Parallel Structure and Performance of the NIMROD Code

Steve Plimpton

MS 1111, Sandia National Laboratories, Albuquerque, NM 87185

Alice Koniges and Paul Covello

MS L-630, Lawrence Livermore National Laboratories, Livermore, CA 94550

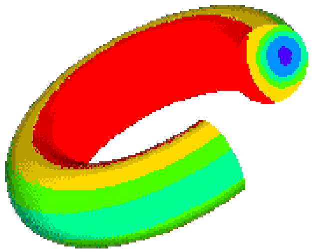
Carl Sovinec, Dan Barnes

MS K717, Los Alamos National Laboratories, Los Alamos, NM 87545

Tom Gianakon, Cadarache

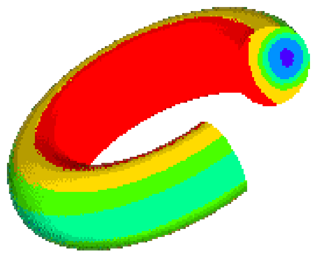
and the NIMROD TEAM

Sherwood Theory Meeting, April 1997, Madison, WI



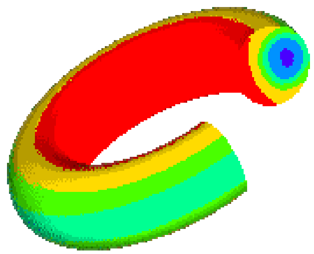
ACKNOWLEDGMENTS

- 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



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 essentially all current-generation massively parallel machines. In this poster, we describe how the NIMROD kernel is structured to enable efficient parallelization and highlight its performance on several parallel machines including the new Cray T3E at NERSC. NIMROD represents the poloidal simulation plane as a collection of adjoining grid blocks; the toroidal discretization is pseudo-spectral. Within a single poloidal block the grid is topologically regular to enable the usual 2-D stencil operations to be performed efficiently. Blocks join each other in such a way that individual grid lines are continuous across block boundaries. Within these constraints, quite general geometries can be gridded, and parallelization is achieved by assigning one or more blocks (with their associated toroidal modes) to each processor. In parallel, the only interprocessor communication that is then required is to exchange values for block-edge or block-corner grid points shared by other processors. For general block connectivity, this operation requires irregular, unstructured interprocessor communication. We describe our method of pre-computing the communication pattern and then exchanging values asynchronously, which enables this block-connection operation to execute efficiently and scalably on any number of processors. NIMROD uses implicit timestepping to model long-timescale events and thus requires a robust iterative solver. To date, an explicit time-stepping routine and an iterative solver using conjugate gradient techniques have been implemented and tested in parallel for NIMROD. The iterative method uses simple diagonal (Jacobi) scaling as a matrix preconditioner. A second method (currently under parallel development) directly inverts the portion of the matrix residing on each block as a preconditioning step. Both iterative solvers perform their computations on a block-wise basis and thus work in parallel using the block-connection formalism described above. We present timings that illustrate the performance and convergence of both techniques as a function of (1) the number of blocks used to grid the poloidal plane and (2) the number of processors used. The timings have been run on the T3D at LLNL, T3E's at NERSC and UT Austin, and the C90 at NERSC.

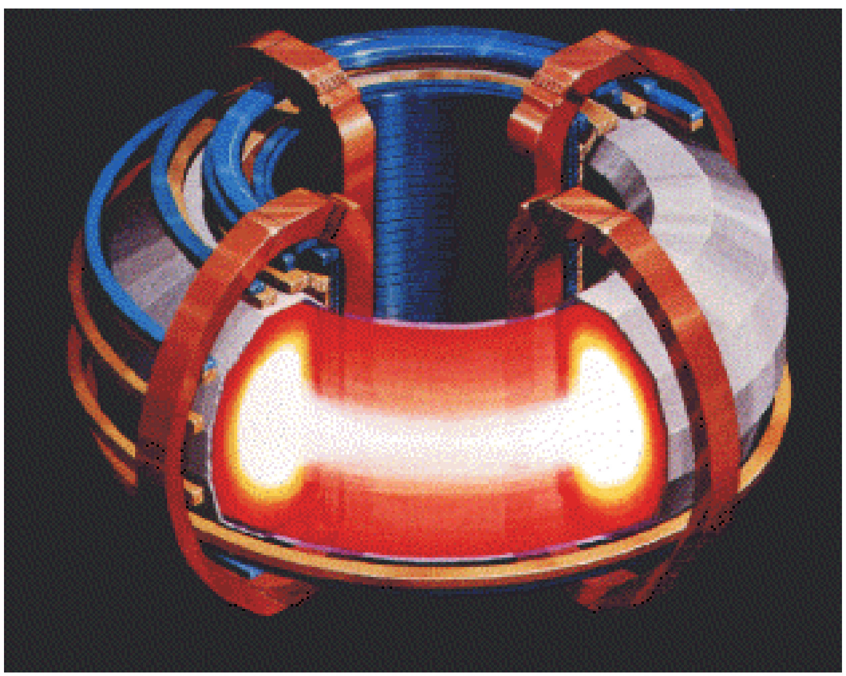


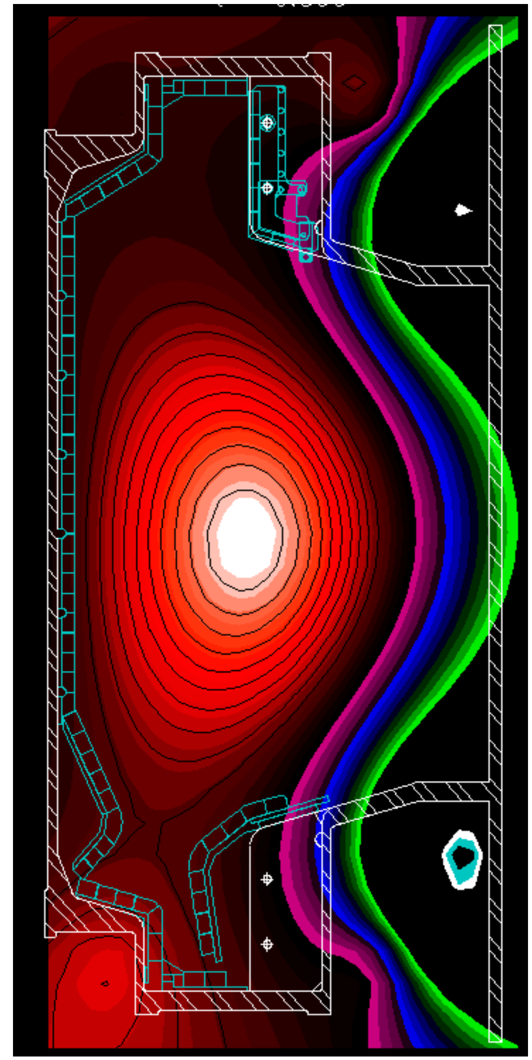
Outline

- The NIMROD Code Development Project
  - Physics Kernel
  - Grid and Finite Elements
  - Graphical User Interface
- Parallel Processing Considerations
- Parallel Processing--results
- Future

