COMMUNICATIONS IN COMPUTATIONAL PHYSICS
Vol. **4**, No. 3, pp. 506-518

Commun. Comput. Phys.
September 2008

A Spectral Element Implementation for the M3D Extended MHD Code

H. R. Strauss[1,*], B. Hientzsch[1] and J. Chen[2]

[1] *New York University, 251 Mercer Street, New York, NY 10012, USA.*
[2] *Princeton Plasma Physics Laboratory, Princeton, NJ 08570, USA.*

Received 29 October 2007; Accepted (in revised version) 20 December 2007

Available online 14 April 2008

Abstract. A spectral element library has been developed and integrated with the M3D extended MHD code. The currently used linear triangular finite element implementation and the new high order quadrilateral spectral element implementation are directly compared on equilibrium, linear stability, and nonlinear evolution calculations run on the same problems.

AMS subject classifications: 65M70

PACS: 52.65.Kj

Key words: Spectral elements, extended MHD, M3D.

1 Introduction

Spectral element methods offer several possible advantages for MHD simulations. They are high order discretizations and offer the possibility of exponential decrease of the error with increasing degree. Since spectral element methods can be implemented with discrete operators that are combinations of tensor product matrices and point-wise operations, they can be implemented efficiently at close-to-peak on modern computer architectures. The resulting global stiffness and system matrices are sparse block matrices in which the blocks are dense and of a special structure. Direct solvers can use static condensation and sparse solvers for the much smaller Schur complement system which leads to a fast and efficient solution algorithm.

M3D [1, 2] is a highly modular code for extended MHD problems. Its modularity allows the implementation of several discretizations and the change from one to another for the same problem. Originally, M3D used a spectral discretization in the toroidal and poloidal angles combined with finite differences in the radial direction. While leading to fast solvers and accurate solutions, that discretization did not allow for complicated

*Corresponding author. *Email addresses:* `strauss@cims.nyu.edu` (H. R. Strauss), `bernhard.hientzsch@wachovia.com` (B. Hientzsch), `jchen@pppl.gov` (J. Chen) geometries of the cross sections such as needed to model stellarators and divertor tokamaks. To handle such geometries, a discretization with linear finite elements (FEM) [3] was introduced, which is still the standard version at present. The implementation of a spectral element discretization, described in this presentation, should allow for complex geometries while at the same time recovering the high accuracy that the original version achieved on simple geometries.

Spectral element (SEL) methods [4–6] have been recently introduced in MHD simulations [7,8]. The SEL approach offers several possible benefits. The discretization can be made accurate to high order, with exponential decrease of error as the order is increased. Static condensation provides an efficient solution method for elliptic problems, in which the Schur complement matrix to be solved is orders of magnitude smaller than the original matrix. Curved isoparametric elements allow alignment with relatively complicated boundary shapes encountered in simulation of magnetic fusion experiments. Spectral elements give a diagonal mass matrix, which is advantageous for the partially implicit M3D time stepping scheme.

Against these benefits, there is a concern that high order methods are only good for smooth problems and will not work for highly nonlinear turbulent MHD flows which can occur in magnetic fusion disruptions and Edge Localized Modes (ELMs).

The implementation of M3D presented here solves exactly the same equations, using the same top level code, with either FEM or SEL discretizations. This makes it possible to compare the two methods. This paper presents a direct comparison of MHD equilibrium, linear stability, and nonlinear evolution calculations using the FEM and SEL discretizations in M3D.

A spectral element discretization of M3D has been parallelized using OpenMP for shared memory computers, and MPI/PETSc for distributed memory computers. The OpenMP version is more restricted in problem size, mainly because no domain decomposition in poloidal planes is employed. Each poloidal plane is assigned to a separate processor. In M3D, all the elliptic solves such as Poisson's equation are two dimensional, and do not couple the poloidal planes. Parallelization consists of solving each poloidal plane simultaneously, giving a linear scaling with number of processors. The MPI/PETSc distributed memory spectral element version is being developed. For now, the standard PETSc solvers are used, without taking advantage of static decomposition. We would expect that parallel scaling with number of processors should scale similarly to the scaling of the present distributed memory M3D implementation. The use of static decomposition would be expected to improve performance of the solvers. This will be addressed in the future.

2 Spectral elements

The implemented SEL elements have C^0 continuity, which is the standard approach. There are C^1 SEL methods, but they are more complicated to implement and have certain restrictions as to their applicability.

