NUCLEAR REACTOR KINETICS

Lecture 5

Basic Transient Solutions
to the 1-D, 2-group Diffusion Equations

Massachusetts
Institute of
Technology

Course Outline

22.213 (22.S904) Calendar					
Lecture #	Date	Topic	LECTURER	Read	Assignment Handed Out
1	5-Sep	Course Overview and First Day Exam	Smith		
2	10-Sep	Review of Delayed Neutrons and Point Kinetics Equations	Smith		PSET # 1: Point Kinetics
3	12-Sep	Review Steady-State Finite-Difference Diffusion Methods (1D, 2D)	Smith		
4	17-Sep	Generalized PKEs from Spatial Finite-Difference Diffusion	Smith		PSET # 2: 2-D Steady-State Diffusion
5	19-Sep	Basic Transient Finite-Difference Direct Solutions	Smith		
6	24-Sep	Testing Various PKE implementations	Smith		PSET # 3: 2-D Fully-Implicit Diffusion
7	26-Sep	Quasi-Static Time-Integration	Smith		
8	1-Oct	Higher-order Time Integration and Runga-Kutta	Smith		PSET # 4: PKE from 2-D Diffusion
9	3-Oct	Time Stepping for Automatic Error Control	Smith		
	8-Oct	Columbus Holiday (8th and 9th)			
10	10-Oct	2D Fully-implicit Iterative Numerical Methods: PJ, GS, SOR	Smith		PSET # 5: PKE Time Step Control
11	15-Oct	Iterative Numerical Methods: CG, GMRES,???	Smith		
12	17-Oct	Coarse Mesh Rebalance & Nonlinear Diffusion Acceleration	Smith		PSET # 6: PKE with Nonlinear Feedback
13	22-Oct	Nodal Methods: Kinetic Distortion and Frequency Transformation	Smith		
	24-Oct	Midterm Exam			
14	29-Oct	Midterm Detailed Exam Solution/2D LRA SS Comparisons	Smith		PSET # 7: CMR and NDA acceleration
15	31-Oct	Multigrid Acceleration Methods	Smith		
16	5-Nov	JFNK for Non-linear Systems	Smith		2-D LRA Rod Ejection Contest
17	7-Nov	Transient Sn	Smith		
	12-Nov	Veterns Day Holiday			
	14-Nov	Special Project Work Period	ANS Meeting		
18	19-Nov	Transient MOC	Smith		
19	21-Nov	Parallel Solver Technologies (PetSc)	Herman/Roberts		
20	26-Nov	So You Want To Be A Professor? Student Lectures	?????		
21	28-Nov	So You Want To Be A Professor? Student Lectures	?????		
22	3-Dec	So You Want To Be A Professor? Student Lectures	?????		
23	5-Dec	So You Want To Be A Professor? Student Lectures	?????		
24	10-Dec	So You Want To Be A Professor? Student Lectures	?????		
25	12-Dec	Last Day of Class General Wrapup, Cats and Dogs, Critique	Smith		
	17-21 Dec	Finals Week - No Exam for 22.S904 (22.213)			

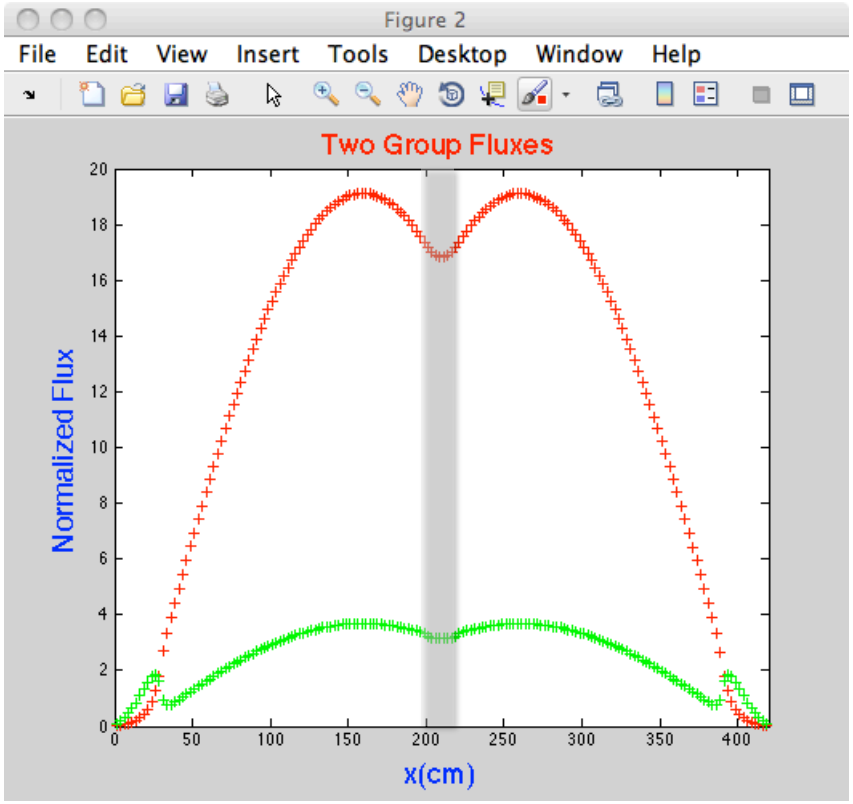
PSET # 2
SOLVING 1-D, 2-GROUP FINITE-DIFFERENCE EQUATIONS

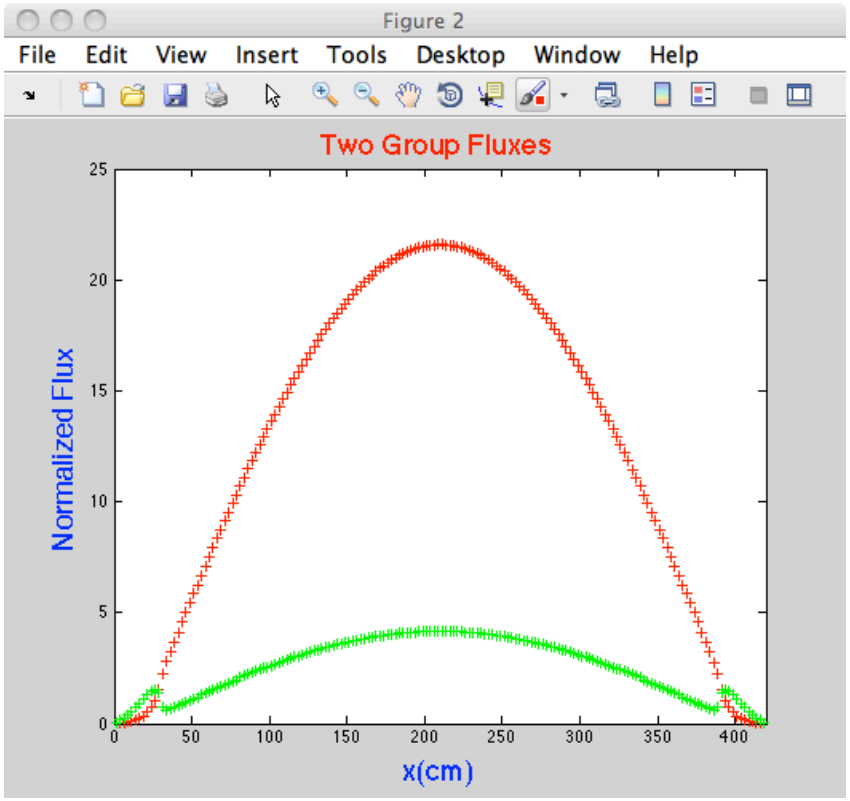
Due: Sept 24, 2012

Massachusetts
Institute of
Technology

Pset 2: Starting Point for Transient Calculations: Steady-State Solutions

Transient Control Rod Withdrawal in a 1-D Reflected Core: Beginning/Ending Shapes





Simulations in 1-D:

Static Solutions

Dynamic Solutions

Point-Kinetics

Generalized Point-Kinetics

Improved Quasi-Static Methods

Pset 2

Solve the Rodded and Unrodded, 1-D, 2-Group Diffusion Problems for the Following Materials, Geometry, and b.c.s

D1, D2, Sigma-a1, Sigma-a2, Sig1→2, Nu*sigma-f1, Nu*sigma-f2

```
%
% define two group cross sections for all potential materials
%
  nmat=5;
  xs(1,1)=1.300; xs(2,1)=0.500; xs(3,1)=0.0098; xs(4,1)=0.114; xs(5,1)=0.022; xs(6,1)=0.006; xs(7,1)=0.1950;  % 3% enriched fuel
  xs(1,2)=1.300; xs(2,2)=0.500; xs(3,2)=0.0105; xs(4,2)=0.134; xs(5,2)=0.022; xs(6,2)=0.008; xs(7,2)=0.2380;  % 4% enriched fuel
  xs(1,3)=1.500; xs(2,3)=0.500; xs(3,3)=0.0002; xs(4,3)=0.010; xs(5,3)=0.032; xs(6,3)=0.000; xs(7,3)=0.0000;  % water
  xs(1,4)=1.300; xs(2,4)=0.500; xs(3,4)=999.99; xs(4,4)=999.9; xs(5,4)=0.020; xs(6,4)=0.000; xs(7,4)=0.0000;  % black absorber
  xs(1,5)=1.300; xs(2,5)=0.500; xs(3,5)=0.0098; xs(4,5)=0.118; xs(5,5)=0.022; xs(6,5)=0.006; xs(7,5)=0.1950;  % 3% enriched + rod
%
% define problem geometry and assign materials:
%
%   #zones=NZONE w(nzone),n(nzone), mat(nzone)
%
%  bc | slab 1| slab 2| slab 3| ..... slab(NZONE) | bc
%

  NZONE=5;                                          % numer of material zones
  w(1)= 30; w(2)=170; w(3)=20;  w(4)=170; w(5)=30;  % width    per zone
  n(1)= 3;  n(2)=17;  n(3)=2;   n(4)=17;  n(5)=3;   % mesh     per zone
  m(1)= 3;  m(2)=1 ;  m(3)=5;   m(4)=1;   m(5)=3;   % material per zone
  n=n*4;                                            % mesh refinement factor

  bc=2;         % 0=zero flux, 1=zero incoming, 2=reflective bc
```

1. b.c. | Reflector | Fuel-1 | Rod | Fuel-1| Reflector | b.c.
2. Cross sections, zone widths, and cross section given above
3. Reflective b.c. on outer surfaces (you will see why later)

Pset 2

Write your own diffusion solver (in any language you choose) using power iteration with P-J and G-S iterative flux inversion

PART A: Difference Equations

- Derive the expression for the first-order finite-difference net current at a nodal interface for the case of variable mesh spacing/material properties.

