde{u}_i(t)\phi_i(x), \tag{16}$$

and use the method of weighted residuals to obtain

$$R(u) = (u_N)_t + u_N(u_N)_x - \nu(u_N)_{xx}.$$

Next we set the inner product $(v_j(x), R)_\Omega = 0$, which gives us the integral equation

$$\int_\Omega v_N(x)(u_N)_t dx + \int_\Omega v_N(x)u_N(u_N)_x dx - \nu \int_\Omega v_N(x)(u_N)_{xx} dx = 0.$$

Now we partition $\Omega = [a,b]$ into Nel elements Ω_e each with step size h_e, and break up the integral into the sum of the integrals over each element.

$$\sum_{e=1}^{Nel} [\int_{\Omega_e} v_N(x)(u_N)_t dx + \int_{\Omega_e} v_N(x)u_N(u_N)_x dx - \nu \int_{\Omega_e} v_N(x)(u_N)_{xx} dx] = 0$$

The important thing to realize is that $\pi_i(x)$ is a P degree Legendre Polynomial, and if we use the $P+1$ scaled Gauss-Legendre-Lobatto points in each element Ω_e to perform the numerical quadrature, we get an exact quadrature on each element for polynomial degrees up to$2P+1$.
After scaling, and applying GLL quadrature rules, we get the system

$$Mu_t(t) + C(u)u(t) + \nu Au(t) = 0. \tag{17}$$

Where M is a block diagonal matrix with elements M_P^e defined as

$$M_{P,ij}^e = \frac{h_e}{2} diag(\rho_i) \tag{18}$$

$\{\rho_i\}_{i=0}^P$ are the GLL quadrature weights. C is the block diagonal matrix with elements $C_P^e(u)$ defined as

$$C_{P,ij}^e(u) = \rho_i u_i D_{P,ij}^{(1)} \tag{19}$$

where $D_{P,ij}$ are the nodal values of the first derivative of the GLL interpolation polynomial. A is the block diagonal matrix with elements A_P^e defined as

$$A_{P,ij}^e = \frac{2}{h_e} \sum_{m=0}^{P} \rho_i D_{P,mi}^{(1)} D_{P,mj}^{(1)} \tag{20}$$

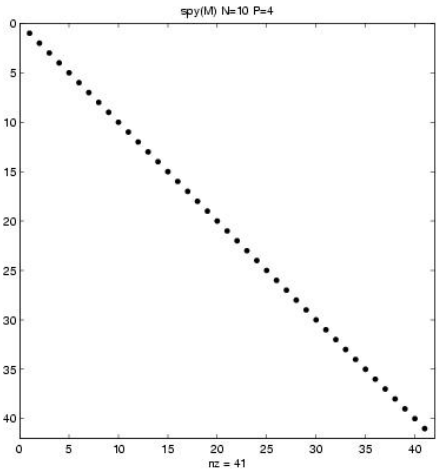

Figure 5: Mass

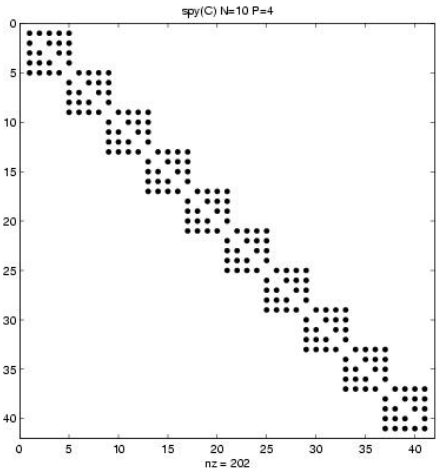

Figure 6: Convection

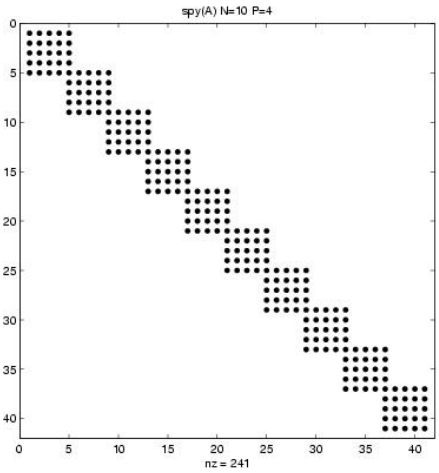

Figure 7: Diffussion

2.5 2D Viscous Burgers' Equation

Going through the same procedure with the 2D viscous Burgers' equation

$$\frac{\partial u}{\partial t} + u \cdot \nabla u = -\nu \Delta u \qquad \text{in } \Omega \in \mathbb{R}^2 \quad t \geq 0 \tag{21}$$

One obtains the same form of the matrix system [2]. The 2D SEM discretization for burgers' equation on element e, using N degree GLL basis functions becomes

$$M^e \dot{u}^e + C^e(u^e)u^e = -\nu A^e u^e \tag{22}$$

Using the Kronecker tensor product, we write the element matrices as

$$M^e = \frac{L_1^e L_2^e}{4}(\hat{M} \otimes \hat{M}) \quad \hat{M} = diag(\rho_i) \qquad i = 0, ...N, \tag{23}$$

$$C^e(u^e)u^e = \frac{L_2^e}{2} U^e \hat{M} \hat{D} U^e \hat{M}^T + \frac{L_1^e}{2} U^e \hat{M} U^e (\hat{M}\hat{D})^T, \tag{24}$$

and

$$A^e u^e = \frac{L_2^e}{L_1^e} \hat{A} U \hat{M}^T + \frac{L_1^e}{L_2^e} \hat{M} U \hat{A}^T. \tag{25}$$

Where L_i^e is the length in the x_i direction for element e, $\hat{D}_{ij}$ is the 1D differentiation matrix on the GLL grid, and

$$\hat{A}_{ij} = \sum_{l=0}^{N} \hat{D}_{li} \rho_l \hat{D} l j \qquad i, j \in 0, 1, ..., N^2 \tag{26}$$

Thus our entire system can be evaluated using only 1D operators. We leave this spatial discretization in this form, because in practice we perform a scatter operation on the global solution u_g to obtain the local contribution u_e on each element. We then solve for u_e^{t+1} on each element, and then perform a gather operation to obtain the global solution u_g. Direct solvers become impractical for $d > 2$, because of the global system size we discuss this in detail in section 3.

Time discretization for convective flows are based on operator splitting schemes [8] [3]. In such schemes the diffusion equation is solved using an implicit method independent of the pure convection problem, which is solved using an explicit scheme that satisfies CFL constraint. Viscous flows, however, can be solved using an implicit scheme coupled with an extrapolation of the convective term. Both schemes lead to $O(\Delta t^3)$ accuracy so not to loose the accuracy gained by this high order spatial discretization. In section 5 we discuss each of these time discretization schemes.

3 Computation Localization

When we began this project, we formed the global 1D system matrices, and then iterated over time. However, as we moved into 2D these system matrices have size $(P+1)^2 N_x N_y$, which, even for coarse meshes are quite large. Thus, we investigated methods for solving local systems.

3.1 Static Condensation

Earlier we gave an illustration of the coupling between elements. By re-ordering the mapping between local and global indices we could effectively decouple the interiors and only solve the coupled system on elemental boundaries. This could all be done with local element matrices of size $(P+1)^2$.
When solving the local system, elemental matrices L^e can be split into blocks containing, boundary and interior contributions. [5]

$$L^e = \begin{bmatrix} L_b^e & L_{bi}^e \\ (L_{bi}^e)^T & L_i^e \end{bmatrix}. \tag{27}$$

Here, L_b^e denotes the parts of L^e formed by boundary-boundary connections, L_{bi}^e denotes the parts of L^e formed by boundary-interior connections, and L_i^e denotes the parts of L^e formed by interior-interior connections. Figure **??** shows how the laplacian operator is decoupled using this ordering. The local to global ordering of nodal values for this method is done my first numbering all boundary nodes, then the boundary-interior nodes, and finally the interior nodes. The advantage of this reordering is that it eliminates communication once the boundary information has been solved at each time step, however, obtaining the solution of the boundary information requires one to compute the Schur compliment

$$[L_b - L_{bi} L_i^{-1} L_{bi}^T \tag{28}$$

This turned us away from using this method, and instead using element-wise operations, in combination with the direct stiffness summation of the local and global solution vectors.