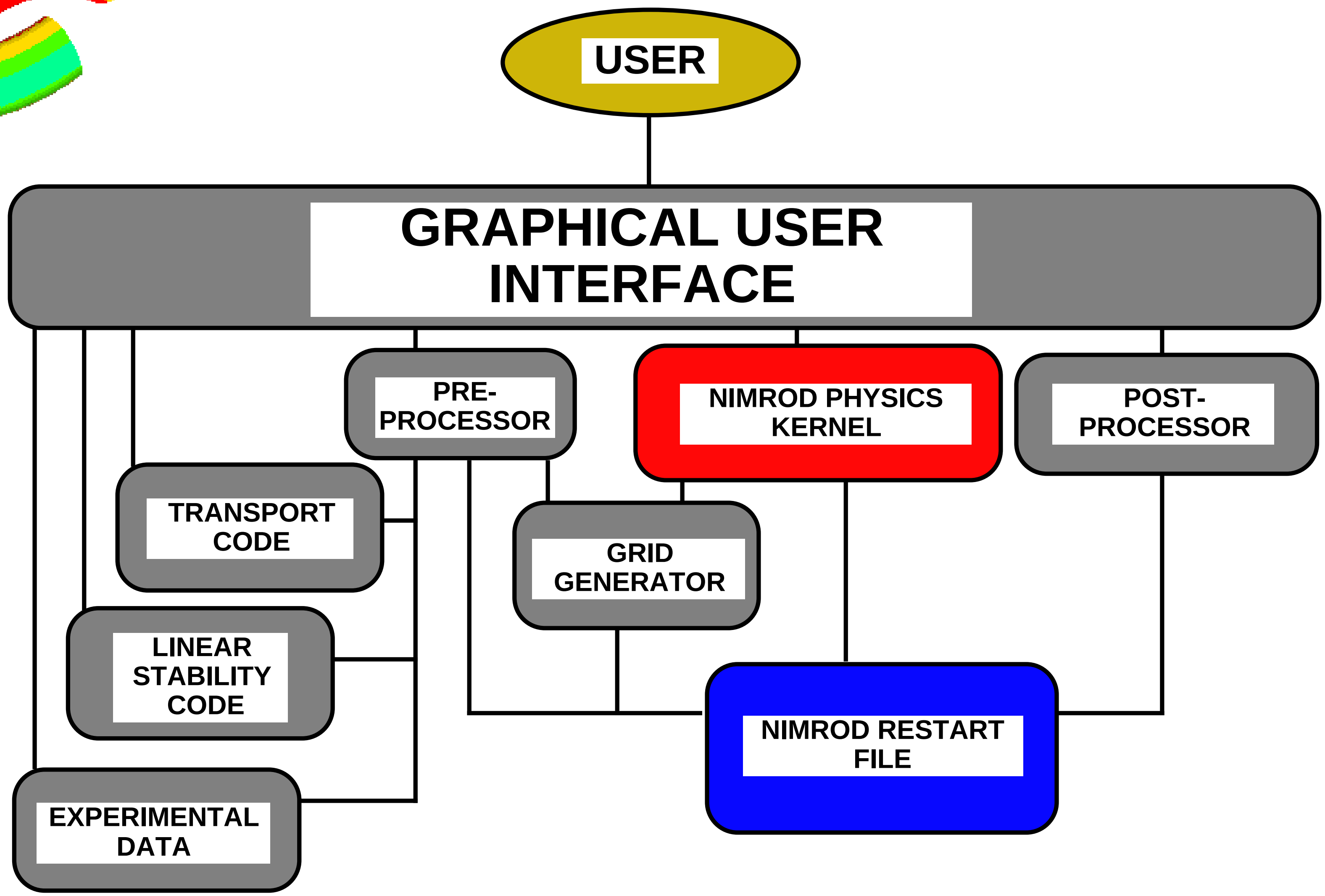
The NIMROD TEAM and THEIR QUEST

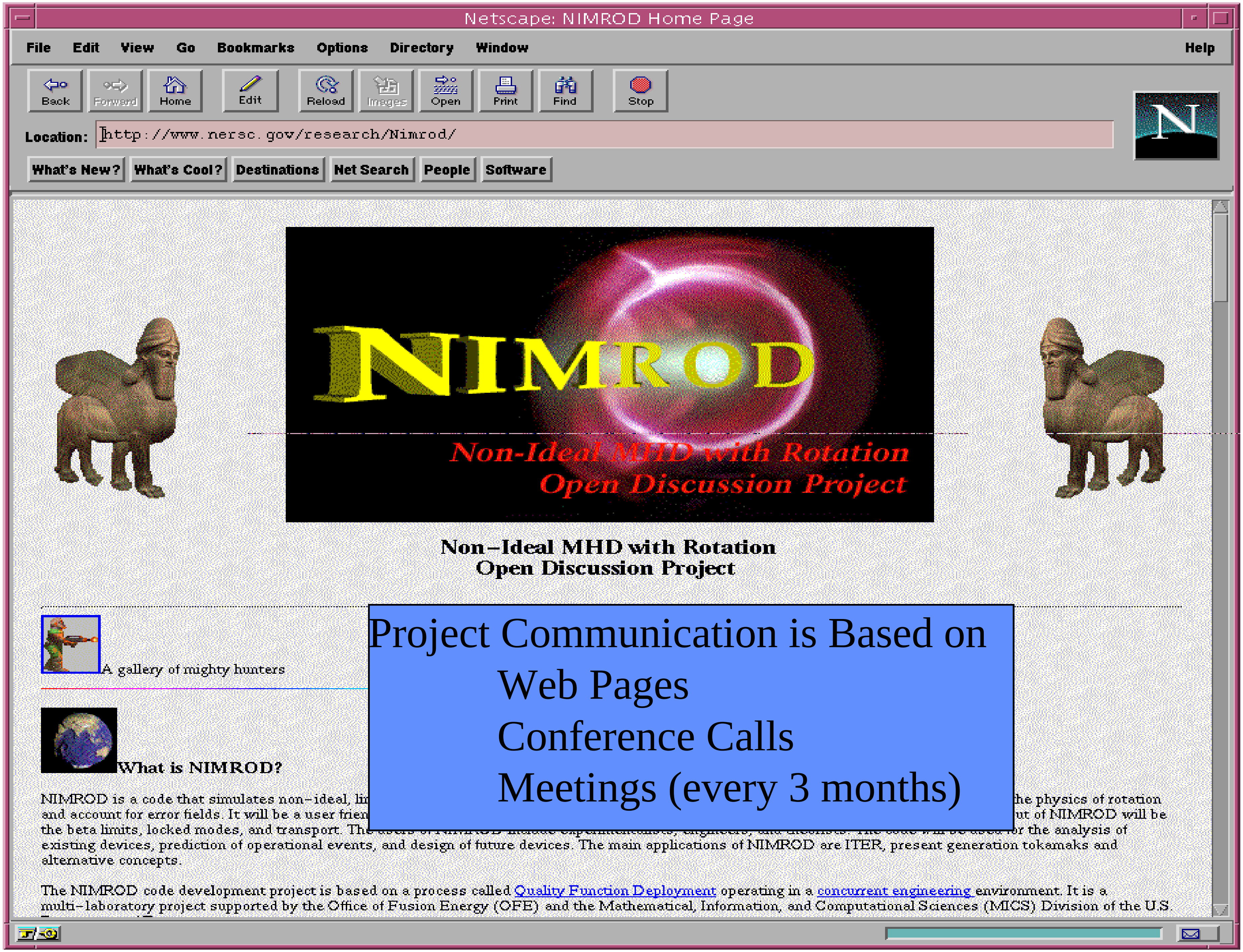
Curt Bolton	DOE/OFE
Ming Chu	GA
Sergei Galkin	Keldysh Inst.
Tom Gianakon	Cadarache
Alan Glasser	LANL
Harsh Karandakar	SAIC
Alice Koniges	LLNL
Scott Kruger	U. Wisc.
Rick Nebel	LANL
Steve Plimpton	SNL
Nina Popova	Moscow
Olivier Sauter	ITER
Dalton Schnack	SAIC
Carl Sovinec	LANL
Alfonso Tarditi	SAIC
Alan Turnbull	GA





THE NIMROD CODE SYSTEM

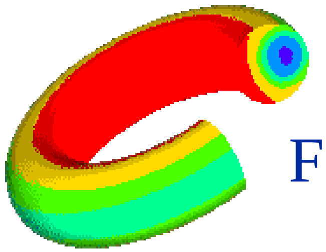




Project Communication is Based on

- Web Pages
- Conference Calls
- Meetings (every 3 months)

NIMROD Physics Kernel



Faraday and Ampere: $\frac{1}{c}\frac{\partial \mathbf{B}}{\partial t} = -\nabla \times \mathbf{E}$ $\quad \nabla \times \mathbf{B} = \frac{4\pi}{c}\mathbf{J} + \frac{1}{c}\frac{\partial \mathbf{E}}{\partial t}$

Equation of motion for two fluid system (α = e,i; $n_e = Zn_i = n$):

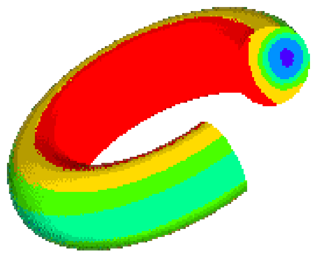
$$m_\alpha n_\alpha \left(\frac{\partial \mathbf{v}_\alpha}{\partial t} + \mathbf{v}_\alpha \cdot \nabla \mathbf{v}_\alpha \right) = -\nabla \cdot \Pi_\alpha + q_\alpha n_\alpha \left(\mathbf{E} + \frac{1}{c}\mathbf{v}_\alpha \times \mathbf{B} \right) + \sum_\beta \mathbf{R}_{\alpha\beta} + \mathbf{S}_\alpha^m$$

Thermodynamics: $\frac{\partial p_\alpha}{\partial t} + \mathbf{v}_\alpha \cdot \nabla p_\alpha = -\frac{3}{2} p_\alpha \nabla \cdot \mathbf{v}_\alpha - \Pi_\alpha : \nabla \mathbf{v}_\alpha - \nabla \cdot \mathbf{q}_\alpha + \mathbf{Q}_\alpha$

Continuity: $\frac{\partial n_\alpha}{\partial t} = -\nabla \cdot (n_\alpha \mathbf{v}_\alpha) + S_\alpha^n$

Constitutive equations:

$$p_\alpha = n_\alpha k_B T_\alpha \qquad \mathbf{J} = \sum_\alpha \mathbf{J}_\alpha = \sum_\alpha n_\alpha q_\alpha \mathbf{v}_\alpha \qquad \mathbf{M} = \sum_\alpha \mathbf{M}_\alpha = \sum_\alpha m_\alpha \mathbf{J}_\alpha / q_\alpha$$



Implicit Field Equation

Cold plasma $\Rightarrow$ $p_\alpha = 0, \qquad n = const.$

Low frequency $\Rightarrow$ Ignore displacement current

$$\frac{\partial \mathbf{J}_\alpha}{\partial t} + \frac{q_\alpha}{m_\alpha c}\mathbf{B}\times\mathbf{J}_\alpha = \frac{nq_\alpha^2}{m_\alpha}\mathbf{E} \qquad \frac{\partial \mathbf{M}_\alpha}{\partial t} = q_\alpha n\mathbf{E} + \frac{1}{c}\mathbf{J}_\alpha \times \mathbf{B}$$

Time differencing, sum over species:

$$\omega_p^2 = \omega_e^2 + \omega_i^2 \qquad \mathbf{J}_\alpha^{n+1} = \mathbf{J}_\alpha^n + \Delta\mathbf{J}_\alpha, \quad 0 < f_\Omega \le 1 \qquad \omega_\alpha^2 = 4\pi n q_\alpha^2 / m_\alpha$$

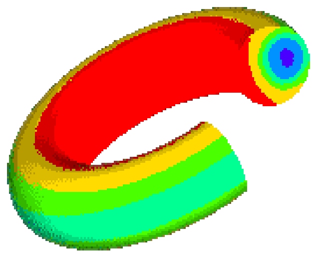
$$\frac{\Delta\mathbf{J}}{\Delta t} + f_\Omega \frac{1}{c}\mathbf{B}\times\sum_\alpha \frac{q_\alpha}{m_\alpha}\mathbf{J}_\alpha^{n+1} = \frac{\omega_p^2}{4\pi}\mathbf{E} + \left(f_\Omega - 1\right)\frac{1}{c}\mathbf{B}\times\sum_\alpha \frac{q_\alpha}{m_\alpha}\mathbf{J}_\alpha^n$$

$$\frac{\Delta\mathbf{M}}{\Delta t} = \frac{1}{c}\left(f_\Omega \Delta\mathbf{J} + \mathbf{J}^n\right)\times\mathbf{B}$$

$$\nu = Z m_e / m_i, \qquad q_e = -e, \qquad q_i = Ze$$

A useful expression:

$$\sum_\alpha \frac{q_\alpha}{m_\alpha}\mathbf{J}_\alpha = \frac{Ze^2}{m_i m_e}\mathbf{M} - \frac{e}{m_e}(1-\nu)\mathbf{J}$$



Implicit Field Equation (cont.)

$$\rho = mn, \quad m = m_e + m_i / Z$$

Solve for $\mathbf{E}$ (generalized Ohm's law) with $\mathbf{E} = \mathbf{E}_{im} + \mathbf{E}_{ex}$

$$\mathbf{E}_{im} = \underbrace{\frac{4\pi}{\omega_p^2}\frac{\Delta\mathbf{J}}{\Delta t}}_{Inertia} + \underbrace{\frac{f_\Omega^2 \Delta t}{\rho c^2}\left(B^2\mathbf{I} - \mathbf{BB}\right)\cdot\Delta\mathbf{J}}_{MHD} - \underbrace{\frac{1-\nu}{1+\nu}\frac{f_\Omega}{nec}\mathbf{B}\times\Delta\mathbf{J}}_{Hall}$$

$$\mathbf{E}_{ex} = \frac{1}{\rho c}\left[\mathbf{B}\times\mathbf{M}^n + \frac{f_\Omega \Delta t}{c}\left(B^2\mathbf{I} - \mathbf{BB}\right)\cdot\mathbf{J}^n\right] - \frac{1-\nu}{1+\nu}\frac{1}{nec}\mathbf{B}\times\mathbf{J}^n$$

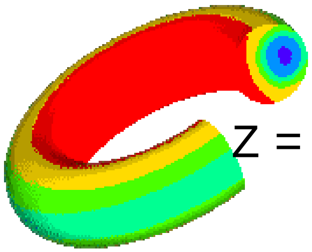
Impedance tensor, $\mathbf{E}_{im} = (4\pi / c)\mathsf{Z}\cdot\Delta\mathbf{J}$:

$$\mathsf{Z} = c\Delta t\left\{\frac{1}{\left(\omega_p \Delta t\right)^2}\hat{\mathbf{b}}\hat{\mathbf{b}} + \left[\frac{1}{\left(\omega_p \Delta t\right)^2} + f_\Omega^2\left(\frac{V_A}{c}\right)^2\right]\left(\mathbf{I} - \hat{\mathbf{b}}\hat{\mathbf{b}}\right) - \frac{1-\nu}{1+\nu}\frac{f_\Omega}{\Omega\Delta t}\left(\frac{V_A}{c}\right)^2\hat{\mathbf{b}}\times\mathbf{I}\right\}$$

Combine with Maxwell $\Rightarrow$ Implicit field equation

$$c\Delta t\nabla\times\mathsf{Z}\cdot\nabla\times\Delta\mathbf{B} + \Delta\mathbf{B} = -c\Delta t\nabla\times\mathbf{E}_{ex}$$

$$\Delta\mathbf{B} = \mathbf{B}^{n+1} - \mathbf{B}^n$$



Must invert operator $\Omega = \mathbf{I} + c\Delta t \nabla \times \mathrm{Z} \cdot \nabla \times \mathbf{I}$

$$\mathrm{Z} = \mathrm{Z}_S + \mathrm{Z}_A \qquad \Omega = \mathbf{S} + \mathbf{A} \qquad \mathrm{Z}_A = c\Delta t \frac{1-\nu}{1+\nu} \frac{f_\Omega}{\Omega \Delta t} \left(\frac{V_A}{c} \right)^2 \hat{\mathbf{b}} \times \mathbf{I}$$

$$\delta I = 0, \qquad I = \frac{1}{2} \int d\mathbf{x} \left\{ |\Delta \mathbf{B}|^2 + c\Delta t \left[(\nabla \times \Delta \mathbf{B}) \cdot \mathbf{S} \cdot (\nabla \times \Delta \mathbf{B}) \right] \right\}$$

