

Multiple timescale calculations of sawteeth and other global macroscopic dynamics of tokamak plasmas

(Submitted Dec 18 2011)

S.C. Jardin¹, N. Ferraro², J. Breslau¹, J. Chen¹

¹ Princeton University Plasma Physics Laboratory

² General Atomics

Abstract

The M3D-C¹ [1] code is designed to perform 3D nonlinear magnetohydrodynamics (MHD) calculations of a tokamak plasma that span the timescales associated with ideal and resistive stability as well as parallel and perpendicular transport. This requires a scalable fully implicit time advance where the time step is not limited by a Courant condition based on the MHD wave velocities or plasma flow but is chosen instead to accurately and efficiently resolve the physics. In order to accomplish this, we make use of several techniques to improve the effective condition number of the implicit matrix equation that is solved each time-step. The split time advance known as the differential approximation [2] reduces the size of the matrix and improves its diagonal structure. A particular choice of velocity variables and annihilation operators approximately split the large matrix into 3 sub-matrices, each with a much improved condition number. A final block-Jacobi preconditioner further dramatically improves the condition number of the final matrix, allowing it to converge in a Krylov solver (GMRES) with a small number of iterations. As an illustration, we have performed transport timescale simulations of a tokamak plasma that periodically undergoes sawtooth oscillations [3]. We specify the transport coefficients and apply a “current controller” that adjusts the boundary loop-voltage to keep the total plasma current fixed. The short-time plasma response depends on the initial conditions, but the long-time behavior depends only on the transport coefficients and the boundary conditions applied.

Submitted to the IOP journal: Computational Science & Discovery

Corresponding author: jardin@pppl.gov

I. Introduction

The 3D resistive magnetohydrodynamic (MHD) equations are a mixed system of equations that have both hyperbolic and parabolic terms. This leads to multiple timescales when applied to high temperature magnetized plasma such as is confined in a modern tokamak. The hyperbolic terms are associated with ideal MHD wave propagation and global instabilities. These are the shortest timescales, typically microseconds or less. The parabolic terms are associated with the diffusion and transport of the magnetic field, current, pressures, and densities. These are the longest timescales, typically tens to hundreds of milliseconds or longer. To calculate both phenomena in a single simulation requires a highly implicit formulation so that the time step is determined by accuracy requirements only, not by numerical stability requirements such as the Courant condition. The implicit solution procedure is complicated by the fact that the multiple timescales present in the physics lead to a very ill-conditioned matrix equation that needs to be solved each time step. In this paper, we describe three techniques used in M3D- C^1 to precondition this matrix equation so that it may be solved efficiently using GMRES [4] or other iterative solvers suitable for non-symmetric matrices. The techniques are: the split implicit method (Section III), the annihilation operators (Section IV), and the block-Jacobi preconditioner (Section V). These three techniques, when used together, enable accurate solutions, even when using fine spatial meshes and large time steps.

The M3D- C^1 code uses a stream-function/potential representation of the magnetic and velocity vector fields and applies corresponding annihilation operators in a way that is similar, but not identical, to that used in M3D [5]. This representation builds on the reduced-MHD representation used in JOEK [6] and BOUT++ [7], but extends it to the full MHD equations. The split implicit method described in Section III is very similar to that used in the NIMROD [8] code and that used as a preconditioner for the Newton-Krylov solve in the PIXIE3D [9] and XTOR [10] codes. However, M3D- C^1 is the only code that uses three-dimensional finite elements with C^1 continuity (continuous first derivatives across element boundaries), and that feature allows these two preconditioning techniques to be combined in a way that would be very inefficient with other representations. The underlying reason is that when the split implicit method is applied to the M3D and M3D- C^1 representation of the vector fields, fourth order spatial derivatives occur in the equations. These fourth order spatial derivatives require C^1 elements when the Galerkin method is applied and two integration-by-parts are performed [11].

The other unique feature of this effort is using the block-Jacobi preconditioner described in Section V to dramatically improve the condition number of the entire matrix by solving for all the in-plane strong couplings directly. This technique relies on the use of finite elements (or finite differences) in the toroidal direction, as opposed to the spectral representation used in NIMROD, JOEK, and XTOR.

II. The equations and their properties

The partial differential equations we are concerned with are the following (SI units) [12]:

$$\frac{\partial n}{\partial t} + \nabla \cdot (n\mathbf{V}) = 0 \quad \text{continuity} \quad (1a)$$

$$\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E} \quad \nabla \cdot \mathbf{B} = 0 \quad \mu_0 \mathbf{J} = \nabla \times \mathbf{B} \quad \text{Maxwell} \quad (1b)$$

$$nM_i \left(\frac{\partial \mathbf{V}}{\partial t} + \mathbf{V} \cdot \nabla \mathbf{V} \right) + \nabla p = \mathbf{J} \times \mathbf{B} - \nabla \cdot \mathbf{\Pi}_{GV} - \nabla \cdot \mathbf{\Pi}_{\mu} \quad \text{momentum} \quad (1c)$$

$$\mathbf{E} + \mathbf{V} \times \mathbf{B} = \eta \mathbf{J} + \frac{1}{ne} (\mathbf{J} \times \mathbf{B} - \nabla p_e - \nabla \cdot \mathbf{\Pi}_e) \quad \text{Ohm's law} \quad (1d)$$

$$\frac{3}{2} \frac{\partial p_e}{\partial t} + \nabla \cdot \left(\frac{3}{2} p_e \mathbf{V}_e \right) = -p_e \nabla \cdot \mathbf{V}_e + \eta J^2 - \nabla \cdot \mathbf{q}_e + Q_{\Delta} \quad \text{electron energy} \quad (1e)$$

$$\frac{3}{2} \frac{\partial p_i}{\partial t} + \nabla \cdot \left(\frac{3}{2} p_i \mathbf{V} \right) = -p_i \nabla \cdot \mathbf{V} - \mathbf{\Pi}_{\mu} : \nabla \mathbf{V} - \nabla \cdot \mathbf{q}_i - Q_{\Delta} \quad \text{ion energy} \quad (1f)$$

These have been color-coded so that the terms in black are those of ideal-MHD, the terms in red are the new terms for resistive MHD, and the terms in blue are those associated with two-fluid MHD (which will not be considered here). The symbols have their normal meanings with $n = n_e = n_i$ being the electron and the ion particle density, \mathbf{V} is the ion fluid velocity, $\mathbf{V}_e \equiv \mathbf{V} - \mathbf{J} / ne$ is the electron fluid velocity, \mathbf{B} and \mathbf{E} are the magnetic and electric fields, \mathbf{J} is the electrical current density, $p = p_e + p_i$ is the total pressure (sum of electron and ion pressures), $\mathbf{\Pi}_{\mu}$ is the viscosity tensor[13], Q_{Δ} is the classical equipartition term, and $\mathbf{q}_e, \mathbf{q}_i$ are the electron and ion heat fluxes, which need to be supplied to close the equations.

