Spectral Elements for MHD

Bernhard Hientzsch
Courant Institute of Mathematical Sciences
New York University
mailto:Bernhard.Hientzsch@na-net.ornl.gov
http://www.math.nyu.edu/~hientzsc

November 14, 2004

CEMM Meeting
46th Annual Meeting of the Division of Plasma Physics
American Physical Society
November 15-19, 2004 in Savannah, Georgia

Postdoc under H.R. Strauss at NYU.

Overview of the talk

- My background
- Why spectral elements: the approach
- Incompressible MHD in 2D: primitive and potentials
- Time discretization (FD), space discretization (SEM), complete algorithm
- One example: tilting mode. Setup (IV/BV). Variables, energies, peak currents, and some growth rates.
- Numerical observations
- Extension of algorithms, next steps.

My background

- M.S.: Multigrid, sparse grids, and some hyperbolic conservation laws.
- Ph.D.: Additive domain decomposition and fast solvers for spectral elements for Maxwell's equation.
- Last: Domain decomposition and fast solvers for Maxwell (3D), spectral elements and nonlinear preconditioning for nonlinear elliptic problems. Also background in PETSc, iterative methods for linear and nonlinear systems.
- Now: Spectral/higher order elements for MHD timestepping (prototypes, later M3D?).

Why spectral elements: the approach

- Exponential convergence for (standard) problems with (piecewise) smooth solution.
- Even the solution is only piecewise smooth, alignment of elements, postprocessing, or filtering can possibly restore higher order convergence.
- Faster solvers/application of element matrices and of subassembled system for rectangular array of elements. (Helmholtz: generalized Sylvester equation.)
- Implementation is relatively straightforward, can be expressed as LAPACK calls and some self-written modules (or as script in MATLAB).
- Runs at (relatively) high percentage of peak at modern computer architectures. (Sparse block matrix with dense blocks. Usually degree 15-20 resolves space enough.)

General philosphy

- Try out spectral element type discretizations on more complex problems.
- Rapid prototype the discretizations and methods. Choose simple and straightforward approaches first to get to into the problem as soon as possible.
- Rapid prototype the programming (First use MATLAB, then modular PETSc code and possibly inserting into M3D or other packages).
- Try out methods first on structured grids where one has fast solvers etc. so that quick turnaround, extensive testing possible, and a slightly higher chance to find bugs and understand what is going on.

Incompressible MHD: primitive variables

∇u is the standard gradient $\nabla u := \left(\frac{\partial u}{\partial x}, \frac{\partial u}{\partial y}\right)$

$\nabla^2 u$ is the standard Laplacian $\nabla^2 u := \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$

$\mathbf{B}$: magnetic field.$\mathbf{v}$: velocity. ρ: density, assumed constant. μ: viscosity.

$$\frac{\partial \mathbf{B}}{\partial t} = \mathbf{curl}(\mathbf{v} \times \mathbf{B}) \tag{1}$$

$$\rho \frac{\partial \mathbf{v}}{\partial t} = -\rho \mathbf{v} \cdot \nabla \mathbf{v} + \mathbf{curl}\,\mathbf{B} \times \mathbf{B} + \rho \mu \nabla^2 \mathbf{v} \tag{2}$$

$$\nabla \cdot \mathbf{v} = 0 \tag{3}$$

$$\nabla \cdot \mathbf{B} = 0 \tag{4}$$

Incompressible MHD: potential form in 2D (vorticity-flux)

$[a,b] := \frac{\partial a}{\partial x}\frac{\partial b}{\partial y} - \frac{\partial b}{\partial x}\frac{\partial a}{\partial y}$. $\mathbf{v} = \mathbf{curl}\,\phi$ with ϕ (velocity flux) and $\mathbf{B} = \mathbf{curl}\,\psi$ with ψ (magnetic flux). Ω: vorticity. C: current density. η: resistivity (new here).

$$\frac{\partial \Omega}{\partial t} = [C,\psi] - [\Omega,\phi] + \mu\nabla^2\Omega \tag{5}$$

$$\frac{\partial \psi}{\partial t} = -[\psi,\phi] + \eta\nabla^2\psi \tag{6}$$

$$\nabla^2\phi = \Omega \tag{7}$$

$$C = \nabla^2\psi \tag{8}$$

Incompressible MHD: potential form in 2D (vorticity-current)

$$\begin{aligned}
\frac{\partial \Omega}{\partial t} &= [C, \psi] - [\Omega, \phi] + \mu \nabla^2 \Omega && (9) \\
\frac{\partial C}{\partial t} &= [\phi, C] + 2\left[\frac{\partial \phi}{\partial x}, \frac{\partial \psi}{\partial x}\right] + 2\left[\frac{\partial \phi}{\partial x}, \frac{\partial \psi}{\partial x}\right] + \eta \nabla^2 C && (10) \\
\nabla^2 \phi &= \Omega && (11) \\
\nabla^2 \psi &= C && (12)
\end{aligned}$$