3.2 Element-wise operations

Instead of using static condensation, one can perform operations on local elements and then cleverly add the proper amount to the global solution U without the cost of static condensation. To

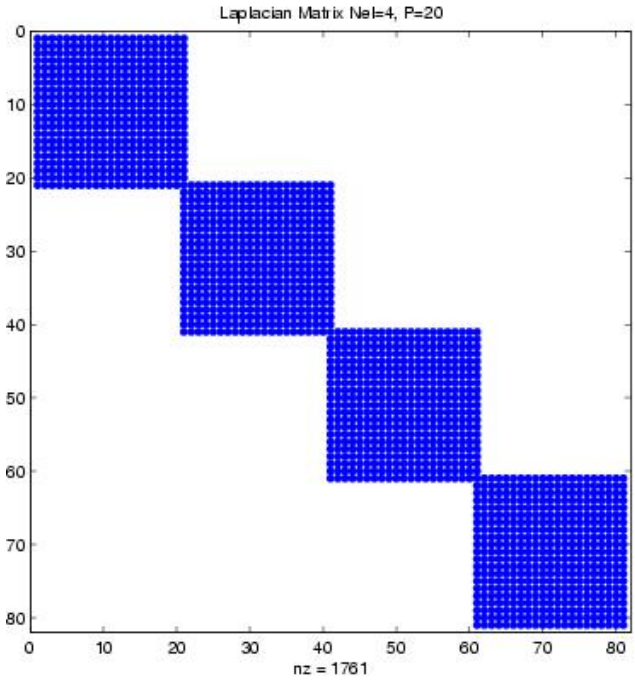

Figure 8: Coupled Diffusion Operator

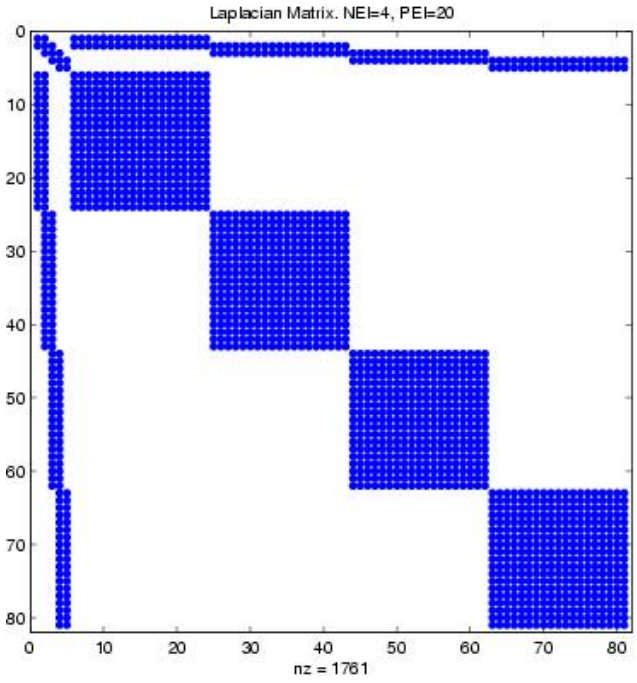

Figure 9: Uncoupled Diffusion Operator

do so, a weighting matrix, W_e, is formed for each element to determine the contribution of the local solution to the global solution. At the beginning of each local solution step, a local element solution vector u_e is obtained from the global solution vector u_g. u_e is then acted on my the local matrix operators defined in the previous sections, in order to obtain u_e at the next time step. Once all the local matrix operations are completed on u_e, it is multiplied by the weighting matrix W_e, and that result is then placed into u_g^{t+1}. After all local matrix operation are calculated, boundary conditions on the boundary nodes of u_g are then enforced. [8]

The weighting matrix W_e is formed by performing direct stiffness summation $\sum'$ on a unit vector e_L of the same length as u_e. This results in a vector w_L which is one where there are no other elements contributing to u_g on $u_e's$ global indices, and r if there are r local elements contributing to the corresponding global index. Thus,

$$W_e = diag(\frac{1}{w_L}). \tag{29}$$

Definition 4 (Direct Stiffness Summation $\sum'$) *A noninvertable local-to-local transformation that sums shared interface variables and then sends them to their original locations leaving interior nodes unchanged.*

Direct stiffness summation is achieved via the mapping between the local and global node ordering. The map corresponding to figure 10, would be implemented as

```
map(1,1:9)=(1, 2, 3, 4, 5, 6, 7, 8, 9)
map(2,1:9)=(7, 8, 9, 10, 11, 12, 13, 14,15)
```

Where the first subscript of map denotes the global index for a particular element. Thus the operation of forming W_e for elements 1 and 2 could be constructed as follows

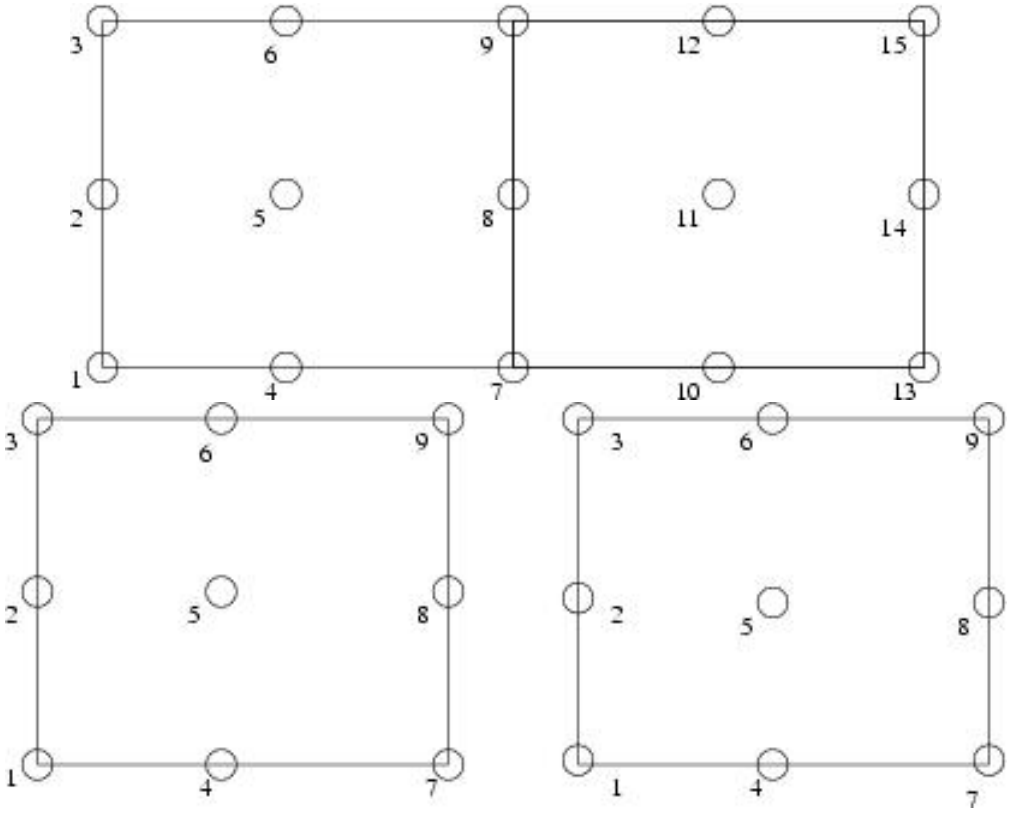

Figure 10: (Top) Global ordering and (Bottom) local ordering

```
allocate(w_L(size(u_g))
w_L=0.0
do i=1,2
   n=El(i)%pdeg+1
   n2=n*n
   allocate(e_L(n2))
   e_L=1.0
   w_L(map(i,1:n2))=w_L(map(i,1:n2))+e_L
   deallocate(e)
end do
...  Inside the time stepping routine for element i
n=El(i)%pdeg+1
n2=n*n
allocate(W(n2,n2))
allocate(r(n2))
W=0.0
r=w_L(vmap2d(i,1:n2))

do j=1,n2
   W(j,j)=1.0/r(j)
end do
```

Advantages for this method include high efficiency for large polynomial degrees, since local cal-

culations can be performed on each element then summed. Also by implementing $\sum'$ using the local to global mapping, the operation is independent of the meshes actual geometry, thus allowing for problems defined on complicated domains.

3.3 Local Data Structures

In either case, these problems scale well to higher dimensions because the global Matrix operators can be written as tensor products.

We construct the local operators for all possible polynomial degree (run time parameter), and store them in an easily accessible data structure. These include, M,A,C, Derivative, Interpolants from $P_n \rightarrow P_{n-1}$ and vice versa. These structures are accessible through a global data module to all subroutines that need access to these operators.

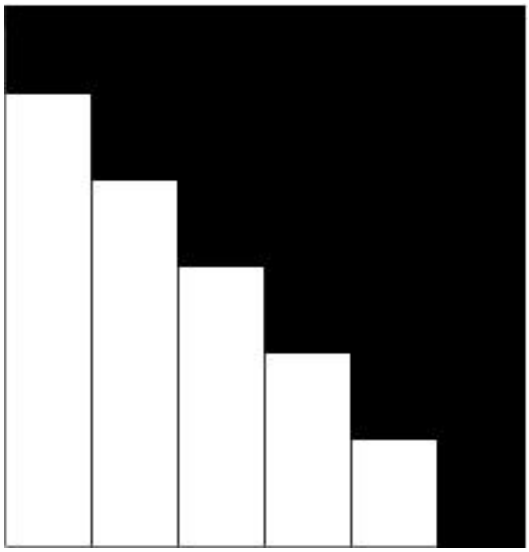

Figure 11: Data structure for hat matrices. Stored as a $(P_{max+1})^2 x P_{max}$ column matrix. The black in a given column denotes non-zero entries for a given P. Here $P_{max} = 6$ thus the 6^{th} column of the matrix which is of size $7^2 x 1$ is full.

4 P-type Refinement

With the local matrices stored for all values of P, and the ability to perform local operations and build the global solution, it is now trivial to compute the derivative of the local solution and perform error analysis with it.

For example, if the slope of our solution at a local element is greater than some user defined value, then we increase the polynomial degree of that element by one. We perform this analysis on each element, interpolate up if needed, and then construct a new local to global mapping with respect to the new local refinements. See figure 12.