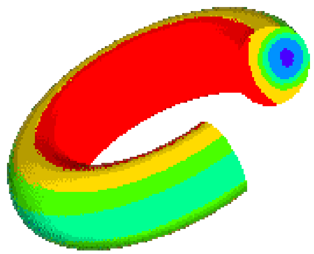
Can invert related symmetric system:

$$\left(\mathbf{S} + \mathbf{S}_A \right) \cdot \left(\Delta \mathbf{B}^{l+1} - \Delta \mathbf{B}^{l} \right) = -c\Delta t \nabla \times \mathbf{E}_{ex} - \Omega \cdot \Delta \mathbf{B}^{l}$$

Where $\mathbf{S}_A$ is symmetric positive definite "semi-implicit" operator

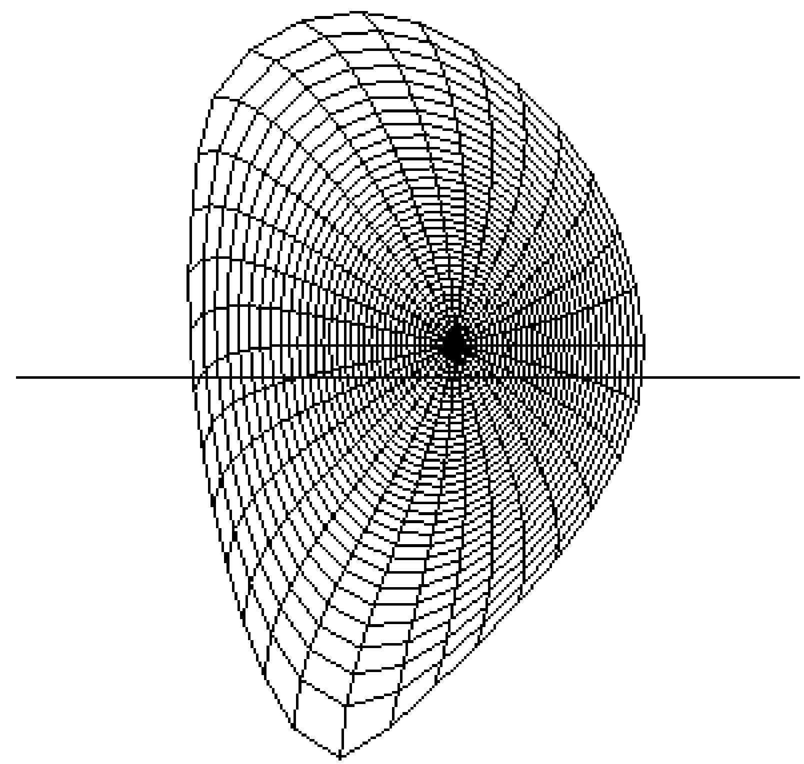
SUMMARY

Poloidal Spatial discretization by finite elements
CG matrix inversion for symmetric system
Parallel Processing focused on CG system

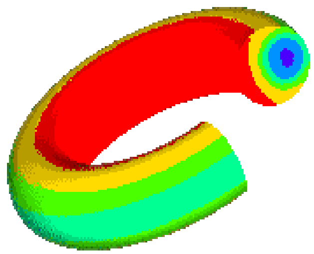


Nimrod Grid

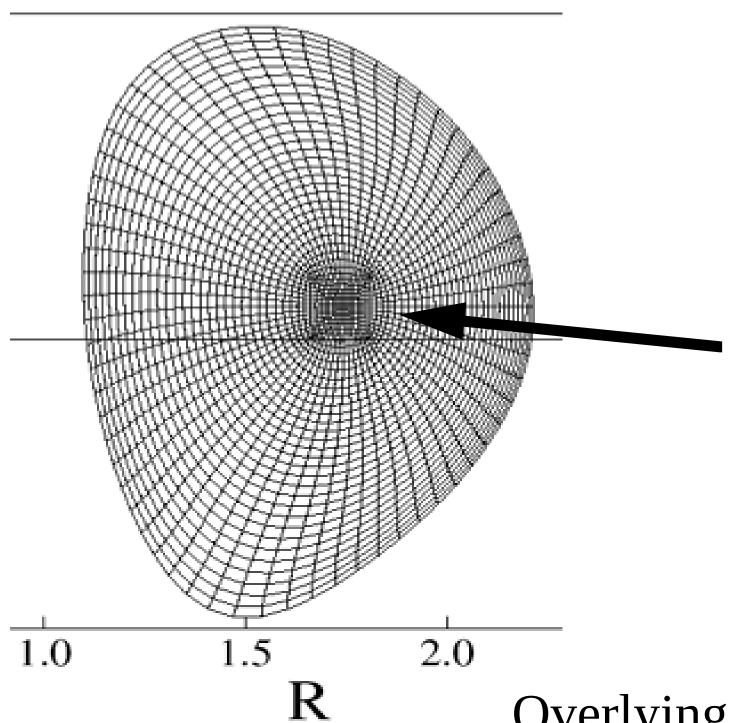
- Spectral in Toroidal Direction
- Unstructured Blocks of Structured Quadrilaterals in Poloidal Plane
- Each Unstructured Block may be a Single Triangle (Patching of Non-Conforming Blocks)
- Outer Boundary can conform to Real Machine Geometry
- Nearly Flux Surface Conforming within Separatrix
- Singularity at magnetic axis
  - Overlying Quadrilateral Grid (1st try)
  - Triangle Elements (Pie-slices) (2nd try)

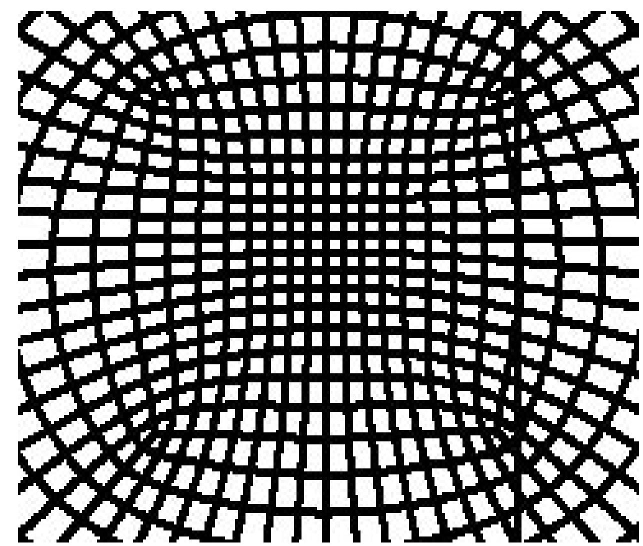


Nimrod Grid



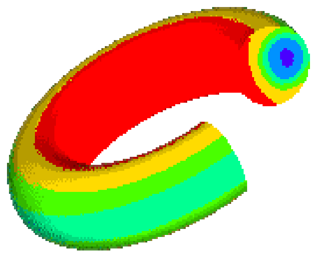
Overlying Quadrilateral Grid led to unphysical results at corners

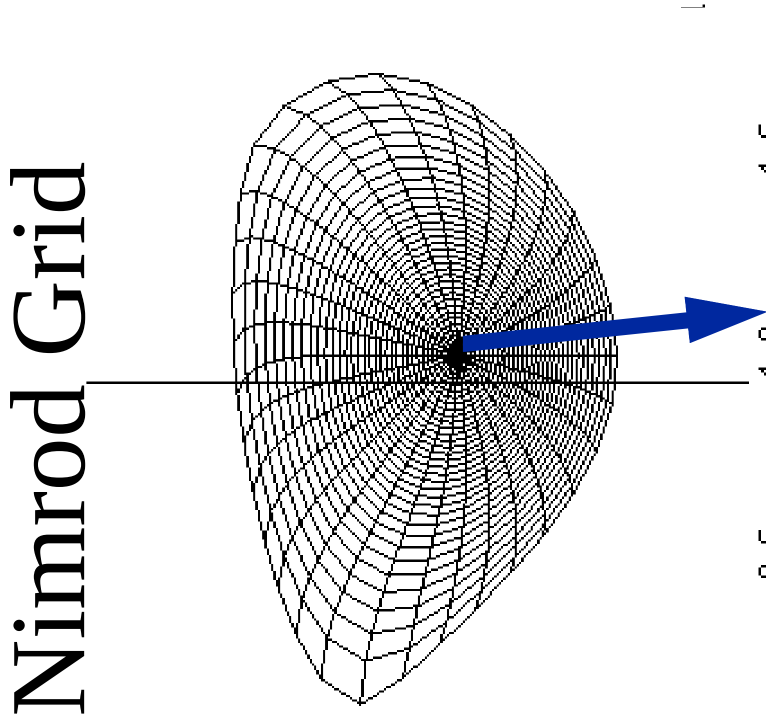


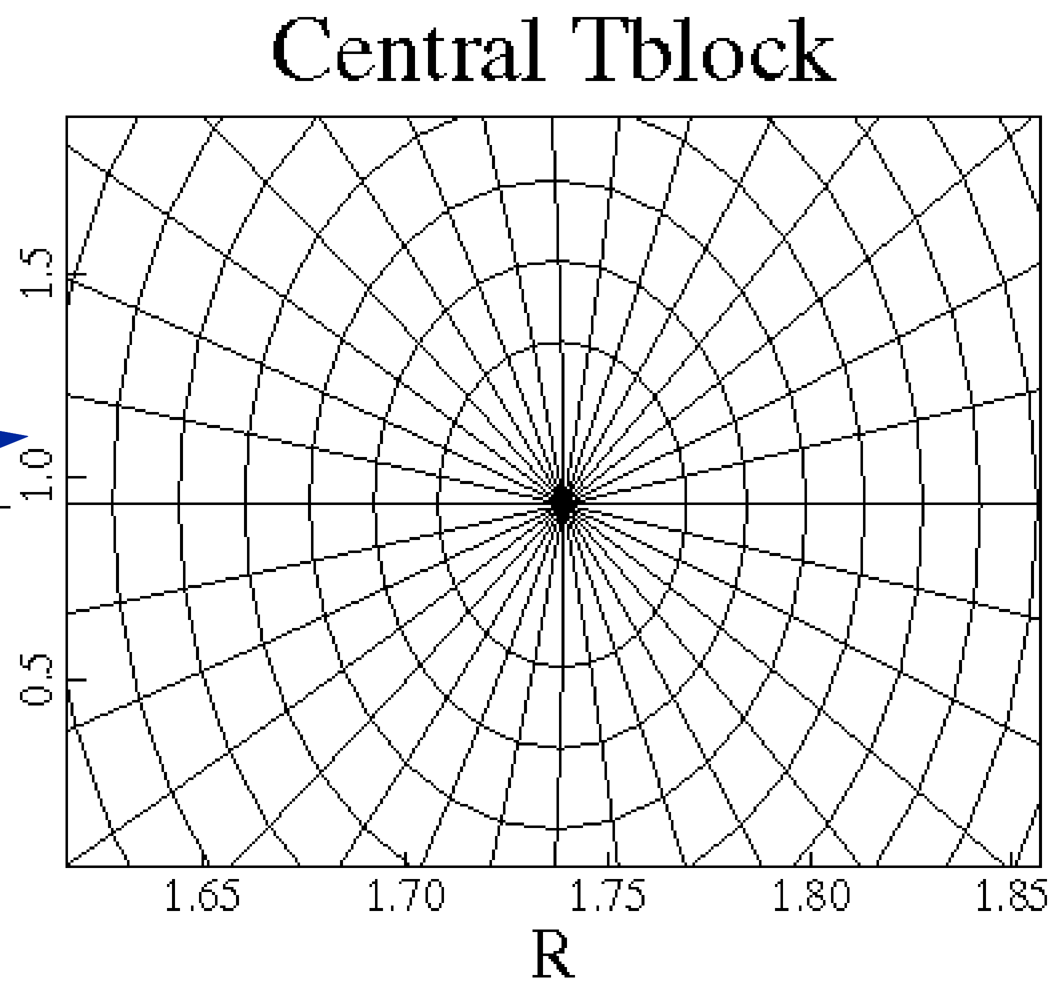


Overlying Quadrilateral Grid
near Magnetic Axis (Avoid Singularity)