PART B: Spatial Convergence

- Plot iteratively-converged eigenvalue and L_2 norm of nodal power error (using 10 cm nodes) vs. mesh spacing until the L_2 norm of error is converged to < 1.e-6 for the rodded and unrodded cores.

PART C: Dominance Ratios

- Plot the asymptotic dominance ratio vs. mesh spacing for the rodded and unrodded cores.

Pset 2

PART D: Iterative Convergence of P-J

- Plot the number of fission source iterations needed to achieve L_2 norm of changes of nodal powers for successive fission source iterations < 1.e-6 vs. flux iteration point-wise L_2 norm for flux convergence criteria of 1.e-1, 1.e-2, 1.e-3, 1.e-4, and 1.e-5 for the rodded and unrodded cores.

PART E: Iterative Convergence of G-S

- Plot the number of fission source iterations needed to achieve L_2 norm of changes of nodal powers for successive fission source iterations < 1.e-6 vs. flux iteration point-wise L_2 norm for flux convergence criteria of 1.e-1, 1.e-2, 1.e-3, 1.e-4, and 1.e-5 for the rodded and unrodded cores.

PART F: Real vs. Adjoint Fluxes

1. What are the spatially and iteratively converged real and adjoint eigenvalues for the rodded and unrodded problems?

2. What is the static rod worth in pcm?

3. Plot the spatially and iteratively converged real and adjoint fluxes for the rodded and unrodded problems.

Today's Lecture: Goals

- Questions about PSet 2 assignment:1-D, 2-group finite-difference
- Derive time-dependent 1-D, 2-group finite-difference diffusion equations
- Derive time-dependent delayed precursor equations
- Motivate fully-implicit (F-I) time integration
- Explore direct solutions of F-I diffusion equations in Matlab
  - Holding steady-state solutions
  - Debugging suggestions
  - Time-dependent solutions
- Examine characteristics of solutions to rod withdrawal/insertion problem
- Discuss upcoming lectures and paths forward

Transient Neutron Diffusion Equations

Continuous energy:

$$\frac{\partial}{\partial t}\left\langle \frac{1}{v(\vec{r},E)}\phi(\vec{r},E,t)\right\rangle = \nabla\cdot D(\vec{r},t)\nabla\phi(\vec{r},E,t) - \Sigma_t(\vec{r},E,t)\,\phi(\vec{r},E,t) + \int dE'\,\Sigma_s(\vec{r},E'\rightarrow E,t)\,\phi(\vec{r},E',t)$$

$$+[1-\beta(\vec{r},t)]\frac{\chi^p(E)}{k_{crit}}\int dE'\,\nu\Sigma_f(\vec{r},E',t)\,\phi(\vec{r},E',t) + \sum_i^I \chi_i^d(E)\,\lambda_i C_i(\vec{r},t)$$

$$\frac{\partial}{\partial t}C_i(\vec{r},t) = -\lambda_i C_i(\vec{r},t) + \frac{\beta_i(\vec{r},t)]}{k_{crit}}\int dE'\,\nu\Sigma_f(\vec{r},E',t)\,\phi(\vec{r},E',t), \quad i=1,\ldots,I$$

Multi-group formulation:

$$\frac{\partial}{\partial t}\left\langle \frac{1}{v_g(\vec{r},t)}\phi_g(\vec{r},t)\right\rangle = \nabla\cdot D_g(\vec{r},t)\nabla\phi_g(\vec{r},t) - \Sigma_{t,g}(\vec{r},t)\,\phi_g(\vec{r},t) + \sum_{g'=1}^{G}\Sigma_{s,g'\rightarrow g}(\vec{r},t)\,\phi_{g'}(\vec{r},t)$$

$$+[1-\beta(\vec{r},t)]\frac{\chi_g^p}{k_{crit}}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\vec{r},t)\,\phi_{g'}(\vec{r},t) + \sum_i^I \chi_{i,g}^d\,\lambda_i C_i(\vec{r},t), \quad g=1,\ldots,G$$

$$\frac{\partial}{\partial t}C_i(\vec{r},t) = -\lambda_i C_i(\vec{r},t) + \frac{\beta_i(\vec{r},t)]}{k_{crit}}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\vec{r},t)\,\phi_{g'}(\vec{r},t), \quad i=1,\ldots,I$$

Transient Neutron Diffusion Equations

Transforming to removal cross sections:

$$\frac{\partial}{\partial t}\left\langle \frac{1}{v_g(\vec{r},t)}\phi_g(\vec{r},t)\right\rangle = \nabla\cdot D_g(\vec{r},t)\nabla\phi_g(\vec{r},t) - \Sigma_{r,g}(\vec{r},t)\,\phi_g(\vec{r},t) + \sum_{g'\neq g}^{G}\Sigma_{s,g'\to g}(\vec{r},t)\,\phi_{g'}(\vec{r},t)$$

$$+[1-\beta(\vec{r},t)]\frac{\chi_g^p}{k_{crit}}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\vec{r},t)\,\phi_{g'}(\vec{r},t) + \sum_{i}^{I}\chi_{i,g}^d\,\lambda_i C_i(\vec{r},t), \quad g=1,\ldots,G$$

$$\frac{\partial}{\partial t}C_i(\vec{r},t) = -\lambda_i C_i(\vec{r},t) + \frac{\beta_i(\vec{r},t)]}{k_{crit}}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\vec{r},t)\,\phi_{g'}(\vec{r},t), \quad i=1,\ldots,I$$

Remove temporal dependency of beta and velocity:

$$\frac{1}{v_g(\vec{r})}\frac{\partial}{\partial t}\left\langle \phi_g(\vec{r},t)\right\rangle = \nabla\cdot D_g(\vec{r},t)\nabla\phi_g(\vec{r},t) - \Sigma_{r,g}(\vec{r},t)\,\phi_g(\vec{r},t) + \sum_{g'\neq g}^{G}\Sigma_{s,g'\to g}(\vec{r},t)\,\phi_{g'}(\vec{r},t)$$

$$+[1-\beta(\vec{r})]\frac{\chi_g^p}{k_{crit}}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\vec{r},t)\,\phi_{g'}(\vec{r},t) + \sum_{i}^{I}\chi_{i,g}^d\,\lambda_i C_i(\vec{r},t), \quad g=1,\ldots,G$$

$$\frac{\partial}{\partial t}C_i(\vec{r},t) = -\lambda_i C_i(\vec{r},t) + \frac{\beta_i(\vec{r})]}{k_{crit}}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\vec{r},t)\,\phi_{g'}(\vec{r},t), \quad i=1,\ldots,I$$

1 index of ν 6 vs. 8 groups

↳ Due to loading

Transient Multigroup Neutron Diffusion Equations

Continuous equation in time:

$$\frac{1}{v_g(\vec{r})}\frac{\partial}{\partial t}\left\langle \phi_g(\vec{r},t)\right\rangle = \nabla\cdot D_g(\vec{r},t)\nabla\phi_g(\vec{r},t) - \Sigma_{r,g}(\vec{r},t)\,\phi_g(\vec{r},t) + \sum_{g'\neq g}^{G}\Sigma_{s,g'\to g}(\vec{r},t)\,\phi_{g'}(\vec{r},t)$$

$$+[1-\beta(\vec{r})]\frac{\chi_g^p}{k_{crit}}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\vec{r},t)\,\phi_{g'}(\vec{r},t) + \sum_{i}^{I}\chi_{i,g}^d\,\lambda_i C_i(\vec{r},t), \quad g=1,\ldots,G$$

$$\frac{\partial}{\partial t}C_i(\vec{r},t) = -\lambda_i C_i(\vec{r},t) + \frac{\beta_i(\vec{r})]}{k_{crit}}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\vec{r},t)\,\phi_{g'}(\vec{r},t), \quad i=1,\ldots,I$$

Is transformed in to first-order temporal difference:

$$\frac{1}{v_g(\vec{r})}\frac{\phi_g(\vec{r},t_{n+1})-\phi_g(\vec{r},t_n)}{t_{n+1}-t_n} = \nabla\cdot D_g(\vec{r},t)\nabla\phi_g(\vec{r},t) - \Sigma_{r,g}(\vec{r},t)\,\phi_g(\vec{r},t) + \sum_{g'\neq g}^{G}\Sigma_{s,g'\to g}(\vec{r},t)\,\phi_{g'}(\vec{r},t)$$

$$+[1-\beta(\vec{r})]\frac{\chi_g^p}{k_{crit}}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\vec{r},t)\,\phi_{g'}(\vec{r},t) + \sum_{i}^{I}\chi_{i,g}^d\,\lambda_i C_i(\vec{r},t), \quad g=1,\ldots,G$$

$$\frac{C_i(\vec{r},t_{n+1})-C_i(\vec{r},t_n)}{t_{n+1}-t_n} = -\lambda_i C_i(\vec{r},t) + \frac{\beta_i(\vec{r})]}{k_{crit}}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\vec{r},t)\,\phi_{g'}(\vec{r},t), \quad i=1,\ldots,I$$

Transient Finite-Difference Neutron Diffusion Equations

Explicit time integration (all rhs fluxes at old time step)

$$\frac{1}{v_g(\vec{r})}\frac{\phi_g(\vec{r},t_{n+1})-\phi_g(\vec{r},t_n)}{t_{n+1}-t_n}=\nabla\cdot D_g(\vec{r},t_n)\nabla\phi_g(\vec{r},t_n)-\Sigma_{r,g}(\vec{r},t_n)\,\phi_g(\vec{r},t_n)+\sum_{g'\neq g}^{G}\Sigma_{s,g'\to g}(\vec{r},t_n)\,\phi_{g'}(\vec{r},t_n)$$

$$+[1-\beta(\vec{r})]\frac{\chi_g^p}{k_{crit}}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\vec{r},t_n)\,\phi_{g'}(\vec{r},t_n)+\sum_{i}^{I}\chi_{i,g}^d\,\lambda_i C_i(\vec{r},t_n),\qquad g=1,\ldots,G$$

$$\frac{C_i(\vec{r},t_{n+1})-C_i(\vec{r},t_n)}{t_{n+1}-t_n}=-\lambda_i C_i(\vec{r},t_n)+\frac{\beta_i(\vec{r})]}{k_{crit}}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\vec{r},t_n)\,\phi_{g'}(\vec{r},t_n),\qquad i=1,\ldots,I$$