Various other error estimators can be contrived depending on the nature of the flow. Higher order local elements, combined with smaller local times steps could be used to achieve proper global accuracy in difficult regions, or perhaps be used to enhance the local accuracy in regions of interest.[6]

We currently have the error estimator above implemented in the 1D portion of the code, time has not permitted us to deal with error estimation in 2D. However, implementing various error

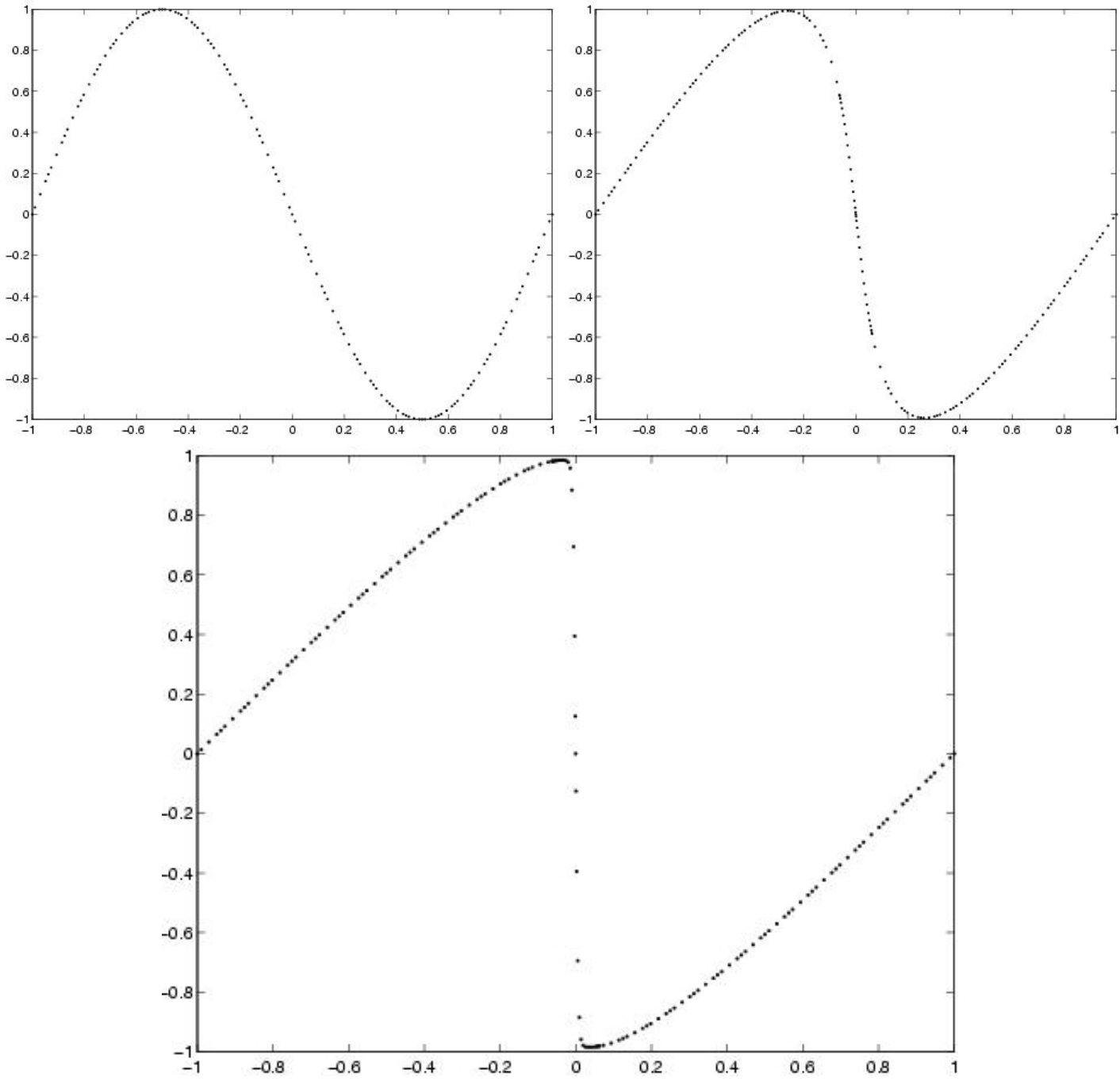

Figure 12: Three solutions to burgers' equation at different times. We illustrate that starting with N=32 and P=4, and refining with $\| \frac{du_e}{dx} \| > 8$, we can resolve the shock at $x = 0$.

estimators to handle various mantle gradients will be a large part of the work stemming from this code, and we have structured the code with with dynamic memory allocation to be able to handle changes in local polynomial degree throughout the 2D code as well.

5 Time Discretization

In order to obtain a stable solution in time, one considers the eigenvalues of the operators acting on u, and makes certain that the time marching scheme is stable in this region. In our system, C and A act on u. [8] [3]

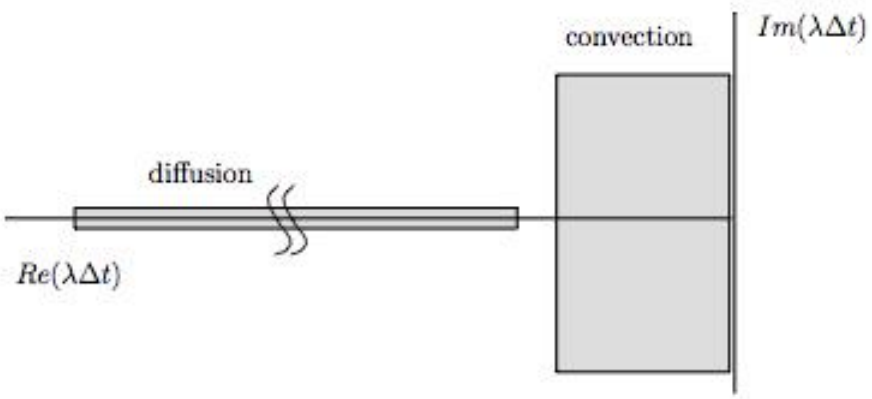

Figure 13: Eigenvalues of the Diffusion and Convection operators [3]

As illustrated in figure 13, spectral methods yield eigenvalues, λ, of the diffusion matrix that are real and negative. The maximum eigenvalue is $O(N^4)$ where N is the maximum polynomial degree. For Spectral Elements, empirical tests show $\lambda \approx O(n_e N^3)$ where n_e is the number of elements. Also illustrated in figure 13, are the eigenvalues, λ, of the convection operator, which have an imaginary part and a negative real part, the largest eigenvalue is $O(N^2)$.

Thus, we want a time discretization which is stable on the negative real axis and the imaginary axis. For viscous dominated flows, one does not need to be as concerned with stability along the imaginary axis included in the time scheme. This is because the affect of C is small, and high order extrapolation of the convection term can be performed at low cost. For convection dominated flows, however, one must choose a scheme which is stable along the imaginary axis to integrate the convection term. We discuss an $O(\Delta t^3)$ time marching scheme for both viscous, and convection dominated in the following sections.

5.1 Crank-Nicholson for 1D Flows

For 1D flows we can afford to use a reduced time step from the convection term to advance the entire solution in time. We choose to use the Crank-Nicholson scheme which is unconditionally stable for our flow (see figure 14).
The time discretization can be written as

$$(\frac{1}{\Delta t}M + .5[C(u^n) + \nu A])u^{n+1} = (\frac{1}{\Delta t}M - .5[C(u^n) + \nu A])u^n \tag{30}$$

In higher dimensions however we must use either extrapolation or a Runge-Kutta solver to deal with the convection term because it is too expensive for the less restrictive diffusion part of the problem to run at the convective time step these issues are addressed in section the following sections.

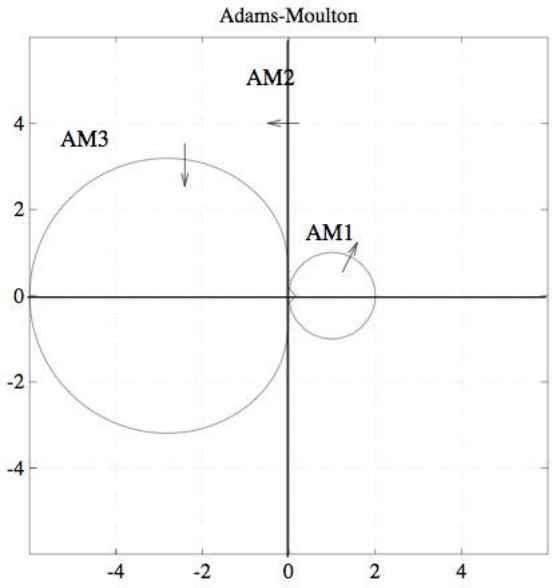

Figure 14: Stability regions for Adams Moulton Schemes. Crank Nicholson is AM2.

5.2 BDF3/EX3 for Viscous Flows

For viscous flows, we achieve 3rd order accuracy, and a stable solution in time using the Backward Difference Formula 3 scheme coupled with a third order extrapolation of the Convection term at each iteration.

$$(\frac{11}{6\Delta t}M + \nu A)v_i^{n+1} = \frac{M}{\Delta t}(3v_i^n - \frac{3}{2}v_i^{n-1} + \frac{1}{3}v_i^{n-2}) - Cv_i^{n+1} \tag{31}$$

where we use 3rd order extrapolation

$$Cv_i^{n+1} = 3Cv_i^n - 3Cv_i^{n-1} + Cv_i^{n-2} + O(\Delta t^3) \tag{32}$$

to obtain $C(v_i^{n+1})$ at each time step. [8]
We show results of this time marching scheme for a viscous an inviscid flow in section 6.2.