Time discretization (vorticity-flux, semi-implicit)

$$\frac{\Omega^{n+1}-\Omega^{n}}{\Delta t} = [C^n,\psi^n]-[\Omega^n,\phi^n]+\mu\nabla^2\Omega^{n+1} \tag{13}$$

$$\nabla^2\phi^{n+1} = \Omega^{n+1} \tag{14}$$

$$\frac{\psi^{n+1}-\psi^{n}}{\Delta t} = -[\psi^n,\phi^{n+1}]+\eta\nabla^2\psi^{n+1} \tag{15}$$

$$C^{n+1} = \nabla^2\psi^{n+1} \tag{16}$$

PDEs to be solved in each time step (vorticity-flux)

$$\begin{aligned}
\Omega^{n+1} - \mu \Delta t \nabla^2 \Omega^{n+1} &= \Omega^n + \Delta t \left\{ [C^n, \psi^n] - [\Omega^n, \phi^n] \right\} && (17)\\
\nabla^2 \phi^{n+1} &= \Omega^{n+1} && (18)\\
\psi^{n+1} - \eta \Delta t \nabla^2 \psi^{n+1} &= \psi^n - \Delta t [\psi^n, \phi^{n+1}] && (19)\\
C^{n+1} &= \nabla^2 \psi^{n+1} && (20)
\end{aligned}$$

(17) and (19) are Helmholtz solves, for the operator $(I + \alpha \nabla^2)$ with $\alpha = -\mu \Delta t$ and $\alpha = -\eta \Delta t$, respectively. For zero viscosity and zero resistivity, respectively, the Helmholtz solves simplify to direct formulae for the new values taking into account possible boundary conditions. (18) and (20) are standard Laplace solves resp. application of Laplace operator.

Spectral elements

- Approximate u by nodal values on GLL grid $\underline{u}.\underline{u}$ also expansion in interpolatory basis. Interpolation between GLL grids of different degrees, differentiation, and integration of SEM function on the grid can be written as matrices I_n^k, D_x, D_y, M.

- $(u,v) = \int uv$ can be approximated by a mass matrix: $(u,v) \approx \underline{v}^T M \underline{u}$. For one dimension, and integration on same grid, $(u,v) = \int uv \approx \sum_i u_i v_i \rho_i = \underline{v}^T M \underline{u}$ with diagonal M.

- $(\nabla u, \nabla v)$: this is a sum of inner products (component by component) $(\nabla u, \nabla v) = \left(\frac{\partial u}{\partial x}, \frac{\partial v}{\partial x}\right) + \left(\frac{\partial u}{\partial y}, \frac{\partial v}{\partial y}\right)$. Approximate $(\cdot,\cdot)$ as before: $(\nabla u, \nabla v) \approx (D_x \underline{v})^T M (D_x \underline{u}) + (D_y \underline{v})^T M (D_y \underline{u}) = \underline{v}^T D_x^T M D_x \underline{u} + \underline{v}^T D_y^T M D_y \underline{u} =: \underline{v}^T K \underline{u}$ with $K = D_x^T M D_x + D_y^T M D_y$.

PDE 1 in timestep - SEM approximation

$$\begin{aligned}\left(\Omega^{n+1} - \mu\Delta t\nabla^2\Omega^{n+1}, v\right) &= \left(\Omega^n + \Delta t\left\{[C^n,\psi^n] - [\Omega^n,\phi^n]\right\}, v\right) \\ \left(\Omega^{n+1}, v\right) + \mu\Delta t\left(\nabla\Omega^{n+1}, \nabla v\right) &= \left(\Omega^n, v\right) + \Delta t\left([C^n,\psi^n], v\right) - \Delta t\left([\Omega^n,\phi^n], v\right)\end{aligned}$$

Already did types (u, v) and $(\nabla u, \nabla v)$, only need

$$([a,b],v) = \left(\frac{\partial a}{\partial x}\frac{\partial b}{\partial y} - \frac{\partial a}{\partial y}\frac{\partial b}{\partial x}, v\right)$$

Approximate $[a,b]$ pointwise by pointwise multiplication $\circledast$: $([a,b],v) \approx \underline{v}^T M\left((D_x\underline{a}) \circledast (D_y\underline{b}) - (D_y\underline{a}) \circledast (D_x\underline{b})\right) =: \underline{v}^T M P(\underline{a},\underline{b})$ with $P(\underline{a},\underline{b}) = (D_x\underline{a}) \circledast (D_y\underline{b}) - (D_y\underline{a}) \circledast (D_x\underline{b})$. (anti-aliasing, other tricks?)