XS at t_{n+1}

Implicit time integration (all rhs fluxes at new time step)

$$\frac{1}{v_g(\vec{r})}\frac{\phi_g(\vec{r},t_{n+1})-\phi_g(\vec{r},t_n)}{t_{n+1}-t_n}=\nabla\cdot D_g(\vec{r},t_{n+1})\nabla\phi_g(\vec{r},t_{n+1})-\Sigma_{r,g}(\vec{r},t_{n+1})\,\phi_g(\vec{r},t_{n+1})+\sum_{g'\neq g}^{G}\Sigma_{s,g'\to g}(\vec{r},t_{n+1})\,\phi_{g'}(\vec{r},t_{n+1})$$

$$+[1-\beta(\vec{r})]\frac{\chi_g^p}{k_{crit}}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\vec{r},t_{n+1})\,\phi_{g'}(\vec{r},t_{n+1})+\sum_{i}^{I}\chi_{i,g}^d\,\lambda_i C_i(\vec{r},t_{n+1}),\qquad g=1,\ldots,G$$

$$\frac{C_i(\vec{r},t_{n+1})-C_i(\vec{r},t_n)}{t_{n+1}-t_n}=-\lambda_i C_i(\vec{r},t_{n+1})+\frac{\beta_i(\vec{r})]}{k_{crit}}\sum_{g'=1}^{G}\nu\Sigma_{f,g'}(\vec{r},t_{n+1})\,\phi_{g'}(\vec{r},t_{n+1}),\qquad i=1,\ldots,I$$

Transient Finite-Difference 2-group Neutron Diffusion Equations

Using the standard definition of 2- group parameters:

$\chi_1 = 1.0$ and $\chi_2 = 0.0$ and $\Sigma_{2\to 1}(\vec{r}) = 0$

$$\frac{\phi_1(\vec{r},t_{n+1})}{v_1(\vec{r})\Delta_t} - \nabla\cdot D_1(\vec{r},t_{n+1})\nabla\phi_1(\vec{r},t_{n+1}) + \Sigma_{r,1}(\vec{r},t_{n+1})\,\phi_1(\vec{r},t_{n+1})$$

$$-[1-\beta(\vec{r})]\frac{1}{k_{crit}}\left[\upsilon\Sigma_{f,1}(\vec{r},t_{n+1})\,\phi_f(\vec{r},t_{n+1}) + \upsilon\Sigma_{f,2}(\vec{r},t_{n+1})\,\phi_2(\vec{r},t_{n+1})\right] - \sum_i^I \lambda_i C_i(\vec{r},t_{n+1}) = \frac{\phi_1(\vec{r},t_n)}{v_1(\vec{r})\Delta_t}$$

$$\frac{\phi_2(\vec{r},t_{n+1})}{v_2(\vec{r})\Delta_t} - \nabla\cdot D_2(\vec{r},t_{n+1})\nabla\phi_2(\vec{r},t_{n+1}) + \Sigma_{a,2}(\vec{r},t_{n+1})\,\phi_2(\vec{r},t_{n+1}) - \Sigma_{1\to 2}(\vec{r},t_{n+1})\,\phi_1(\vec{r},t_{n+1}) = \frac{\phi_2(\vec{r},t_n)}{v_2(\vec{r})\Delta_t}$$

$$C_i(\vec{r},t_{n+1}) = \frac{\beta_i(\vec{r})\Delta_t}{(1+\lambda_i\Delta_t)k_{crit}}\left[\upsilon\Sigma_{f,1}(\vec{r},t_{n+1})\,\phi_1(\vec{r},t_{n+1}) + \upsilon\Sigma_{f,2}(\vec{r},t_{n+1})\,\phi_2(\vec{r},t_{n+1})\right] + \frac{C_i(\vec{r},t_n)}{(1+\lambda_i\Delta_t)}, \qquad i = 1,\ldots,I$$

Transient Finite-Difference 2-group Neutron Diffusion Equations

Dropping the cross section temporal arguments (all at advanced time step)

$$\frac{\phi_1(\vec{r},t_{n+1})}{v_1(\vec{r})\Delta_t} - \nabla\cdot D_1(\vec{r})\nabla\phi_1(\vec{r},t_{n+1}) + \Sigma_{r,1}(\vec{r})\,\phi_1(\vec{r},t_{n+1})$$

$$-[1-\beta(\vec{r})]\frac{1}{k_{crit}}\left[\upsilon\Sigma_{f,1}(\vec{r})\,\phi_f(\vec{r},t_{n+1}) + \upsilon\Sigma_{f,2}(\vec{r})\,\phi_2(\vec{r},t_{n+1})\right] - \sum_i^I \lambda_i C_i(\vec{r},t_{n+1}) = \frac{\phi_1(\vec{r},t_n)}{v_1(\vec{r})\Delta_t}$$

$$\frac{\phi_2(\vec{r},t_{n+1})}{v_2(\vec{r})\Delta_t} - \nabla\cdot D_2(\vec{r})\nabla\phi_2(\vec{r},t_{n+1}) + \Sigma_{a,2}(\vec{r})\,\phi_2(\vec{r},t_{n+1}) - \Sigma_{1\to 2}(\vec{r})\,\phi_1(\vec{r},t_{n+1}) = \frac{\phi_2(\vec{r},t_n)}{v_2(\vec{r})\Delta_t}$$

$$C_i(\vec{r},t_{n+1}) = \frac{\beta_i(\vec{r})\Delta_t}{(1+\lambda_i\Delta_t)k_{crit}}\left[\upsilon\Sigma_{f,1}(\vec{r})\,\phi_1(\vec{r},t_{n+1}) + \upsilon\Sigma_{f,2}(\vec{r})\,\phi_2(\vec{r},t_{n+1})\right] + \frac{C_i(\vec{r},t_n)}{(1+\lambda_i\Delta_t)}, \quad i=1,\dots,I$$

need to include

Discrete operator allows us to do this

Formulate a 10 block matrix equations from 2 group fluxes and 8 delayed precursor equations?

No, substitute precursor equations into group 1 flux equation, so we only have a 2 block matrix equations to solve for advanced time step fluxes.

Precursor equations can then be solve one mesh point and group at a time for advanced time step precursors.

Transient Finite-Difference 2-group Neutron Diffusion Equations

Substituting delayed precursor into the nodal balance equations:

$$\frac{\phi_1(\vec{r},t_{n+1})}{v_1(\vec{r})\Delta_t} - \nabla\cdot D_1(\vec{r})\nabla\phi_1(\vec{r},t_{n+1}) + \Sigma_{r,1}(\vec{r})\,\phi_1(\vec{r},t_{n+1})$$

$$-\left(\frac{[1-\beta(\vec{r})]}{k_{crit}} + \sum_i^I \lambda_i \frac{\beta_i(\vec{r})\Delta_t}{(1+\lambda_i\Delta_t)k_{crit}}\right)\left[\upsilon\Sigma_{f,1}(\vec{r})\,\phi_1(\vec{r},t_{n+1}) + \upsilon\Sigma_{f,2}(\vec{r})\,\phi_2(\vec{r},t_{n+1})\right] = \frac{\phi_1(\vec{r},t_n)}{v_1(\vec{r})\Delta_t} + \sum_i^I \lambda_i \frac{C_i(\vec{r},t_n)}{(1+\lambda_i\Delta_t)}$$

$$\frac{\phi_2(\vec{r},t_{n+1})}{v_2(\vec{r})\Delta_t} - \nabla\cdot D_2(\vec{r})\nabla\phi_2(\vec{r},t_{n+1}) + \Sigma_{a,2}(\vec{r})\,\phi_2(\vec{r},t_{n+1}) - \Sigma_{1\to 2}(\vec{r})\,\phi_1(\vec{r},t_{n+1}) = \frac{\phi_2(\vec{r},t_n)}{v_2(\vec{r})\Delta_t}$$

$$C_i(\vec{r},t_{n+1}) = \frac{\beta_i(\vec{r})\Delta_t}{(1+\lambda_i\Delta_t)k_{crit}}\left[\upsilon\Sigma_{f,1}(\vec{r})\,\phi_1(\vec{r},t_{n+1}) + \upsilon\Sigma_{f,2}(\vec{r})\,\phi_2(\vec{r},t_{n+1})\right] + \frac{C_i(\vec{r},t_n)}{(1+\lambda_i\Delta_t)}, \quad i = 1,\ldots,I$$

Transient Finite-Difference 2-group Neutron Diffusion Equations

If we redefine "pseudo fission" and "pseudo removal" cross sections

$$\upsilon\hat{\Sigma}_{f,g}(\vec{r}) \equiv \left(\frac{[1-\beta(\vec{r})]}{k_{crit}} + \sum_i^I \frac{\beta_i(\vec{r})\lambda_i\Delta_t}{(1+\lambda_i\Delta_t)k_{crit}} \right) \upsilon\Sigma_{f,g}(\vec{r})$$

$$\hat{\Sigma}_{r,1}(\vec{r}) \equiv \left\langle \Sigma_{r,1}(\vec{r}) + \frac{1}{v_1(\vec{r})\Delta_t} \right\rangle \qquad \hat{\Sigma}_{r,2}(\vec{r}) \equiv \left\langle \Sigma_{a,2}(\vec{r}) + \frac{1}{v_2(\vec{r})\Delta_t} \right\rangle$$

$$S_1 \equiv \left\langle \frac{\phi_1(\vec{r},t_n)}{v_1(\vec{r})\Delta_t} + \sum_i^I \lambda_i \frac{C_i(\vec{r},t_n)}{(1+\lambda_i\Delta_t)} \right\rangle \qquad S_2 \equiv \left\langle \frac{\phi_2(\vec{r},t_n)}{v_2(\vec{r})\Delta_t} \right\rangle$$

Kinetic delayed term

And collect some terms:

$$-\nabla\cdot D_1(\vec{r})\nabla\phi_1(\vec{r},t_{n+1}) + \left(\hat{\Sigma}_{r,1}(\vec{r}) - \upsilon\hat{\Sigma}_{f,1}(\vec{r})\right)\phi_1(\vec{r},t_{n+1}) - \upsilon\hat{\Sigma}_{f,2}(\vec{r})\,\phi_2(\vec{r},t_{n+1}) = S_1$$

$$-\nabla\cdot D_2(\vec{r})\nabla\phi_2(\vec{r},t_{n+1}) + \hat{\Sigma}_{r,2}(\vec{r})\phi_2(\vec{r},t_{n+1}) - \Sigma_{1\to 2}(\vec{r})\,\phi_1(\vec{r},t_{n+1}) = S_2$$

$$C_i(\vec{r},t_{n+1}) = \frac{\beta_i(\vec{r})\Delta_t}{(1+\lambda_i\Delta_t)k_{crit}}\left[\upsilon\Sigma_{f,1}(\vec{r})\,\phi_1(\vec{r},t_{n+1}) + \upsilon\Sigma_{f,2}(\vec{r})\,\phi_2(\vec{r},t_{n+1})\right] + \frac{C_i(\vec{r},t_n)}{(1+\lambda_i\Delta_t)}, \qquad i = 1,\ldots,I$$