A Galerkin discretization of the extended MHD equations [2] is used. The solution is expanded in terms of tensor products of one dimensional Lagrange basis functions, and evaluated by collocation on the Gauss-Lobatto-Legendre (GLL) nodal points. The GLL points s_i^N, $i=0,..N$ for degree N are the zeros of derivatives of the Legendre polynomials L_N of degree N,

$$\left(1-\left(s_i^N\right)^2\right)L_N'(s_i^N)=0. \tag{2.1}$$

A nodal representation is used. Each basis function is zero except at a single nodal point. The Lagrange interpolatory basis functions are

$$l_i^N(s)=-\frac{(1-s^2)L_N'(s)}{N(N+1)L_N(s_i^N)(s-s_i^N)}. \tag{2.2}$$

They have the property

$$l_i^N(s_j^N)=\delta_{ij}. \tag{2.3}$$

Hence the product of two basis functions, evaluated by collocation, is zero if the basis functions are different.

In mapped spectral elements, the domain Ω is discretized as the union of K spectral elements E_i, $\Omega=\cup_{i=1}^K E_i$. There is a mapping ϕ^E from the reference element $\hat{E}=[-1,1]^2$ to each element E, $\phi^E:(s,t)\mapsto(x^E(s,t),y^E(s,t))$ with an inverse $\psi^E:(x,y)\mapsto(s^E(x,y),t^E(x,y))$. Such mappings allow the mesh to be aligned with more or less arbitrary boundaries or features in the solutions or right hand sides.

The solution u^E on each element E is sought as a bivariate polynomial in $P^{N,N}(\hat{E})$ (polynomials up to degree N in each variable separately), parameterized by its nodal values on the GLL grid of degree N, and written in an interpolatory tensor basis as

$$\begin{aligned} u^E(x^E(s,t),y^E(s,t)) &= \sum_{i=0}^N\sum_{i=0}^N \tilde{u}_{i,j}^E l_i^N(s)l_j^N(t) \\ &= \sum_{i=0}^N\sum_{i=0}^N \tilde{u}_{i,j}^E l_i^N(s^E(x,y))l_j^N(t^E(x,y)) \end{aligned} \tag{2.4}$$

with $\tilde{u}_{i,j}^E=u(x^E(s_i^N,s_j^N),y^E(s_i^N,s_j^N))$. As shown below, this allows for an efficient high order discretization with implementations running close to peak on modern computer architectures.

Depending on the circumstances it will be convenient to either write the nodal values of the solution as matrix U^E with elements $U_{i,j}^E=\tilde{u}_{i,j}^E$ or as vector u^E with elements $u_{ind(i,j)}^E=\tilde{u}_{i,j}^E$ for some index mapping $ind(i,j)$ mapping from two-dimensional to one-dimensional indices.

In parametric spectral elements, the mapping ϕ^E (and therefore the functions x^E and y^E) are also chosen as polynomials from a space $P^{M,M}$, in isoparametric spectral elements

M is chosen to be equal N:

$$x^E(s,t)=\sum_{i=0}^{N}\sum_{i=0}^{N}\tilde{x}_{i,j}^E l_i^N(s) l_j^N(t), \tag{2.5}$$

$$y^E(s,t)=\sum_{i=0}^{N}\sum_{i=0}^{N}\tilde{y}_{i,j}^E l_i^N(s) l_j^N(t), \tag{2.6}$$

with

$$\tilde{x}_{i,j}^E=x^E(s_i^N,s_j^N), \quad \tilde{y}_{i,j}^E=y^E(s_i^N,s_j^N).$$

As above, these values are either written as matrices X^E and Y^E or as vectors x^E and y^E.

In our implementation, we considered three different choices for the specification of the x^E and y^E. In the first, only the coordinates of the mapped vertices are given, leading to straight-line quadrilateral elements and simplified mappings. In the second, the coordinates of the mapped GLL points on the boundary are given, and the coordinates of the interior points are computed by bilinear blending. In the third, the coordinates of all mapped GLL points are given. The initial computations were done with straight-line elements, currently the bilinear blending approach is used, and the complete mapping case is being implemented.

To ensure C^0 continuity, a point (x,y) belonging to two elements E and F must receive the same function value in both elements, that is $u^E(x,y)$ must be equal to $u^F(x,y)$. This can be ensured by the GLL points for the elements on the boundary (which coincide for adjacent elements) receiving the same value in both elements.

