Parallel Structure and Performance of the NIMROD Code

Perfomance of the NIMROD Code

Parallel Structure and
ACKNOWLEDGMENTS

• DoE MICS Office supported SNL work on solvers and parallelization for the NIMROD project.
• Work at LLNL for DOE under Contract W7405-ENG-48.
• Computer time provided by NERSC, LLNL, and Univ. Texas, Austin.
• DoE MICS Office supported SNL work on
ABSTRACT

The computationally intensive physics kernel is written so as to run without modification on single processors or any platform that supports a message-passing style of programming. This includes workstations, traditional vector supercomputers, and the CRAY C90 at NERSC.

been run on the T3D at LLNL, T3E's at NERSC and UT Austin, and the C90 at NERSC.

function of (1) the number of blocks used to grid the poloidal plane and (2) the number of processors used. The timings have

formulation described above. We present timings that illustrate the performance and convergence of both techniques as a

iterative solvers perform their computations on a block-wise basis and thus work in parallel using the block-connection

NIMROD. The iterative method uses simple diagonal (Jacobi) scaling as a matrix preconditioner. A second method (currently

routine and an iterative solver using conjugate gradient techniques have been implemented and tested in parallel for

NIMROD. In this block-connection operation, the exchange of values between grids is simplified, which enables

describe our method of pre-compiling the communication pattern and then exchanging values asynchronously, which enables

communication that is then required is to exchange values for block-edge or block-corner grid points shared by other

assigning one or more blocks (with their associated toroidal modes) to each processor in parallel, the only interprocessor

block boundaries. Within these constraints, the general geometries can be gridded, and parallelization is achieved by

discretization is pseudo-spectral. Within a single poloidal block the grid is topologically regular to enable the usual 2-D schemes

solution to enable efficient parallelization and high-fidelity performance on several parallel machines including the new CRAY

essentially all current-generation massively parallel machines. In this poster, we describe how the NIMROD kernel is

The computationally intensive physics kernel is written so as to run without modification on single processors or any platform


The NIMROD Code Development Project

Outline

• The NIMROD Code Development Project
• Parallel Processing Considerations
  – Graphical User Interface
  – Grid and Finite Elements
  – Physics Kernel
• Parallel Processing Results
• Future
THE NIMROD CODE SYSTEM

PRE-PROCESSOR

NIMROD PHYSICS KERNEL

GRID GENERATOR

TRANSPORT CODE

LINEAR STABILITY CODE

EXPERIMENTAL DATA

USER INTERFACE

GRAPHICAL USER INTERFACE

NIMROD RESTART FILE

USER
Project Communication is Based on
Web Pages
Conference Calls
Meetings (every 3 months)
Open Discussion Project
Non-Ideal MHD with Rotation
Faraday and Ampere:
\[
\frac{\mu_0 I}{c} \times \Delta = \mathbf{B} \times \Delta
\]
\[
\mathbf{E} \times \Delta^* = \frac{\mu_0 I}{c} \frac{\sigma}{\kappa} \frac{\mathbf{B}}{\mathbf{e} \times \Delta}
\]

NichR0D Physics Kernel

**Continuity:**
\[
\frac{\partial}{\partial t} \rho + \nabla \cdot \mathbf{v} = 0
\]

**Constitutive Equations:**
\[
\mu \nabla \times \mathbf{E} + \frac{\sigma}{\kappa} \frac{\mathbf{B}}{\mathbf{e} \times \Delta} = \frac{\mu_0 I}{c} \frac{\sigma}{\kappa} \frac{\mathbf{B}}{\mathbf{e} \times \Delta}
\]

**Thermodynamics:**
\[
\frac{\partial}{\partial t} \rho \mathbf{v} + \nabla \cdot \mathbf{p} \mathbf{v} = \nabla \cdot \mathbf{q} + \mathbf{R}
\]

**Equation of motion for two fluid system (\( \alpha = e, i \); \( \mathbf{n} = Z \mathbf{n}_i \)):**
\[
\frac{\partial}{\partial t} \rho \mathbf{v} + \nabla \cdot \mathbf{p} \mathbf{v} = \nabla \cdot \mathbf{q} + \mathbf{R}
\]
\( \mathbf{J} (\lambda - 1) \frac{\partial \mathbf{w}}{\partial \vartheta_\lambda} - \mathbf{w} \frac{\partial \mathbf{W}}{\partial \vartheta_\lambda} = \frac{\partial \mathbf{w}}{\partial \tau} \int \frac{\mathbf{w}}{\vartheta_\lambda \alpha} \mathbf{Z} \)