Transient Finite-Difference 2-group Neutron Diffusion Equations

Now, making the first-order finite-difference spatial approximation and integrating over mesh "m":

$$-\hat{D}_1^{m-1.m}\phi_1^{m-1}(t_{n+1})+\left[\left(\hat{\Sigma}_{r1}^m-\upsilon\hat{\Sigma}_{f1}^m\right)\Delta_x+\hat{D}_1^{m-1.m}+\hat{D}_1^{m,m+1}\right]\phi_1^m(t_{n+1})-\hat{D}_1^{m,m+1}\phi_1^{n+1}(t_{n+1})-\upsilon\hat{\Sigma}_{f2}^m\Delta_x\phi_1^m(t_{n+1})=S_1^m\Delta_x$$

$$-\hat{D}_2^{m-1.m}\phi_2^{m-1}(t_{n+1})+\left[\hat{\Sigma}_{r2}^m\Delta_x+\hat{D}_2^{m-1.m}+\hat{D}_2^{n,n+1}\right]\phi_2^m(t_{n+1})-\hat{D}_2^{m,m+1}\phi_2^{n+1}(t_{n+1})-\Sigma_{12}^m\Delta_x\phi_1^m(t_{n+1})=S_2^m\Delta_x$$

$$C_i^m(t_{n+1})=+\frac{\beta_i^m(\bar{r})\Delta_t}{(1+\lambda_i\Delta_t)k_{crit}}\left[\upsilon\Sigma_{f1}^m\phi_1^m(t_{n+1})+\upsilon\Sigma_{f2}^m\phi_2^m(t_{n+1})\right]+\frac{C_i^m(t_n)}{(1+\lambda_i\Delta_t)},\qquad i=1,\ldots,I$$

Comparing back to earlier steady-state equations with a fixed sources:

$$-\hat{D}_1^{m-1.m}\phi_1^{m-1}+\left[\Sigma_{r1}^m\Delta+\hat{D}_1^{m-1.m}+\hat{D}_1^{n,n+1}\right]\phi_1^m-\hat{D}_1^{m,m+1}\phi_1^{m+1}=\Sigma_{f1}^m\Delta\phi_1^m+\nu\Sigma_{f2}^m\Delta\phi_2^m+S_1^m\Delta$$

$$-\hat{D}_2^{m-1.m}\phi_2^{m-1}+\left[\Sigma_{a2}^m\Delta+\hat{D}_2^{m-1.m}+\hat{D}_2^{m,m+1}\right]\phi_2^n-\hat{D}_2^{m,m+1}\phi_2^{m+1}=\Sigma_{s12}^m\Delta\phi_1^m+S_2^m\Delta$$

We can manipulate the steady-state equations in to the form:

$$-\hat{D}_1^{m-1.m}\phi_1^{m-1}+\left[\left(\Sigma_{r1}^m-\Sigma_{f1}^m\right)\Delta_x+\hat{D}_1^{m-1.m}+\hat{D}_1^{m,m+1}\right]\phi_1^m-\hat{D}_1^{m,m+1}\phi_1^{m+1}-\nu\Sigma_{f2}^m\Delta_x\phi_2^m=S_1^m\Delta_x$$

$$-\hat{D}_2^{m-1.m}\phi_2^{m-1}+\left[\Sigma_{a2}^m\Delta_x+\hat{D}_2^{m-1.m}+\hat{D}_2^{m,m+1}\right]\phi_2^m-\hat{D}_2^{m,m+1}\phi_2^{m+1}-\Sigma_{s12}^m\Delta_x\phi_1^m=S_2^m\Delta_x$$

So the transient equations are identical in form to steady-state equations!

Transient vs. Steady State Finite-Difference 2-group Matrix Equations

$$-\hat{D}_1^{m-1.m}\phi_1^{m-1}(t_{n+1})+\left[\left(\hat{\Sigma}_{r1}^m-\upsilon\hat{\Sigma}_{f1}^m\right)\Delta_x+\hat{D}_1^{m-1.m}+\hat{D}_1^{m,m+1}\right]\phi_1^m(t_{n+1})-\hat{D}_1^{m,m+1}\phi_1^{n+1}(t_{n+1})-\upsilon\hat{\Sigma}_{f2}^m\Delta_x\phi_1^m(t_{n+1})=S_1^m(t_n)\Delta_x$$

$$-\hat{D}_2^{m-1.m}\phi_2^{m-1}(t_{n+1})+\left[\hat{\Sigma}_{r2}^m\Delta_x+\hat{D}_2^{m-1.m}+\hat{D}_2^{n,n+1}\right]\phi_2^m(t_{n+1})-\hat{D}_2^{m,m+1}\phi_2^{n+1}(t_{n+1})-\Sigma_{12}^m\Delta_x\phi_1^m(t_{n+1})=S_2^m(t_n)\Delta_x$$

But recall from previous lecture the matrix form of steady-state equations:

$$\begin{bmatrix}[L+D+U]_1 & [0]\\ -[T]_2 & [L+D+U]_1\end{bmatrix}\begin{bmatrix}[\phi_1]\\ [\phi_2]\end{bmatrix}=\begin{bmatrix}[M]_1 & [M]_2\\ [0] & [0]\end{bmatrix}\begin{bmatrix}[\phi_1]\\ [\phi_2]\end{bmatrix}+\begin{bmatrix}[S_1]\\ [S_2]\end{bmatrix}$$

$$\begin{bmatrix}[L+\hat{D}+U]_1 & -[M]_2\\ -[T]_2 & [L+D+U]_1\end{bmatrix}\begin{bmatrix}[\phi_1]\\ [\phi_2]\end{bmatrix}=\begin{bmatrix}[S_1]\\ [S_2]\end{bmatrix}$$

$$C_i^m(t_{n+1})=+\frac{\beta_i^m(\vec{r})\Delta_t}{(1+\lambda_i\Delta_t)k_{crit}}\left[\upsilon\Sigma_{f1}^m\phi_1^m(t_{n+1})+\upsilon\Sigma_{f2}^m\phi_1^m(t_{n+1})\right]+\frac{C_i^m(t_n)}{(1+\lambda_i\Delta_t)},\qquad i=1,\ldots,I$$

Precursors need not be matrix equations, as there is no spatial coupling.

Transient vs. Steady State Finite-Difference 2-group Matrix Equations

$$-\hat{D}_1^{m-1.m}\phi_1^{m-1}(t_{n+1})+\left[\left(\hat{\Sigma}_{r1}^m-\upsilon\hat{\Sigma}_{f1}^m\right)\Delta_x+\hat{D}_1^{m-1.m}+\hat{D}_1^{m,m+1}\right]\phi_1^m(t_{n+1})-\hat{D}_1^{m,m+1}\phi_1^{n+1}(t_{n+1})-\upsilon\hat{\Sigma}_{f2}^m\Delta_x\phi_1^m(t_{n+1})=S_1^m(t_n)\Delta_x$$

$$-\hat{D}_2^{m-1.m}\phi_2^{m-1}(t_{n+1})+\left[\hat{\Sigma}_{r2}^m\Delta_x+\hat{D}_2^{m-1.m}+\hat{D}_2^{n,n+1}\right]\phi_2^m(t_{n+1})-\hat{D}_2^{m,m+1}\phi_2^{n+1}(t_{n+1})-\Sigma_{12}^m\Delta_x\phi_1^m(t_{n+1})=S_2^m(t_n)\Delta_x$$

Each group flux matrix equation is tri-diagonal (ignoring group transfer terms):

$$\begin{bmatrix}\left[L+\hat{D}+U\right]_1 & -[\hat{M}]_2\\ -[T]_2 & \left[L+\hat{D}+U\right]_1\end{bmatrix}\begin{bmatrix}[\phi_1]\\ [\phi_2]\end{bmatrix}^{t_{n+1}}=\begin{bmatrix}[S_1]\\ [S_2]\end{bmatrix}^{t_n}$$

$$C_i^m(t_{n+1})=\frac{\beta_i^m(\vec{r})\Delta_t}{(1+\lambda_i\Delta_t)k_{crit}}\left[\upsilon\Sigma_{f1}^m\phi_1^m(t_{n+1})+\upsilon\Sigma_{f2}^m\phi_2^m(t_{n+1})\right]+\frac{C_i^m(t_n)}{(1+\lambda_i\Delta_t)},\qquad i=1,\ldots,I$$

Each precursor matrix equation is strictly diagonal (no spatial coupling):

$$[C_i]=[D_i]\left\{[M]_1[\phi_1]^{t_{n+1}}+[M]_2[\phi_1]^{t_{n+1}}\right\}+[Q_i]^{t_n},\qquad i=1,\ldots,I$$

