RELAP5-3D Kinetics Upgrades

2013 International RELAP5 Users Group Meeting

Idaho Falls, Idaho

September 12-13, 2013

Doug Barber
Information Systems Laboratories, Inc.

Presentation Outline

- RELAP5-3D Nodal Kinetics Upgrades
  - Face-Dependent Albedos
    - Purpose: Replace global albedos with face- and energy-dependent albedos.
  - Composition-Dependent Neutron Velocity
    - Purpose: Replace global neutron velocities with composition- and energy-dependent velocities.
  - Asynchronous Nodal Advancement
    - Purpose: Provide automatic time step control for the kinetics along with nonlinear nodal solver logic.
  - Parallel Processing with Domain Decomp
    - Purpose: Resurrect parallel domain decomposition logic for kinetics.

Face-Dependent Albedos

- Purpose:
  - Albedos, $\alpha = J/\varphi$, are used for specifying the flux boundary condition, e.g.,
    - $\alpha = 0$, zero current BC
    - $\alpha = \infty$, zero flux BC
    - $\alpha = 0.5$, zero net incoming current BC
  - Albedos previously entered on an energy-dependent basis which was the same for the entire boundary.
  - To account for heterogeneity in the neutron leakage at the boundary, face-dependent albedos are now input.
- Status:
  - Completed
  - Not yet in latest version of RELAP5-3D

Face-Dependent Albedos

- Approach:
  - New card series 37IIJJKKF
    - II = x-direction index
    - JJ = y-direction index
    - KK = z-direction index
    - F = face index
  - Used in conjunction with BC specified on W7-W10 of 30000003 card
  - Used in conjunction with energy-dependent albedos specified on 30000021 card
  - 37IIJJKKF card is <u>not</u> required for every boundary face
    - If card is present, albedos will be taken from the input card
    - Otherwise, albedos are determined from 30000003 and 30000021 cards
  - Only for Krylov solver, not LSOR solver
  - Used for both Cartesian and Hexagonal geometries
  - Use for both CMFD and Nonlinear-Nodal (NEM, TPEN)

Face-Dependent Albedos

- Verification Testing
  - Test cases designed to test functionality
    - Both Cartesian and Hexagonal test cases
    - Both CMFD and NEM test cases
    - Compared to global albedo cases
    - Error checking tested
  - Regression analysis on code versions before and after face-dependent albedo logic
    - Comparison of statistical parameters
    - Comparison of output files
  - 122 nodal kinetics test cases
    - Cartesian & hexagonal geometries
    - 2- & 4-group cases
    - Krylov and LSOR CMFD solvers
    - NEM & TPEN nodal solvers
    - BiCGStab & GMRES linear system solvers
    - With and without rod decusping
    - Restart cases

Face-Dependent Albedos

- Results of Regression Testing
  - Statistical Parameters
    - 20 parameters examined, including reactor power and core k-eff
    - No differences in any of the parameters except CPU time
  - Output File Comparison
    - Zero differences
- CPU Timing
  - Threshold: greater than 10sec with +/- 5% difference
    - 4 cases with a CPU degradation
      - max: +11.9% (k3200nk.i)

Comp-Dependent Neutron Velocity

- Purpose:
  - Neutron velocities are used in the transient fixed-source problem:

$$\frac{1}{v_g}\frac{\partial \phi_g}{\partial t} = \sum_{g'=1}^{G} \Sigma_{g'g}\phi_{g'} + (1-\beta)\chi_g^p \sum_{g'=1}^{G} \nu\Sigma_{fg'}\phi_{g'} + \chi_g^d \sum_{i=1}^{m} \lambda_i C_i - (\nabla \cdot J_g + \Sigma_{tg}\phi_g) + S_g^{ext}$$

  - Neutron velocities were previously entered on an energy-dependent basis, v_g, which was the same for the entire core.
  - However, velocities can vary significantly with composition due to spectral shift.
  - Therefore, composition-dependent velocities are now input.
  - Also accounts for control rod position.
- Status:
  - Completed
  - Not yet in latest version of RELAP5-3D