A useful expression:

\( \vartheta_\lambda = \lambda \mathbf{b} \quad \vartheta_\mu = \mu \mathbf{b} \quad \vartheta_\lambda + \vartheta_\mu = \lambda \mathbf{b} \)

\[ \mathbf{b} \times \left( \mu \mathbf{J} + \mu \mathbf{\nabla} \mathbf{f} \right) \frac{\partial \mathbf{c}}{\partial \tau} = \frac{\mu \nabla}{\mu \nabla} \]

\[ \frac{\partial \mathbf{w}}{\partial \tau} \int \frac{\mathbf{w}}{\vartheta_\mu} \mathbf{b} \times \frac{\partial \mathbf{c}}{\partial \mathbf{f}} (1 - \vartheta_\mathbf{f}) + \\mathbf{E} \frac{\partial \mathbf{f}}{\partial \mathbf{w}} = \frac{\partial \mathbf{w}}{\partial \tau} \int \frac{\mathbf{w}}{\vartheta_\mu} \mathbf{b} \times \frac{\partial \mathbf{c}}{\partial \mathbf{f}} + \frac{\mu \nabla}{\mu \nabla} \]

\[ \frac{\partial \mathbf{w}}{\partial \tau} \int \frac{\mathbf{w}}{\vartheta_\mu} \mathbf{b} \times \frac{\partial \mathbf{c}}{\partial \mathbf{f}} = \vartheta_\mathbf{w} \quad 1 \leq \vartheta_\mathbf{f} > 0 \quad \frac{\partial \nabla}{\partial \mathbf{w}} + \frac{\partial \mathbf{f}}{\partial \mathbf{w}} = \frac{\partial \mathbf{w}}{\partial \mathbf{w}} \frac{\partial \mathbf{w}}{\partial \mathbf{w}} \]

Time differencing, sum over species:

\[ \mathbf{b} \times \int \frac{\mathbf{c}}{\partial \tau} + \mathbf{E} \frac{\mathbf{u}}{\partial \mathbf{w}} = \frac{\mathbf{I} \mathbf{e}}{\mathbf{w} \mathbf{e}} \quad \mathbf{E} \frac{\partial \mathbf{w}}{\partial \mathbf{w}} = \int \frac{\mathbf{w}}{\partial \mathbf{w}} \times \frac{\mathbf{c}}{\partial \mathbf{w}} - \frac{\mathbf{I} \mathbf{e}}{\mathbf{w} \mathbf{e}} \]

Low frequency \( \leftarrow \) Ignore displacement current

Cold plasma \( \leftarrow \) const.

Implicit Field Equation
\[ u_{n+1} \mathbf{B}_n = \mathbf{B} \nabla \nabla \times \mathbf{E} = \mathbf{E} \times \mathbf{B} + \mathbf{B} \nabla \times \nabla \times \mathbf{A} \times \mathbf{Z} \times \mathbf{E} \times \mathbf{A} \]

Combine with Maxwell\(\Leftrightarrow\) Implicit field equation

\[
\begin{cases}
\mathbf{J} \times \mathbf{B} - \frac{c}{\sqrt{\gamma}} \frac{\nabla \varphi + \frac{1}{\gamma - 1} \left[ \mathbf{J} \mathbf{J} - \mathbf{J} \right] \left( \frac{c}{\sqrt{\gamma}} \frac{\nabla \varphi}{\gamma - 1} + \frac{1}{\gamma - 1} \frac{\nabla \varphi}{\gamma - 1} \right] \left( \frac{c}{\sqrt{\gamma}} \frac{\nabla \varphi}{\gamma - 1} + \frac{1}{\gamma - 1} \frac{\nabla \varphi}{\gamma - 1} \right) + \mathbf{J} \mathbf{J} \right] \mathbf{p} \mathbf{c} \mathbf{d}}{\gamma - 1}
\end{cases}
\]

\[ \mathbf{J} \nabla \times \mathbf{E} = \mathbf{E} \times \mathbf{J} \]

Impedance tensor, \(\mathbf{Z} = \frac{w}{\sqrt{\gamma} \mathbf{J}}\)

\[ \mathbf{J} \nabla \cdot \mathbf{Z} \mathbf{J} = \mathbf{E} \times \mathbf{E} \]