The global spectral element solution is represented by its values on the global GLL grid

$$\Xi_N=\cup_E\left\{\left(\tilde{x}_{i,j}^E,\tilde{y}_{i,j}^E\right)\right\}_{i=0,j=0}^{N}. \tag{2.7}$$

The values on the global GLL grid are either represented by a long vector u, with mappings R_E which return the vector of values within each element, $u^E=R_E u$ (the "flat" vector representation) or by a vector of matrices $U=(U^{E_1},U^{E_2},\dots,U^{E_K})$ (the "distributed" representation). The flat representation allows easier global operations, such as the forming of Schur complements, global solves, and scatter-gather operations; the distributed representation is redundant — values on the interfaces are represented more than once and have to be forced to be the same - but it allows fast implementations of element-wise operations. Mappings between the two representations can be implemented efficiently.

One-dimensional Gaussian quadrature on the GLL points $\{s_i^N\}_{i=0}^N$ leads to integration weights $\{\rho_i^N\}$, and the inner product is approximated by

$$\begin{aligned}(u,v) &= \int_{-1}^{+1} u(t)v(t)dt \approx \sum_{i=0}^{N}\rho_i^N u(s_i^N)v(s_i^N)\\ &= v^T\hat{M}u=v^T\Lambda(\{\rho_i^N\}_{i=0}^N)u=v^T\Lambda(\rho^N)u.\end{aligned} \tag{2.8}$$

$\Lambda(x)$ stands here for a diagonal matrix with diagonal entries x. For two-dimensional mapped elements, the inner product is approximated by

$$(u,v)=\int_\Omega uv=\sum_{i=1}^K\int_{E_i}uv\approx\sum_{i=1}^K v^T R_{E_i}^T M_{E_i} R_{E_i} u,$$

where on each element

$$\begin{aligned}&\int_E u^E(x,y)v^E(x,y)dxdy=\int_{\hat{E}}u^E(s,t)v^E(s,t)|J^E|dsdt\\ &=\sum_{i=0}^N\sum_{j=0}^N u_{ij}^E v_{ij}^E J_{ij}^E \rho_i^N\rho_j^N=v^{E,T}(\Lambda(\rho^N)\otimes\Lambda(\rho^N))\Lambda(J^E)u^E,\end{aligned}\tag{2.9}$$

with

$$J^E=\frac{\partial x}{\partial s}\frac{\partial y}{\partial t}-\frac{\partial x}{\partial t}\frac{\partial y}{\partial s},$$

evaluated on the GLL grid points. As can be seen, the resulting mass matrix is diagonal and trivial to invert, allowing a large speedup in a partially explicit method as in M3D. $C=A\otimes B$ is the tensor product of A and B,

$$C_{ind(i,j),ind(k,l)}=A_{ik}B_{jl}.$$

The matrix form of Cu^E can be computed as AU^EB^T, where U^E is the matrix form of u^E. Also, $\Lambda(A)u^E$ (in matrix form) corresponds to an point-wise multiplication of the entries of A and U^E, denoted $A\circledast U^E$.

Differentiation of univariate polynomials can be written as a matrix D^N, $\frac{\partial}{\partial t}u=D^Nu$. On the reference element $\hat{E}$, differentiation can be written as (I being the identity matrix of appropriate size)

$$\frac{\partial}{\partial s}u=(D^N\otimes I)u,\qquad \frac{\partial}{\partial t}u=(I\otimes D^N)u.$$

To differentiate on the mapped elements, one uses the chain rule to obtain on each element

$$\left(\frac{\partial}{\partial x}u\right)^E=\left(\Lambda\left(\frac{\partial s}{\partial x}\right)(D^N\otimes I)+\Lambda\left(\frac{\partial t}{\partial x}\right)(I\otimes D^N)\right)u,\tag{2.10}$$

$$\left(\frac{\partial}{\partial y}u\right)^E=\left(\Lambda\left(\frac{\partial s}{\partial y}\right)(D^N\otimes I)+\Lambda\left(\frac{\partial t}{\partial y}\right)(I\otimes D^N)\right)u.\tag{2.11}$$

The derivatives appearing in the diagonal matrices, also known as geometry factors, can be computed by solving a 2×2 system resulting from the chain rule:

$$\begin{aligned}\frac{\partial s}{\partial x}&=\frac{\partial y}{\partial t}/J, & \frac{\partial s}{\partial y}&=-\frac{\partial x}{\partial t}/J,\\ \frac{\partial t}{\partial x}&=-\frac{\partial y}{\partial s}/J, & \frac{\partial t}{\partial y}&=\frac{\partial x}{\partial s}/J.\end{aligned}$$