The objective of the M3D-C¹ project is to solve these equations as accurately as possible over long timescales in 3D toroidal geometry with realistic boundary conditions without making further approximations. The solution procedure is optimized for a low- β torus (where $\beta \equiv 2\mu_0 p / B^2$ is the ratio of fluid to magnetic pressure) with a strong toroidal magnetic field such as exists in a modern tokamak. The equations describe ideal MHD phenomena that occur over timescales τ_I , magnetic reconnection phenomena that occur over timescales τ_R , and the transport of particles, heat, momentum, and the magnetic field that occur over timescales τ_T . The timescales for these distinct phenomena satisfy the inequalities:

$$\tau_I \ll \tau_R \ll \tau_T \quad (2)$$

This is the primary source of the multiple timescales that must be dealt with.

However, even within ideal MHD as applied to a tokamak there are a wide range of timescales. These stem from the three wave solutions in a low- β magnetized plasma that become nearly orthogonal and largely distinct [12]. The slow wave, with velocity V_S only propagates parallel to \mathbf{B} , only compresses the fluid in the parallel direction, and does not perturb the magnetic field. The Alfvén wave, with velocity V_A , also only propagates parallel to \mathbf{B} , is incompressible, and only bends the magnetic field (does not compress it). The fast wave, with velocity V_F is the only wave that can propagate perpendicular to \mathbf{B} . It only compresses the fluid in the perpendicular direction. This is the wave that

makes the equations stiff. When applied to tokamak geometry for studying MHD instabilities which propagate nearly perpendicular to the magnetic field, $\mathbf{k} \cdot \mathbf{B} \approx 0$, the three wave velocities satisfy the inequalities :

$$V_F \gg V_A \gg V_S. \quad (3)$$

This leads to multiple timescales even within ideal MHD.

The multiple timescales require using an implicit numerical algorithm so that the time step can be large compared to the Courant condition for the fastest wave and still maintain numerical stability. The implicit solution requires evaluating the spatial derivatives at the new time level. If we discretize the equations in space (using finite differences, finite elements, or spectral decomposition) and linearize them about the present time level, the time advance equations to go from time step n to time step $n+1$ take the form:

$$\mathbf{A} \cdot \mathbf{X}^{n+1} = \mathbf{R}(\mathbf{X}^n) \quad (4)$$

Here \mathbf{X}^{n+1} is the complete solution $(n, \mathbf{V}, \mathbf{B}, p)$ at the new time level, the right side is a function only of the old time level, and the matrix \mathbf{A} is a very large (typically $10^7 \times 10^7$) non-diagonally dominant, non-symmetric, ill-conditioned sparse matrix. The ill-conditioning is a direct result of the fact that this matrix contains all of the MHD wave phenomena in it. We next describe the three physics-based preconditioning techniques used to transform this equation into one that can be efficiently solved with an iterative sparse matrix Krylov solver.

III. The split implicit method

The split implicit method is a technique for reducing the size of the matrix \mathbf{A} in Eq. (4) and making it more symmetric and diagonally dominant by algebraically eliminating the new time level magnetic field and pressure, \mathbf{B}^{n+1} and p^{n+1} in terms of the new time level velocity \mathbf{V}^{n+1} . As an example, consider the simple 1D wave equation for the velocity V and pressure p :

$$\left. \begin{aligned} \frac{\partial V}{\partial t} &= c \frac{\partial p}{\partial x} \\ \frac{\partial p}{\partial t} &= c \frac{\partial V}{\partial x} \end{aligned} \right\} \frac{\partial^2 V}{\partial t^2} - c^2 \frac{\partial^2 V}{\partial x^2} = 0 \quad (5)$$

A fully implicit finite centered in space finite difference method would evaluate the spatial derivatives at the new time level. Letting $t = n \delta t$, $x = j \delta x$, and $s \equiv c \delta t / \delta x$, we have

$$V_j^{n+1} = V_j^n + s(p_{j+1/2}^{n+1} - p_{j-1/2}^{n+1}) \quad (6a)$$

$$p_{j+1/2}^{n+1} = p_{j+1/2}^n + s(V_{j+1}^{n+1} - V_j^{n+1}) \quad (6b)$$

The split implicit method uses Eq. (6b) to eliminate the new time pressure from Eq. (6a) in favor of higher spatial derivative of the new time velocity. We thus obtain the pair of equations:

$$V_j^{n+1} - s^2 (V_{j+1}^{n+1} - 2V_j^{n+1} + V_{j-1}^{n+1}) = V_j^n + s (p_{j+1/2}^n - p_{j-1/2}^n) \quad (7a)$$

$$p_{j+1/2}^{n+1} = p_{j+1/2}^n + s (V_{j+1}^{n+1} - V_j^{n+1}) \quad (7b)$$

The finite difference equations (7a,b) will give exactly the same answers as equations (6a,b), but can be solved sequentially. First, Eq. (7a) is solved for the new time velocities V_j^{n+1} and then Eq. (7b) is solved for the new time pressure p_j^{n+1} . Also the matrix corresponding to Eq. (7a) is seen to be half the size of the matrix corresponding to Eq. (6a,b) (since it involves only V^{n+1} and not p^{n+1}) and is symmetric and diagonally dominant. The matrix corresponding to Eq. (7b) is just the identity matrix since everything on the right side is known.

