3D Solves in NIMROD

Carl Sovinec

for the

NIMROD Team

CEMM Meeting

October 29, 2006

Pre-APS, Philadelphia, PA

Linear solver performance is historically the most important issue for NIMROD’s computational effectiveness.

- This was the case in the early days with bilinear elements and 2D matrices.
    - We tried standard approaches with block-based preconditioning.
    - Solvers that worked well on standard problems were not very effective on our (anisotropic) MHD matrices. [Recall the AZTEC comparisons.]
    - Additive Schwarz with 1D global solves over the poloidal plane outperformed others.
- With better modeling, came the need for 3D ‘matrix-free’ solves.
    - ‘Full’ continuity and anisotropic thermal conduction produce 3D systems with MHD.
    - Preconditioning over the poloidal plane alone was effective.
- Interaction with TOPS led us to SuperLU and modern sparse parallel direct solves, in general.
    - Direct solves handle increasing condition numbers arising from high-order polynomial bases.
    - Parallel scaling seems to limit quickly, especially when communication is ‘off-node.’
- Hall physics and the move toward peta-scale computing require new efforts.

In the nonlinear two-fluid ELM computation, preconditioner performance was the limiting factor.

- The band of unstable modes immediately produces toroidal coupling that increases in strength as the perturbation amplitude became large.
- All solves are 3D and nonsymmetric, but the magnetic advance with Hall has the worst condition number (judging by the iteration).
- With the 20×120 mesh of biquintic elements and 43 Fourier components, the solves have algebraic vectors as large as 7.5×10^6 complex elements.
- GMRES orthogonalizes these large iterates.
    - Large iteration counts are costly--120 vectors kept but magnetic iterations went as large as 200. [Other solves took about 10 iterations.]
        - Convergence was not obtained with 50 vectors.
        - Once it got to high 200s, it wouldn't converge with 120.
        - It seemed too costly to keep more.
        - Time-step was severely limited (sub nano-seconds) just to improve condition numbers and not let the iteration exceed ~200 for each magnetic solve.
    - The nonlinear computation only ran about 2000 total steps. [15 segments on 43 nodes of Bassi.]

The timing output shows that the matrix-free part of the calculation is not the dominant factor.

```
Seam  time =      4.26048E+02 1.02460E+00
Seg   time =      1.29540E+03 3.11531E+00
I/O   time =      4.66289E+01 1.12138E-01
Iteration time =  3.03820E+04 7.30656E+01
Factoring time =  4.29958E+03 1.03401E+01
Line comm time =  0.00000E+00 0.00000E+00
FFT   time =      6.66910E+03 1.60385E+01
FE matrix time =  5.89415E+03 1.41748E+01
FE rhs time =     1.00635E+04 2.42017E+01
Static con time = 2.68359E+02 6.45377E-01
```

• FE rhs time is much less than iteration time.

• SLU factoring time is small, despite new matrices at each step, so SLU solve time cannot account for iteration time either.

• Orthogonalization is the culprit.

• Dan is considering a BLAS replacement for the present orthogonalization method, but better preconditioning is critical.

The Fourier representation leads to dense submatrices over the toroidal angle, but FFTs provide efficiency.

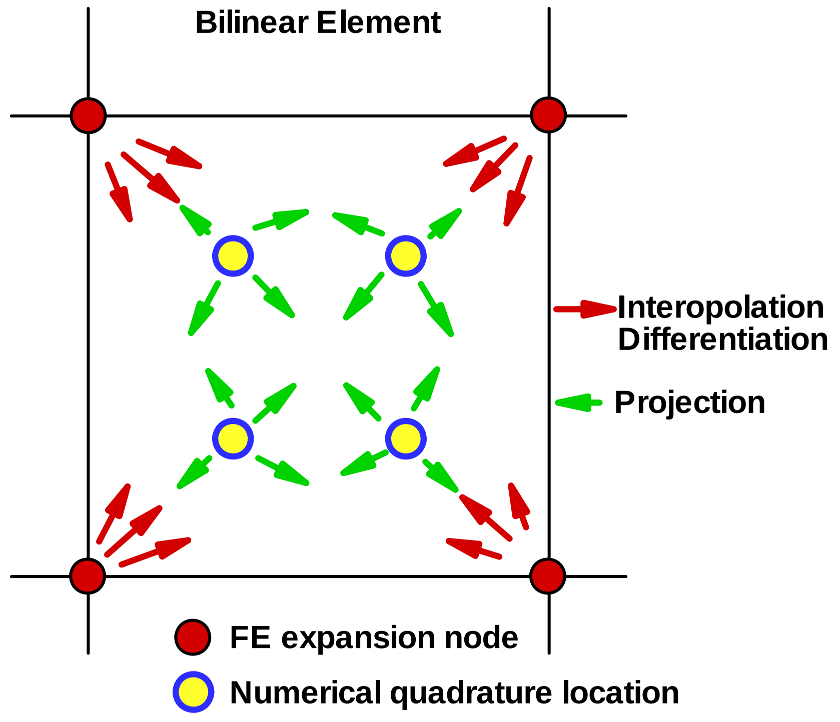
- Linear finite element operations can be broken into a set of distinct steps:

$$\begin{pmatrix}\text{Linear}\\ \text{Operator}\end{pmatrix}\cdot\begin{pmatrix}\text{coef1}\\ \text{coef2}\\ \vdots\end{pmatrix}=\begin{pmatrix}\text{Projection}\\ \text{Operation}\end{pmatrix}\begin{pmatrix}\text{Local-in-}k\\ \text{Algebra 2}\end{pmatrix}\begin{pmatrix}\text{Forward}\\ \text{FFT}\end{pmatrix}$$

$$\cdot\begin{pmatrix}\text{Local-in-}\phi\\ \text{Algebra}\end{pmatrix}\begin{pmatrix}\text{Inverse}\\ \text{FFT}\end{pmatrix}\begin{pmatrix}\text{Local-in-}k\\ \text{Algebra 1}\end{pmatrix}\begin{pmatrix}\text{Interpolation}\\ \text{Differentiaion}\end{pmatrix}\cdot\begin{pmatrix}\text{coef1}\\ \text{coef2}\\ \vdots\end{pmatrix}$$

Schematic of finite element used in the poloidal plane.

Finite Fourier series is used for the perpendicular (toroidal) direction.



Our present preconditioning strategy uses just one possible simplification to produce approximations to the matrices.

• Omitting toroidal coupling, the matrix is composed of the following steps that lead to sparse matrices.

$$\begin{pmatrix}\text{Linear}\\ \text{Appox 1}\end{pmatrix} = \begin{pmatrix}\text{Projection}\\ \text{Operation}\end{pmatrix}\begin{pmatrix}\text{Local - in - k}\\ \text{Algebra}\end{pmatrix}\begin{pmatrix}\text{Interpolation}\\ \text{Differentiaion}\end{pmatrix}$$

• Another approximation omits poloidal operations:

$$\begin{pmatrix}\text{Linear}\\ \text{Appox 2}\end{pmatrix} \cong \begin{pmatrix}\text{Local - in - }k\\ \text{Algebra 2}\end{pmatrix}\begin{pmatrix}\text{Forward}\\ \text{FFT}\end{pmatrix}\begin{pmatrix}\text{Local - in - }\phi\\ \text{Algebra}\end{pmatrix}\begin{pmatrix}\text{Inverse}\\ \text{FFT}\end{pmatrix}\begin{pmatrix}\text{Local - in - }k\\ \text{Algebra 1}\end{pmatrix}$$

• Finding a set of inverse operations would be straightforward:

$$\begin{pmatrix}\text{Linear}\\ \text{Appox 2}\end{pmatrix}^{-1} \cong \begin{pmatrix}\text{Local - in - }k\\ \text{Algebra 1}\end{pmatrix}^{-1}\begin{pmatrix}\text{Forward}\\ \text{FFT}\end{pmatrix}\begin{pmatrix}\text{Local - in - }\phi\\ \text{Algebra}\end{pmatrix}^{-1}\begin{pmatrix}\text{Inverse}\\ \text{FFT}\end{pmatrix}\begin{pmatrix}\text{Local - in - }k\\ \text{Algebra 2}\end{pmatrix}^{-1}$$

• A full poloidal/toroidal preconditioning step could then be done by additive or multiplicative Schwarz:

$$\begin{pmatrix}\text{Precon}\\ \text{Add}\end{pmatrix}^{-1} = \begin{pmatrix}\text{Linear}\\ \text{Appox 1}\end{pmatrix}^{-1} + \begin{pmatrix}\text{Linear}\\ \text{Appox 2}\end{pmatrix}^{-1} \qquad \begin{pmatrix}\text{Precon}\\ \text{Mult}\end{pmatrix}^{-1} = \begin{pmatrix}\text{Linear}\\ \text{Appox 1}\end{pmatrix}^{-1} \cdot \begin{pmatrix}\text{Linear}\\ \text{Appox 2}\end{pmatrix}^{-1}$$

The brainstorming session in Seattle produced a number of ideas for improving the scalability of our preconditioning.

- Multi-level with SuperLU (possibly only) at the highest level.
- Link to a multi-grid package such as HYPRE (via PETSc?).
- Give SuperLU only numerically large matrix elements, like a threshold-incomplete factorization strategy.
- Skip static condensation during preconditioning so that SuperLU can better overlap computation with communication.
- Perform alternating-direction-implicit operations with the toroidal direction.