SEM for time step PDEs

$$\begin{aligned}
\underline{v}^T \left(M\underline{\Omega}^{n+1} + \mu\Delta t K\underline{\Omega}^{n+1}\right) &= \underline{v}^T M \left(\underline{\Omega}^n + \Delta t P(\underline{C}^n, \underline{\psi}^n) - \Delta t P(\underline{\Omega}^n, \underline{\phi}^n)\right) \\
\underline{v}^T K \underline{\phi}^{n+1} &= \underline{v}^T M \underline{\Omega}^{n+1} \\
\underline{v}^T \left(M\underline{\psi}^{n+1} + \eta\Delta t K\underline{\psi}^{n+1}\right) &= \underline{v}^T M \left(\underline{\psi}^n - \Delta t P(\underline{\psi}^n, \underline{\phi}^{n+1})\right) \\
\underline{v}^T M \underline{C}^{n+1} &= \underline{v}^T K \underline{\psi}^{n+1}
\end{aligned}$$

$$\begin{aligned}
M\underline{\Omega}^{n+1} + \mu\Delta t K\underline{\Omega}^{n+1} &= M \left(\underline{\Omega}^n + \Delta t P(\underline{C}^n, \underline{\psi}^n) - \Delta t P(\underline{\Omega}^n, \underline{\phi}^n)\right) \\
K \underline{\phi}^{n+1} &= M \underline{\Omega}^{n+1} \\
M\underline{\psi}^{n+1} + \eta\Delta t K\underline{\psi}^{n+1} &= M \left(\underline{\psi}^n - \Delta t P(\underline{\psi}^n, \underline{\phi}^{n+1})\right) \\
M \underline{C}^{n+1} &= K \underline{\psi}^{n+1}
\end{aligned}$$

Time stepping algorithm

This means that a time-stepping algorithm can be implemented like

$$\begin{aligned}
\underline{\Omega}^{n+1} &= \text{HHSolve}(-\mu\Delta t, \underline{\Omega}^n + \Delta t P(\underline{C}^n, \underline{\psi}^n) - \Delta t P(\underline{\Omega}^n, \underline{\phi}^n)) \\
\underline{\phi}^{n+1} &= \text{LapSolve}(\underline{\Omega}^{n+1}) \\
\underline{\psi}^{n+1} &= \text{HHSolve}(-\eta\Delta t, \underline{\psi}^n - \Delta t P(\underline{\psi}^n, \underline{\phi}^{n+1})) \\
\underline{C}^{n+1} &= \text{ApplyLap}(\underline{\psi}^{n+1})
\end{aligned}$$

Some optimization is possible in the computation of the right hand side by saving terms occuring at several places. Right hand side assembly needs only modules for the two types of terms if modularity is more important.

Tilting mode problem - Setup

Introduce polar coordinates $x = r\cos\theta$, $y = r\sin\theta$ and use seperable form in polar coordinates,

$$\psi(t=0) = \psi_{0,rad}(r)\cos\theta \qquad \Omega(t=0) = \epsilon\Omega_{0,pert}(r)$$

with the radial functions (k being the first positive zero of J_1, $k \approx 3.8317$)

$$\psi_{0,rad}(r) = \begin{cases} \frac{2J_1(kr)}{kJ_0(k)} & \text{for } r \leq 1 \\ \frac{r^2-1}{r} & \text{for } r \geq 1 \end{cases} \qquad \Omega_{0,pert}(r) = 4(r^2-1)\exp(-r^2)$$

The following boundary conditions were used in this problem:

$$\phi = 0 \qquad C = 0 \qquad \frac{\partial\psi}{\partial t} = 0 \qquad \Omega = 0$$

General setting, algorithmic choices

- MATLAB: time-integrating on one element (spectral method) or on a rectangular array: explicit, semi-implicit, ODE suite.
- Sometime scaled version of problem to compute only on $[-1,1]^2$.
- Split rectangle into $M \times M$ elements of degree $N \times N$.
- Use the tensor product structure of Helmholtz and Laplace equations for fast solvers. (Leftovers: Fast diagonalization methods/block diagonalization methods/Hessenberg-Schur methods for Generalized Sylvester equations.)
- Typical: $\mu = 0.005$, $\eta = 0$ or $\eta = 0.005$, $\epsilon = 0.0001$ or $\epsilon = 0.001$, integration up to time $t = 2, 4, 10$.