So in this case, the algebraic substitution takes one from having to solve a $2N \times 2N$ anti-symmetric system that has large off-diagonal elements to sequentially solving a $N \times N$ symmetric system that is diagonally dominant, and following this with a trivial update that does not involve a matrix solve. These systems are mathematically equivalent and will give the same answers for infinite precision arithmetic, but the second system is better suited for iterative solvers. This same technique is generalized to include the magnetic field and three variables representing the velocity field and used in M3D-C¹ to solve first for the velocity field at the new time level and then for the pressure and magnetic field variables using the new time velocities [14].

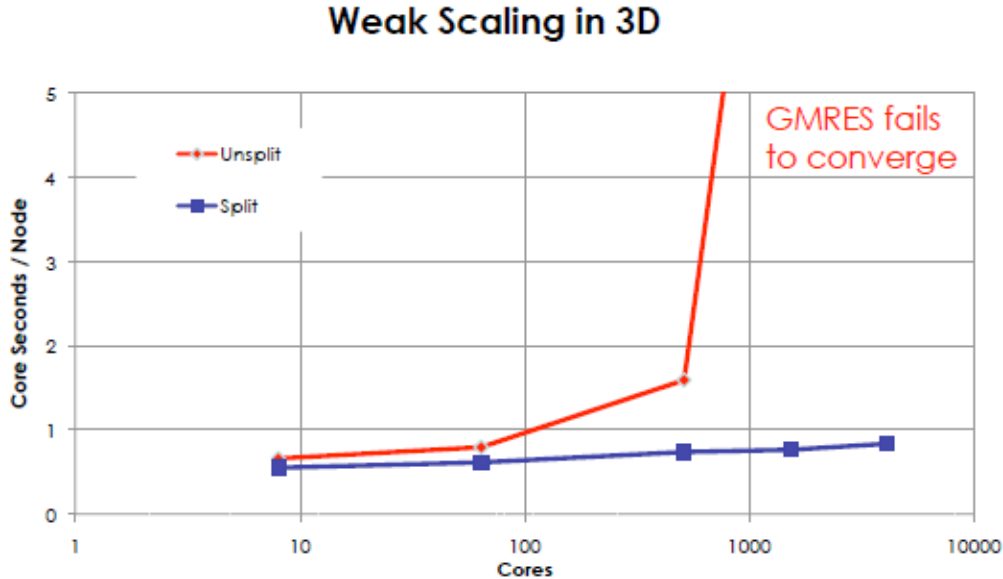


Figure 1: Comparison of the amount of CPU time required for one time step using the split and unsplit formulations for a typical problem. For sufficiently large problem size, the unsplit method fails to converge.

As an illustration of the effectiveness of this technique, we show in figure 1 a comparison weak scaling study where this transformation was (split) and was not (unsplit) performed, but that the same preconditioning techniques as described in Sections IV and V are applied. It is clear that when using the GMRES Krylov solver to solve the 3D matrix equation, the split method always converges in fewer iterations than does the unsplit method. This difference is more pronounced for large numbers of mesh and processor nodes, and for large enough systems, the unsplit method fails to converge at all.

IV. The annihilation operators

In order to further improve the properties of the matrix \mathbf{A} , we express the 3D velocity vector in terms of 3 scalar fields (U, ω, χ) as follows using cylindrical (R, ϕ, Z) coordinates:

$$\mathbf{V} = R^2 \nabla U \times \nabla \phi + R^2 \omega \nabla \phi + \frac{1}{R^2} \nabla_{\perp} \chi. \quad (8)$$

The operator ∇_{\perp} denotes the gradient in the (R, Z) plane; i.e. orthogonal to $\nabla \phi$. The first term, U , is associated mainly with the shear Alfvén wave. It is constructed in a way that it does not compress the strong toroidal magnetic field. The second term, ω , is mainly associated with the slow wave and also does not compress the toroidal magnetic field. The last term, χ , is associated mainly with the fast wave. This term alone does compress the toroidal field. It has been demonstrated that this form for the velocity vector can yield very accurate solutions for linear ideal MHD instabilities in low- β tokamak plasmas [15].

To obtain the scalar time-advancement equations, we apply annihilation projections to the momentum equation as it is modified by the split-implicit method described in the last section. Thus, instead of taking Cartesian projections of the vector momentum equation 1c (as modified by the split implicit method), we form three scalar projections by successively operating with the three projection operators

$$\nabla \phi \cdot \nabla_{\perp} \times R^2 \quad (9a)$$

$$R^2 \nabla \phi \cdot \quad (9b)$$

$$-\nabla_{\perp} \cdot R^{-2} \quad (9c)$$

These three projections, together with the form of the velocity field, have the effect of approximately separating the dynamics associated with each of the three MHD waves into separate diagonal blocks. The block associated with the projection (9a) contains the shear Alfvén wave dynamics and multiplies the velocity variable U . The block associated with (9b) contains the slow wave dynamics and multiplies the velocity variable ω . Similarly, the block associated with (9c) contains the bulk of the fast wave dynamics and multiplies the velocity variable χ .

Schematically, the effect of the annihilation operators on the matrix \mathbf{A} is as shown in Eq. (10),

$$\left[\begin{array}{c} \\ \\ \end{array} \right] \rightarrow \left[\begin{array}{ccc} [] & & \\ & [] & \\ & & [] \end{array} \right] . \quad (10)$$

Since the matrix is transformed to one with three approximate diagonal blocks, each multiplying a separate component of the velocity field, it is now only the condition number of each of the diagonal blocks that is relevant for the efficiency of the iterative solver, not the condition number of the entire matrix. This structure with the dominant terms appearing only in the three diagonal blocks is maintained in the Krylov solver because the Krylov basis space is formed by repeated multiplication of the matrix in Eq. (10).

It is worth noting that the M3D- C^1 code can be run with one, two, or three velocity variables. Keeping only one velocity variable, U , corresponds to 2-variable reduced MHD, and just keeps the upper left diagonal block in Eq. (10). Keeping both U and ω corresponds to 4-variable reduced MHD, and keeps the upper two diagonal blocks in Eq. (10). Keeping all three velocity variables corresponds to full MHD, and keeps all three diagonal blocks.