Solve for \(\mathbf{E}\) (generalized Ohm's law) with \(\mathbf{E} = \mathbf{E} + \mathbf{E}^\text{im}\)

\[ \mathbf{Z} / \mathbf{J} w + \mathbf{m} = \mathbf{m} \]

Implicit Field Equation (cont.)
Parallel Processing focused on CG system
CG matrix inversion for symmetric system
Poloidal Spatial discretization by finite elements

SUMMARY

Here is symmetric positive definite „semi-implicit operator
\[ \mathbf{D} \cdot \mathbf{A} - \mathbf{E} \times \Delta \mathbf{C} = (\mathbf{D} - \mathbf{E} + \mathbf{I}) \cdot (\mathbf{A} + \mathbf{S}) \]

Can invert related symmetric system:
\[ \{[(\mathbf{D} \times \Delta) \cdot \mathbf{S} \cdot (\mathbf{D} \times \Delta)] \mathbf{C} + \mathbf{D} \mathbf{I} \mathbf{C} \} \mathbf{x} \int_{\frac{\pi}{4}} = \mathbf{I} \quad \text{for} \quad 0 = \mathbf{S} \]

Must invert operator
\[ \mathbf{I} \times \mathbf{q} \left( \frac{\mathbf{C}}{\mathbf{a}} \right) \mathbf{I} \mathbf{C} + \mathbf{I} = \mathbf{a} \quad \mathbf{A} + \mathbf{S} = \mathbf{a} \quad \mathbf{Z} + \mathbf{S} \mathbf{Z} = \mathbf{Z} \]
Nimrod Grid

- Spectral in Toroidal Direction
- Unstructured Blocks of Structured Quadrilaterals in Poloidal Plane
  - Each Unstructured Block may be a Single Triangle
  - Singularity at magnetic axis
  - Overlying Quadrilateral Grid (1st try)
  - Triangle Elements (Pie-slices) (2nd try)

- Outer Boundary can conform to Real Machine Geometry
- Nearly Flux Surface Conforming within Separatrix
- Singularity at Magnetic Axis
- Overlying Quadrilateral Grid (1st try)
- Conforming Blocks (2nd try)
- Single Triangle (Patching of Non-Conforming Blocks)
Overlying Quadrilateral Grid led to unphysical results at corners.

Nimrod Grid

Overlying Quadrilateral Grid near Magnetic Axis (Avoid Singularity)
Triangle Block Patch appears to fix problem
Patching Triangles also allows for variation in Grid Resolution (to be implemented).

Patching Triangles also...
NIMROD GUI

- Written in tcl/tk
- Controls interaction between user and NIMROD
  - Problem setup, Dynamic, Diagnostics
  - Runtime, Postprocessors, Preprocessors
- Controls interaction between NIMROD and other user codes, pre/post processors
- Written in tcl/tk
GUI Configuration
NIMROD GUI
STEPS of PARALLEL CODE DEVELOPMENT

Code design to avoid bottle necks
- Block domain decomposition of 2D toroidal mesh
- Blocks seam together
- Blocks or multiple blocks assigned to processors
- Block domain decomposition of 2D toroidal mesh
- Code design to avoid bottle necks

Multiple Processor Optimization
- Overlap communication and computation
- FFT's in third dimension restricted to block
- Blocks or multiple blocks assigned to processors
- Blocks seam together
- Communication between blocks via Message Passing Interface (MPI)

Single Processor Optimization

Iterative Solver Design Issues
- Overlap communication and computation
- Multiple Processor Optimization

- Overlap communication and computation
INHERENT PARALLELISM

in NIMROD

• Each processor owns 1 or more “blocks” and their associated “seams”.

• Computations can be done on each block independently.

• Only connection/communication with other processors is via “seams”.

in NIMROD
NIMROD Parallel Coding Choices

- Message-passing parallelism with F90/MP
- F90 provides dynamic memory, data structures
- MPI provides portability to any machine with a F90 compiler
- MPI allows irregular, asynchronous communication
- Same code will run on workstation, Cray C90, or Cray T3D/E, IBM SP2

Future: benchmark vs. loop parallel (DEC cluster)

- Workstation Clusters
- Cray T3D/E, IBM SP2

Parallel Platforms:

Message-passing parallelism with F90/MP
Grid Structure of NIMROD

- NIMROD grid is a general collection of joined sub-blocks mapped to the poloidal plane.
- Edge points of adjacent blocks join exactly.

Grid Structure of NIMROD
Sub-blocking with associated seams.

Each edge point has "image" points in other blocks/seams.