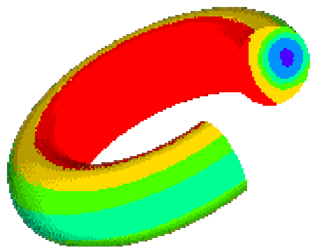
Triangle Block Patch appears to fix problem

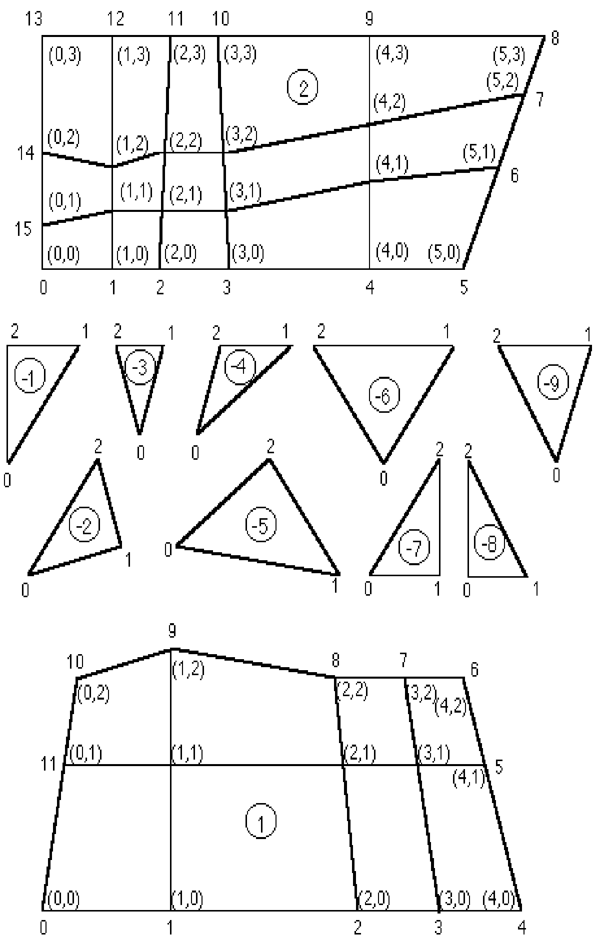


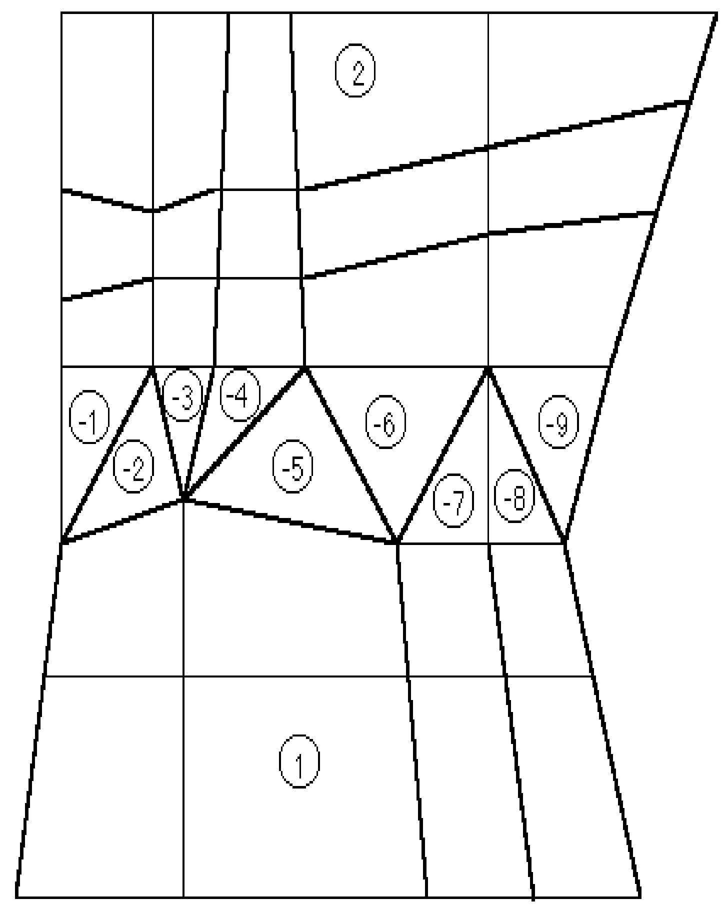




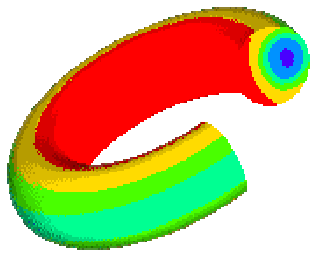
Patching Triangles also Allows for Variation in Grid Resolution (to be implemented)