We can illustrate how effective these annihilation projections are by forming the equivalent of the matrix \mathbf{A} of Eq. (4) (but after the split implicit method and the annihilation operators have been applied) and calculating all of its eigenvalues. The ratio of the largest to the smallest will give the condition number. We do this for a typical tokamak equilibrium state and form three matrices corresponding to keeping one, two, or three velocity variables. We then calculate all the eigenvalues of each of these three matrices and plot the result in figure 2. In order to make this analysis tractable, we perform these operations on a linearized 2D form of the matrix \mathbf{A} where we have assumed a ϕ dependence of $\exp[in\phi]$ with $n = 1$.

We see that when keeping only one velocity variable, U (far left in the figure), the eigenvalues range from about 10^{-2} to 10^3 , a range of about 10^5 . These are the eigenvalues associated with the shear Alfvén wave. We then reform the matrix keeping two velocity variables and two annihilation projections this time and re-compute the eigenvalues and plot them in the middle column of figure 1. It is seen that the eigenvalues associated with the shear Alfvén wave are almost unchanged, but that a new set of eigenvalues appear in the range of about 10^{-6} to 10^1 , a range of about 10^7 . These are associated with the slow wave. Finally, we reform the matrix keeping all three velocity variables and all three annihilation projections and again calculate the eigenvalues and plot them on the right of figure 3. We see that again the eigenvalues associated with the shear Alfvén wave are almost unchanged, those associated with the slow wave are modified slightly, and a new set of eigenvalues appear that are associated with the fast wave and are in the range 10^2 to 10^7 .

Thus, we see that the annihilation operators have led to a matrix which is block-diagonally dominant (i.e. there are three diagonal blocks, one for each type of eigenmodes), whereas using different projections of the equations would lead to a less dominantly block-diagonal structure. Each of the three

blocks has a condition number greatly reduced from the condition number of the full matrix. Since these diagonal blocks are approximately decoupled from one another in the Krylov iterations, we expect that the block diagonal matrix so formed will converge in many fewer iterations than the original one. This assertion cannot be directly tested in M3D-C¹ because the form of the velocity vector and annihilation operators are built into the formulation at a fundamental level and cannot be changed.

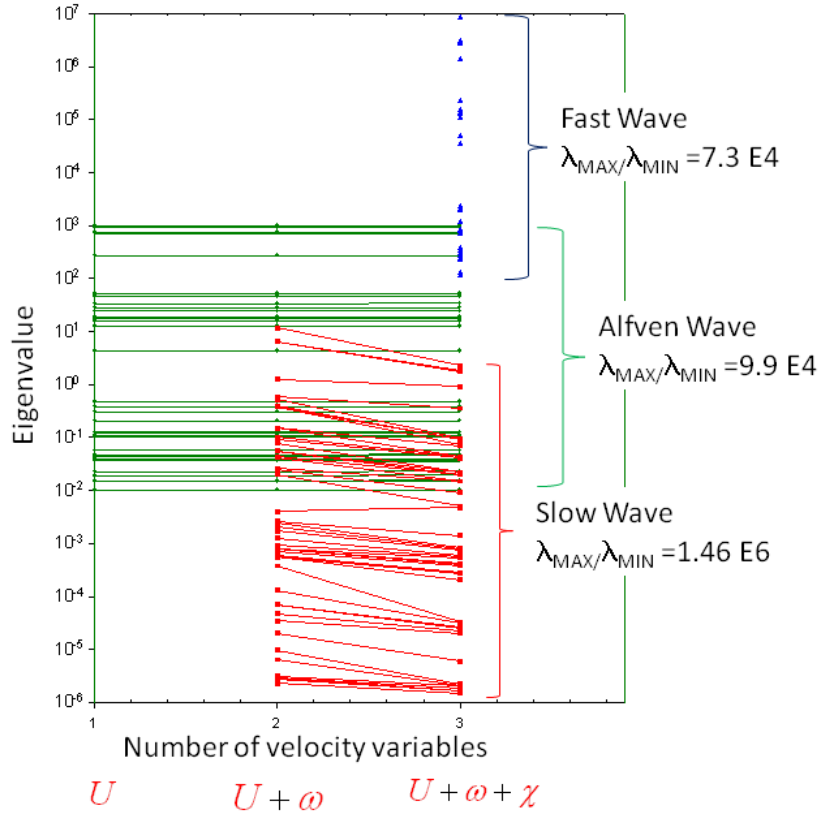


Figure 2: Illustration of eigenvalues of the matrix A from Eq. (4) when keeping 1,2,or 3 velocity variables. It is seen that the eigenvalues split into three groups corresponding to the Alfven, slow, and fast waves respectively. This analysis was performed for a 2D A matrix assuming ϕ dependence $\exp[i n \phi]$ with $n=1$.

V. Block Jacobi preconditioner and parallel scaling

The M3D-C¹ code uses a triangular wedge high order finite element as shown in figure 3 that is a tensor product of a reduced quintic triangular element in the (R,Z) plane and a Hermite cubic element in the toroidal angle ϕ . Because the element ordering is structured in the toroidal angle and each plane is coupled only to the two neighboring planes, the final matrix we obtain after performing the Galerkin integrations takes on a block triangular structure:

$$\mathbf{A}_j \cdot \mathbf{Y}_{j-1} - \mathbf{B}_j \cdot \mathbf{Y}_{j-1} + \mathbf{C}_j \cdot \mathbf{Y}_{j+1} = \mathbf{D}_j \quad (11)$$

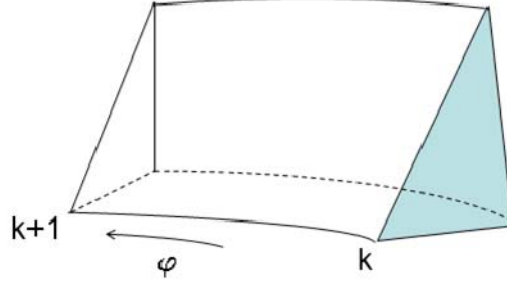


Figure 3: Triangular wedge integration volume is the tensor product of a reduced quintic triangle in (R,Z) and a Hermite cubic element in ϕ .