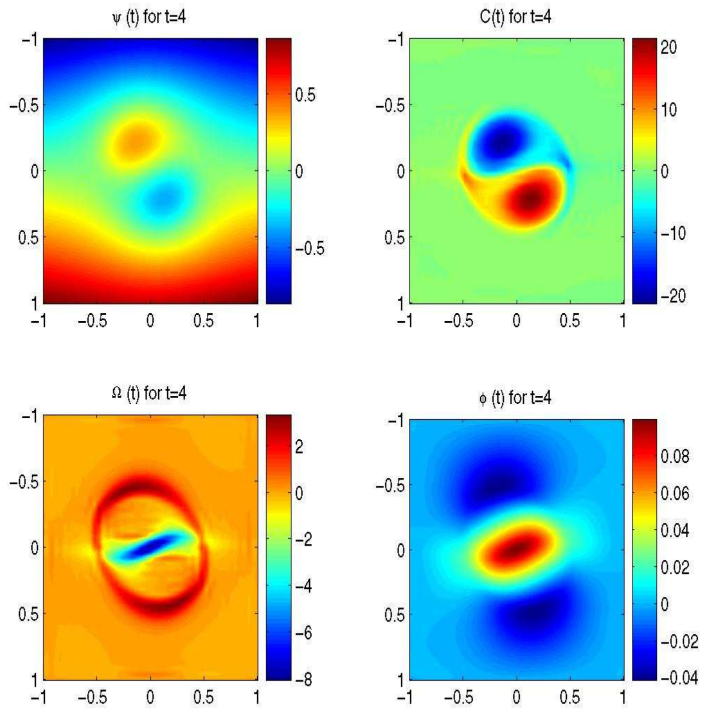
Numerical experiments

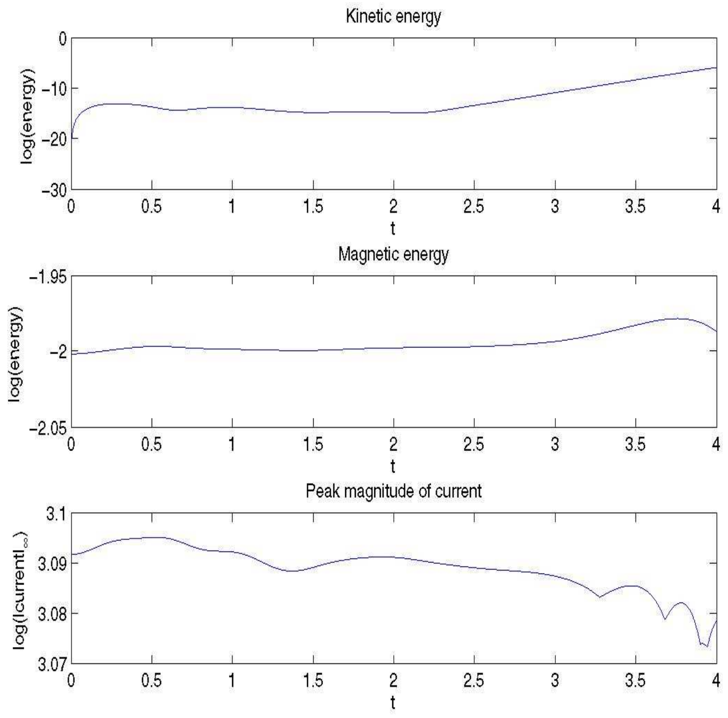
Here only results of semi-implicit vorticity-flux integrations. Variables are interpolated by either the natural spectral element interpolation or a piecewise Hermite interpolation onto a uniform grid of PPE by PPE points in each element. Kinetic energy and magnetic energy are approximated by directly approximating the defining integral as an inner product with the already computed massmatrices. Peak current is just largest magnitude over the computational GLL grid. Growth rates are computed by least-square fitting of eyeball determined region of exponential growth and appropriate scaling. Pictures both by MATLAB (beautiful but slow, especially remotely), gnuplot (fast, not as many options and not as automatic).

We will present some lower order results on many elements and some results for increasing degrees on 10 by 10 and 5 by 5 elements.