Using the above and the notation $A \oslash B$ for the matrix resulting from the element-wise division of matrices A and B, the Jacobian and the geometry factors on each element can be computed as

$$\begin{aligned} &J^E = D^N X^E \circledast Y^E D^{N,T} - X^E D^{N,T} \circledast D^N Y^E, \\ &\left(\frac{\partial s}{\partial x}\right)^E = Y^E D^{N,T} \oslash J^E, \quad \left(\frac{\partial s}{\partial y}\right)^E = -X^E D^{N,T} \oslash J^E, \\ &\left(\frac{\partial t}{\partial x}\right)^E = -D^N Y^E \oslash J^E, \quad \left(\frac{\partial t}{\partial y}\right)^E = D^N X^E \oslash J^E. \end{aligned} \tag{2.12}$$

The isoparametric element mappings allow one to manipulate the element mappings exactly in the same way one would manipulate the spectral element functions, leading to easier implementations.

With these forms of the mass and differentiation matrices, variational formulations of partial differential operators can easily be derived. For instance, the variational form of the Laplacian can be written on each element as

$$-(\Delta u^E, v^E) = (\nabla u^E, \nabla v^E) = v^{E,T} \tilde{K}^E u^E$$

with

$$\begin{aligned} \tilde{K}_E = &(D^{N,T} \otimes I)\Lambda(w_{11})(D^N \otimes I) + (D^{N,T} \otimes I)\Lambda(w_{12})(I \otimes D^N) \\ &+ (I \otimes D^{N,T})\Lambda(w_{12})(D^N \otimes I) + (I \otimes D^{N,T})\Lambda(w_{22})(I \otimes D^N) \end{aligned}$$

with appropriate geometry factors w_{11}, w_{12}, and w_{22} involving derivatives of the element mappings and the Jacobian.

Some operators in the extended MHD equations lead to computations of inner products of polynomials with degrees that no longer can be integrated exactly on the GLL grid on which the solution is represented. This happens, for instance, for the Poisson bracket $([u,v],w)$. To gain flexibility in the discretization and to allow exact integration for such operators, interpolation to GLL grids of higher degrees has been implemented, involving tensor products of one-dimensional interpolation matrices as well as Gaussian integration on such higher degree grids.

The used nodal representation can easily be transformed into a modal representation which might lend itself to easier spatial filtering or truncation error analysis. In this implementation we use no special filtering, but instead rely on adequate physically based dissipation by viscosity, thermal conductivity, and resistivity.

The SEL and FEM discretizations are employed in the poloidal planes of constant toroidal angle. In the periodic, toroidal direction, the representation of the solution is pseudospectral.

3 Coupling of spectral elements to M3D

M3D is a highly modular code, which makes it possible to change the underlying discretization of the solution. The right hand side of the equations is built up by calling a set of subroutines which implement various operations, such as derivatives in the three coordinate directions, Poisson brackets, inner product of derivatives, and the Laplacian. The left hand side of the equations involves solving a set of two dimensional elliptic operators in poloidal planes, such as Poisson and Helmholtz equations. The high level driver part of the M3D code, which is for the most part Fortran legacy code, does not explicitly contain mesh information. It simply calls functions from the SEL library of solvers, differential operators, and integral operators. The physical variables, as well as many auxiliary variables, are stored in common blocks in the driver code. It is assumed that the mesh is unstructured, and the variables are simply listed in two index arrays. The first index refers to the location in the poloidal plane, and the second refers to the toroidal angle. When an operator is called, the variables in its arguments are passed to functions in the SEL library, which is written in C. The variables must be mapped to temporary C arrays which have a different layout than the "flat" Fortran arrays. The C arrays are organized by elements, and include storage for the nodal values interior to each element, and the boundary values of each element. This arrangement is convenient for the implementation of element-wise operations.

Static condensation solves the matrix equation

$$Ax=b \tag{3.1}$$

by splitting the vector x into element interior and edge values. The vector x_i consists of element interior values, while the vector x_e consists of element edge values.

$$x=\begin{bmatrix} x_i \\ x_e \end{bmatrix}, \tag{3.2}$$

$$A=\begin{bmatrix} A_{ii} & A_{ie} \\ A_{ei} & A_{ee} \end{bmatrix}. \tag{3.3}$$