MATLAB coding of transient case: start from previous steady-state solver

```
function OneD_diffusion_rodtransient
%
% define two group cross sections for all potential materials
%
  nmat=5;
  xs(1,1)=1.300; xs(2,1)=0.500; xs(3,1)=0.0098; xs(4,1)=0.114; xs(5,1)=0.022; xs(6,1)=0.006; xs(7,1)=0.1950;  % 3% enriched fuel
  xs(1,2)=1.300; xs(2,2)=0.500; xs(3,2)=0.0105; xs(4,2)=0.134; xs(5,2)=0.022; xs(6,2)=0.008; xs(7,2)=0.2380;  % 4% enriched fuel
  xs(1,3)=1.500; xs(2,3)=0.500; xs(3,3)=0.0002; xs(4,3)=0.010; xs(5,3)=0.032; xs(6,3)=0.000; xs(7,3)=0.0000;  % water
  xs(1,4)=1.300; xs(2,4)=0.500; xs(3,4)=999.99; xs(4,4)=999.9; xs(5,4)=0.020; xs(6,4)=0.000; xs(7,4)=0.0000;  % black absorber
  xs(1,5)=1.300; xs(2,5)=0.500; xs(3,5)=0.0098; xs(4,5)=0.118; xs(5,5)=0.022; xs(6,5)=0.006; xs(7,5)=0.1950;  % 3% enriched + rod
%
% define problem geometry and assign materials:
%
%   #zones=NZONE w(nzone),n(nzone), mat(nzone)
%
%  bc | slab 1| slab 2| slab 3| ..... slab(NZONE) | bc
%
  NZONE=5;                                          % numer of material zones
  w(1)= 30; w(2)=170; w(3)=20;  w(4)=170; w(5)=30;  % width    per zone
  n(1)= 3;  n(2)=17;  n(3)=2;   n(4)=17;  n(5)=3;   % mesh     per zone
  m(1)= 3;  m(2)=1 ;  m(3)=5;   m(4)=1;   m(5)=3;   % material per zone
  n=n*2;                                            % mesh refinement factor
  bc=2;                                             % 0=zero flux, 1=zero incoming, 2=reflective bc

  [A,M,x,NP,L,h,matvec] = Diffusion_setup (xs,w,n,m,NZONE,bc);
%
%  solve steady-state problem
%
   eps=1.e-6; maxouter=5000; [phi,keff] = OneD_matlab (A,M,x,NP,L,eps,maxouter);
```

```
function [phi,keff] = OneD_matlab(A,M,x,NP,L,criteria,maxouter,keff,phi)
%
% solve eignevalue problem iteratively, Matlab inversion of flux matrix
%
  fprintf ('\n\n%s',' Iterative Fission Source Solution');
  epsold=10.; phi(1:2*NP,1)=rand(2*NP,1); sold=(M*phi); sumold=sum(sold);
  sold=sold/(sumold/NP); sumold=NP; epssave([1:10000])=0; drsave([1:10000])=0.0;
  for iter=1:maxouter
      phi=A^-1*sold;
      snew=M*phi; sumnew=sum(snew); keff=sumnew/sumold;
      eps=0.0;for np=1:NP; if snew(np) > 0; eps=eps+((sold(np)*keff)/snew(np)-1)^2; end; end; eps=(eps/NP)^.5;
      sold=snew/(sumnew/NP); approxdr=eps/epsold; epsold=eps;
      if iter > 5 && eps < criteria;  break; end
      epssave(iter)=eps; drsave(iter)=approxdr;
  end
  fprintf ('%s','  # FS iters, keff, eps, dr'); fprintf ('%14.7f',iter, keff,eps,approxdr);
  flux1=phi(1:NP); flux2=phi(NP+1:2*NP);
  max1=max(flux1); max2=max(flux2);
  mylinplot (1,x,flux1,flux2,'Initial: 2-Group Fluxes','x(cm)','Normalized Flux',0,L,0,0);
end
```

MATLAB coding of transient case: set kinetics parameters

```
%
% kinetics parameters
%
betai  = [.000218 .001023 .000605 .00131 .00220 .00060 .000540 .000152];   % no units, 8-group data
halfli = [55.6      24.5    16.3    5.21   2.37  1.04   0.424   0.195 ];   % half-life in seconds
decayi = log(2)./halfli;                                                   % sec-1
vel1   = 2200.*100.*(.100 /.0253)^.5; vel2   = 2200.*100.*(.100e6/.0253)^.5; % cm/sec
beta= sum(betai);
```

MATLAB coding of transient case: Setup Transient Matrix and Solve

```
%
%  solve transient problem
%
   deltat=0.1  ;  tend=50.0 ; nstep=tend/deltat;                    % set time integration parameters
   pertsl=2.0  ; pertel=4.0 ; pertxsl=+1.00*(xs(4,1)-xs(4,5))/(pertel-pertsl);  % set perturbation1 start, end del-xsa2
   power([1:nstep])=0; time([1:nstep])=0; C([1:8,1:2*NP])=0; S([1:2*NP])=0;

   M0=M/keff; FS=(M0*phi); power0=sum(FS);                          % save real fission matrix
   for i=1:8
       C(i,:)=(FS')*betai(i)/decayi(i);                             % initialize precursors [volume in M0, so in C(i)]
   end
   for n=1:NP;
      A(n,n)       =A(n,n)       +h(n)/(vel1*deltat);               % add pseudo absorption to group 1 removal
      A(NP+n,NP+n)=A(NP+n,NP+n)+h(n)/(vel2*deltat);                 % add pseudo absorption to group 2 removal
   end
   factor=(1.-beta)/keff; for i=1:8; factor=factor+betai(i)*decayi(i)*deltat/((1.+decayi(i)*deltat)*keff); end
   M=M*factor;                                                      % create pseduo fission matrix
   AA=A-M;                                                          % create base iteration coefficient matrix
   for step=1:nstep; time(step)=deltat*step;                        % loop over time steps
      for n=1:NP
        if (matvec(n) == 3)
           if (time(step) > pertsl && time(step)<= pertel)
             AA(NP+n,NP+n)=AA(NP+n,NP+n)+h(n)*pertxsl*deltat;       % move control rod out
           end
        end
        S(NP+n)=h(n)*phi(NP+n)/(vel2*deltat);                       % compute group 2 source vector
        S(n   )= h(n)*phi(n    )/(vel1*deltat);                     % compute group 1 source vector
        for i=1:8;
            S(n)=S(n)+decayi(i)*C(i,n)/(1.+decayi(i)*deltat);
        end
      end
      phi=(AA^(-1))*S';                                             % evaluate new time step flux vector
      FS=(M0*phi); power(step)=sum(FS);                             % compute normalize core poweer
      for i=1:8
         C(i,:)=(C(i,:)+(FS')*betai(i)*deltat)/(1.+decayi(i)*deltat);   % new time step precursor vector
      end
   end
```

$$\upsilon\hat{\Sigma}_{f,g}(\vec{r}) \equiv \left(\frac{[1-\beta(\vec{r})]}{k_{crit}} + \sum_i^I \frac{\beta_i(\vec{r})\lambda_i\Delta_t}{(1+\lambda_i\Delta_t)k_{crit}}\right)\upsilon\Sigma_{f,g}(\vec{r})$$

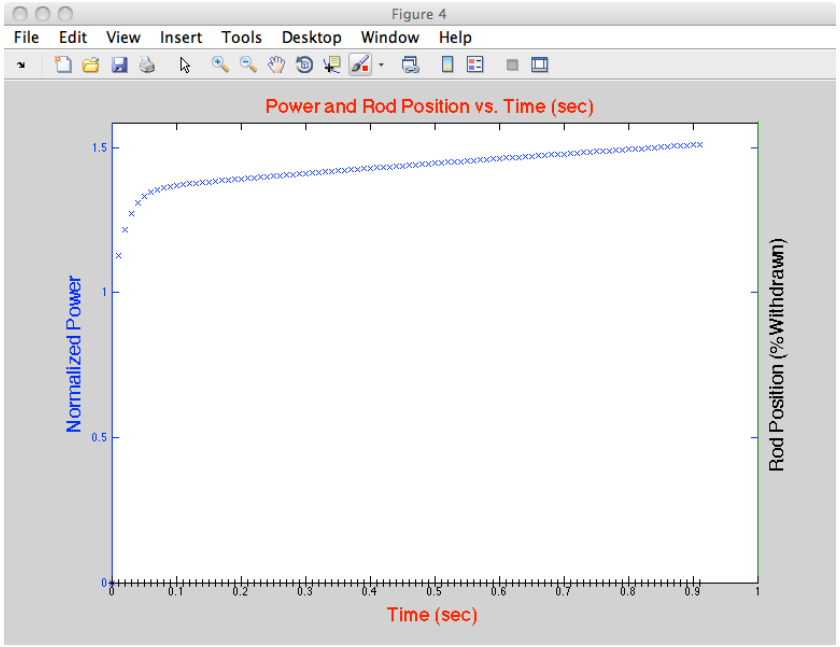
$$\hat{\Sigma}_{r,1}(\vec{r}) \equiv \left\langle \Sigma_{r,1}(\vec{r}) + \frac{1}{v_1(\vec{r})\Delta_t} \right\rangle \qquad \hat{\Sigma}_{r,2}(\vec{r}) \equiv \left\langle \Sigma_{a,2}(\vec{r}) + \frac{1}{v_2(\vec{r})\Delta_t} \right\rangle$$

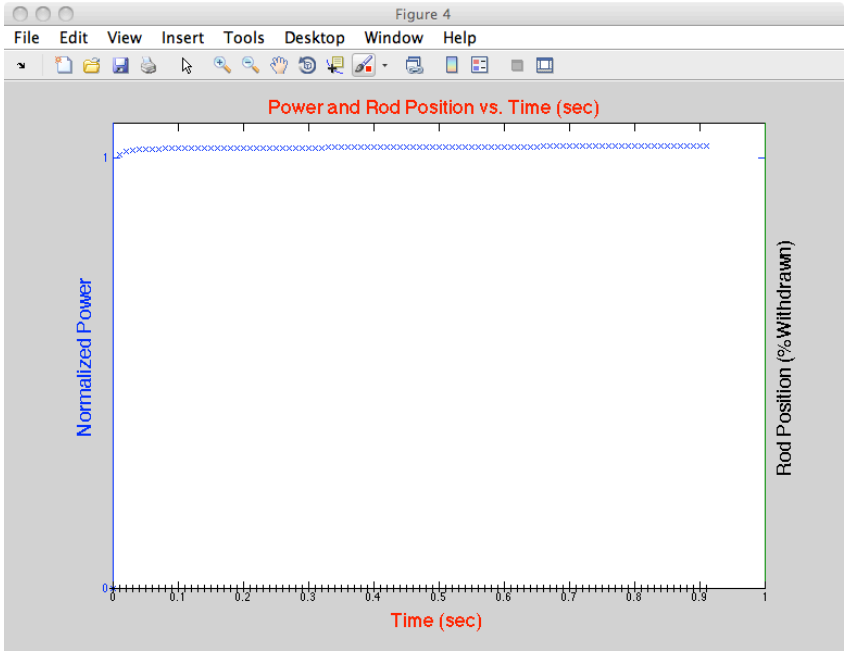
$$S_1 \equiv \left\langle \frac{\phi_1(\vec{r},t_n)}{v_1(\vec{r})\Delta_t} + \sum_i^I \lambda_i \frac{C_i(\vec{r},t_n)}{(1+\lambda_i\Delta_t)} \right\rangle \qquad S_2 \equiv \left\langle \frac{\phi_2(\vec{r},t_n)}{v_2(\vec{r})\Delta_t} \right\rangle$$

$$\begin{bmatrix} [L+\hat{D}+U]_1 & -[\hat{M}]_2 \\ -[T]_2 & [L+\hat{D}+U]_1 \end{bmatrix} \begin{bmatrix} [\phi_1] \\ [\phi_2] \end{bmatrix}^{t_{n+1}} = \begin{bmatrix} [S_1] \\ [S_2] \end{bmatrix}^{t_n}$$

$$[C_i] = [D_i]\left\{[M]_1[\phi_1]^{t_{n+1}} + [M]_2[\phi_1]^{t_{n+1}}\right\} + [Q_i]^{t_n}, \qquad i = 1, \ldots, I$$

Sanity Check: Does Unperturbed Transient Solution Hold Steady-State?