The large sparse matrices $\mathbf{A}_j, \mathbf{B}_j, \mathbf{C}_j$ represent all the couplings within plane j and with the two adjacent planes. While in principle, one could solve Eq. (11) with the block tridiagonal algorithm; this direct approach does not scale well. Instead, what we do is to precondition the large 3D matrix by multiplying each block row by the inverse of the diagonal block \mathbf{B}_j^{-1} . This is done in terms of a block Jacobi preconditioner using SuperLU_DIST [16] or MUMPS[17] within PETSc [18] so that each diagonal block matrix needs to be factored but never actually inverted. Physically, this preconditioner is motivated by the small zone size and hence the strong couplings within a plane. We thus precondition the 3D matrix by effectively directly inverting the components within each poloidal plane simultaneously. It is shown in the appendix that for a model problem, this can dramatically improve the condition number of the matrix. For typical tokamak parameters, this improvement can be by a factor of 1000 or more as shown in Table A1.

The results of a weak scaling study in which these preconditioning techniques are used are shown in Fig. 4 and in Table 1. The approximate linear size of the zones that make up the unstructured triangles within a poloidal plane varied from 0.08 to 0.02 in this study which corresponds to a factor of about 16 in the number of zones in the cross-sectional area. The number of toroidal planes varied from 8 to 64. The time step was held fixed at $\Delta t = 20 \tau_A$. We see that as we increase both the number of zones and the number of processors by a factor of about 128, the cycle time increases only by about 70% and the number of iterations required for the velocity matrix iterative solve increases only by a factor of 5. Even at the largest problem size, the time spent in the solvers (including the block Jacobi preconditioner) did not dominate the loop time. Had we not performed the block-Jacobi preconditioner, the GMRES iteration would fail to converge even with a 1000 iteration maximum.

Table 1: Data for scaling study of Figure 4. Size is the approximate linear size of the triangles. Planes is the number of toroidal planes. C/PL are the number of processors per plane. Loop_T is the total time for 1 timestep (s). Solve_T is the total time spent in the preconditioners and solvers per timestep. #ITER is the number of iterations used in GMRES for solving the preconditioned velocity matrix for 1 timestep.

	SIZE	PLANES	C/PL	LOOP_T	SOLVE_T	#ITER
A	.08	8	12	200	12	54
B	.08	16	12	202	14	100
C	.04	16	48	236	31	84
D	.04	32	48	242	37	141
E	.03	40	72	270	50	173
F	.03	60	72	275	54	226
G	.02	16	192	325	100	77
H	.02	24	192	332	108	105
I	.02	32	192	335	110	122
J	.02	64	192	340	140	256

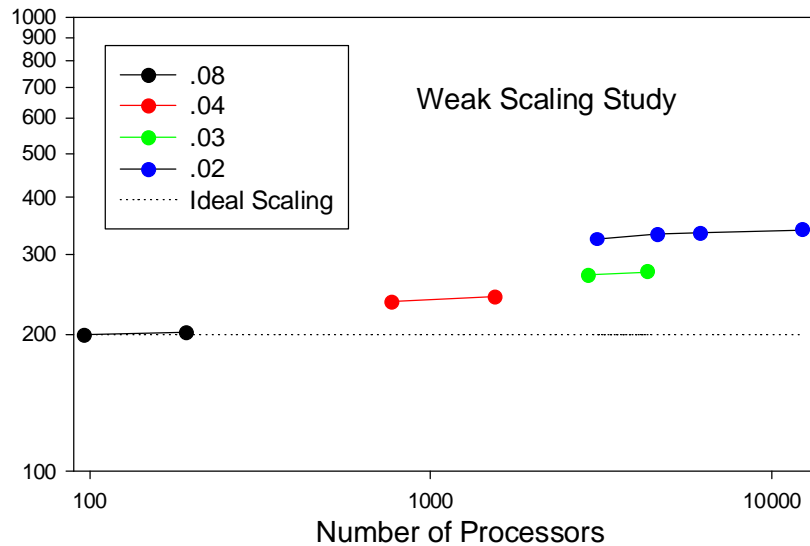


Figure 4: Weak scaling study on a Cray XE6 (Hopper) at NERSC. We plot the wall clock time for 1 time step vs. the number of processors (and elements). The approximate linear triangle size varied from 0.02 to 0.08 and the number of toroidal planes varied from 8 to 64. The number of elements per processor remained constant from 96 to 12288 processors.

VI. Results

As an illustration of the effectiveness of this method, we present in Fig. 5 the results of a multiple timescale tokamak simulation that spans the transport, reconnection, and stability timescales. For simplicity, we took the plasma density to be constant in time and space. In this simulation, we specify a transport model by applying an approximate Spitzer resistivity model, $\eta \sim p^{-3/2}$ and an approximate Braginskii [19] perpendicular thermal conductivity, $\kappa_{\perp} \sim p^{-1/2}$ which varied from 5.7×10^{-6} to 16×10^{-6} as the pressure p varied from 0.0007 to 0.0077. In the normalized units used in this calculation where the toroidal magnetic field and minor radius of the device were both unity, a constant viscosity is used with $\nu = 10^{-4}$ and a uniform parallel thermal conductivity of $\kappa_{\parallel} = 10^6$. The central value of the resistivity was 10^{-6} .

A current controller is included that is in a feedback loop to provide a loop voltage at the boundary to maintain the plasma current its initial value. The loop voltage provides thermal energy through ohmic heating. As the calculation proceeds, the current density periodically peaks in the central cross section of the torus, becomes unstable, reconnects, and broadens. Each time it becomes unstable and reconnects the kinetic energy of the system increases and then decreases when the reconnection is complete. This sequence is illustrated in Fig. 5. The details of the first reconnection event are dependent on the initial conditions, but after many reconnection events, the system takes on a periodic cycle and the calculation is stopped. The time step used in these runs was $\Delta t = 40\tau_A$. Because the total simulation time is so long compared to the Alfvén wave transit time, we are able to compute the stability behavior of the plasma with self-consistent profiles that are determined solely by the transport model and boundary conditions. These calculations have spanned the timescales between ideal MHD and plasma transport.