5.3 OIFS for Convective Flows

For SEM a harsh condition is placed on Δt in order to satisfy the CFL criteria [8]. For basis functions of degree $N-1$,

$$\Delta t \leq \frac{6.5}{\nu}\frac{\pi^2}{N^4} \tag{33}$$

However, we do not have to integrate our entire system at this time step, since the convection term is the dominant limiting factor [8]. For convection dominate flows, we use an explicit Runge-Kutta 4 scheme who's stability region is given in figure 16 to solve the convection part of the flow $\hat{u}_{n-2}, \hat{u}_{n-1}, \hat{u}_n$. These are then used on the right hand side of the BDF3 scheme where we solve the diffusion system for u^{n+1} . Thus the OIFS method can be written as

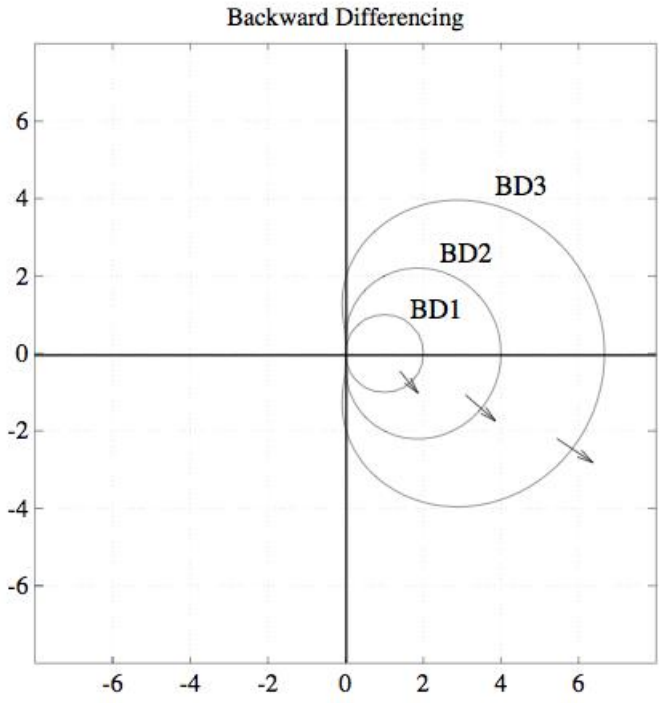

Figure 15: Stability region arrows denote stability outside the corresponding curve for the Backward Difference time marching scheme. [3]

Start with u^{n-2}, u^{n-1}, u^n, solve the IVP

$$\begin{cases} M\frac{d}{ds}\hat{u}_j(s) = -ReC(\hat{u}_j(s))\hat{u}_j(s), & s \in (0, j\gamma\Delta s]\hat{u}_j(t^{n+1-j} = u_j^{n+1-j} \end{cases} \tag{34}$$

with time steps $\Delta s_j = \Delta/\gamma$ where gamma is chosen such that Δs satisfies the CFL condition. Each iteration of the RK4 scheme yields $\hat{u}_1^{n+1}, \hat{u}_2^{n+1}, \hat{u}_3^{n+1}$ respectively.
After $\hat{u}_1^{n+1}, \hat{u}_2^{n+1}, \hat{u}_3^{n+1}$ are obtained, we use the BDF3 scheme to advance the diffusion contributions of the system.

$$(\frac{11}{6\Delta t}M + \nu A)u_i^{n+1} = \frac{M}{\Delta t}(3\hat{u}_1^{n+1} - \frac{3}{2}\hat{u}_2^{n+1} + \frac{1}{3}\hat{u}_3^{n+1}) \tag{35}$$

u^{n-2}, u^{n-1}, and u^n are then updated for the next RK4 solve.

6 Validation/Results

6.1 1D Burgers' Equation

To test our 1D code, we use the test case performed by [8]. Where we start with the initial boundary problem

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \nu\frac{\partial^2}{\partial x^2} \tag{36}$$

$$u(t,-1) = u(t,1) = 0 \tag{37}$$

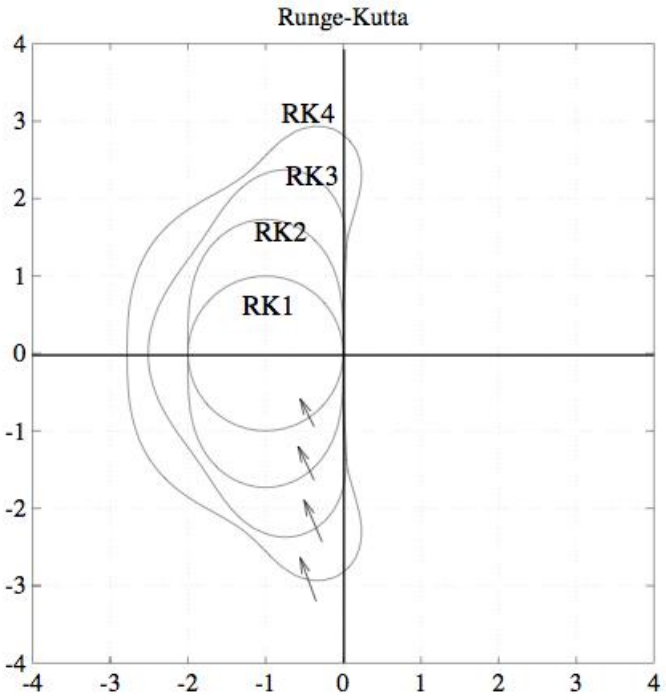

Figure 16: Stability region arrows denote stability inside the corresponding curve for Runge Kutta time marching scheme. [3]

with initial conditions

$$u(0,x) = u^0(x) := -sin(\pi x) \tag{38}$$

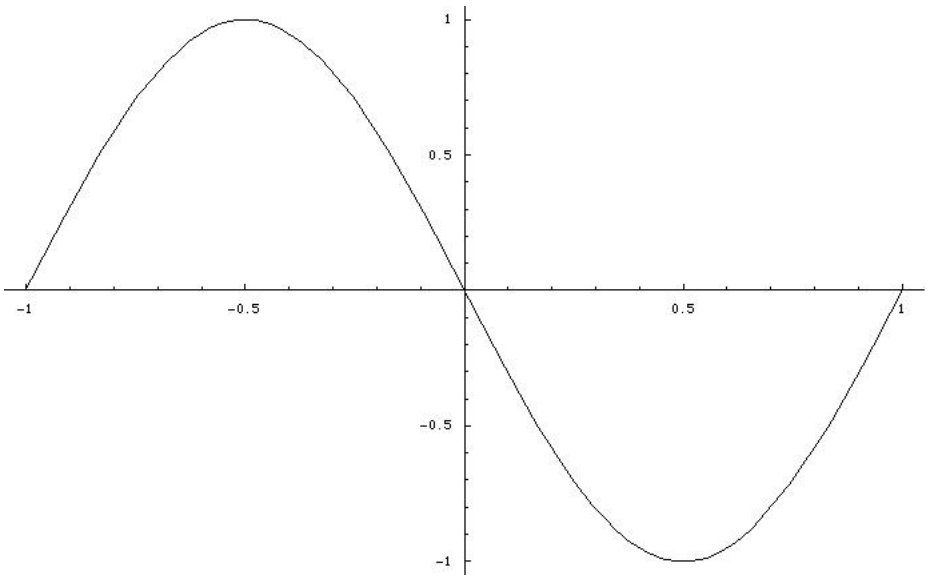

Figure 17: Initial values for 1D test problem

We note that for small ν, a discontinuity will develop at $x = 0$, whereas, for large ν this wave will simply diffuse over time.
The solution to the corresponding system has analytical solution that can be written

$$u(t,x) = 4\pi\nu \frac{\sum_{n=1}^{\infty} n a_n e^{-\nu n^2 \pi^2 t} sin(n\pi x)}{a_0 + 2\sum_{n=1}^{\infty} n a_n e^{-\nu n^2 \pi^2 t} cos(n\pi x)} \tag{39}$$

where a_n are given by

$$a_n = (-1)^n I_n(\frac{1}{2\pi\nu}) \tag{40}$$

with $I_n(z)$ denoting the modified Bessel function of the first kind and order n. We note that we are using this test case, not because the analytical solution is easy to compare with, in fact, we will show in many cases it is not, but rather we it because there are published results, from similar numerical results for us to compare against.

6.1.1 Analytical Comparisons with Diffusive Flows

For the analytical test cases below, we used a 16 element grid with polynomial degree 4. For time marching we used the unconditionally stable Crank-Nicholson (AM2) scheme (see figure 14), with a CFL number of .001. We can afford to run with this restricted time step 1D allowing us to use Crank-Nicholson.
For small values of t and ν equation 39 is difficult to evaluate because for large values of z, $I_n(z)$ behaves asymptotically as $e^z(2\pi z)^{\frac{-1}{2}}$, independent of the value of n [8]. We illustrate this in figures 33 and 29, by using *Mathematica* to evaluate $u(x,t)$ for small ν, and t. We take the 10^{th} partial sums of u, which give differences of machine precision for the 50^{th} partial sums.

In order to validate our code against the the analytical solution, we used large viscosity values, .1, .5, and 1. We attempted to run at smaller viscosities, but, in order for the analytical solution to become evaluatable, the system would have already become quite diffusive. We note in the resulting plots, that for $\nu = .1$, the evaluation of the analytical solution is not stable for small t, thus the discrepancy. Figures 31 through 33 illustrate that the evaluation of the analytical solution becomes more accurate when ν or t are increased.