Methods to help debugging transient solutions: test unit consistency and limits of individual important variables:

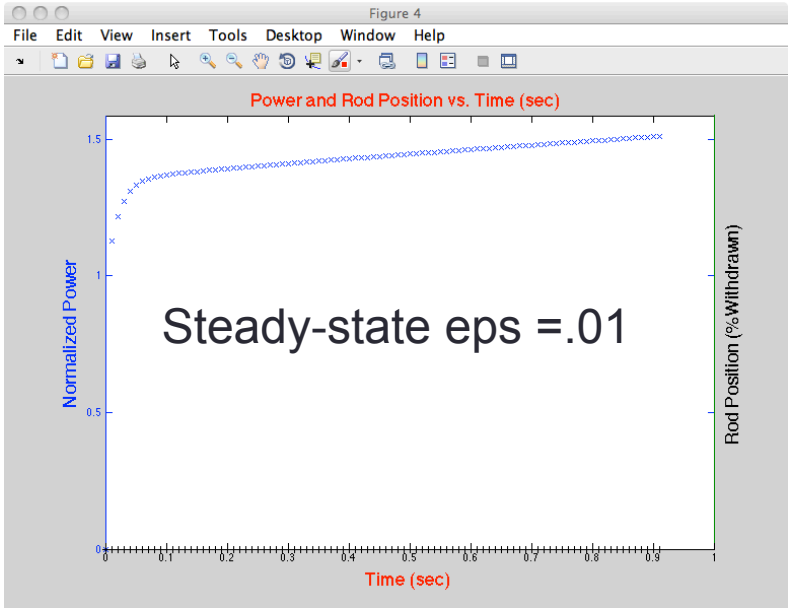
- Spatial mesh = 1.0
- Time step size = 1.0
- Betas → 0.0
- Lambda → infinity
- Velocity*(time step) → infinity

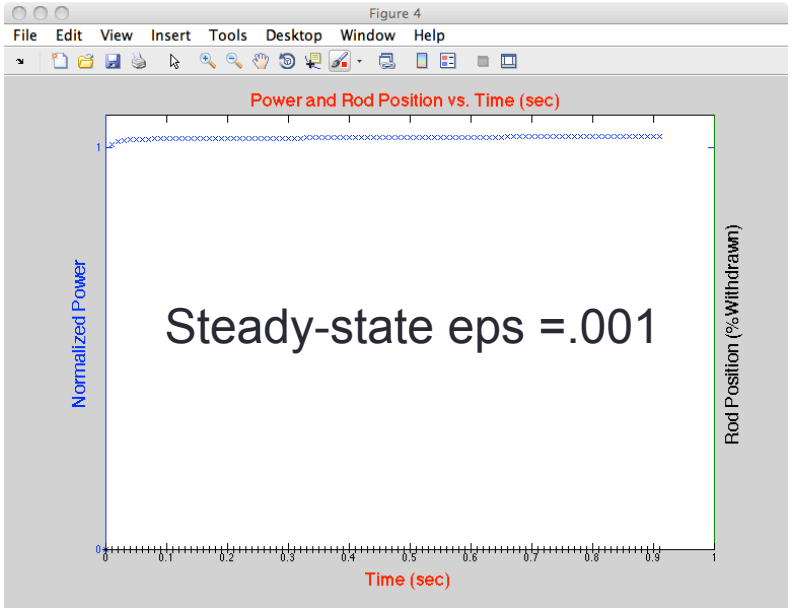
$$\upsilon\hat{\Sigma}_{f,g}(\vec{r}) \equiv \left(\frac{[1-\beta(\vec{r})]}{k_{crit}} + \sum_i^I \frac{\beta_i(\vec{r})\lambda_i\Delta_t}{(1+\lambda_i\Delta_t)k_{crit}} \right) \upsilon\Sigma_{f,g}(\vec{r})$$

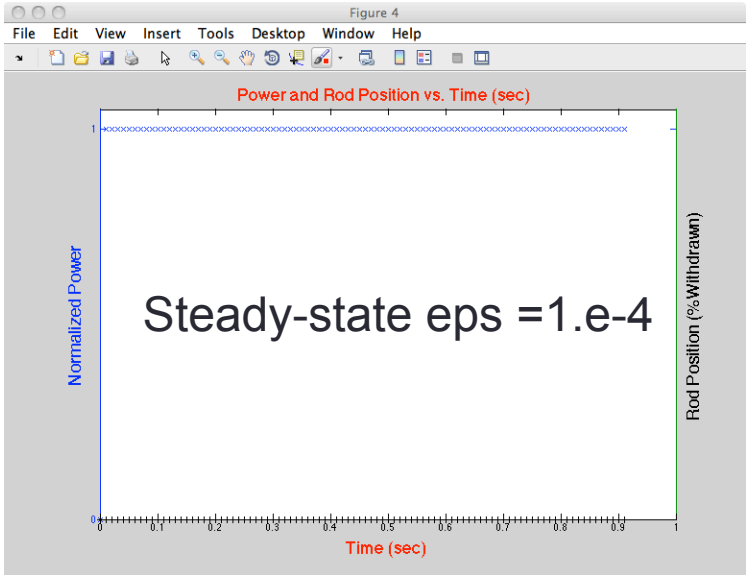
$$\hat{\Sigma}_{r,1}(\vec{r}) \equiv \left\langle \Sigma_{r,1}(\vec{r}) + \frac{1}{v_1(\vec{r})\Delta_t} \right\rangle \qquad \hat{\Sigma}_{r,2}(\vec{r}) \equiv \left\langle \Sigma_{a,2}(\vec{r}) + \frac{1}{v_2(\vec{r})\Delta_t} \right\rangle$$

$$S_1 \equiv \left\langle \frac{\phi_1(\vec{r},t_n)}{v_1(\vec{r})\Delta_t} + \sum_i^I \lambda_i \frac{C_i(\vec{r},t_n)}{(1+\lambda_i\Delta_t)} \right\rangle \qquad S_2 \equiv \left\langle \frac{\phi_2(\vec{r},t_n)}{v_2(\vec{r})\Delta_t} \right\rangle$$

Sanity Check: Is Steady-State Converged Well Enough?



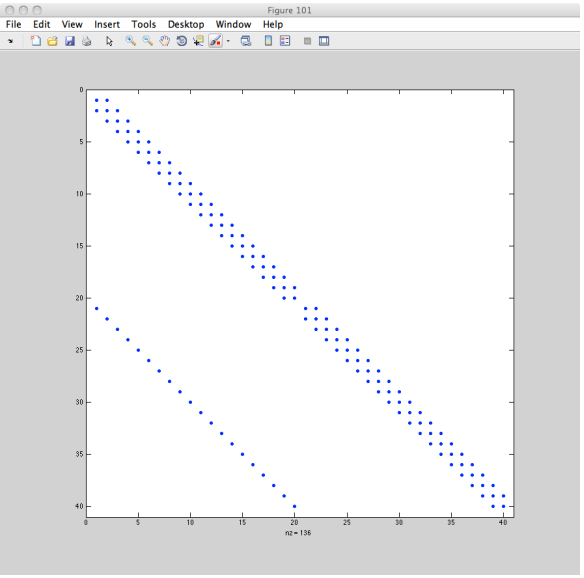




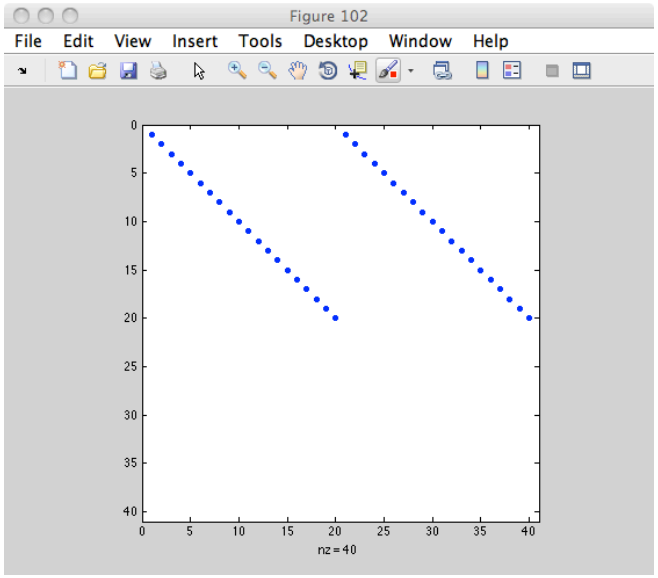
Never move on until steady state is absolutely rock solid!

We never, never, never, drive transient equations to achieve a steady state!!