VII. Summary

Solving the 3D MHD equations in a highly magnetized high temperature plasma is difficult because of the multiplicity of time scales associated with ideal MHD wave propagation and stability, magnetic reconnection, and transport. If one casts this as an implicit system, the implicit matrix contains a large range of eigenvalues associated with the three different MHD wave types.

We have presented a three stage preconditioning technique for solving the implicit resistive MHD equations in a tokamak that allows a large time step to be used. The split-implicit method reduces the matrix size by 2 and makes the matrix near symmetric and diagonally dominant. The annihilation operators approximately split the matrix into 3 diagonal blocks, each with a greatly reduced condition number. A block Jacobi preconditioner dramatically reduces the condition number of each of the diagonal blocks. The final preconditioned matrix is then given to GMRES and converges in tens or a few hundreds of iterations even for a fine zoned problem. We presented an application showing repeating sawteeth that demonstrates the ability to calculate on multiple timescales. The periodic stability behavior is directly dependent on the transport model specified.

Acknowledgements

This project has benefited greatly from software developed and provided by M. Shephard and the SCOREC team at Rensselaer Polytechnic Institute, and by the modifications to the PETSc software by Hong Zhang to allow parallel block Jacobi preconditioning. This work was supported by U.S. DOE Contract No. DE-AC02-76CH03073, and the SciDAC Center for Extended Magnetohydrodynamic Modeling (CEMM). Calculations were performed at the PPPL Computer Center and at the National Energy Research Supercomputer Center (NERSC).

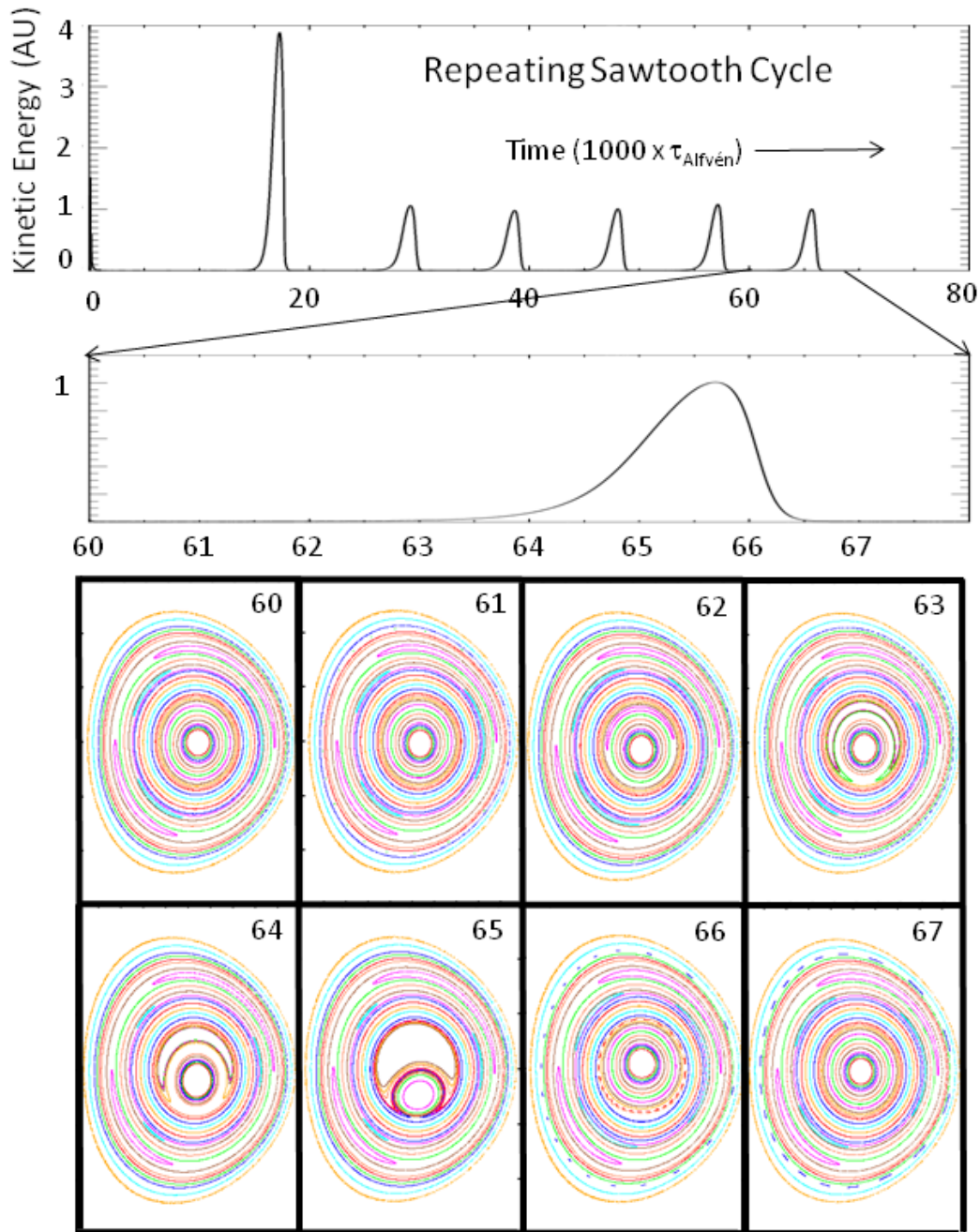


Figure 5: Repeating sawtooth cycle. Top is kinetic energy vs. time. Bottom is a sequence of Poincaré plots of the magnetic field during a single cycle.