50×50 **elements of degree** 2×2

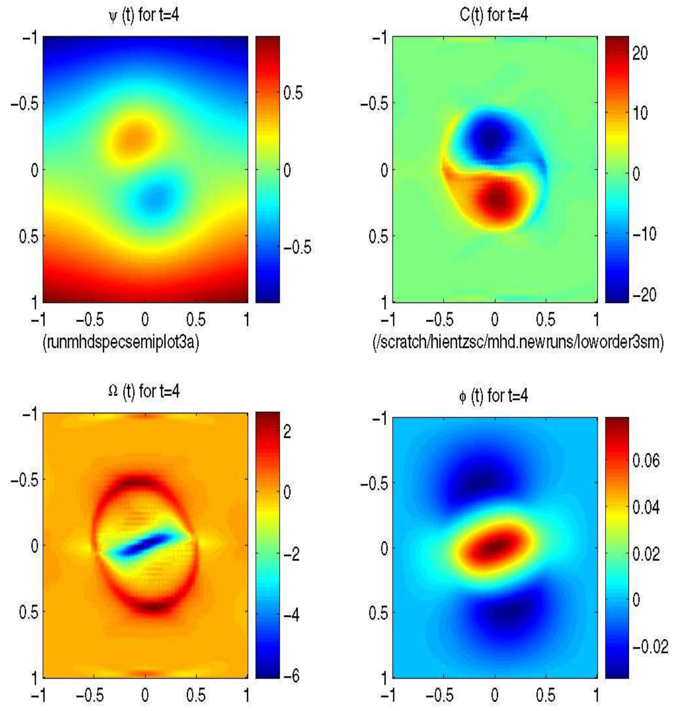
$$PPE = 10,\ \Delta = 0.001,\ t_{final} = 4.0.$$

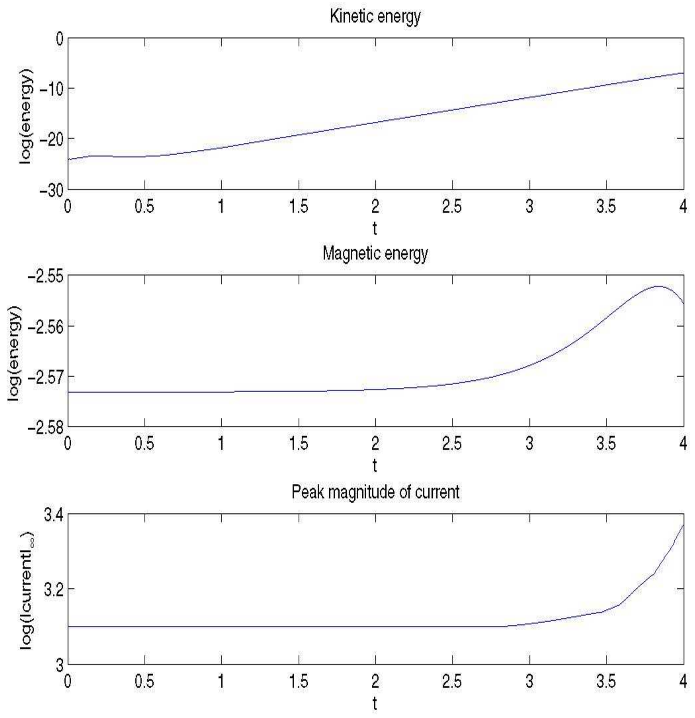




50×50 elements of degree 3×3

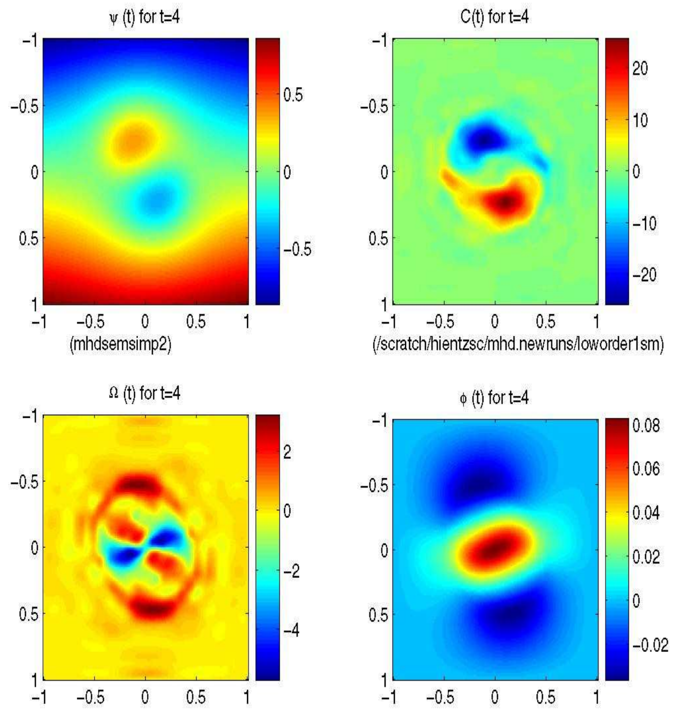
$$PPE = 10,\ \Delta = 0.001,\ t_{final} = 4.0.$$