The interior values can be eliminated in terms of the edge values

$$x_i=-A_{ii}^{-1}A_{ie}x_e+A_{ii}^{-1}b_i. \tag{3.4}$$

The edge values can be determined from the Schur complement problem

$$\hat{A}x_e=b_e-A_{ei}A_{ii}^{-1}b_i, \tag{3.5}$$

where, using (3.4), the system matrix is

$$\hat{A}=A_{ee}-A_{ei}A_{ii}^{-1}A_{ie}. \tag{3.6}$$

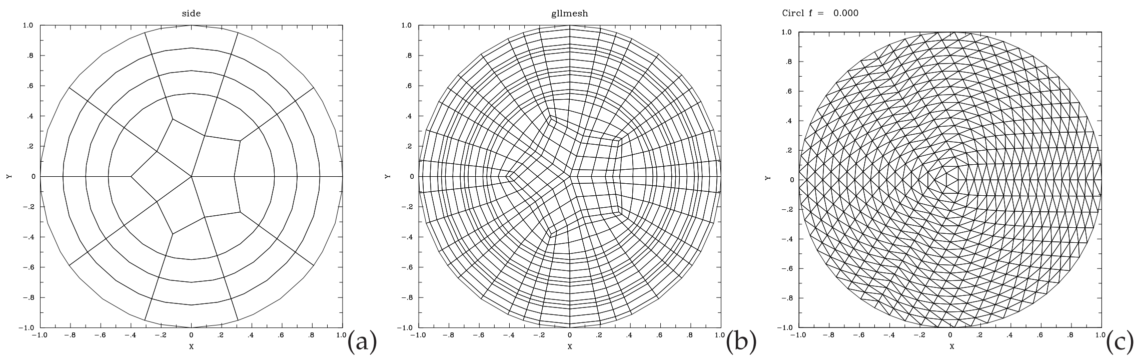

Figure 1: (a) skeleton mesh for spectral elements (b) spectral element mesh (c) Linear finite element mesh.

The Schur complement matrix $\hat{A}$ is generally much smaller than the original matrix A. (3.5) is solved with a direct sparse solver. A general interface for such solvers has been implemented, allowing the use of a variety of packages. The package LDL has been used for most calculations [9]. The inversion of the local element interior matrices and the local solves (using (3.4)) are implemented with LAPACK. Highly optimized libraries providing those LAPACK and BLAS functions are linked for optimal performance. The exact choice of libraries depends on the particular machine.

Mesh generation is initiated in the driver code. A skeleton mesh is generated, which consists of the GLL points lying on the curved quadrilateral element boundaries. An example skeleton mesh is shown in Fig. 1(a). This information is passed to the SEL library, which generates the full mesh, which consists of the GLL points on lines connecting the boundary points. The interior points lie on curves which are linearly blended from the element boundary points. An example, using fourth order elements, is shown in Fig. 1(b). It has 45 elements and 741 meshpoints. For comparison, a piecewise linear finite element mesh with a comparable number of meshpoints is shown in Fig. 1(c). It has 1081 elements and 570 meshpoints.

4 Equilibrium, linear stability, and nonlinear comparison of spectral element and FEM implementations

The same M3D driver code can be compiled with either a linear finite element (FEM) library, or the new spectral element (SEL) library. This permits direct comparison of equilibrium, linear stability, and nonlinear evolution calculations using linear triangular finite elements and high order quadrilateral spectral elements. The equilibria were produced by choosing the same non-equilibrium initial state for both versions and relaxing to a two dimensional equilibrium. A simple case was chosen which was used to benchmark the FEM version to the initial spectral M3D. The equilibria are in excellent agreement for sufficiently high resolution. The SEL runs used the skeleton mesh shown in Fig. 1, with

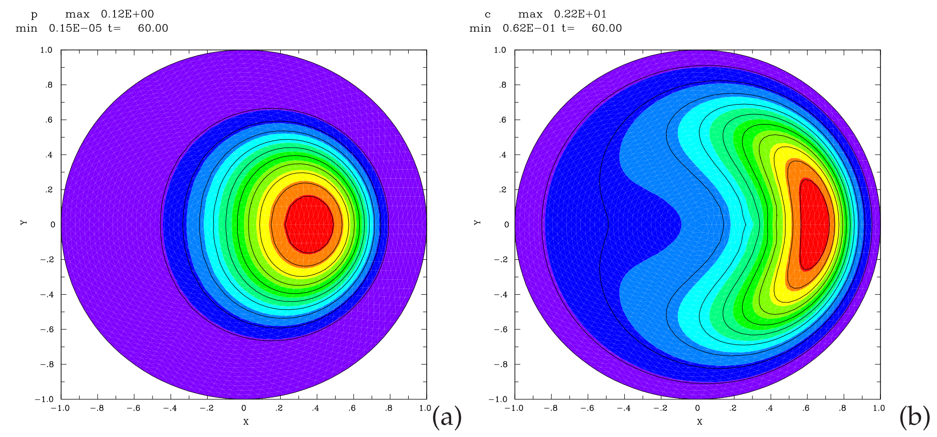

Figure 2: Equilibrium (a) pressure (b) toroidal current density calculated with linear finite elements.