We note that for $\nu = 1$ there is very little error for small t thus we feel confident that our code is performing as it should in the early stages for $\nu = .1$ and $\nu = .5$. The dots represents values from our code, the solid lines represent the analytical evaluation of the solution.

6.1.2 Numerical Comparisons with Advective Flows

For the advective test case we chose $\nu = 10^{-2}/\pi$ and compared the results against Dr. Paul Fischer's spectral element code, which was used to produce the figures for this problem in [8]. We ran both codes with 32 elements and polynomial degree 8. We can see from figure 18, our codes perform match.

We also considered how our adaptive scheme performs when using a 32 element grid with initial polynomial degree 4, and refining up to degree 16, several solution curves for this run are in figure 12. We compared the slope of the solution at $x = 0$, obtaining a maximum amplitude of 152.2265 at $t = .5100$, with the analytical value being 152.0051 at t=.5105. Thus we are pleased with the results of our 1D SEM scheme.

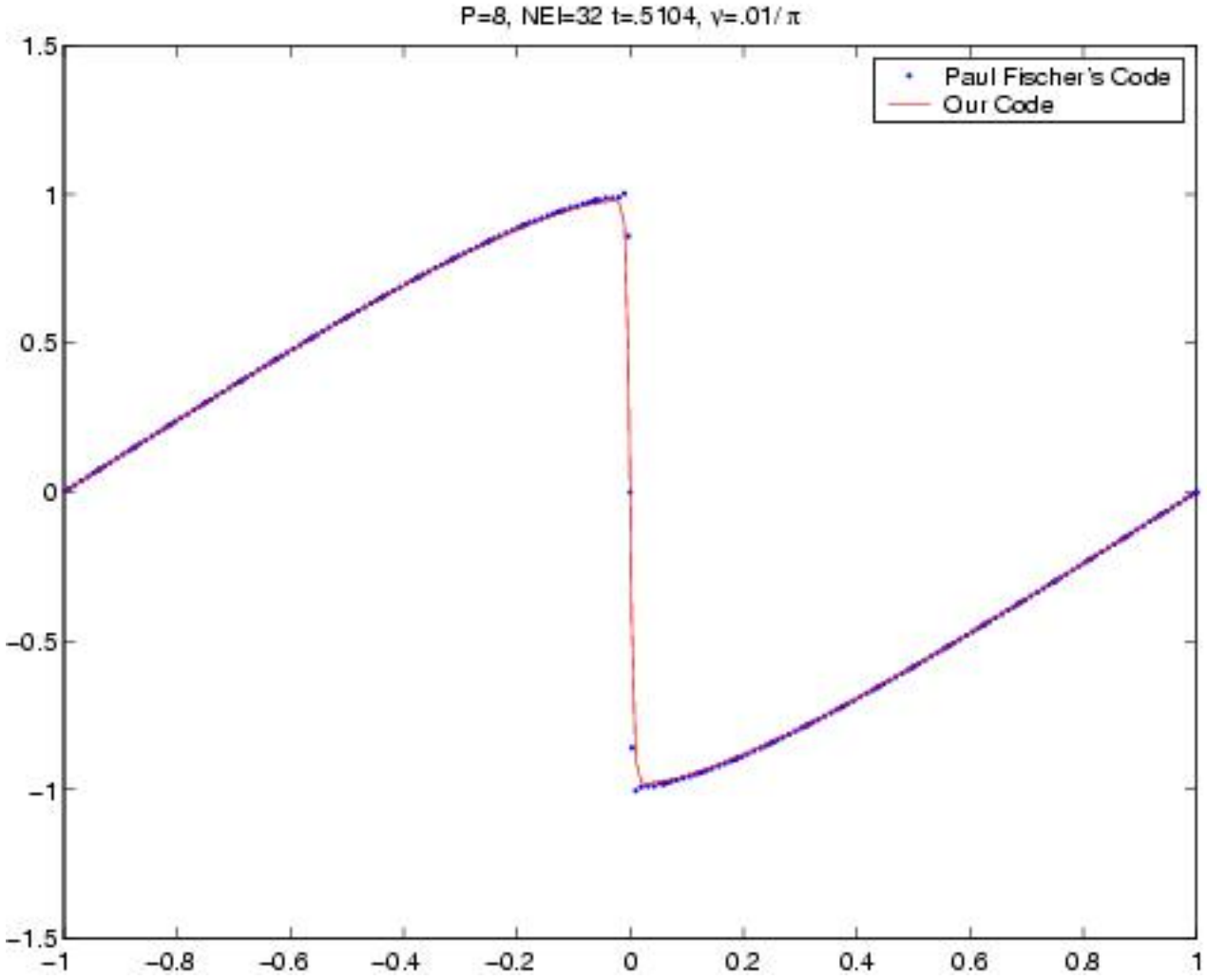

Figure 18: Comparison of Published code (blue) and Our code (red) at time of peak shock. [8]

6.2 2D Burgers' Equation

In order to validate our code we need to have the BDF3/RK4 scheme working properly for advection diffusion flows. We currently have ran, but not tested our BDF3/EX3 code for a diffusive case, we show in this section our validation of the RK4 scheme used for constant advection only flows.

Currently, our RK4 method will only work with flows with constant rate, we are currently working on expanding it to have advection $\vec{c}$ dependent on time, space, and u. Once this is done we will couple it with with BDF3 and test BDF3/RK4 against BDF3/EX3 on our viscous flow case to see that the BDF3/EX3, is performing as it should.

6.2.1 Viscous Burgers' BDF3/EX3

We illustrate the result of our 2D viscous burgers' equation with a diffusive flow. With initial condition

$$u(x,y,0) = .01^{4(x^2+y^2)} \quad \text{on}[-1,1]^2 \tag{41}$$

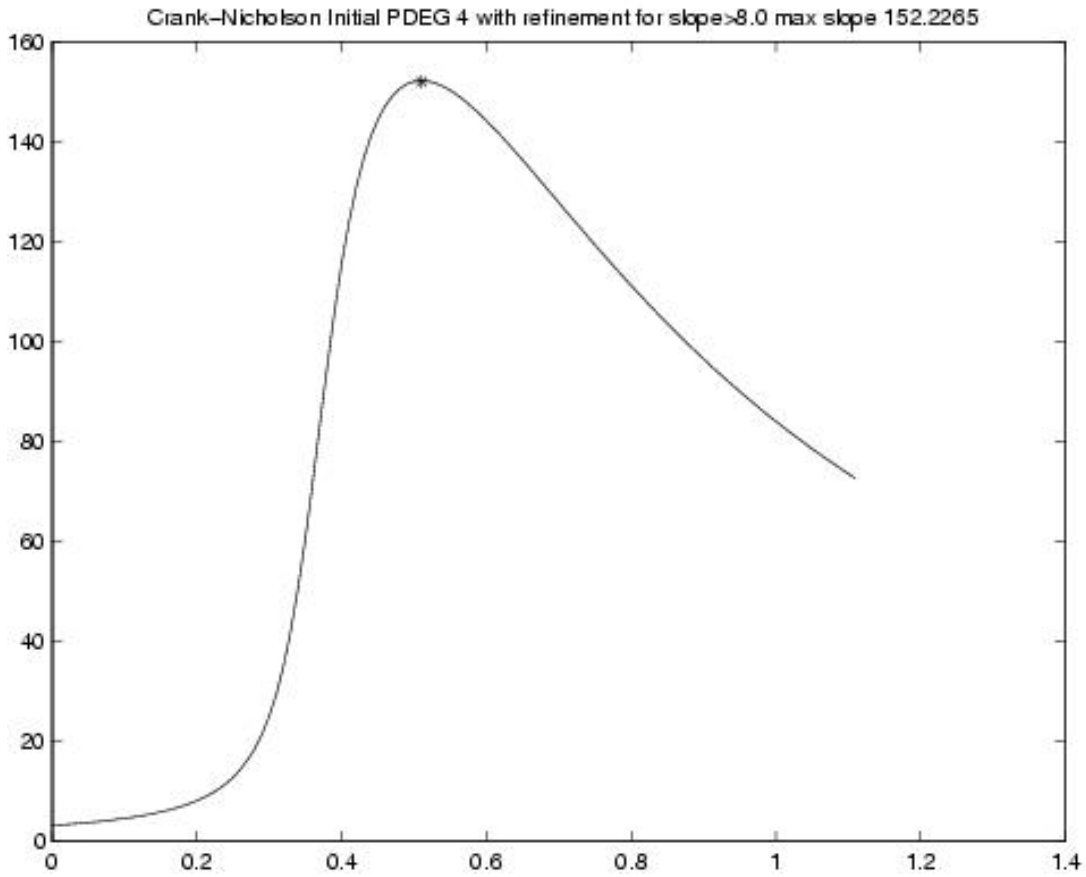

Figure 19: Comparison of results between our code and actual maximum amplitude of slope. We have a value of 152.2265 at t=.5100 analytical value is 152.0051 at t=.5105 (* in the figure). The compares very well with other published numerical methods.[8]

and use periodic boundary conditions. We choose our number of elements in both directions to be 4 and use polynomial degree 8. Our viscosity is chosen to be $\nu = .01$ we take a time step of .0031. Figure 35 shows 6 timesteps for this run.