Comp-Dependent Neutron Velocity

- Approach:
  - New card series 38CCC0GN1-9 or 38CCCC0GN
    - CCC or CCCC is the composition number
    - G is the energy group index
    - N is the kinetics parameter, which for now is only "1"
      - 1 = neutron velocity
      - Future expansion for N = 2 to 9
  - Extension to the 32CCC0GN1-9 / 32CCCC0GN cards
    - Can only enter a 38 series card if composition previously entered
    - Not required; if not entered velocities taken from 30000005
  - Uncontrolled and Controlled values input
  - Velocity and inverse velocity now use 3D arrays to store values for each node / energy group

Comp-Dependent Neutron Velocity

- Verification Testing
  - Test cases designed to test functionality
    - Total of 7 test cases
    - Test neutron velocity input for all compositions
    - Test neutron velocity input for only a couple compositions
    - Error checking tested
  - Regression analysis on code versions before and after composition-dependent neutron velocity logic
    - Comparison of statistical parameters
    - Comparison of output files
  - 122 nodal kinetics test cases
    - Cartesian & hexagonal geometries
    - 2- & 4-group cases
    - Krylov and LSOR CMFD solvers
    - NEM & TPEN nodal solvers
    - BiCGStab & GMRES linear system solvers
    - With and without rod decusping
    - Restart cases

Comp-Dependent Neutron Velocity

- Results of Regression Testing
  - Statistical Parameters
    - 20 parameters examined, including reactor power and core k-eff
    - Most had no differences in any of the parameters except CPU time
    - 6 cases showed minor differences
      - Reactor power (0.057%), core reactivity (0.243%) and/or mass error (-3.558%)
  - Output File Comparison
    - 6 cases showed differences
      - (1) neacrp-c1-4node-krlv-nem, (2) neacrp-c1-alb-krlv-nem,
      - (3) neacrp-c1-krlv-nem, (4) smart330-c1g4-tr-krlv-tpen-020cusp0,
      - (5) smart330-c1g4-tr-krlv-tpen-020cusp1, and (6) smart330-c1g4-tr-krlv-tpen-020cusp2
  - Differences due to movement of control rods
    - Weighted linear combination of controlled and uncontrolled neutron velocities
    - Previously, effect of control rod was not considered
    - Differences confirmed to be numerical
- CPU Timing
  - Threshold: greater than 10sec with +/- 5% difference
    - 3 cases with a CPU degradation
      - max: +7.0% (k3200nk.i)

Asynchronous Nodal Advancement

- Purpose:
  - Nodal kinetics solution can be computationally intensive
  - Need dynamic time step control
    - Use small time steps when conditions are changing
    - Use large time steps when conditions are quasi-steady
  - Implement automatic time step prediction based on change in:
    - Absorption + removal cross section
    - Delayed neutron source
  - Also implement a criteria for determining if nonlinear-nodal updates should be performed.
- Status:
  - In work: testing and debugging

Asynchronous Nodal Advancement

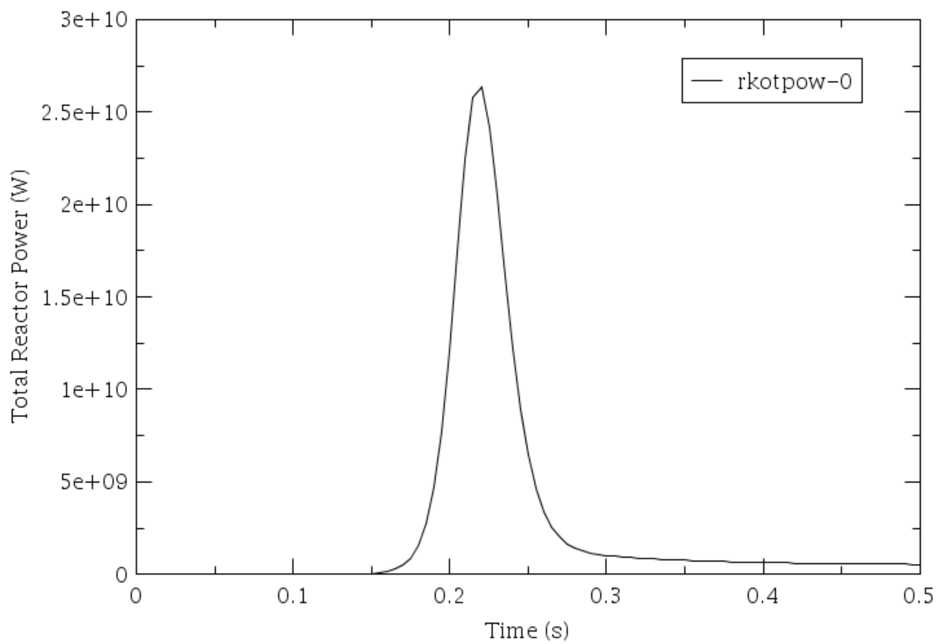
$$\tau_\theta = \frac{1}{\left|\frac{1}{\theta}\frac{\partial \theta}{\partial t}\right|}$$