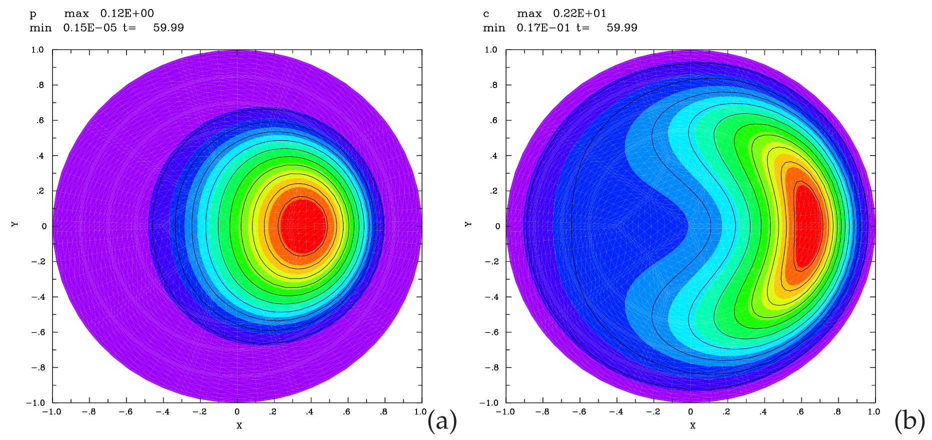

Figure 3: Equilibrium (a) pressure (b) toroidal current density calculated with 8th order spectral elements.

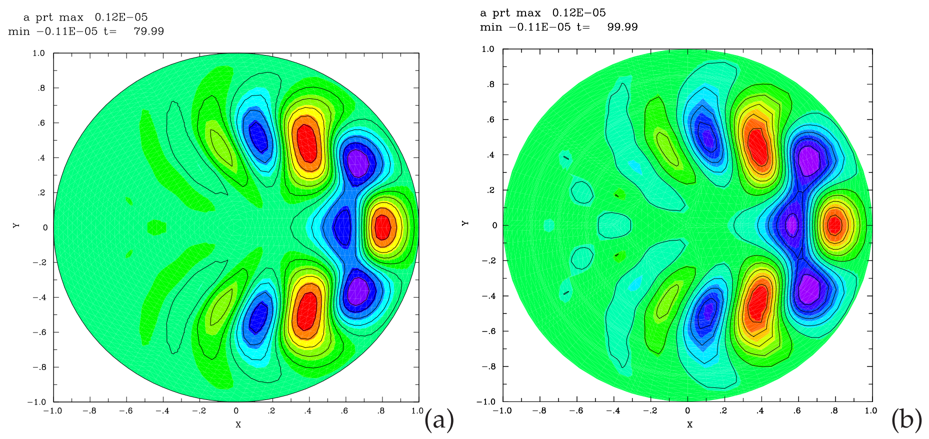

Figure 4: perturbed magnetic flux function ψ (a) calculated with linear finite elements (b) calculated with 8th order spectral elements.

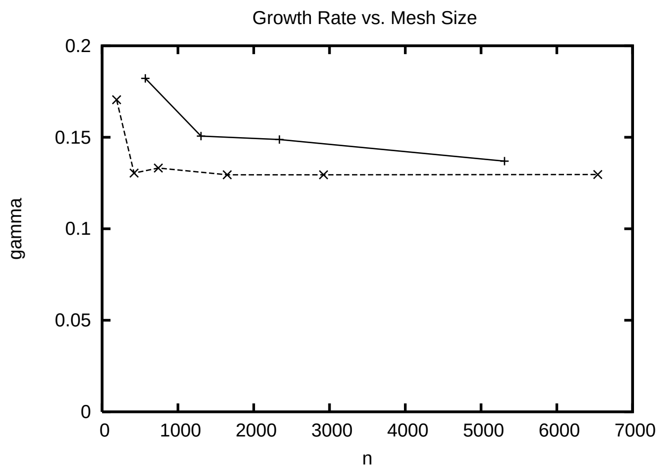

Figure 5: Growth rate as a function of number of mesh points. Solid curve: Finite elements; Dotted curve, spectral elements. Note the expected improved convergence of the growth rate with spectral elements. The number of mesh points in the SEL mesh was changed by varying the polynomial order from 2 to 12.