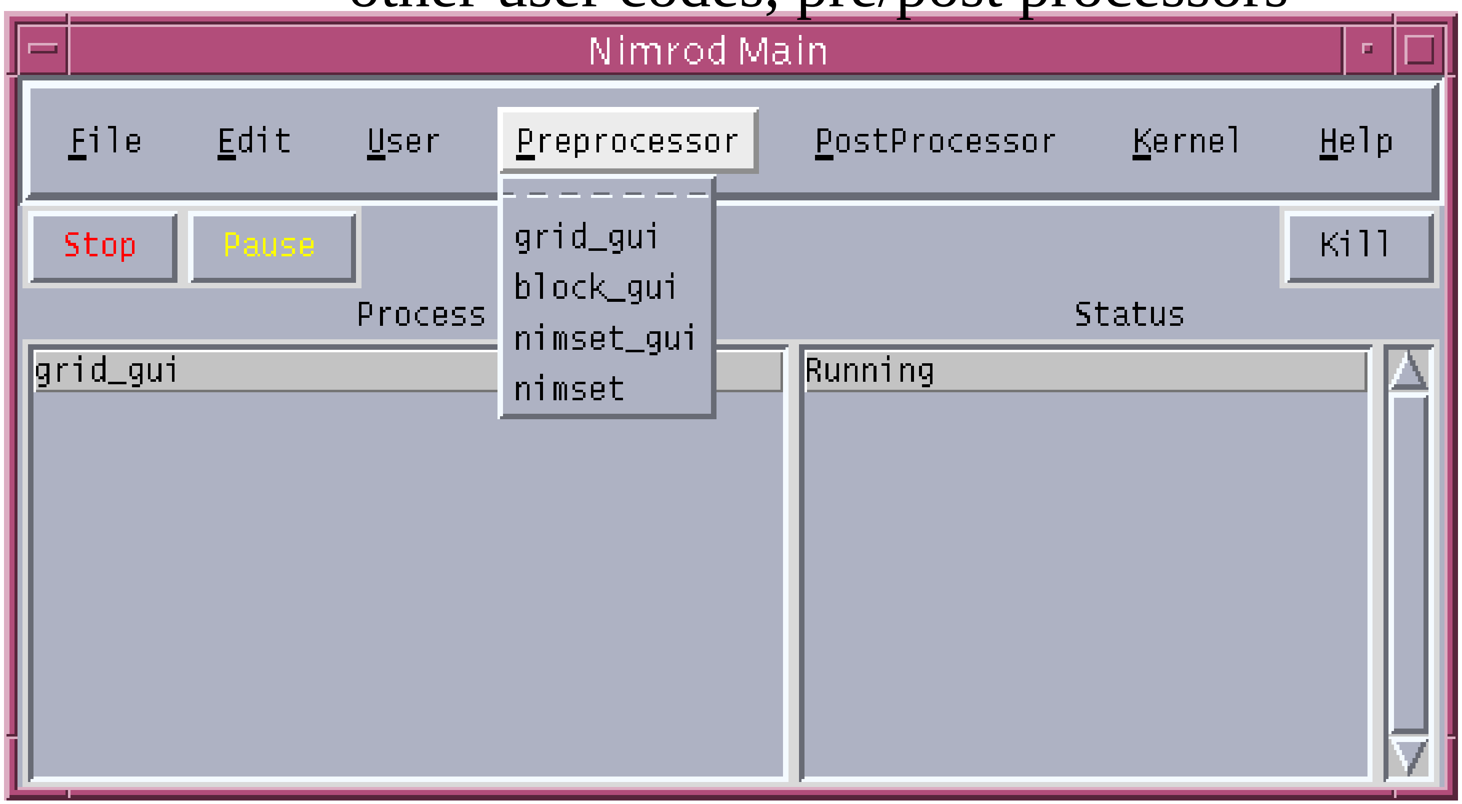




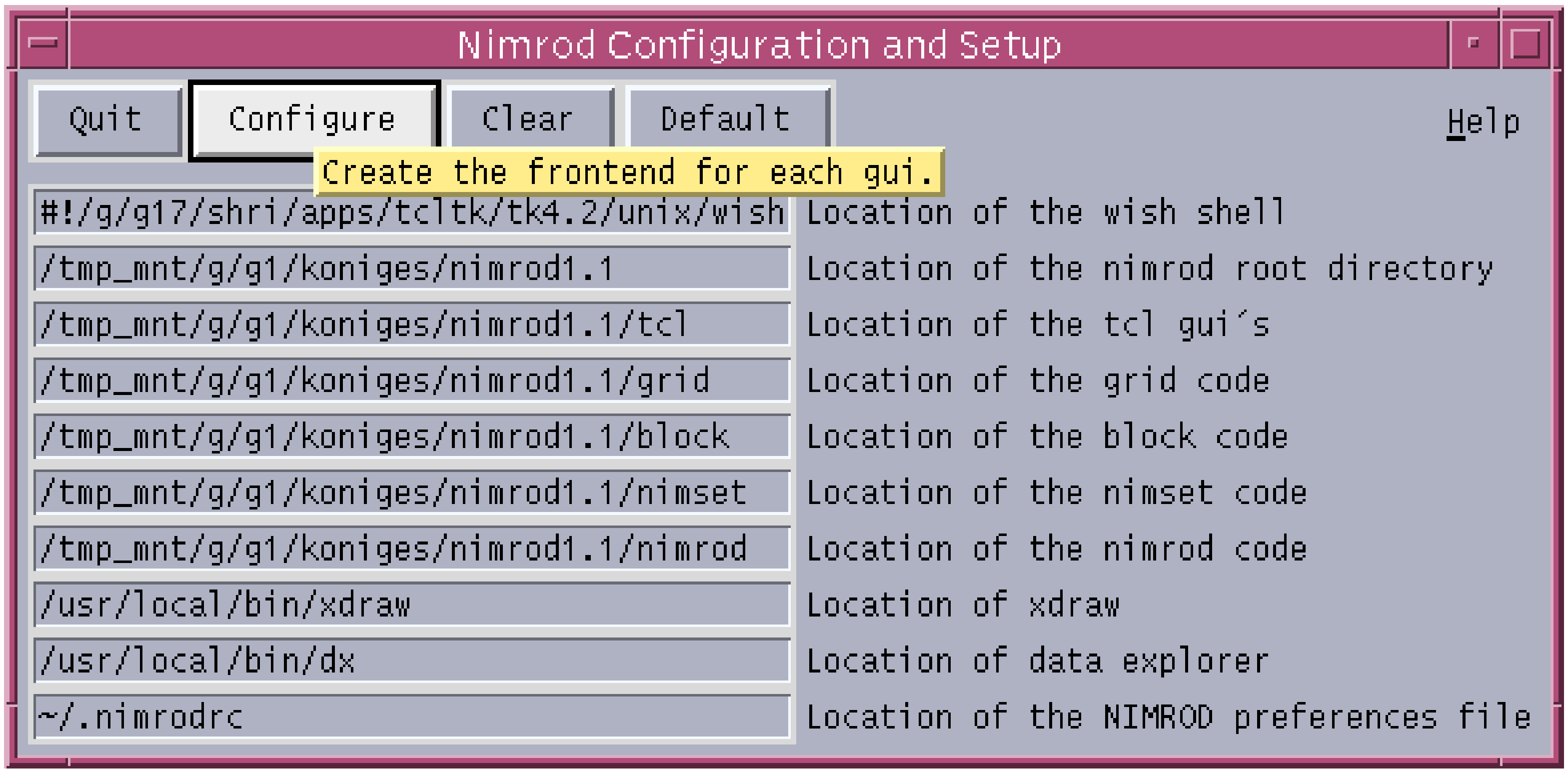
NIMROD GUI



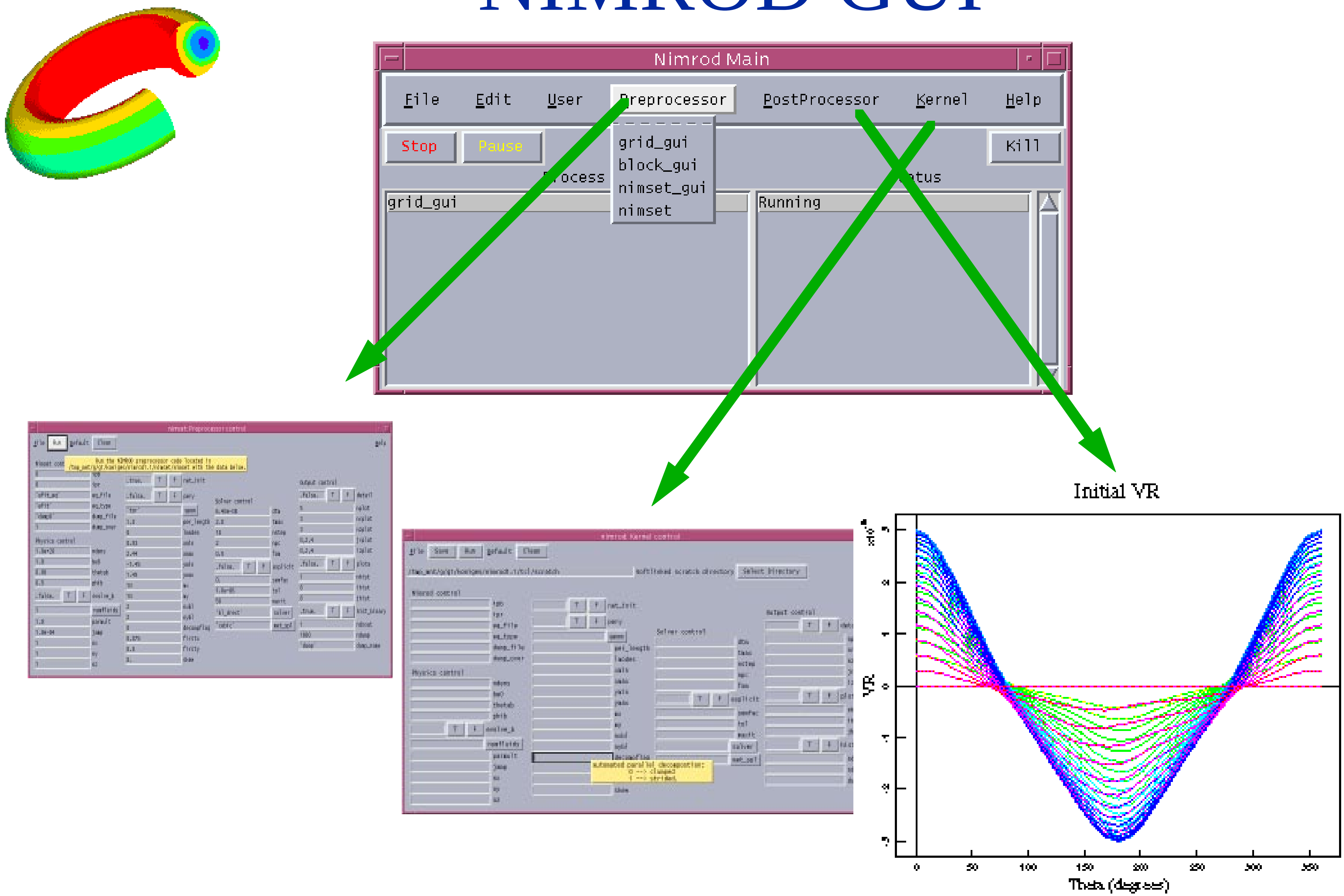
- Written in tcl/tk
- Controls interaction between user and NIMROD
  - Problem setup, Dynamical integration, Runtime dianostics
- Controls interaction between NIMROD and
  - other user codes, pre/post processors



GUI Configuration

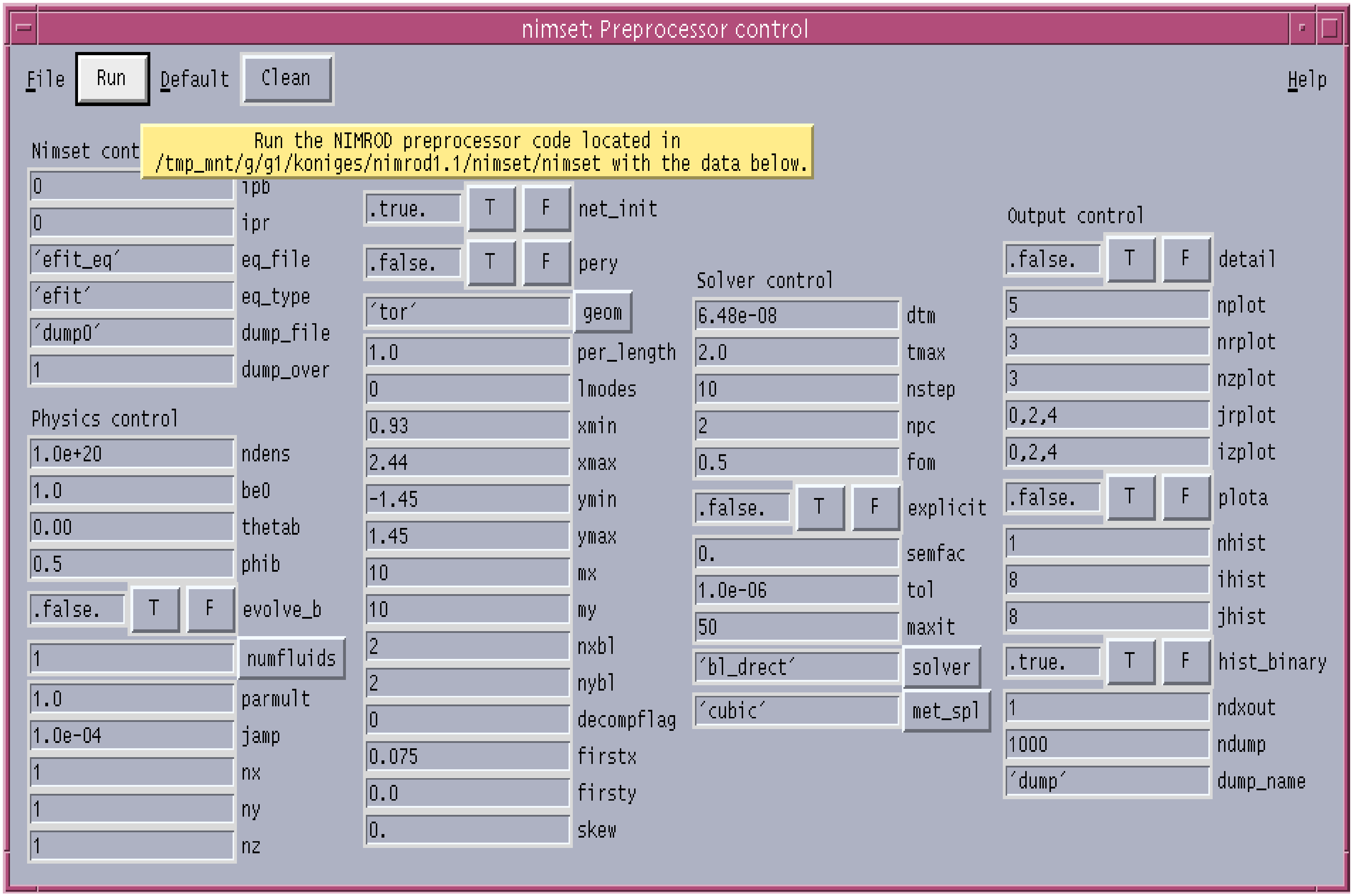


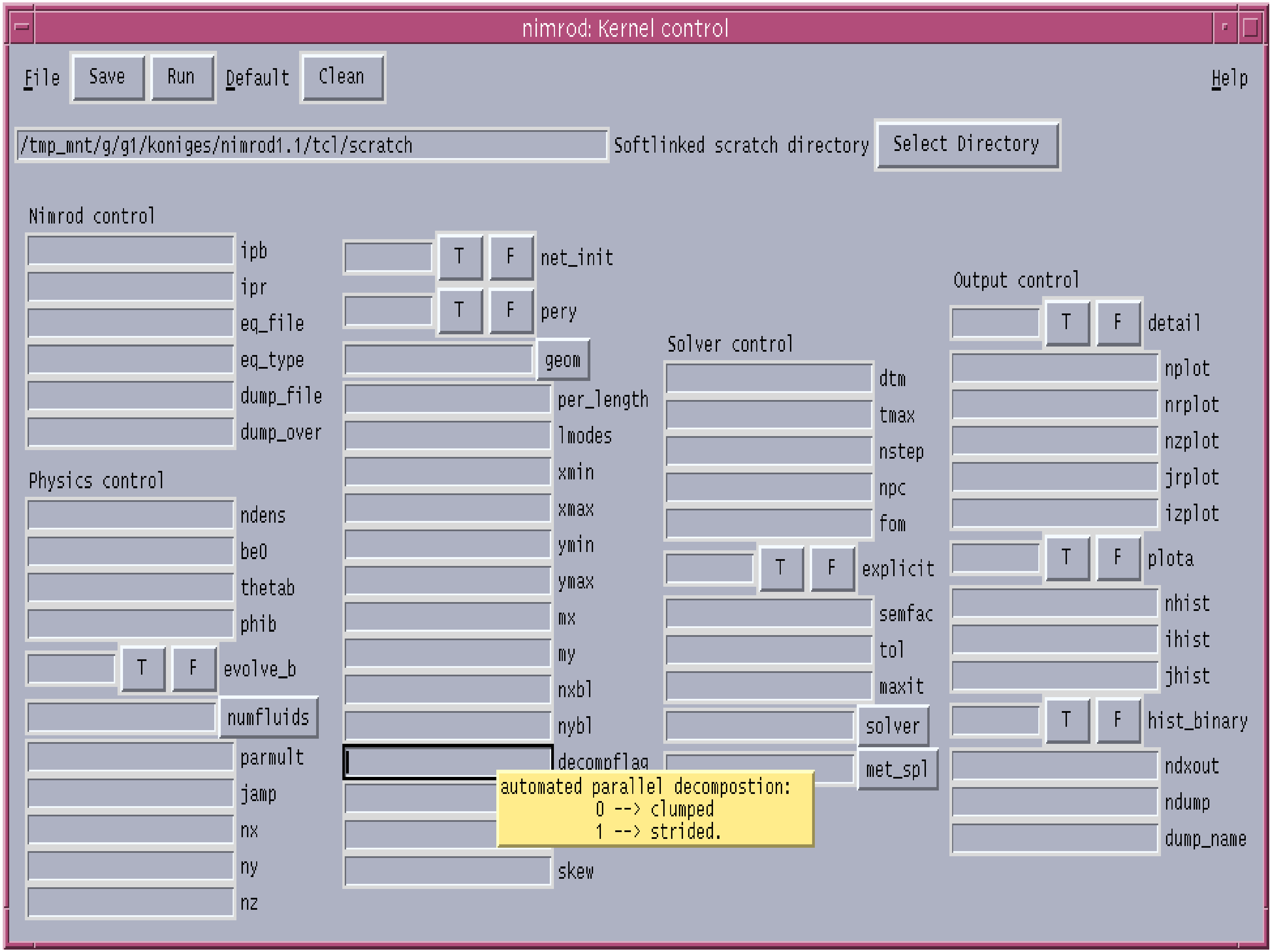
NIMROD GUI

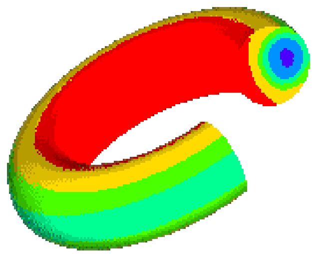


Nimrod Main
File
Edit
User
Preprocessor
PostProcessor
Kernel
Help
Stop
Pause
Kill
grid_gui
block_gui
nimset_gui
nimset
grid_gui
Running
Initial VR
VR
Theta (degrees)