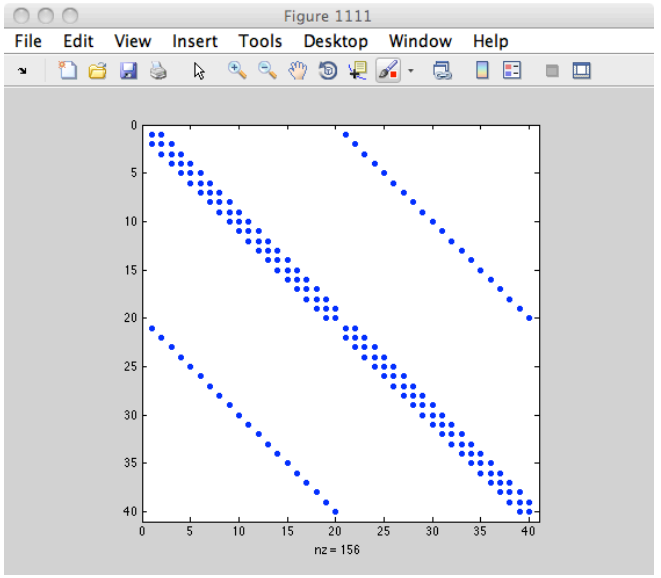
Flux Matrix Inversion for Fission Source and Transient Cases



$[A]$



$[M]$



$[A]^{t_{n+1}}$

$$\begin{bmatrix} [L+D+U]_1 & [0] \\ -[T]_2 & [L+D+U]_2 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} = \begin{bmatrix} [M]_1 & [M]_2 \\ [0] & [0] \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Steady-state Eigenvalue Problem

$$\begin{bmatrix} [L+(\hat{D}-\hat{M})+U]_1 & -[\hat{M}]_2 \\ -[T]_2 & [L+\hat{D}+U]_1 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix}^{t_{n+1}} = \begin{bmatrix} S_1 \\ S_2 \end{bmatrix}^{t_n}$$

One Transient Time Step

Remember Gauss-Seidel Iterative Flux Matrix Inversion?

$$[A][\phi]^{i+1} = \frac{1}{k_{eff}}[M][\phi]^{i} \qquad [A] = [L] + [D] + [U]$$

$$\left\langle [L] + [D] + [U] \right\rangle [\phi]^{i+1} = \frac{1}{k_{eff}}[M][\phi]^{i} \qquad k_{eff} = \frac{[M][\phi]^{i+1}}{[M][\phi]^{i}}$$

$$[\phi]^{k+1} = \left\langle [L] + [D] \right\rangle^{-1} \left\langle \frac{1}{k_{eff}}[M][\phi]^{i} - [U][\phi]^{k} \right\rangle$$

Each step in a transient is similar to flux matrix inversion in fission source iteration:

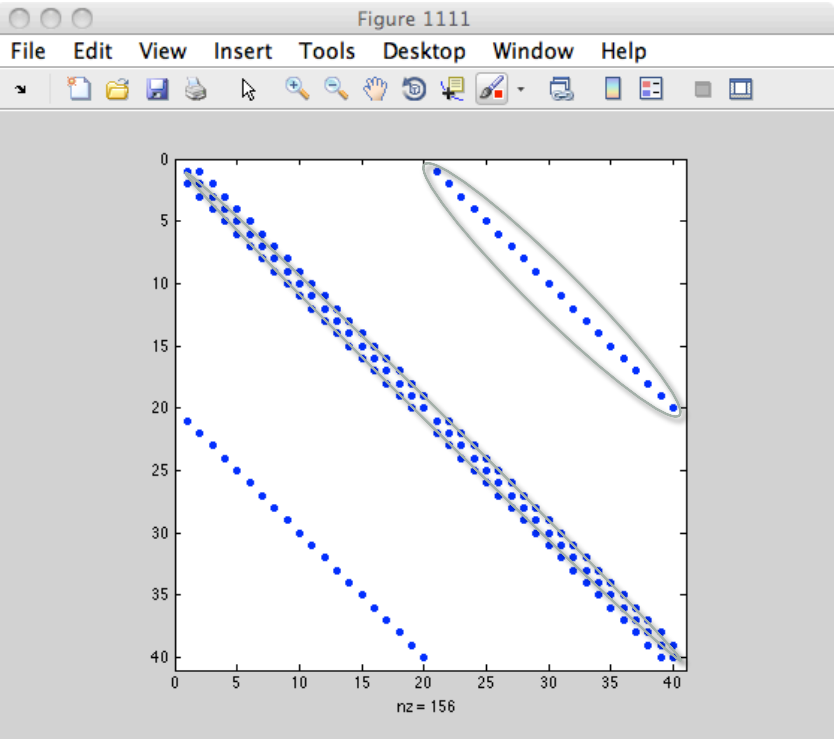
$$-\hat{D}_1^{m-1.m}\phi_1^{m-1}(t_{n+1}) + \left[\left(\hat{\Sigma}_{r1}^{m} - \upsilon\hat{\Sigma}_{f1}^{m}\right)\Delta_x + \hat{D}_1^{m-1.m} + \hat{D}_1^{m,m+1}\right]\phi_1^{m}(t_{n+1}) - \hat{D}_1^{m,m+1}\phi_1^{n+1}(t_{n+1}) - \upsilon\hat{\Sigma}_{f2}^{m}\Delta_x\phi_1^{m}(t_{n+1}) = S_1^{m}(t_n)\Delta_x$$

$$-\hat{D}_2^{m-1.m}\phi_2^{m-1}(t_{n+1}) + \left[\hat{\Sigma}_{r2}^{m}\Delta_x + \hat{D}_2^{m-1.m} + \hat{D}_2^{n,n+1}\right]\phi_2^{m}(t_{n+1}) - \hat{D}_2^{m,m+1}\phi_2^{n+1}(t_{n+1}) - \Sigma_{12}^{m}\Delta_x\phi_1^{m}(t_{n+1}) = S_2^{m}(t_n)\Delta_x$$

Same solver will work to solve transient equations at each time step:

$$\begin{bmatrix} a_{11} & & & & \\ a_{21} & a_{22} & & & \\ a_{31} & a_{32} & a_{33} & & \\ a_{41} & a_{42} & a_{43} & a_{44} & \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{bmatrix} \begin{bmatrix} \phi_1^{k+1} \\ \phi_2^{k+1} \\ \phi_3^{k+1} \\ \phi_4^{k+1} \\ \phi_5^{k+1} \end{bmatrix} = \begin{bmatrix} S_1^{i} \\ S_2^{i} \\ S_3^{i} \\ S_4^{i} \\ S_5^{i} \end{bmatrix} - \begin{bmatrix} a_{12} & a_{13} & a_{14} & a_{15} & \\ & a_{23} & a_{24} & a_{25} & \\ & & a_{34} & a_{35} & \\ & & & a_{45} & \\ & & & & \end{bmatrix} \begin{bmatrix} \phi_1^{k} \\ \phi_2^{k} \\ \phi_3^{k} \\ \phi_4^{k} \\ \phi_5^{k} \end{bmatrix}$$

Gauss-Seidel Iterative Flux Matrix Inversion for Transient Case



$$\begin{bmatrix} \left[L+\left(\hat{D}-\hat{M}\right)+U\right]_1 & -[\hat{M}]_2 \\ -[T]_2 & \left[L+\hat{D}+U\right]_1 \end{bmatrix} \begin{bmatrix} [\phi_1] \\ [\phi_2] \end{bmatrix}^{t_{n+1}} = \begin{bmatrix} [S_1] \\ [S_2] \end{bmatrix}^{t_n}$$

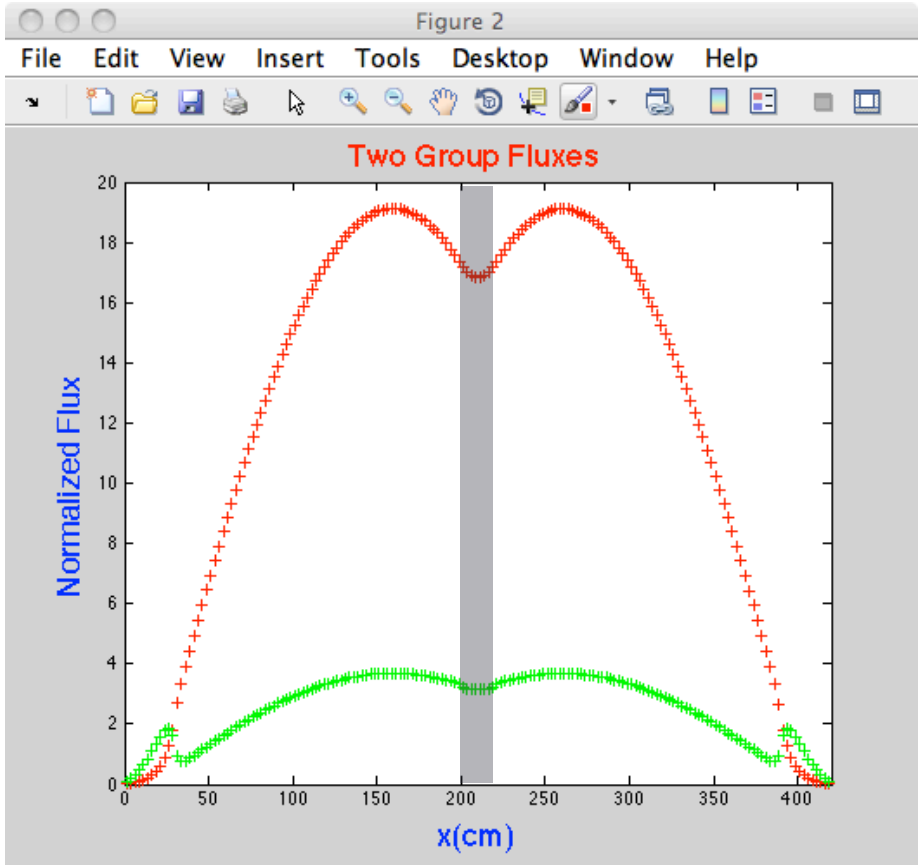
$$[A]=[L]+[D]+[U] \Rightarrow \langle [L]+[D] \rangle [\phi]^{k+1} = [S]-[U][\phi]^k$$

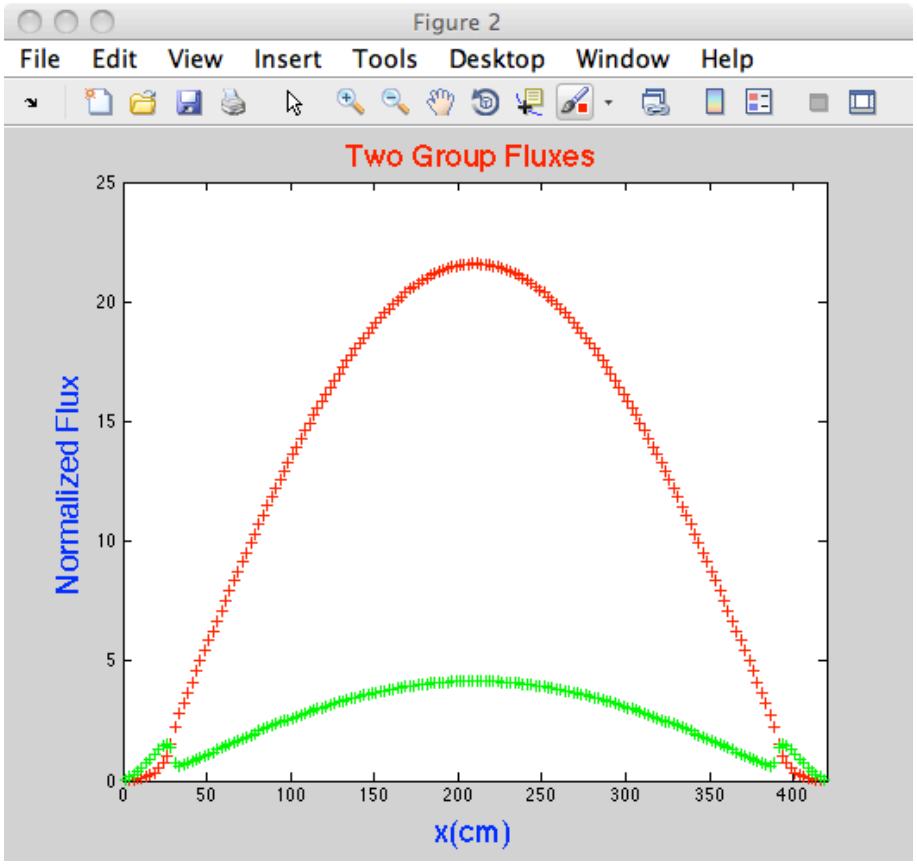
$$[\phi]^{k+1} = \langle [L]+[D] \rangle^{-1} \langle [S]-[U][\phi]^k \rangle$$

"k" is flux iteration index

Solving iteratively for advanced time step fluxes is identical mechanically to performing the flux matrix inversion at each fission source iteration used in solution of the steady-state eigenvalue problem.