2nd to 12th order polynomials. The FEM runs used several different mesh sizes. Fig. 2(a) shows the equilibrium pressure, and Fig. 2(b) shows the toroidal current density, which is much more sensitive to the discretization, for a FEM run with 1050 meshpoints. Fig. 3(a) shows the pressure, and Fig. 3(b) shows the toroidal current density, for a SEL run with 8th order polynomials. All the equilibrium, stability, and nonlinear simulations used the same parameters: a fixed time step $dt = 0.01\tau_A$, where $\tau_A = R/v_A$ is the toroidal Alfvén transit time, R is the major radius, $R = 3a$, with a being the minor radius, and v_A is the Alfvén speed. The normalized resistivity η, the perpendicular viscosity μ, and the perpendicular thermal conductivity κ are all $10^{-4} a v_A$.

These equilibria were perturbed with toroidal mode number $n = 3$ perturbations and were advanced in time until the solution was dominated by unstable eigenfunctions. The eigenfunctions show some small differences in detail. This can be seen in Fig. 4, where (a) is the FEM and (b) is the SEL solution. The figures show the perturbed poloidal magnetic flux function ψ. The convergence of the growth rate with the number of mesh points in shown in Fig. 5. The solid curve shows the growth rate as a function of number of mesh points for the FEM, and the dashed curve shows the same for the SEL. The SEL calculations all used the skeleton mesh of Fig. 1(a), but the polynomial order was varied from 2 to 12. Except for order 2, the growth rate does not seem to vary very much with order. This is an example of the improved convergence expected of SEL methods.

The equilibria were perturbed with linear eigenmodes and allowed to evolve nonlinearly until saturation of the instability. The nonlinear pressure is shown in Fig. 6(a), calculated with FEM. The initial pressure was initially aligned with flux surfaces shown in Fig. 2(a), and develops into a highly nonlinear "crab" shape. The toroidal current, shown in Fig. 6(b), is highly fragmented compared to its initial state in Fig. 2(b). Some of the structure of the linear eigenmode Fig. 4 is visible in the nonlinear current density.

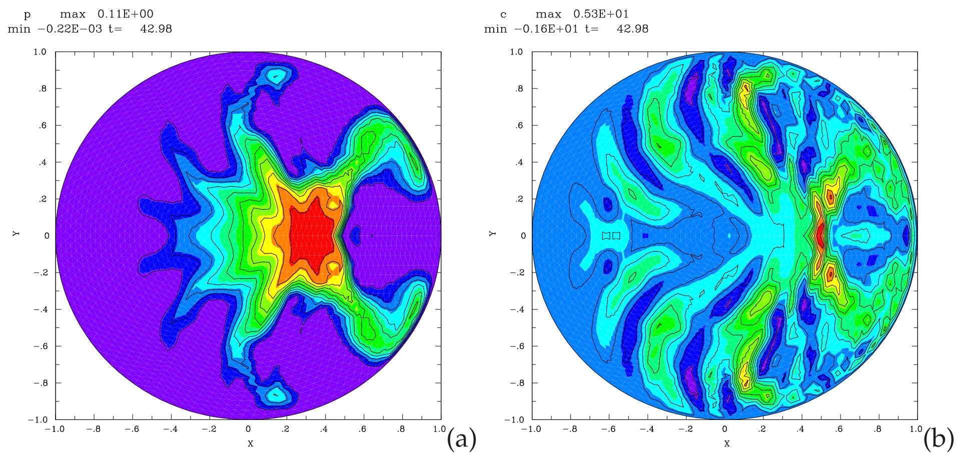

Figure 6: (a) nonlinear pressure (b) nonlinear current density calculated with linear finite elements.

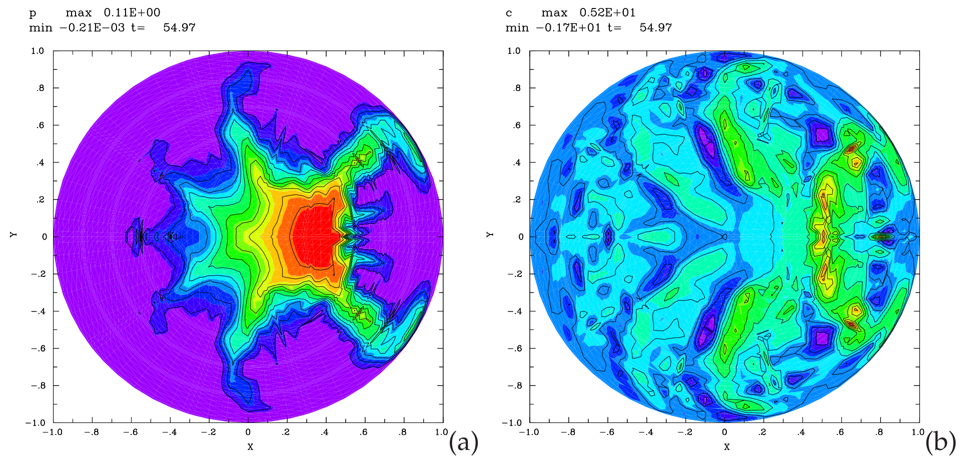

Figure 7: (a) nonlinear pressure (b) nonlinear current density calculated with 8th order spectral elements.