1-d seams

Image pts

Multi-block grid
If 2 adjacent blocks are on different processors, a data exchange is needed to complete the integration.

FE integration stencil for block interior and across block and/or processor boundaries.

Across boundaries

Interior
Parallel Design

- Assignment of blocks to processors (load-balancing)
- Setup of data structures for parallel seaming
- Knit seams between blocks
- Dot-products for CG-solver
  - used in matrix-vector multiply of CG-solver
  - used in explicit time stepper
  - (balancing)
- Assignment of blocks to processors
Serial Seam Connection

1) Copy from block-edge grid points to seams
2) Loop over images of each seam point, sum image values to block-edge grid points
3) Apply external boundary conditions.
SERIAL VERSION: 1) Copy from block-edge grid points to my seams.
2) For seam points where I own both image pairs, sum image values to my block-edge grid points.
3) Receive incoming image data from other processors sum it to my block-edge grid points.
4) Apply external boundary conditions.
5) Apply external boundary conditions.

PARALLEL VERSION: 1) Send my seam data to neighboring processors.
2) For seam points where I own both image pairs, sum image values to block-edge grid points.
3) Apply external boundary conditions.
4) Apply external boundary conditions.
5) Copy from block-edge grid points to seams.
Attributes of Parallel Seam Connection routine

- Uses asynchronous communication in irregular pattern of connectivity between processors.
- Overlaps communication and computation (steps 2-4).
- Pre-computes data structures to optimally pack/unpack messages being exchanged with other processors.
- Fast!
  - Seam communication is only small fraction of block computation time.
Timing Results for Parallel Seam Connection on T3E

• 1.02 million grid cells, 174 blocks, 51200 seam points, 3 values/grid-cell

• CPU seconds for 1 seam-operation:

<table>
<thead>
<tr>
<th>Processes</th>
<th>1</th>
<th>2</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time</td>
<td>0.024</td>
<td>0.033</td>
<td>0.081</td>
<td>0.12</td>
<td>0.25</td>
<td>0.64</td>
</tr>
</tbody>
</table>

• Scales roughly linearly with size of grid and number of processors

1.02 million grid cells, 174 blocks, 51200 seam points, 3 values/grid-cell

• Timing Results for Parallel Seam Connection on T3E
Timing Results for Explicit Nimrod T3D Calculation

- CPU seconds for 200 timesteps on the T3D shows excellent scalability as problem size increases.

<table>
<thead>
<tr>
<th>Blocks/Cell</th>
<th>1 PE</th>
<th>2 PEs</th>
<th>4 PEs</th>
<th>8 PEs</th>
<th>16 PEs</th>
<th>32 PEs</th>
</tr>
</thead>
<tbody>
<tr>
<td>198.7</td>
<td>206.2</td>
<td>400.3</td>
<td>790.8</td>
<td>1531.8</td>
<td>95.2</td>
<td></td>
</tr>
<tr>
<td>94.7</td>
<td>101.3</td>
<td>48.4</td>
<td>95</td>
<td>759</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12.5</td>
<td>101.3</td>
<td>47.3</td>
<td>759</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Blocks are 10x10, Cells are poloidal cells (same real time as 50 implicit time steps)

Experiments performed on the T3D show excellent scalability as problem size increases.
Timing Results for Explicit Nimrod T3E Calculation

- CPU seconds for 200 timesteps on the T3E shows excellent scalability as problem size increases.

<table>
<thead>
<tr>
<th>Blocks/Cell</th>
<th>1PEC90</th>
<th>32 PEs</th>
<th>16 PEs</th>
<th>8 PEs</th>
<th>4 PEs</th>
<th>2 PEs</th>
<th>1 PE</th>
</tr>
</thead>
<tbody>
<tr>
<td>198.7</td>
<td>238.8</td>
<td>62.3</td>
<td>117.4</td>
<td>11.7</td>
<td>11.9</td>
<td>11.2</td>
<td>11.2</td>
</tr>
<tr>
<td>16.0</td>
<td>3.0</td>
<td>30.8</td>
<td>8.1</td>
<td>1.4</td>
<td>1.2</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>8.1</td>
<td>58.3</td>
<td>29.2</td>
<td>7.58</td>
<td>1.4</td>
<td>1.2</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>32 PEs</td>
<td>112.1</td>
<td>58.3</td>
<td>30.8</td>
<td>16.0</td>
<td>198.7</td>
<td>111.0</td>
<td>256.2</td>
</tr>
<tr>
<td>16 PEs</td>
<td>256.0</td>
<td>64.0</td>
<td>56.8</td>
<td>14.2</td>
<td>136.8</td>
<td>11.0</td>
<td>64.0</td>
</tr>
<tr>
<td>8 PEs</td>
<td>64.0</td>
<td>56.8</td>
<td>28.8</td>
<td>14.2</td>
<td>136.8</td>
<td>11.0</td>
<td>64.0</td>
</tr>
<tr>
<td>4 PEs</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>2 PEs</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>1 PE</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>