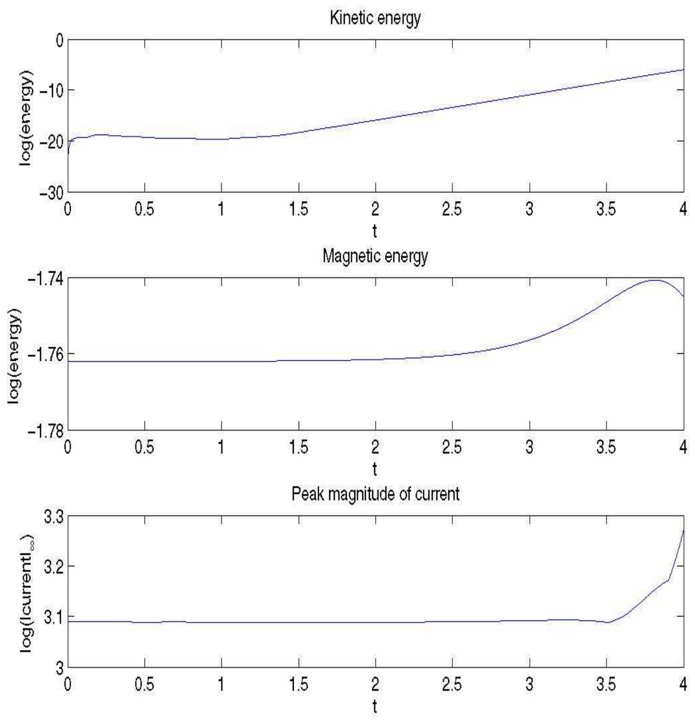




15×15 **elements of degree** 4×4

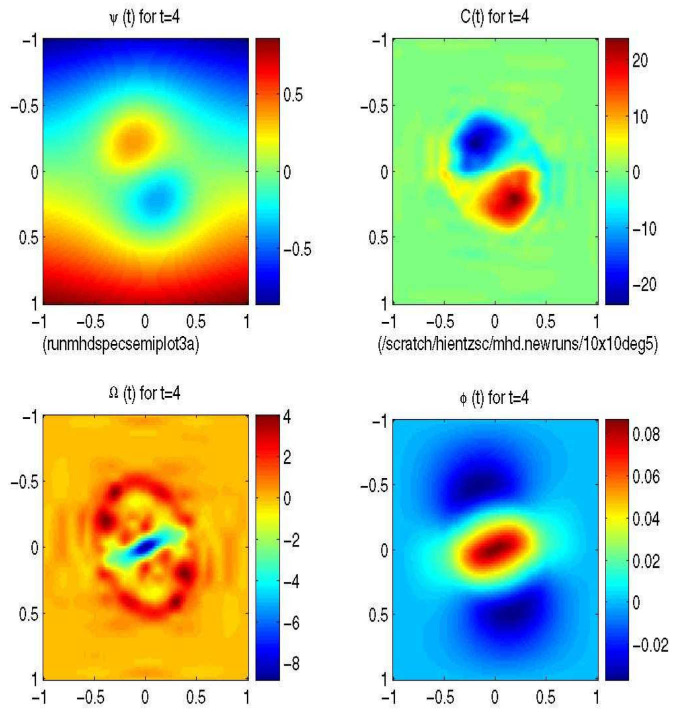
$$PPE = 10,\ \Delta = 0.001,\ t_{final} = 4.0.$$