Preprocessor Control

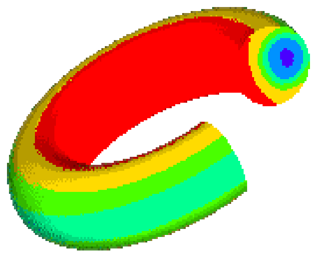



nimrod: Kernel control
File
Save
Run
Default
Clean
Help
/tmp_mnt/g/g1/koniges/nimrod1.1/tcl/scratch
Softlinked scratch directory
Select Directory
Nimrod control
ipb
ipr
eq_file
eq_type
dump_file
dump_over
Physics control
ndens
be0
thetab
phib
evolve_b
numfluids
parmult
jamp
nx
ny
nz
net_init
pery
geom
per_length
lmodes
xmin
xmax
ymin
ymax
mx
my
nxbl
nybl
decompflag
skew
automated parallel decompostion:
0 --> clumped
1 --> strided.
Solver control
dtm
tmax
nstep
npc
fom
explicit
semfac
tol
maxit
solver
met_spl
Output control
detail
nplot
nrplot
nzplot
jrplot
izplot
plota
nhist
ihist
jhist
hist_binary
ndxout
ndump
dump_name



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
  - FFT's in third dimension restricted to block
  - Communication between blocks via Message Passing Interface (MPI)
- Single Processor Optimization
  - see "Parallelizing Code for Real Applications on the T3D," A.E. Koniges and K.R. Lind, Computers in Physics 9, 399 (1995)
- Multiple Processor Optimization
  - overlap communication and computation
- Iterative Solver Design Issues

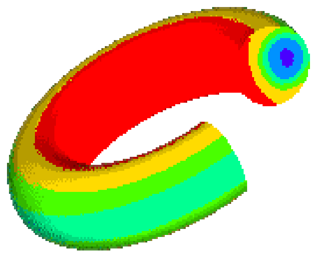


INHERENT PARALLELISM in NIMROD

- 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”

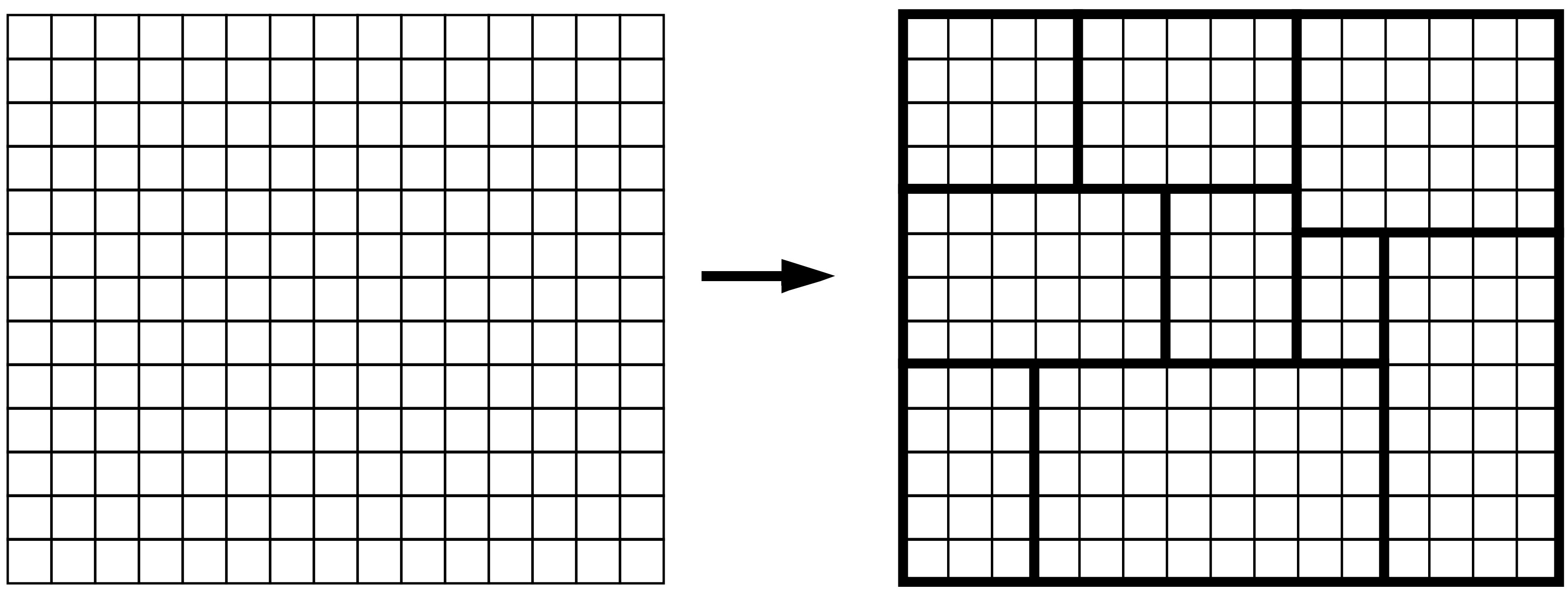
NIMROD Parallel Coding Choices

- Message-passing parallelism with F90/MP
- F90 provides dynamic memory, data structures
- MPI provides portability to any machine with a single-processor F90 compiler
- MPI allows irregular, asynchronous communication
- Same code will run on workstation, Cray C90, or parallel platforms:
  - Cray T3D/E, IBM SP2
  - Workstation Clusters
- Future: benchmark vs. loop parallel (DEC cluster)

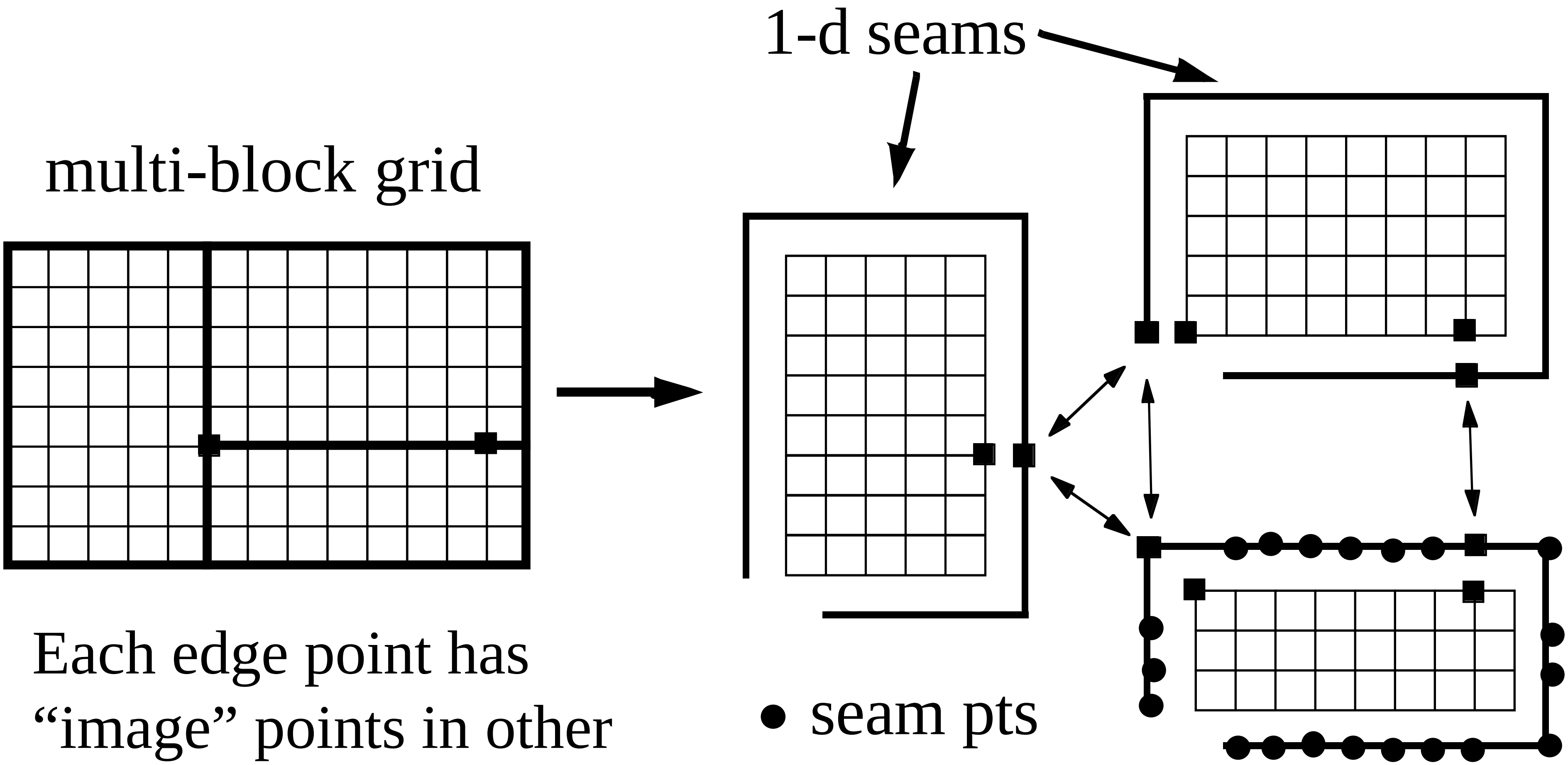


Grid Structure of NIMROD

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

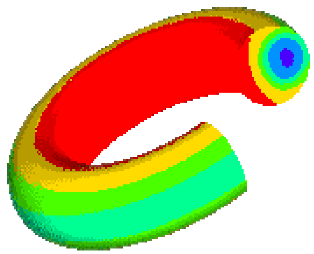


Sub-blocking with associated seams

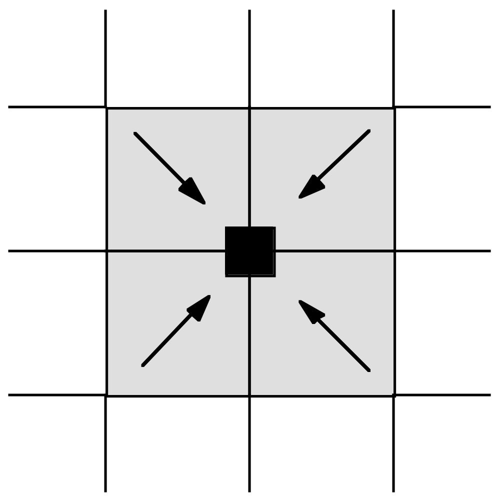


Each edge point has "image" points in other blocks/seams.