6.2.2 Pure Advection RK4

For this case, we start with the same initial conditions as before, except our viscosity $\nu = 0$, thus we have a purely advective flow, and our advective term $\vec{c} = (-.05, -.05)$. We see from figure 21 that after one spatial period the error is less than 10^{-3}. Further work needs to be done to show convergence as the polynomial degree, and number of elements are increased. Several plots of the solution for the first period of advection are in figure 36.

7 Future Directions

The purpose of this project was to solve the 1D, and 2D viscous burgers' equation using an adaptive Spectral Element Method so that we can investigate the use of P refinement in the mantle convection equations, as well as other Navier-Stokes type flows. We hope to soon add 2D adaptivity to this code.
We also plan to implement the full unsteady incompressible Navier Stokes equations. This will

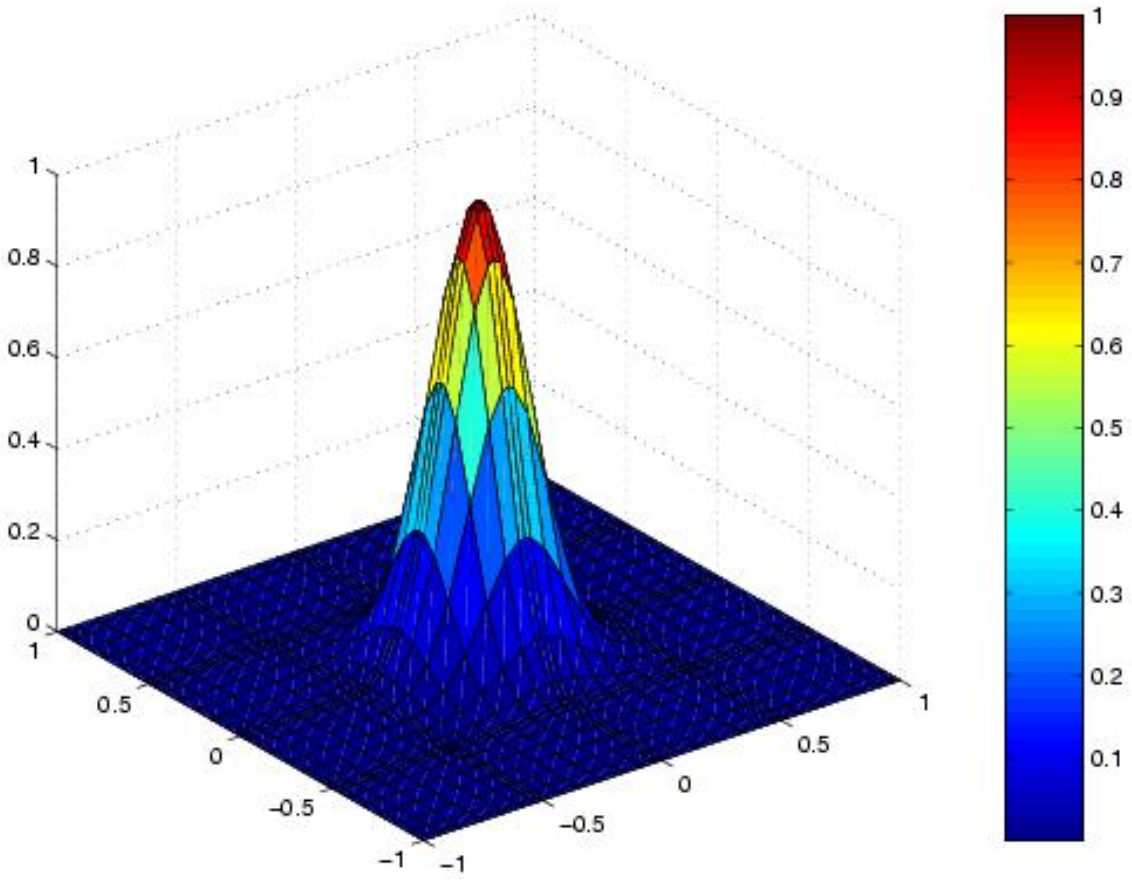

Figure 20: Initial conditions for 2D test cases.

involve adding a pressure term to the Burgers' equation, and an incompressibility constraint, thus far have formed the pressure grid, and interpolation operators that go between the pressure space and the velocity space.

Another feature soon to be implemented is parallelism on at least 2 high performance computing architectures. This will be done as summer work at NASA Goddard under supervision of Dr. Tom Clune and Dr. Anil Deane.

We will also implement a Preconditioned Conjugate Gradient scheme to solve local element systems. PCG is ideal for SEM because both M and A are symmetric positive definite, and diagonally dominant, leading to fast convergence. One set of preconditioners that are of particular interest are the KLESW preconditioners, which have yet to be tested with a Spectral Element Discretization.

8 Conclusions/Summary

We have created a 1D and 2D test bed to begin working on testing adaptive schemes for High Order Spectral Element Methods on advection diffusion equations, for both convection dominated and viscous dominated flows. We have taken into consideration computational localization, to provide for a smooth transition into developing a parallel version of the code.

During the course of this project, we have learned invaluable lessons in writing a large piece of code that are often not as prevalent in smaller projects. Such as forming a large interwoven frame-

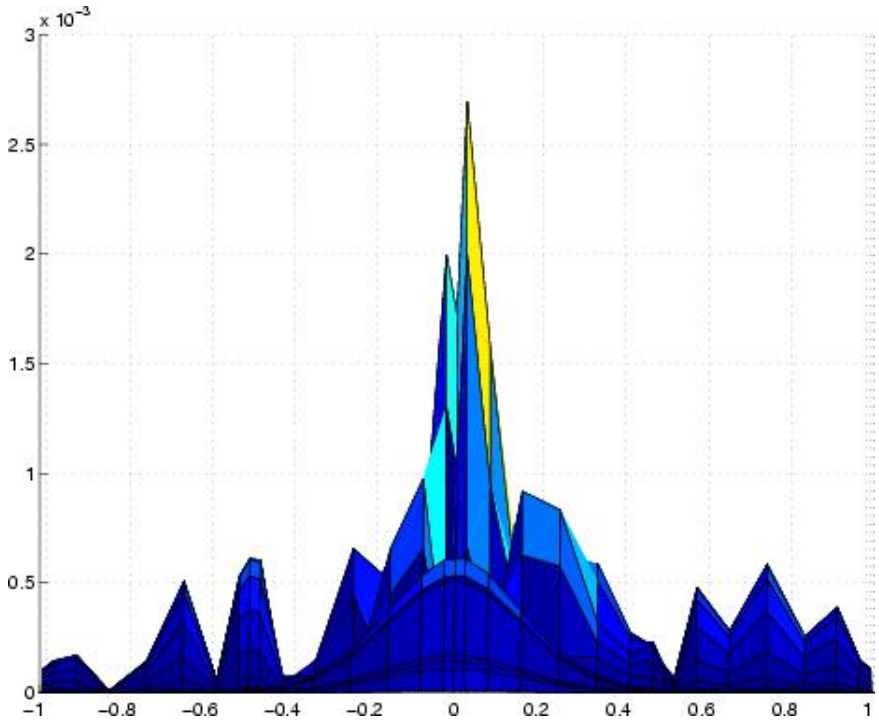

Figure 21: Plot of $|u(x,y,0) - u(x,y,t)| < 3 \times 10^{-3}$ at one period $t = 40$ for 2D advective test case.

work, debugging, validating code, and passing over the numerous conceptual, and technological hurdles that one must pass over in order to produce a working code.

References

[1] Anil Deane. Spectral and spectral-element methods: Lecture notes in high performance computational physics. *NASA Contractor Report 203877*, 1997.

[2] P.F. Fischer. An overlapping schwarz method for spectral element solution of the incompressible navier-stokes equations. *Journal of Computational Physics*, 1997.

[3] P.D. Minev F.N. van de Vosee. Spectral element methods: theory and applications. http://www.mate.tue.nl/people/vosse/docs/vosse96b.pdf.

[4] P. Olson G. Schubert, D. Turcotte. *Mantle Convection in the Earth and Planets*. Cambridge University Press, Cambridge, 2001.

[5] S.J. Sherwin G.E. Karniadakis. *Spectral/hp Element Methods for CFD*. Numerical Mathematics and Scientific Computation. Oxford University Press, Oxford, 1999.

[6] Rainald Löhner. An adaptive finite element scheme for transient problems in cfd. *Computational Methods in Applied Mechanics and Engineering*, 61:323–338, 1987.

[7] P. Aaron Lott. Project website. http://www.lcv.umd.edu/˜palott/research/graduate/663/.

[8] E.H. Mund M.O. Deville, P.F. Fischer. *High-Order Methods for Incompressible Fluid Flows*. Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, Cambridge, 2002.