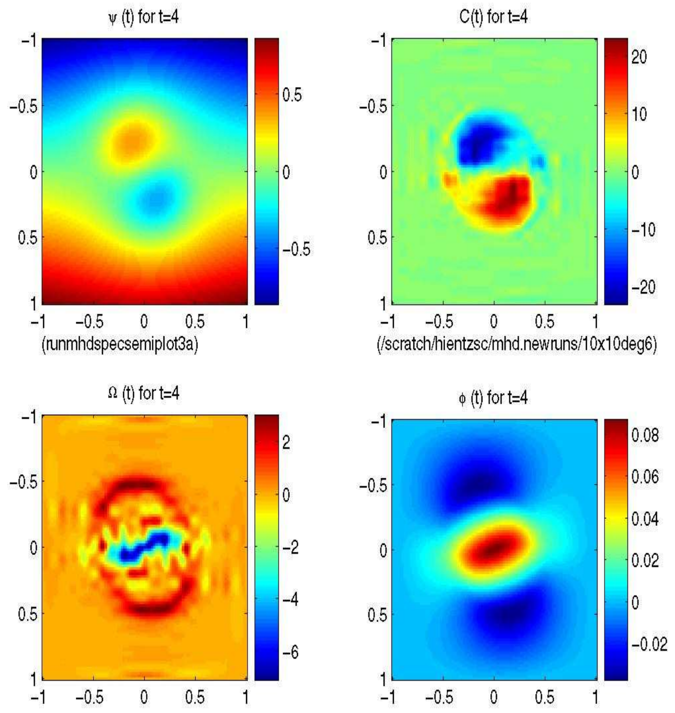




10×10 **elements: degrees** 5×5**,** 6×6

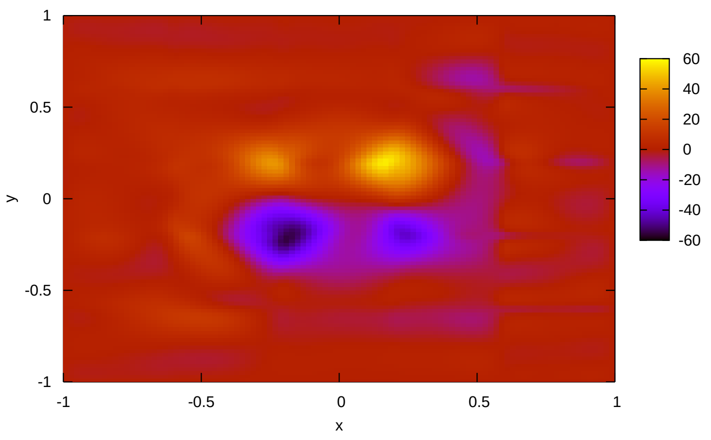
$$PPE = 10,\ \Delta = 0.001,\ t_{final} = 4.0.$$

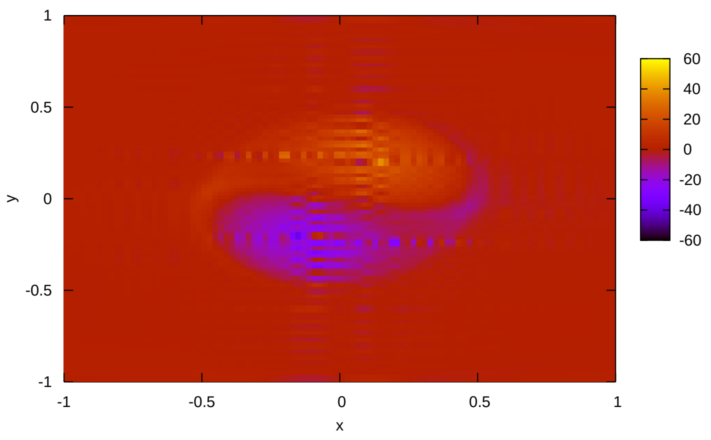




5×5 elements: Final C for degrees 5 and 20

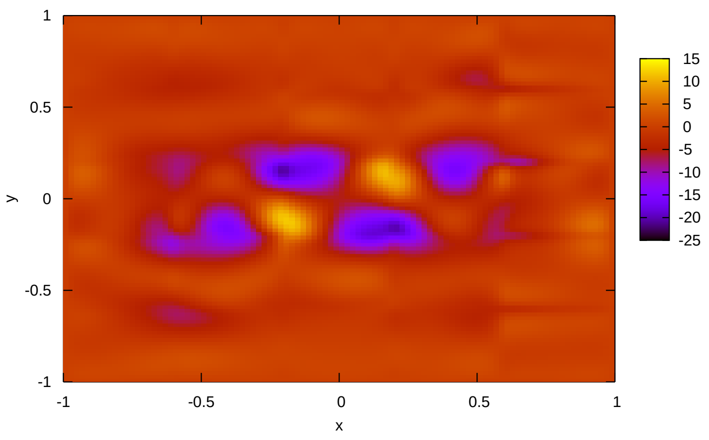
$$PPE = 20,\ \Delta = 0.001,\ t_{final} = 4.0.$$