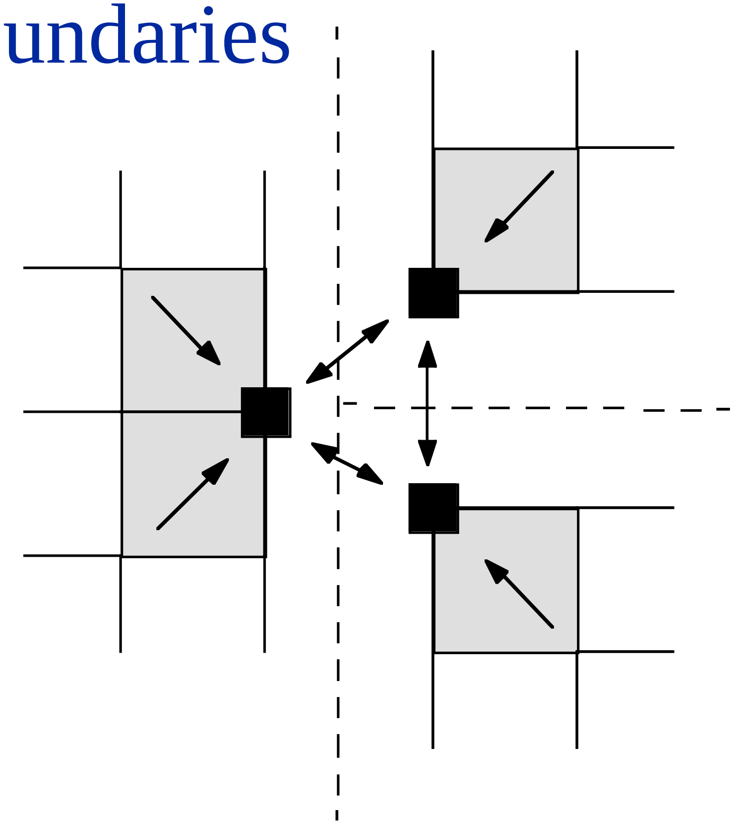
- ● seam pts
- ■ image pts



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

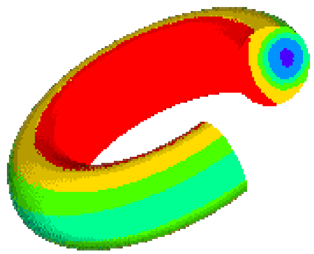


Interior



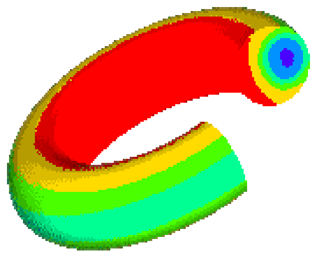
Across boundaries

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



Parallel Design

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

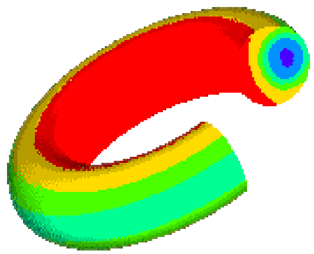


Serial Seam Connection

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.



Parallel Seam Connection

SERIAL VERSION: 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.

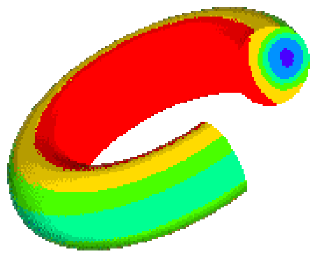
PARALLEL VERSION: 1) Send my seam data to neighboring processors.

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.

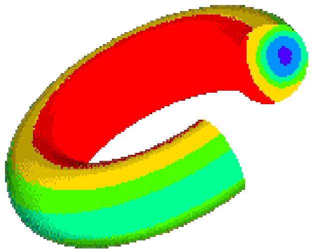
4) Apply external boundary conditions.

5) Copy from my block-edge grid points to my 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.

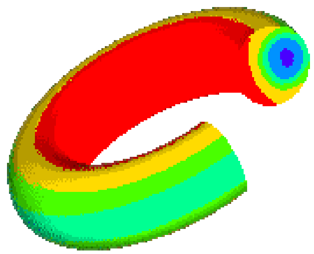


Timing Results for Parallel Seam Connection on T3E

- 1.02 million grid cells, 174 blocks, 51200 seam points, 3 values/grid-cell
- CPU seconds for 1 seam-operation:

Procs	1	2	5	10	20	30
Time	0.64	0.25	0.12	0.081	0.033	0.024

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

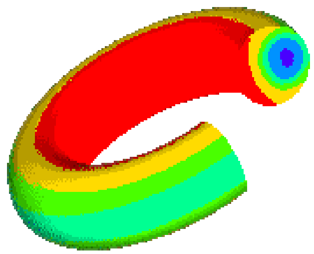


Timing Results for Explicit Nimrod T3D Calculation

- CPU seconds for 200 timesteps on the T3D shows excellent scalability as problem size increases. (same real time as 50 implicit time steps)

Blocks/Cell	1 PE	2 PEs	4 PEs	8 PEs	16 PEs	32 PEs	1peC90
4/400	94.6	47.3	24.2				12.5
16/1600	381.3	192.5	95.2	48.4			49.7
64/6400	1497.9	759	390.5		101.3	50.2	198.7
256/25,600			1531.8	790.8	400.3	206.2	
1024/102,400							

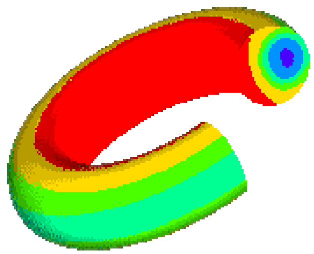
Blocks are 10X10, Cells are poloidal cells



Timing Results for Explicit Nimrod T3E Calculation

- CPU seconds for 200 timesteps on the T3E shows excellent scalability as problem size increases. (same parameters as previous table for T3D)

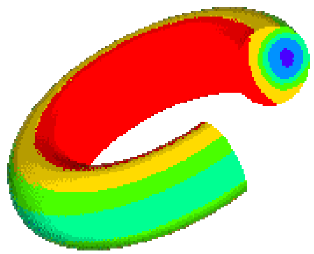
Blocks/Cell	1 PE	2 PEs	4 PEs	8 PEs	16 PEs	32 PEs	1peC90
4/400	28.8	14.2	7.58				12.5
16/1600	111.0	56.8	29.2	14.8	8.1		49.7
64/6400			112.1	58.3	30.8	16.0	198.7
256/25,600					117.4	62.3	
1024/102,400						238.8	



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 for CG solver require more study

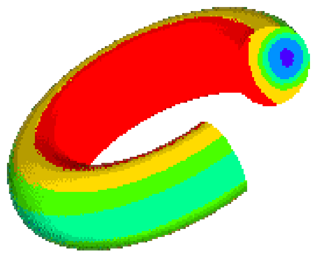
Blocks/Cell	1 PE	2 PEs	4 PEs	8 PEs	16 PEs	32 PEs	N-iter
4/400	216.3	114	57.4				2170
16/1600	870.1	419.8	216.6	110.5	60.2		1949
64/6400	2939	1454.8	743.5	336.3	192.1	104.5	1553
256/25,600			2948.2	1579.1	791.1	397.2	1565
1024/102,400							1378



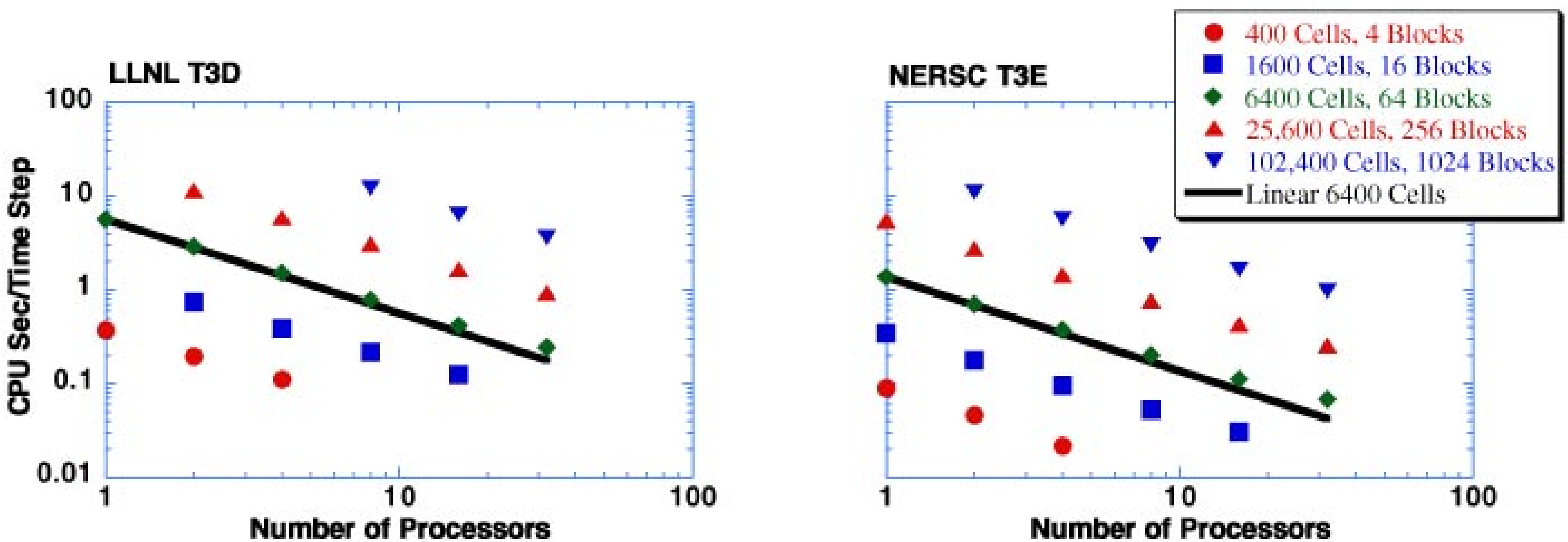
Timing Results for Implicit Nimrod T3E Calculation

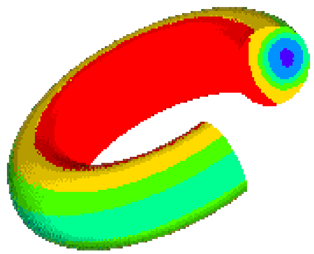
- CG solver with diagonal preconditioning
- 50 timesteps, roughly 30-40 CG iterations per step
  - time proportional to iterations
- Preconditioning methods for CG solver require more study