References

- [1] J. Breslau, N. Ferraro, S. Jardin, *Physics of Plasmas* **16** 092503 (2009)
- [2] E. Caramana, *J. Comput. Phys.* **96** 484 (1991)
- [3] S. Von Goeler, W. Stodiek, and N. Sauthoff, *Phys. Rev. Lett.* **33**, 1201 (1974)
- [4] Y. Saad and M. H. Schultz, *SIAM J. Sci. Stat. Comput.* **7** pp. 856-869 (1986)
- [5] W. Park and D. Monticello, *Nucl. Fusion* **30** 2413-2418 (1990)
- [6] G. Huysmans, S. Pamela, et al., *Plasma Phys. Control. Fusion* **51** 124012 (2009)
- [7] B. D. Dudson, M. Umansky, X. Xu, et al, *Comp. Phys. Comm* **180** 1467-1480 (2009)
- [8] C. R. Sovinec, J. R. King, *J. Comput Phys.* **229** 5803-5819 (2010)
- [9] L. Chacon, *Phys. Plasmas* **15** 056103 (2008)
- [10] H. Lutgens and J. Luciani, *J. Comput. Phys.* **229** 8130-8143 (2010)
- [11] C. Strang and G. Fix, *An Analysis of the Finite Element Method*, Englewood Cliffs, NJ; Prentice-Hall (1973)
- [12] S. C. Jardin, *Computational Methods in Plasma Physics*, Taylor and Francis, Boca Raton (2010)
- [13] Landau and Lifschitz, *Fluid Mechanics*, Butterworth-Heinemann; 2nd edition (1987)
- [14] S. C. Jardin, *J. Comput. Phys.* **231** 822-838 (2012)
- [15] N. M. Ferraro, S. C. Jardin, P. B. Snyder, *Phys. Plasmas* **17**, 102508 (2010)
- [16] X. S. Li and J. W. Demmel, *ACM Trans. Mathematical Software*, **29** 110-140 (2003)
- [17] P. R. Amestoy, I. S. Duff and J.-Y. L'Excellent, *Methods in Appl. Mech. Eng.*, **184**, 501-520 (2000)
- [18] see <http://www.mcs.anl.gov/petsc>
- [19] S. I. Braginskii, "Transport Processes in a Plasma", *Reviews of Plasma Physics*, Vol. 1 (Consultants Bureau, New York, 1965)

Appendix: Block Jacobi Preconditioner

We adopt the standard finite difference notation where $x = j\Delta x$, $j = 0 \cdots N_x$, etc. and the centered finite difference operator is defined as $\delta_x^2 \Phi = \Delta x^{-2} (\Phi_{j+1} - 2\Phi_j + \Phi_{j-1})$, etc. Consider the 3D Helmholtz equation in discretized form:

$$\left[1 - s(\delta_x^2 + \delta_y^2 + \delta_z^2) \right] \Phi_{jkl} = r_{jkl} \quad (\text{A1})$$

The discrete eigenvalues of Eq. (A1) are

$$\lambda_{m,n,p} = 1 - \frac{2s}{\Delta x^2} \left(\cos \frac{m\pi}{N_x} - 1 \right) - \frac{2s}{\Delta y^2} \left(\cos \frac{n\pi}{N_y} - 1 \right) - \frac{2s}{\Delta z^2} \left(\cos \frac{p\pi}{N_z} - 1 \right) \quad m, n, p = 0, N-1. \quad (\text{A2})$$

These are seen to approach the eigenvalues of the continuous equation:

$$\lambda_{m,n,p} = 1 + s \left[\left(\frac{m\pi}{L_x} \right)^2 + \left(\frac{n\pi}{L_y} \right)^2 + \left(\frac{p\pi}{L_z} \right)^2 \right], \quad m, n, p = 0, \dots, \infty. \quad (\text{A3})$$

Expanding out the centered finite difference operator in z gives:

$$\left[1 - s\delta_x^2 - s\delta_y^2 + 2\frac{s}{\Delta z^2} \right] \Phi_{jkl} - \frac{s}{\Delta z^2} \Phi_{jkl+1} - \frac{s}{\Delta z^2} \Phi_{jkl-1} = r_{jkl}. \quad (\text{A4})$$

A block Jacobi preconditioner corresponds to inverting the operator in brackets for each l value independently. To accomplish this, we apply a finite Fourier transform in the x and y directions:

$$\tilde{\Phi}_{mnl} = \frac{1}{N_x N_y} \sum_{j=0}^{N_x-1} \sum_{k=0}^{N_y-1} \Phi_{jkl} \exp \left(\frac{2\pi i j m}{N_x} + \frac{2\pi i k n}{N_y} \right) \quad (\text{A5a})$$

$$\Phi_{jkl} = \sum_{m=0}^{N_x-1} \sum_{n=0}^{N_y-1} \tilde{\Phi}_{mnl} \exp \left(-\frac{2\pi i j m}{N_x} - \frac{2\pi i k n}{N_y} \right) \quad (\text{A5b})$$

Applying the transform given in (A5b) to (A4) gives

$$\left[1 - \frac{2s}{\Delta x^2} \left(\cos \frac{m\pi}{N_x} - 1 \right) - \frac{2s}{\Delta y^2} \left(\cos \frac{n\pi}{N_y} - 1 \right) + 2\frac{s}{\Delta z^2} \right] \tilde{\Phi}_{mnl} - \frac{s}{\Delta z^2} \tilde{\Phi}_{mnl+1} - \frac{s}{\Delta z^2} \tilde{\Phi}_{mnl-1} = \tilde{r}_{mnl} \quad (\text{A6})$$

Or, upon defining and dividing by

$$D_{m,n} \equiv \left[1 - \frac{2s}{\Delta x^2} \left(\cos \frac{m\pi}{N_x} - 1 \right) - \frac{2s}{\Delta y^2} \left(\cos \frac{n\pi}{N_y} - 1 \right) + 2\frac{s}{\Delta z^2} \right] \quad (\text{A7})$$

We have

$$\tilde{\Phi}_{mnl} - \frac{s}{D_{m,n} \Delta z^2} \tilde{\Phi}_{mnl+1} - \frac{s}{D_{m,n} \Delta z^2} \tilde{\Phi}_{mnl-1} = \frac{\tilde{r}_{mnl}}{D_{m,n}} \quad (\text{A8})$$