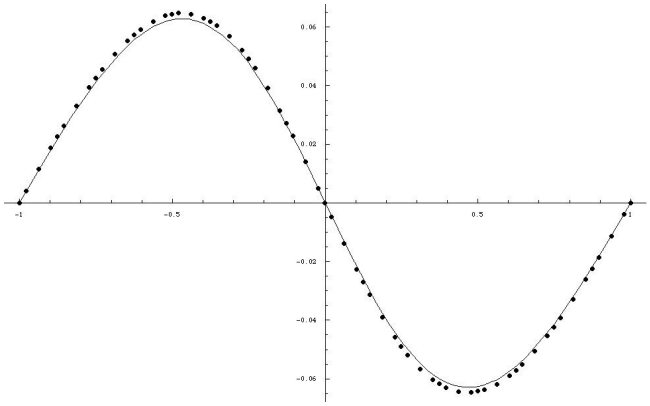

Figure 22: Numerical and Analytical Solutions at $t = 2.55237$, $\nu = .1$

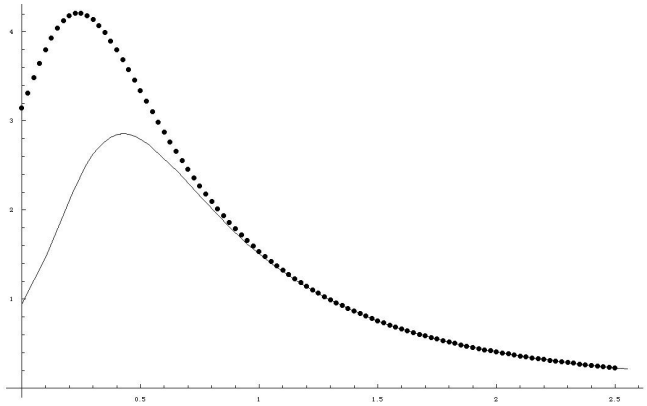

Figure 23: Numerical and Analytical evalutations of $\|\frac{du}{dx}\|$ at $x = 0$, from $t = 0$ to $t = 2.55237$, $\nu = .1$

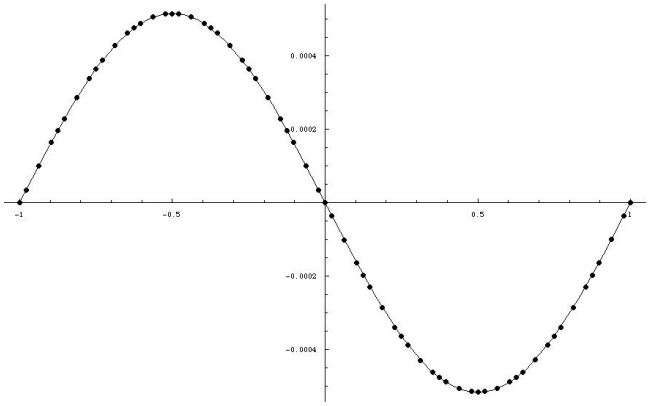

Figure 24: Numerical and Analytical Solutions at $t = 1.531422$, $\nu = .5$

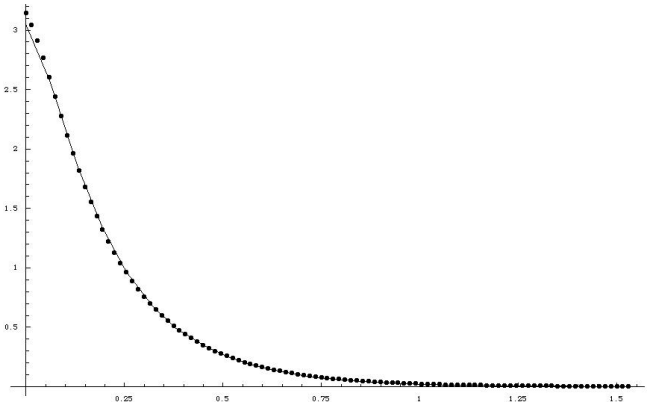

Figure 25: Numerical and Analytical evalutations of $\|\frac{du}{dx}\|$ at $x = 0$, from $t = 0$ to $t = 1.531422$, $\nu = .5$

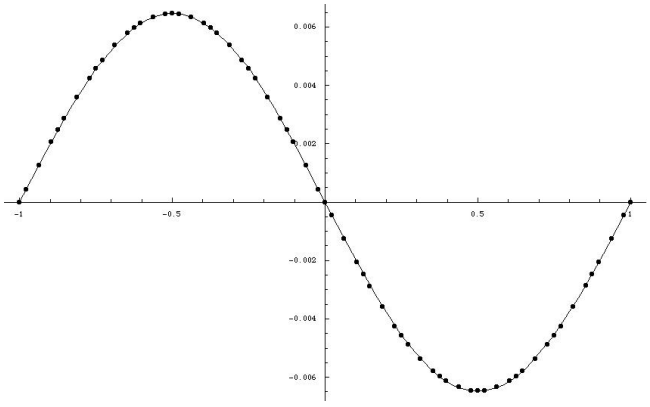

Figure 26: Numerical and Analytical Solution at $t = 0.510474$, $\nu = 1$

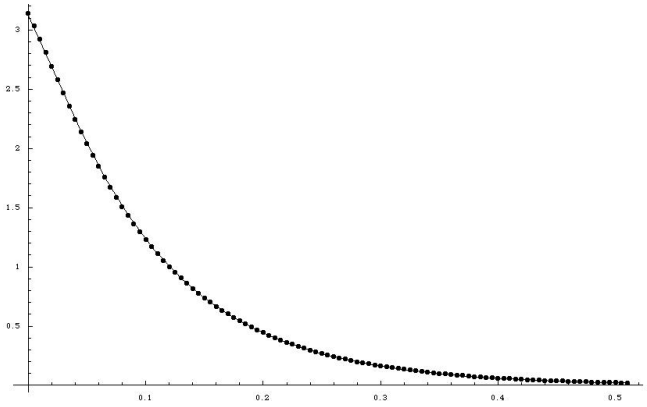

Figure 27: Numerical and Analytical evalutations of $\|\frac{du}{dx}\|$ at $x = 0$, from $t = 0$ to $t = 1.531422$, $\nu = 1$

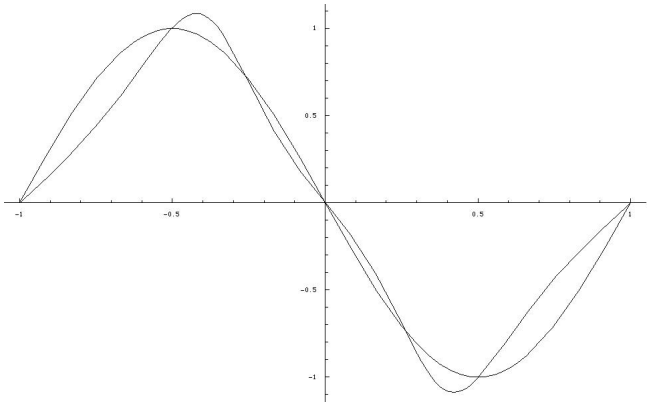

Figure 28: $t = 0$, $\nu = .1$, First order errors

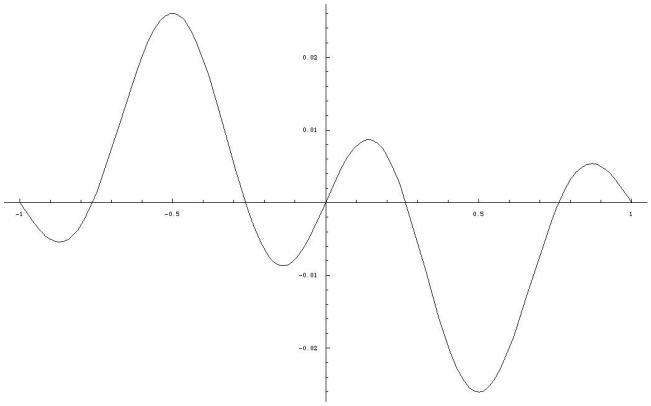

Figure 29: $t = 0$, $\nu = .5$, 3×10^{-2} error

Figure 30: Comparison between the analytical solution at time $t = 0$ and the initial solution

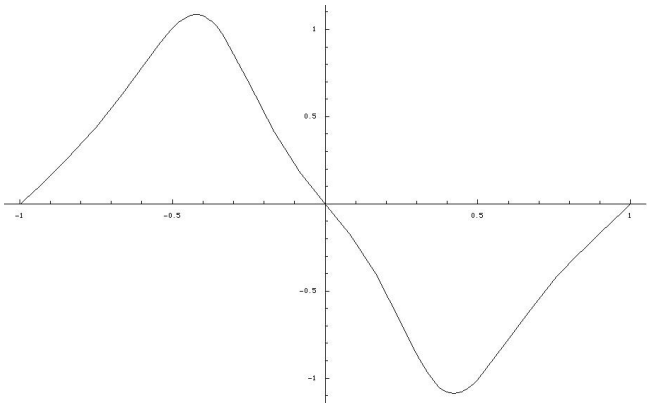

Figure 31: $t = .2$, $\nu = .1$, solution beginning to stabilize

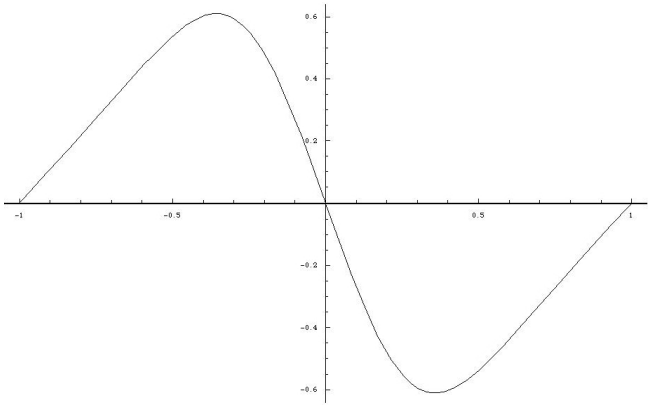

Figure 32: $t = .5$, $\nu = .1$, solution finally taking proper form

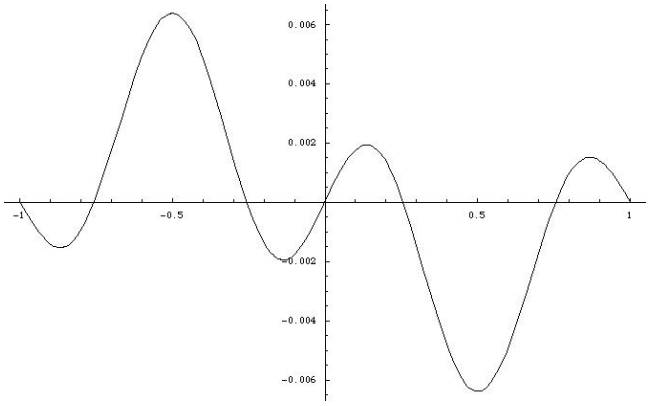

Figure 33: $t = 0$, $\nu = 1$, 6×10^{-3} errors

Figure 34: Illustration of larger t and ν affecting the ability to evaluate the analytical solution