The nonlinear SEL pressure contours are shown in Fig. 7(a). There are some differences when compared to Fig. 6(a). The FEM solution is smoother. The SEL solution shows some structure which is aligned with element boundaries. The nonlinear current shown in Fig. 7(b) shows even more differences from Fig. 6(b). The reason is the solution inside each element is beginning to lose convergence. Other simulations show that this can be ameliorated using higher order elements or more dissipation.

M3D uses a potential representation of the magnetic field, which ensures that the divergence of the magnetic field vanishes, but also introduces higher derivatives into the equations to be solved. We see that satisfactory methods of solving the equations have been developed with both low order and high order discretizations. The high order method performs better for smooth problems, such as equilibrium and linear stability, while the low order method is better for highly nonlinear problems. The tendency of spectral methods to lose convergence at element boundaries is not primarily caused by the potential representation of M3D. It has been observed in fluid dynamics simulations using primitive variables [10]. It was shown that filtering could stabilize unwanted oscillations at element boundaries. Application of filtering methods to M3D will be considered in further research.

5 Conclusions

A spectral element library has been developed for the extended MHD code M3D. The same driver code can be used with a linear triangular finite element discretization and a high order quadrilateral spectral element discretization. This permits direct comparison of equilibrium, linear stability, and nonlinear evolution calculations using the two approaches.

The equilibrium and linear calculations using FEM and SEL libraries are in good agreement. The growth rate calculated with SEL appears to converge at a lower resolution than the growth rate calculated with FEM.

The low order FEM is more robust in nonlinear simulations, because it is inherently smoother. An important lesson seems to be that adequate smoothing must be included, by means of viscous dissipation coefficients such as viscosity and resistivity, to keep the solution sufficiently smooth for spectral elements.

An important lesson also seems to be that while high order spectral elements are very good, linear finite elements seem to also work quite well. Further comparisons will be carried out in future work to assess the advantages and disadvantages of high order SEL for extended MHD computation.

Acknowledgments

This work was supported by the U.S.D.O.E.

References

[1] W. Park, E. V. Belova, G. Y. Fu, X. Tang, H. R. Strauss and L. E. Sugiyama, Plasma simulation studies using multilevel physics models, Phys. Plasmas, 6 (1999), 1796.

[2] L. E. Sugiyama and W. Park, A nonlinear two-fluid model for toroidal plasmas, Phys. Plasma, 7 (2000), 4644.

[3] H. R. Strauss and W. Longcope, An adaptive finite element method for magnetohydrodynamics, J. Comput. Phys., 147 (1998), 318.

[4] A. T. Patera, A spectral element method for fluid dynamics - laminar flow in a channel expansion, J. Comput. Phys., 54 (1984), 463.

[5] G. E. Karniadakis and S. Sherwin, Spectral/hp Element Methods for Computational Fluid Dynamics, 2nd edition, Oxford University Press, 2005.

[6] P. F. Fischer and A. T. Patera, Parallel spectral element solution of the Stokes problem, J. Comput. Phys., 92(2) (1991), 380.

[7] A. H. Glasser and X. Z. Tang, The SEL macroscopic modeling code, Comput. Phys. Commun., 164 (2004), 237.

[8] C. R. Sovinec, A. H. Glasser, T. A. Gianakon, D. C. Barnes, R. A. Nebel, S. E. Kruger, D. D. Schnack, S. J. Plimpton, A. Tarditi and M. S. Chu, Nonlinear magnetohydrodynamics simulation using high-order finite elements, J. Comput. Phys., 195 (2004), 355.

[9] T. A. Davis, Algorithm 849: A concise sparse Cholesky factorization package, ACM Trans. Math. Software, 31 (2005), 587.

[10] P. F. Fischer and J. S. Mullen, Filter-based stabilization of spectral element methods, Comptes Rendus de l'Académie des sciences Paris, Série I-Analyse numérique, 332 (2001), 265.