Now, transform back using (A5a)

$$\begin{aligned} \Phi_{jkl} - \frac{s}{\Delta z^2} \sum_{m=0}^{N_x-1} \sum_{n=0}^{N_y-1} \frac{1}{D_{m,n}} \tilde{\Phi}_{mnl+1} \exp\left(-\frac{2\pi ijm}{N_x} - \frac{2\pi ikn}{N_y}\right) \\ - \frac{s}{\Delta z^2} \sum_{m=0}^{N_x-1} \sum_{n=0}^{N_y-1} \frac{1}{D_{m,n}} \tilde{\Phi}_{mnl-1} \exp\left(-\frac{2\pi ijm}{N_x} - \frac{2\pi ikn}{N_y}\right) = \sum_{m=0}^{N_x-1} \sum_{n=0}^{N_y-1} \frac{1}{D_{m,n}} \tilde{r}_{mnl} \exp\left(-\frac{2\pi ijm}{N_x} - \frac{2\pi ikn}{N_y}\right) \end{aligned}$$

Or, using (A5a) again

$$\begin{aligned} \Phi_{jkl} - \frac{s}{\Delta z^2 N_x N_y} \sum_{m=0}^{N_x-1} \sum_{n=0}^{N_y-1} \frac{1}{D_{m,n}} \sum_{j'=0}^{N_x-1} \sum_{k'=0}^{N_y-1} \Phi_{j'k'l+1} \exp\left(\frac{2\pi i(j'-j)m}{N_x} + \frac{2\pi i(k'-k)n}{N_y}\right) \\ - \frac{s}{\Delta z^2 N_x N_y} \sum_{m=0}^{N_x-1} \sum_{n=0}^{N_y-1} \frac{1}{D_{m,n}} \sum_{j'=0}^{N_x-1} \sum_{k'=0}^{N_y-1} \Phi_{j'k'l-1} \exp\left(\frac{2\pi i(j'-j)m}{N_x} + \frac{2\pi i(k'-k)n}{N_y}\right) \\ = \sum_{m=0}^{N_x-1} \sum_{n=0}^{N_y-1} \frac{1}{D_{m,n}} \tilde{r}_{mnl} \exp\left(-\frac{2\pi ijm}{N_x} - \frac{2\pi ikn}{N_y}\right) \end{aligned}$$

Next, define the matrices:

$$\mathbf{A}_{j'k'}^{jk} \equiv \frac{1}{N_x N_y} \sum_{m=0}^{N_x-1} \sum_{n=0}^{N_y-1} \frac{1}{D_{m,n}} \exp\left(\frac{2\pi i(j'-j)m}{N_x} + \frac{2\pi i(k'-k)n}{N_y}\right) \quad (\text{A9})$$

We have

$$\Phi_{jkl} - \frac{s}{\Delta z^2} \sum_{j'=0}^{N_x-1} \sum_{k'=0}^{N_y-1} \mathbf{A}_{j'k'}^{jk} \cdot [\Phi_{j'k'l+1} + \Phi_{j'k'l-1}] = \sum_{m=0}^{N_x-1} \sum_{n=0}^{N_y-1} \frac{1}{D_{m,n}} \tilde{r}_{mnl} \exp\left(-\frac{2\pi ijm}{N_x} - \frac{2\pi ikn}{N_y}\right) \quad (\text{A10})$$

Now, transform in the z direction (l index),

$$\Phi_{jkl} = \sum_{p=0}^{N_z-1} \bar{\Phi}_{jkp} \exp(-2\pi ilp / N_z) \quad (\text{A11a})$$

$$\bar{\Phi}_{jkp} = \frac{1}{N_z} \sum_{p=0}^{N_z-1} \Phi_{jkl} \exp(+2\pi ilp / N_z) \quad (\text{A11b})$$

With this, Eq. (A10) becomes

$$\left[\mathbf{I} - \frac{2s}{\Delta z^2} \sum_{j'=0}^{N_x-1} \sum_{k'=0}^{N_y-1} \mathbf{A}_{j'k'}^{jk} \cos\left(\frac{2\pi p}{N_z}\right) \right] \cdot \bar{\Phi}_{jkp} = \sum_{l=0}^{N_z-1} \sum_{m=0}^{N_x-1} \sum_{n=0}^{N_y-1} \frac{1}{D_{m,n}} \tilde{r}_{mnl} \exp 2\pi i \left(-\frac{jm}{N_x} - \frac{kn}{N_y} + \frac{lp}{N_z} \right) \quad (\text{A12})$$

So, the eigenvalues satisfy:

$$\lambda \bar{\Phi}_{jkp} = \left[\mathbf{I} - \frac{2s}{\Delta z^2} \sum_{j'=0}^{N_x-1} \sum_{k'=0}^{N_y-1} \mathbf{A}_{j'k'}^{jk} \cos\left(\frac{2\pi p}{N_z}\right) \right] \bar{\Phi}_{jkp} \quad (\text{A13})$$

Or,

$$\left| \left(1 - \lambda\right) \mathbf{I} - \frac{2s}{\Delta z^2} \sum_{j'=0}^{N_x-1} \sum_{k'=0}^{N_y-1} \mathbf{A}_{j'k'}^{jk} \cos\left(\frac{2\pi p}{N_z}\right) \right| = 0 \quad (\text{A14})$$

The condition number is the ratio of the largest to smallest eigenvalue. We compare the condition number of the original system, given by Eq.(A2) to the preconditioned system, given by Eq. (A14) in Table A1. We see that for these parameters, the condition number can be reduced by a factor of over 10^3 by the block Jacobi preconditioning.

Table A1: Condition Numbers for Select Parameters

s	Nx=Ny	Nz	Lx=Ly	Lz	original	final	ratio
10	10	4	1	10	8.07E3	7.4	1.09E3
20	10	4	1	10	1.60E4	13.8	1.16E3
100	10	4	1	10	8.00E4	65.	1.23E3
1000	10	4	1	10	8.00E5	641	1.24E4
1000	20	4	1	10	3.2E6	641	4.9E3
1000	20	8	1	10	3.2E6	2.56E3	1.2E3