$$\Delta t_\theta^n = \eta_{dyn}\left\lfloor \min \tau_{\theta,i,j}^n \right\rfloor$$

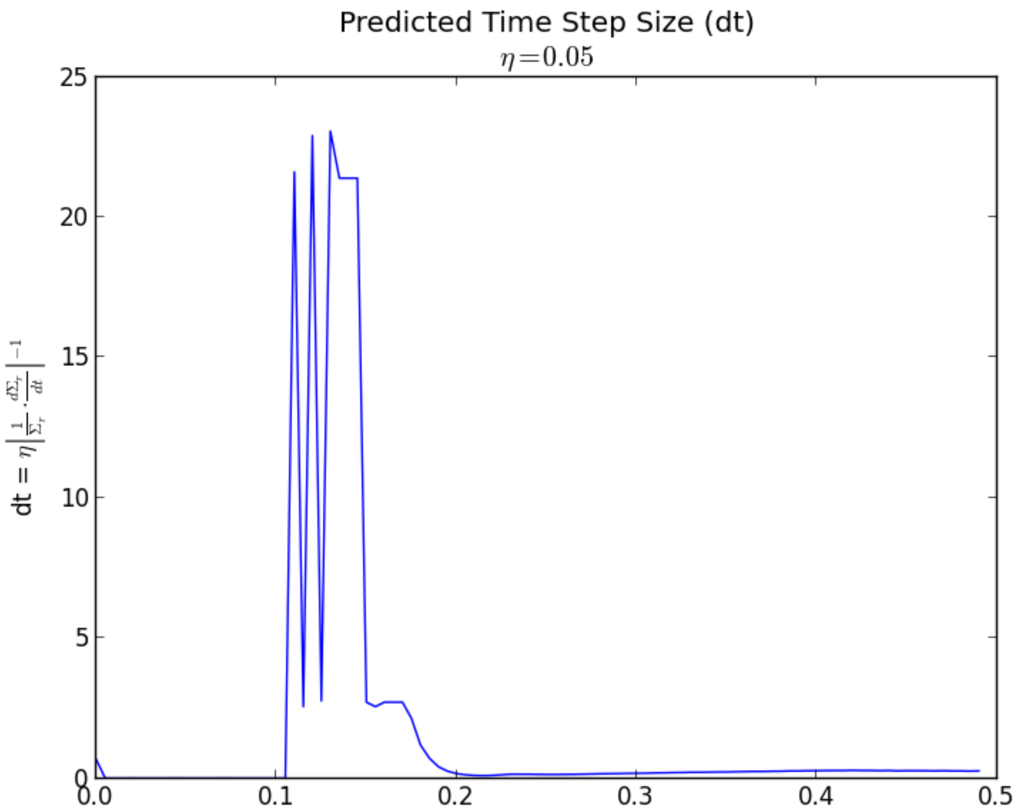
- Approach:
  - Use dynamic time scale
- Apply user-defined fractional allowable change
  - Ratio of kinetics and T/H time step size is restricted to be a rational number
  - Initial testing demonstrated viability of scheme
    - Predicted time step sizes for the NEACRP-C1 rod eject case is in line with expected values
    - 1-2 ms during rod eject; 100-250ms during asymptotic phase
  - Option for using exponential extrapolation when kinetics is supercycling T/H
  - Potentially unstable
    - Synchronization will be key
    - User input min/max kinetics time step size may help