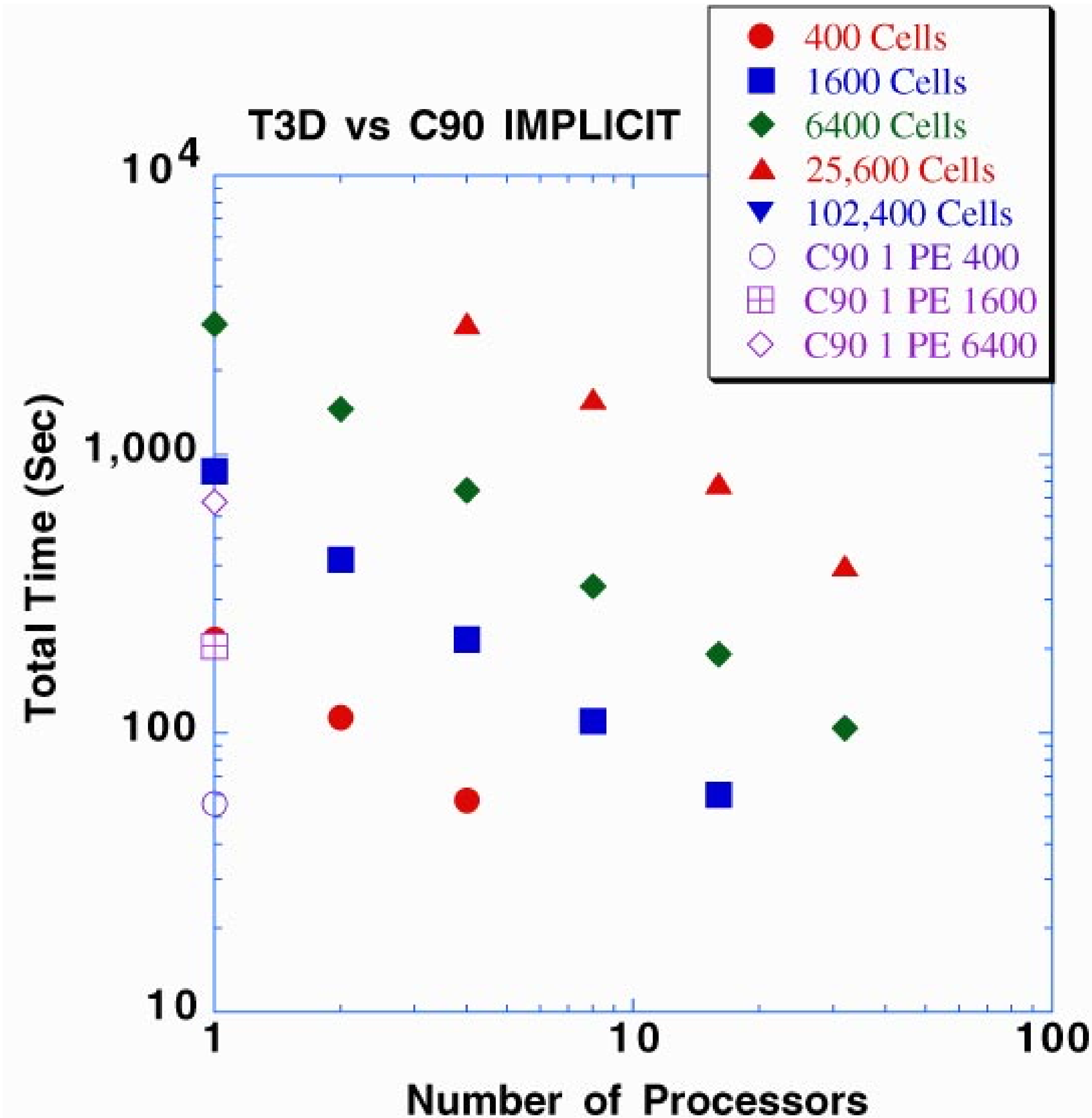
Blocks/Cell	1 PE	2 PEs	4 PEs	8 PEs	16 PEs	32 PEs	1peC90
4/400	68.2	35.5	19.0				55.9
16/1600	261.8	131.1	67.8	35.6	20.3		205.7
64/6400			229.4	120.4	61.6	34.8	678.1
256/25,600					242.8	125.0	
1024/102,400						449.7	

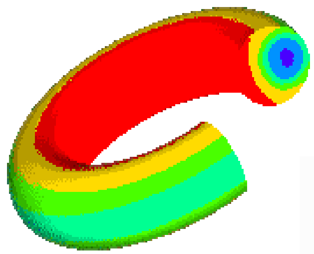


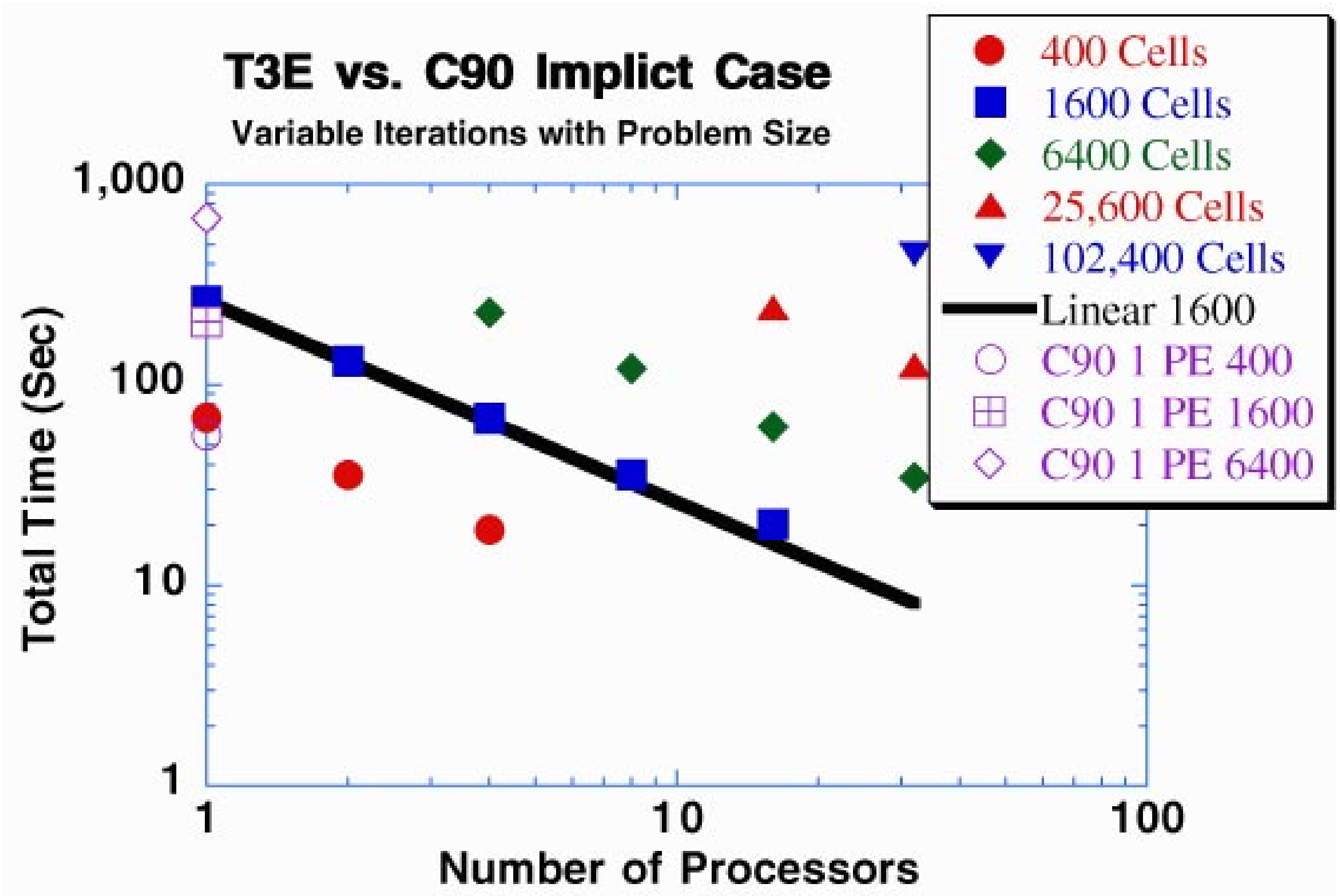
Performance Results Show Nearly Ideal Speed-up for Explicit Case (even for fixed problem size)





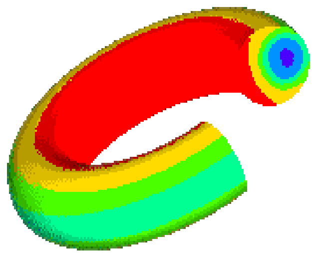

T3D vs C90 IMPLICIT
Total Time (Sec)
Number of Processors
10^4
1,000
100
10
1
10
100
400 Cells
1600 Cells
6400 Cells
25,600 Cells
102,400 Cells
C90 1 PE 400
C90 1 PE 1600
C90 1 PE 6400



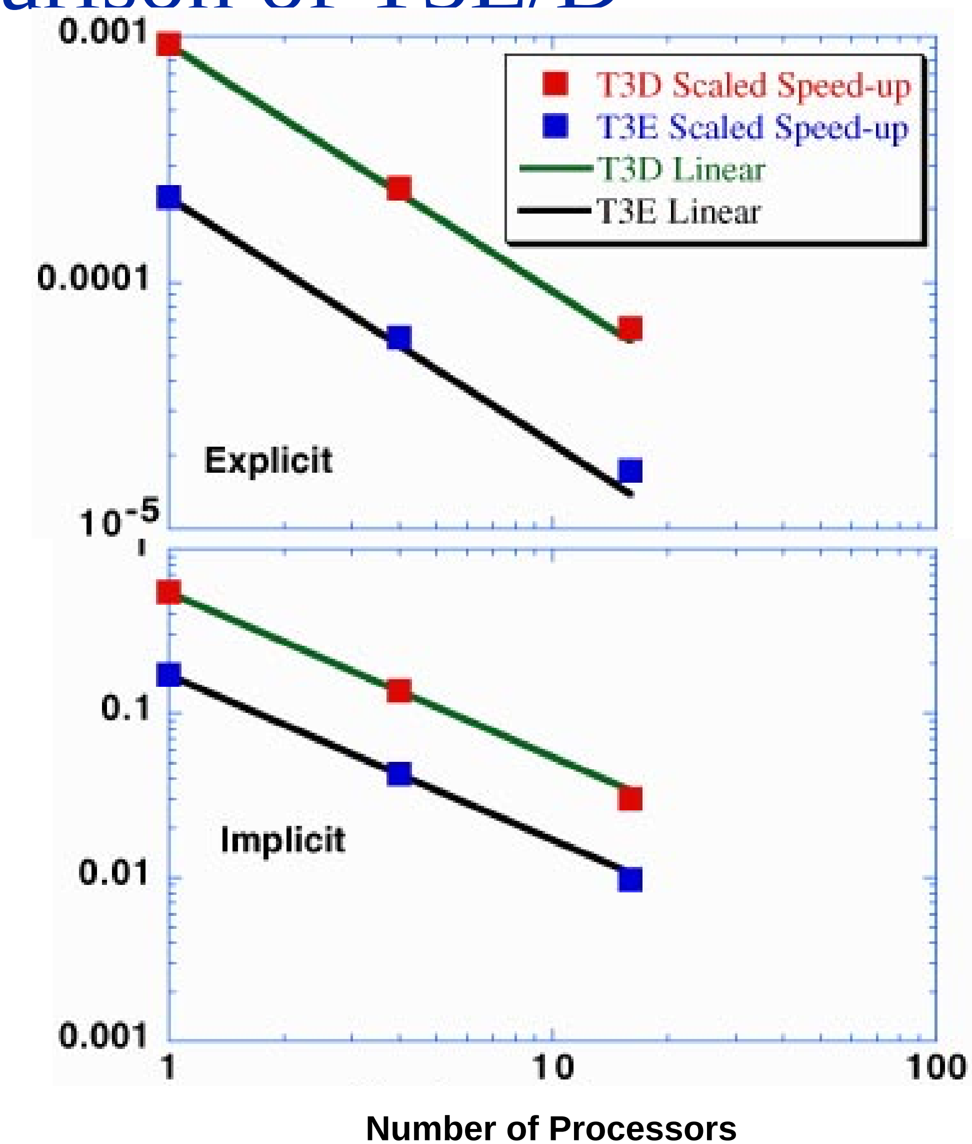


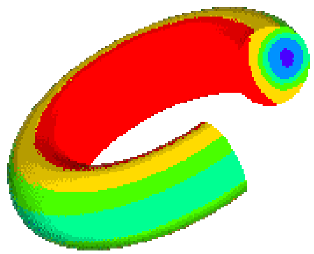
T3E vs. C90 Implict Case
Variable Iterations with Problem Size
Total Time (Sec)
1,000
100
10
1
1
10
100
Number of Processors
400 Cells
1600 Cells
6400 Cells
25,600 Cells
102,400 Cells
Linear 1600
C90 1 PE 400
C90 1 PE 1600
C90 1 PE 6400

Scaled Speed-up Comparison of T3E/D



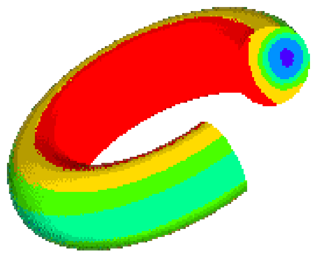
- Scaled speed-up is speedup/problem size
- T3E is roughly a factor of 4 faster
  - 2X processor speed
  - chaining
  - cache effects
- Scalability is virtually linear for both machines





Parallel Conclusions

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



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