(Same parameters as previous table for T3D)
Timing Results for Implicit Nimrod T3D Calculation

- CG solver with diagonal preconditioning
- 50 timesteps, roughly 30-40 CG iterations per step
- Time proportional to iterations
- Preconditioning methods require more study

<table>
<thead>
<tr>
<th>Blocks/Cell</th>
<th>1 PE</th>
<th>2 PE</th>
<th>4 PE</th>
<th>8 PE</th>
<th>16 PE</th>
<th>32 PE</th>
<th>N-iter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>4/400</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16/1600</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>64/6400</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>256/25,600</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1024/102,400</td>
</tr>
</tbody>
</table>

C more study

Preconditioning methods for CG solver require

- Time proportional to iterations
- Roughly 30-40 CG iterations per step
- CG solver with diagonal preconditioning
Timing Results for Implicit Nimrod T3E Calculation

- CG solver with diagonal preconditioning
  - 50 timesteps, roughly 30-40 CG iterations per step
    - time proportional to iterations
  - Preconditioning methods for CG solver require more study

<table>
<thead>
<tr>
<th>Blocks/Cell</th>
<th>1 PE</th>
<th>2 PE</th>
<th>4 PE</th>
<th>8 PE</th>
<th>16 PE</th>
<th>32 PE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1PEC90</td>
<td>1024/1024</td>
<td>256/256</td>
<td>64/64</td>
<td>16/16</td>
<td>4/4</td>
<td></td>
</tr>
<tr>
<td>4/400</td>
<td>449.7</td>
<td>242.8</td>
<td>125.0</td>
<td>102.8</td>
<td>95.9</td>
<td></td>
</tr>
<tr>
<td>64/6400</td>
<td>229.4</td>
<td>153.6</td>
<td>87.6</td>
<td>49.8</td>
<td>34.8</td>
<td></td>
</tr>
<tr>
<td>1PE</td>
<td>84/84</td>
<td>61.6</td>
<td>35.6</td>
<td>20.3</td>
<td>19.0</td>
<td></td>
</tr>
<tr>
<td>16/1600</td>
<td>261.8</td>
<td>131.1</td>
<td>67.8</td>
<td>35.6</td>
<td>20.3</td>
<td></td>
</tr>
<tr>
<td>32 PEs</td>
<td>68.2</td>
<td>35.5</td>
<td>19.0</td>
<td>55.9</td>
<td>35.5</td>
<td></td>
</tr>
<tr>
<td>4/400</td>
<td>1024/1024</td>
<td>256/256</td>
<td>64/64</td>
<td>16/16</td>
<td>4/4</td>
<td></td>
</tr>
</tbody>
</table>

More study required on preconditioning methods for CG solver.
- Time proportional to iterations
- Roughly 30-40 CG iterations per step
Performance Results Show Nearly Ideal Speed-up for Explicit Case (even for fixed problem size)
Scaled speed-up is speedup/problem size

- T3E is roughly a factor of 4 faster
  - 2X processor speed
  - chaining
  - cache effects
- Scalability is virtually linear for both machines

Comparison of T3E/D Scaled Speed-up
Parallel Conclusions

- Blockwise-design of NIMROD enables rapid message-passing
- Explicit and diagonal-preconditioned CG solver run well in parallel
- T3E outperforms T3D, but both perform well
  - Cache, Processor speed
  - F90: Great language
  - Texas Machine vs. NERSC (problem with streams?)
  - Acceptable on T3D, but performance tools need improvement
  - Good on T3E, but libraries still missing
  - Terrible compilers in general
  - Does it produce fast code? (open question)

Blockwise-design of NIMROD enables rapid message-passing
Future Parallel Work

- Implement 2nd NIMROD CG solver (block-invert preconditioner) in parallel (almost complete)
- Test convergence and performance of solvers as a function of number-of-blocks, number-of-processors
- Try new iterative solvers
- Optimize code performance
- Try new iterative solvers
- Implement 2nd NIMROD CG solver (block-invert preconditioner) in parallel (almost complete)