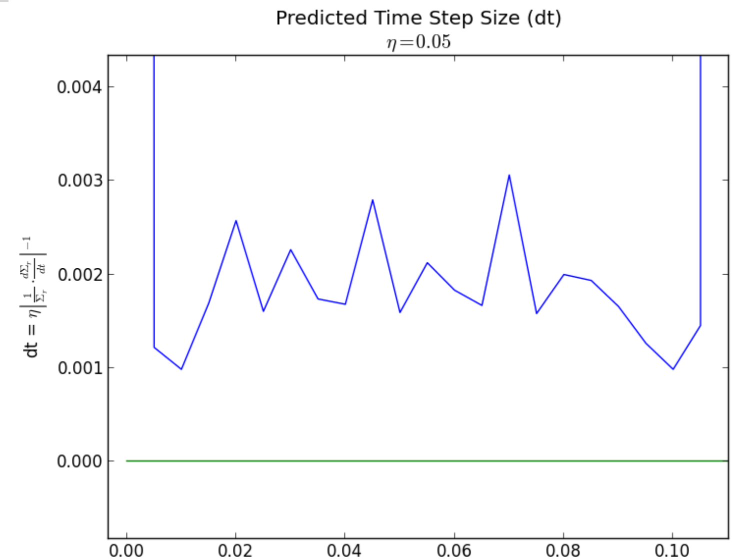
Asynchronous Nodal Advancement

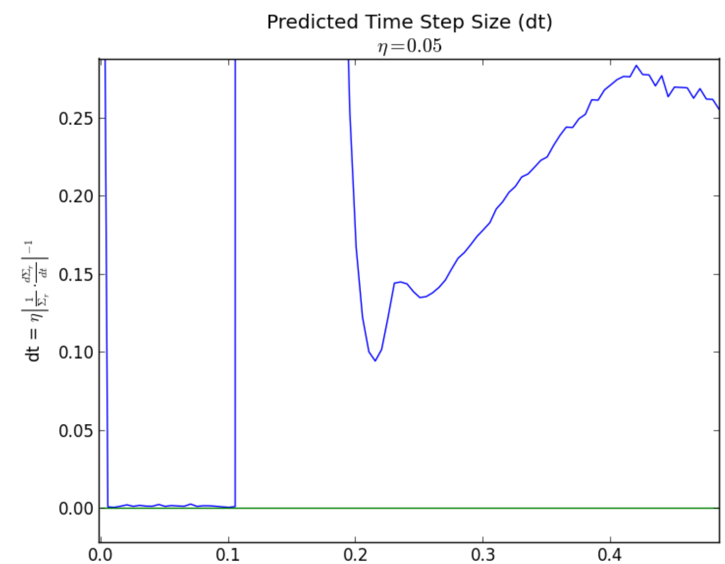
- <u>NEACRP C1 Rod Eject Benchmark</u>
  - Peripheral Rod Ejection
  - 0.1s ejection
  - Peak Power @ 0.22s
  - Time step analysis
    - Expected dt=0.001s during ejection
    - dt=0.100s – 0.250s during asymptotic phase



Asynchronous Nodal Advancement







Parallel Processing

- Purpose:
  - Nodal kinetics solution is largely parallelizable.
    - This work was done 15 years ago, but coding has not been maintained
  - Resurrect parallel processing logic for the nodal kinetics.
  - Utilize axial domain decomposition.
  - Maximum of 4 axial subdomains solved in parallel.
  - Expect near 100% efficiency for 2 processors and slightly less for 4 processors.
- Status:
  - Just started: Initial scoping being performed.

Parallel Processing

- Parallel Coarse Mesh Finite Difference (CMFD)
  - Requires extra solution at the interface
  - Incomplete Domain Decomposition (IDD) Preconditioner is utilized
  - Near 100% efficiency is possible
- Parallel Nonlinear Nodal Solver
  - Two-node solutions are perfectly parallelizable
  - Super-speedups are expected since memory fetch times are reduced
- Support Calculations
  - e.g., cross section evaluation, linear system setup, etc.
  - Inherently parallel
  - Should see 100% efficiency

Parallel Processing

- Planar Incomplete Domain Decomposition (PIDD)