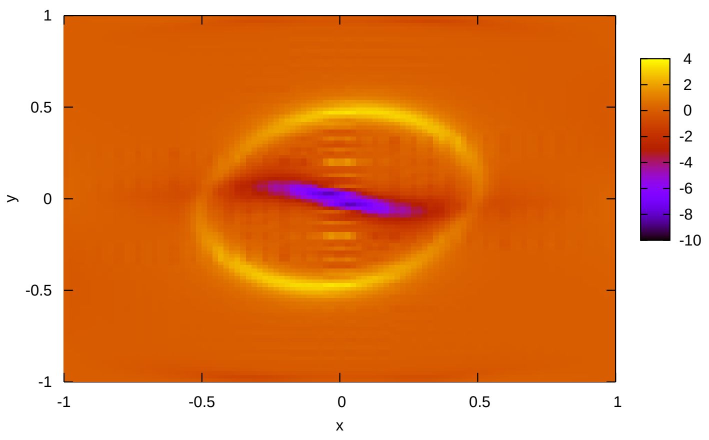




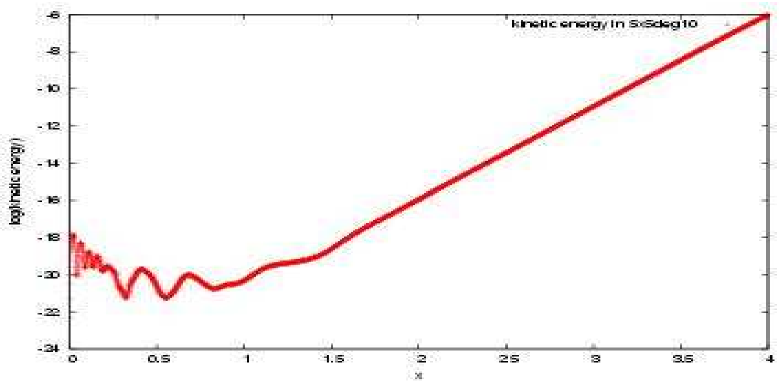
5×5 elements: Final Ω for degrees 5 and 20

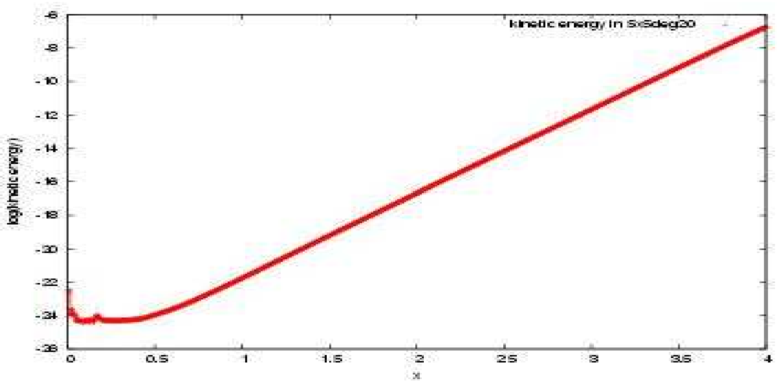
$$PPE = 20,\ \Delta = 0.001,\ t_{final} = 4.0.$$





5×5 elements: Kinetic energies





5×5 elements: Growth rates (+elapsed time)

4000 time steps. MATLAB on Sun Blade workstation.

degree	estimated growth rate	elapsed time
5	1.2065	243.3s
6	1.2543	265.4s
10	1.2398	494.4s
20	1.2417	8843s

Numerical observations

- For some degrees and some kinds of interpolation, see stronger gradients in the vicinity of element interfaces. Stable, relatively local effect. What it that? (Saw it also in results of others.)

- Representing some intermediate variables as continuous or discontinuous (such as the derivatives in the poisson brackets) does not seem to make a difference. What about other treatments of this nonlinear term?

- Explicit scheme seems to be have a stability bound behaving like $1/N^2$, semi-implicit scheme does not seem to be so bad. (Check CFLs!)

- Current advance seems to behave similar, maybe slightly better, but still debugging.

Extension of algorithms

- Fully implicit? Use some leftover tricks for fast implementation of Jacobian for Newton method?
- Mapped elements. (Straight line quadrangles as code, but not in the production version.)
- PETSc version. Or, alternatively, try to convince NERSC to accept parallel MATLAB jobs on seaborg.
- Other problems: tearing mode, ...
- It would be interesting to see how other approaches handle element interfaces. Discontinuous Galerkin, SUPG, C^1 elements.
- ...