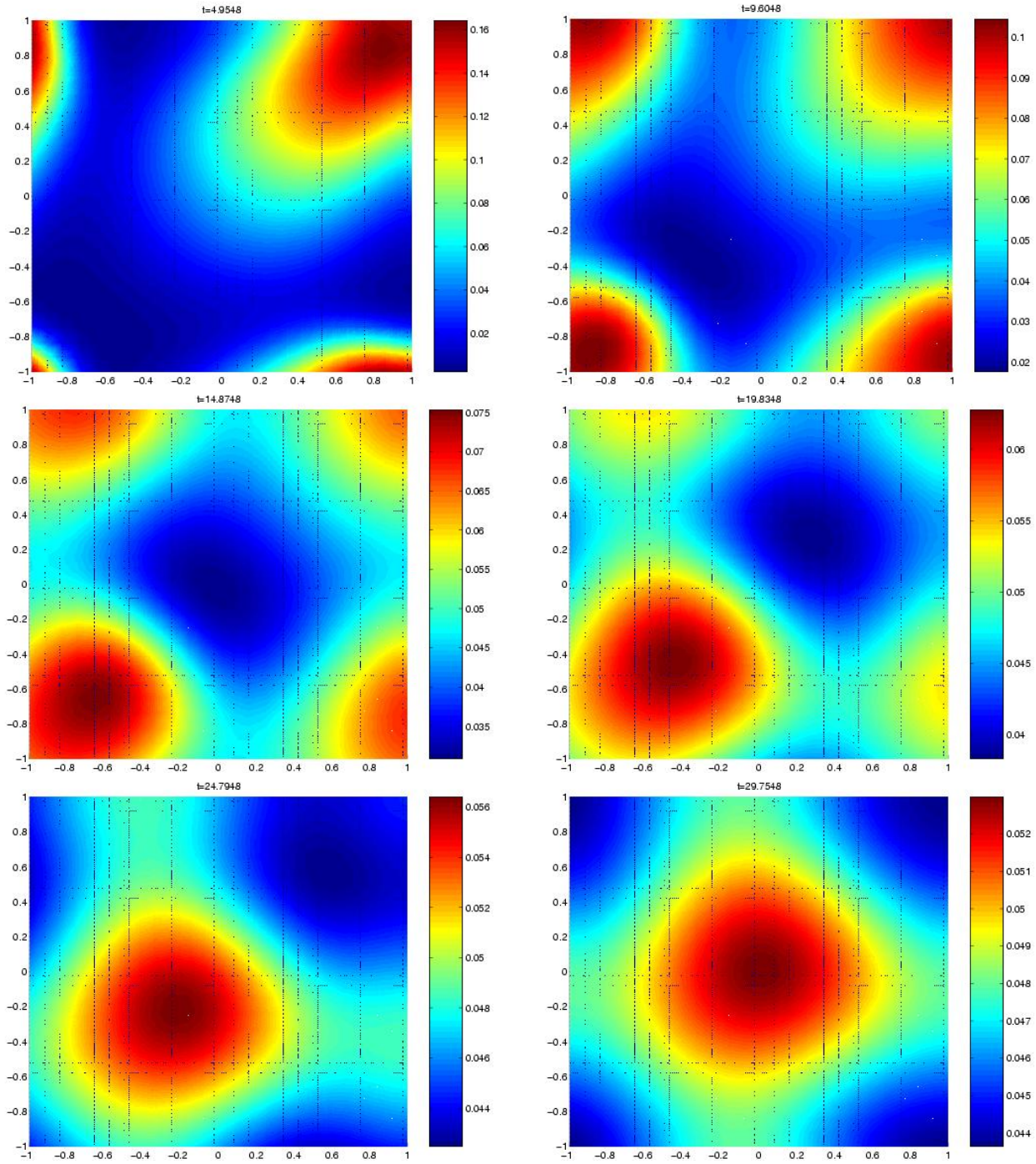

Figure 35: Illustration of Viscous 2D Burgers' Equation test case, at times 4.948, 9.6048, 14.8748,19.8348,24.7948, and 29.7548. Left to right top to bottom. Flow moves to the positive x and y directions since $u(x,y,0) > 0$

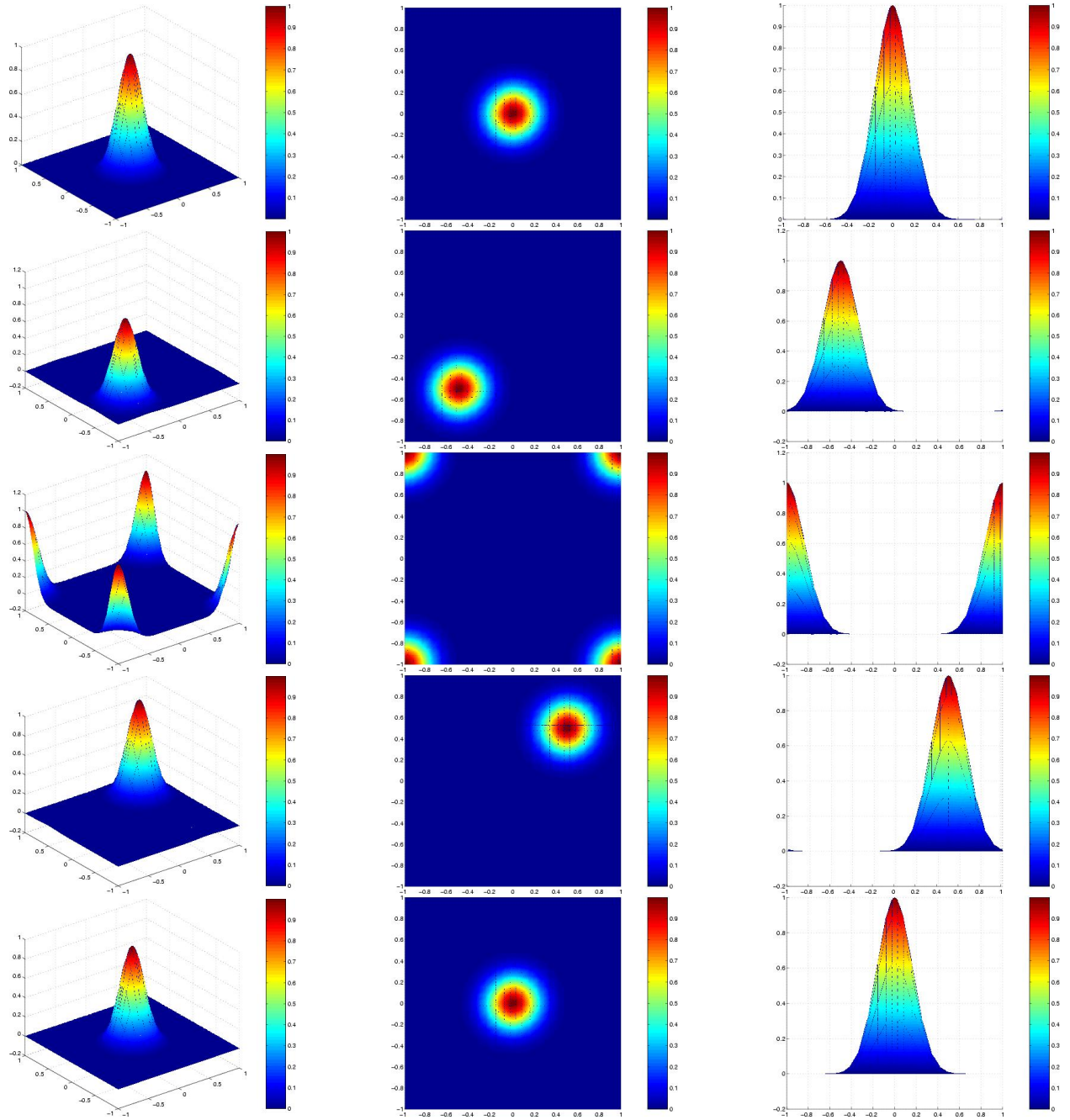

Figure 36: 2D Pure advection test case. Front, top and side views of the flow at $t = 0, 10, 20, 30, 40$ time advancing top to bottom.