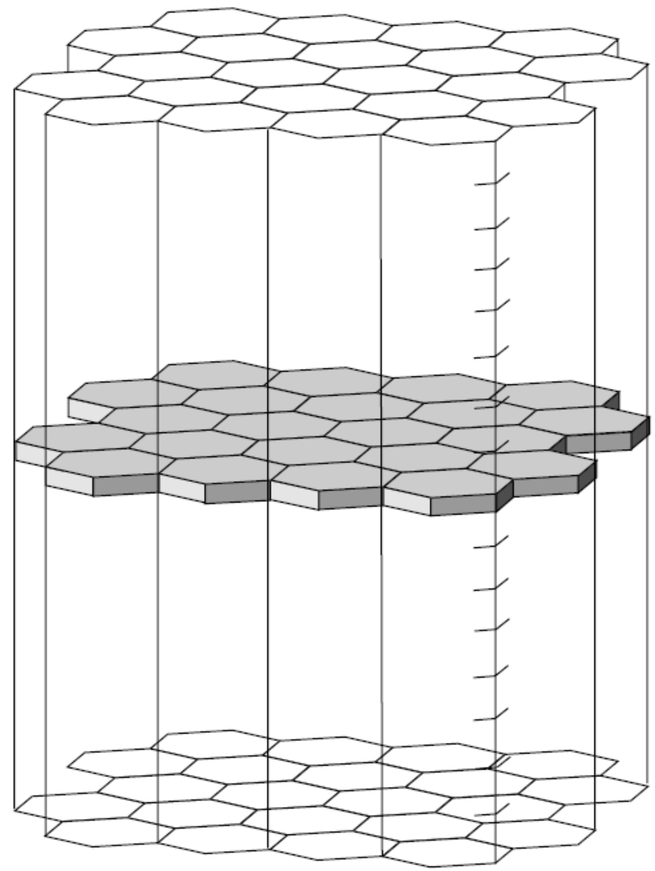
$$\begin{bmatrix} P_1 & U \\ L & P_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \quad (x_1, x_2) \in \mathbb{R}^{N/2}$$

- Costly to invert, so approx.
  - Solve 2-plane linear system
  - Only use block diagonals

$$\begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix} = \begin{bmatrix} \Gamma_1^{-1} & \\ & \Gamma_2^{-1} \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \quad (\tilde{x}_1, \tilde{x}_2) \in \mathbb{R}^{N/k}$$

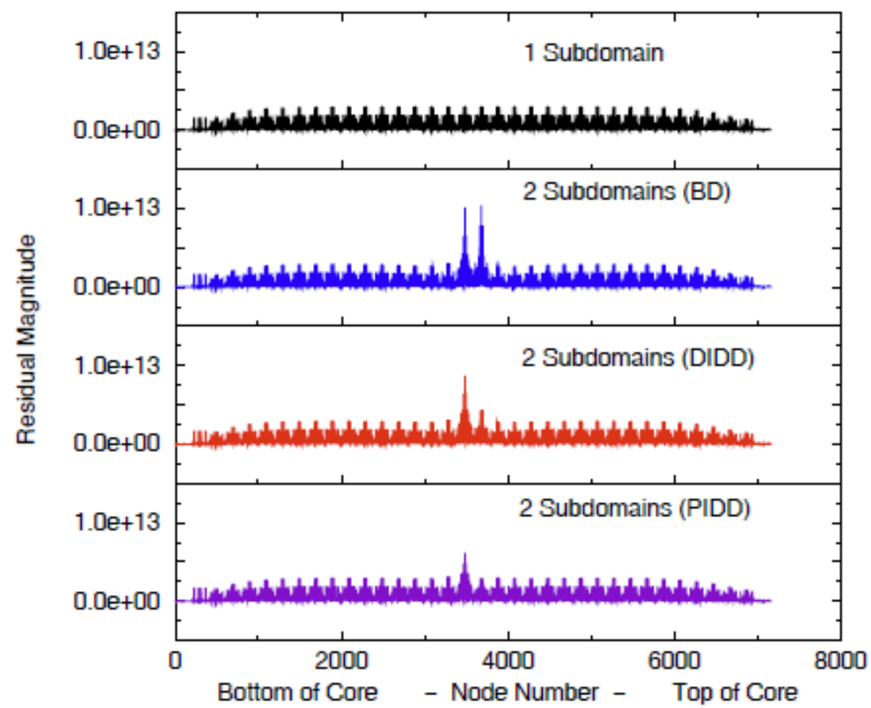
- Substituting yields:

$$P = \begin{bmatrix} I_1 & & & & & \\ & \ddots & & & & \\ & & I_{k/2} & -U_{k/2}\Gamma_2^{-1} & & \\ & & -L_{k/2+1}\Gamma_1^{-1} & I_{k/2+1} & & \\ & & & & \ddots & \\ & & & & & I_k \end{bmatrix}^{-1} \begin{bmatrix} P_1 & U_1 & & & & & & \\ L_2 & \ddots & \ddots & & & & & \\ & \ddots & \ddots & U_{k/2-1} & & & & \\ & & L_{k/2} & P_{k/2} & & & & \\ & & & & P_{k/2+1} & U_{k/2+1} & & \\ & & & & L_{k/2-2} & \ddots & \ddots & \\ & & & & & \ddots & \ddots & U_{k-1} \\ & & & & & & L_k & P_k \end{bmatrix}$$



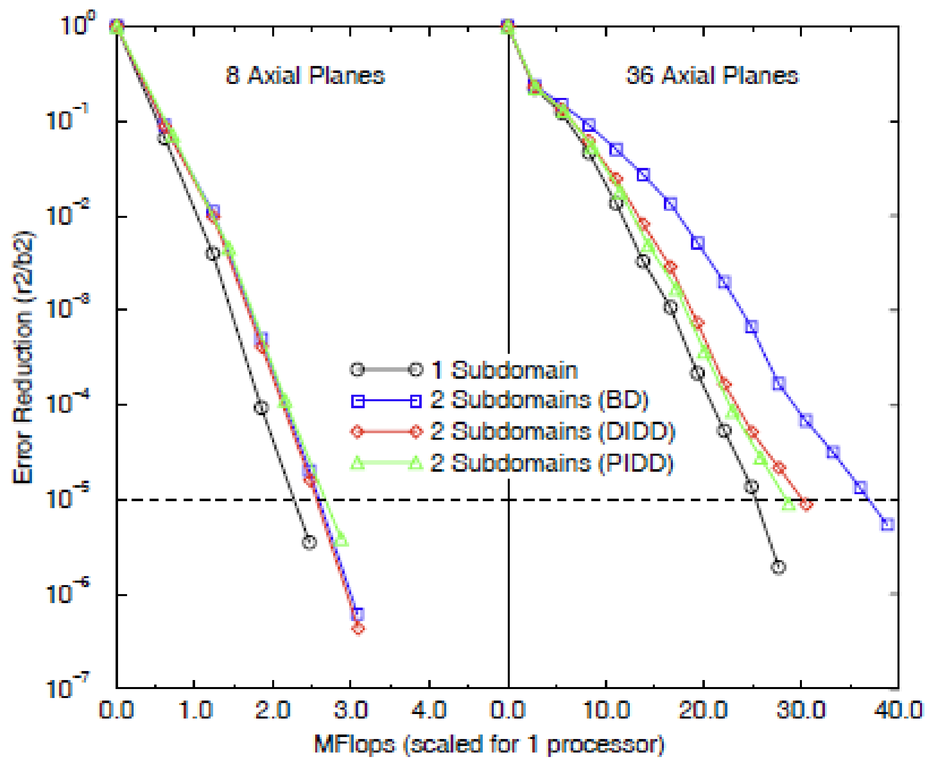
Parallel Processing

- Comparison of 3 IDD Preconditioners
  - Block Diagonal (BD)
  - Diagonal IDD (DIDD)
  - Planar IDD (PIDD)
- PIDD shows best residual performance



Parallel Processing

- Comparison of 3 IDD Preconditioners



Parallel Processing

- Phase I Work
  - Examining code for existing parallel structure
  - Assessing state of the code
  - Beginning simple parallelization of ancillary routines
  - Using OpenMP
  - Write Software Requirements Specification (SRS)
- Phase II Work
  - Implement IDD Preconditioner
  - Test CMFD Parallelization
  - Parallelize 2-node solver and test
  - Parallelize support calculations and test