Transient Control Rod Withdrawal in1-D Reflected Core

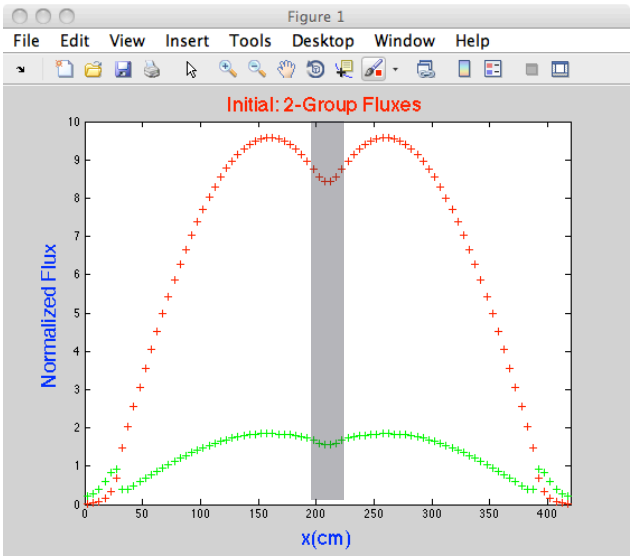


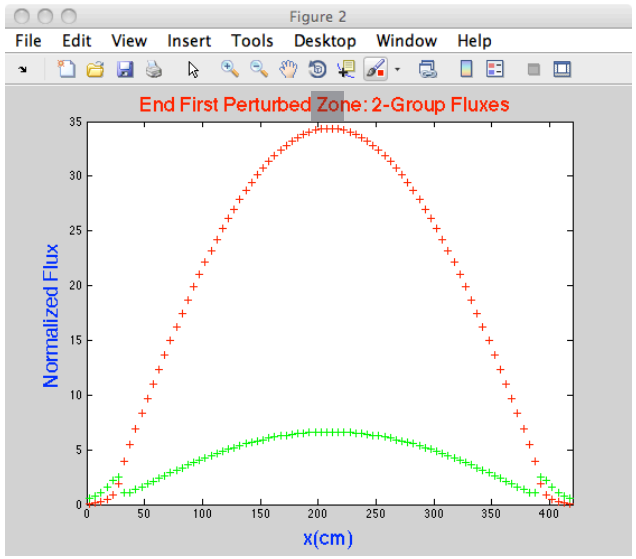


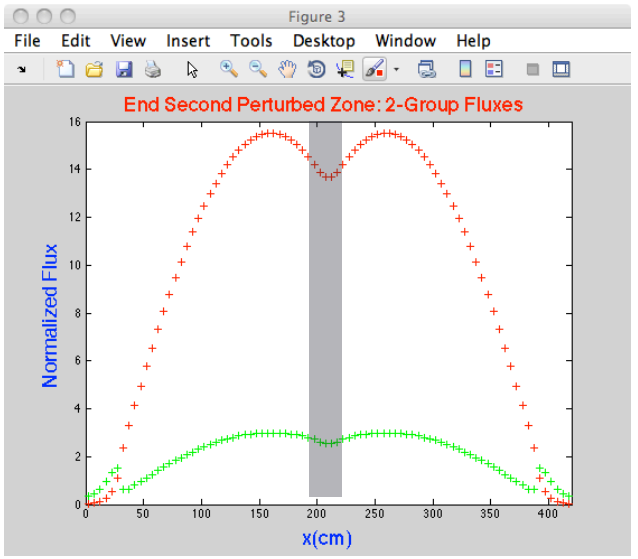
Simulations in 1-D

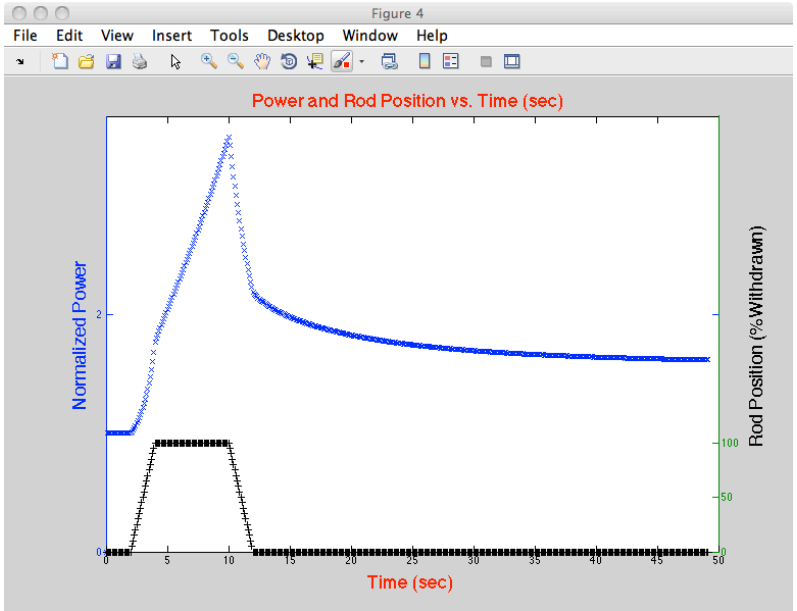
Dynamic Simulation

Transient Control Rod Withdrawal/Insertion in1-D Reflected Core

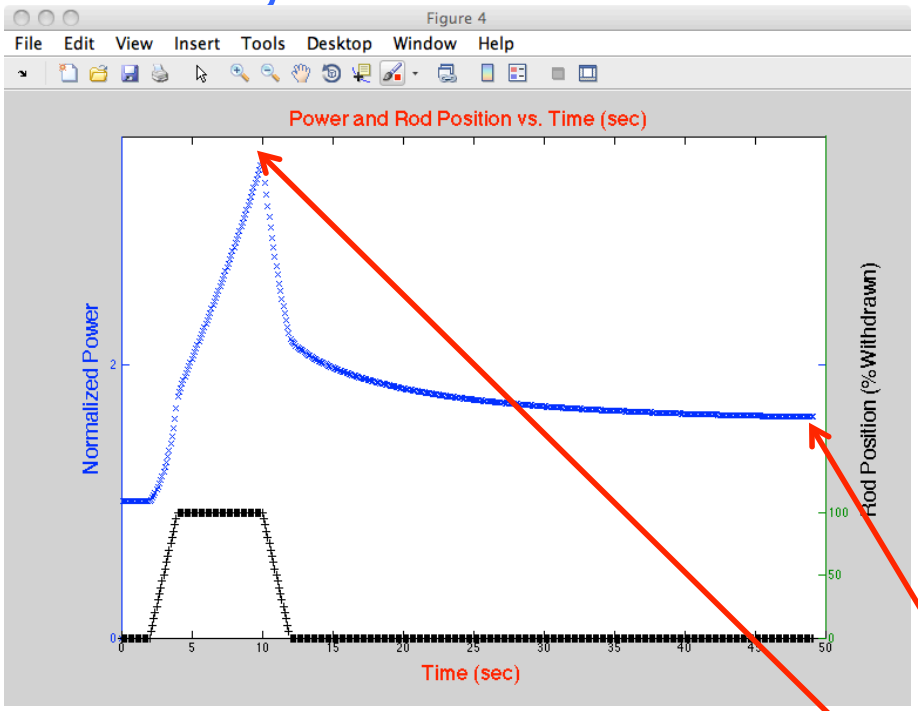




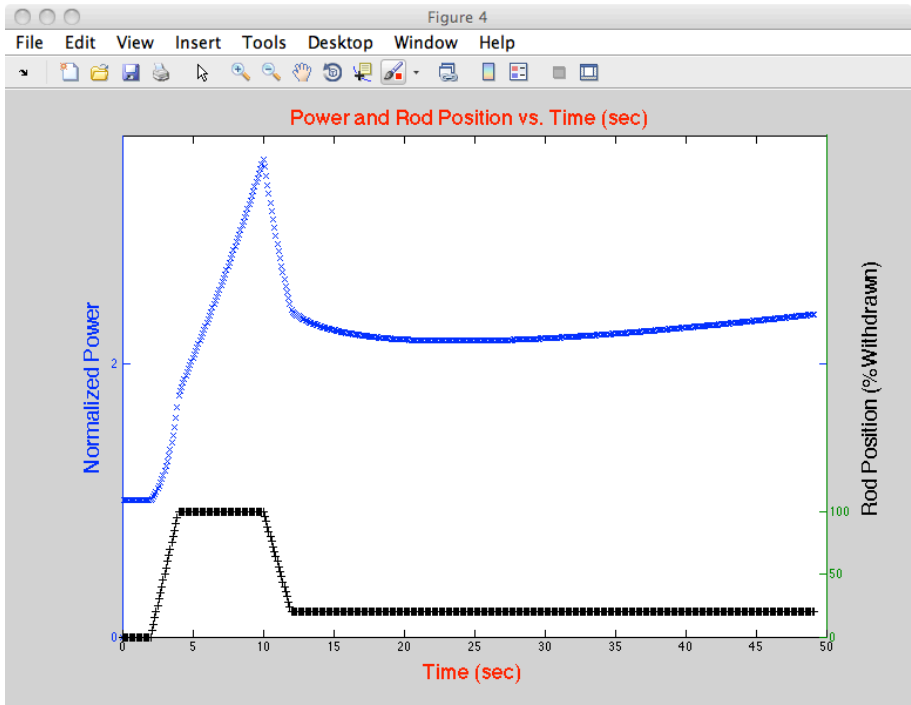




Sanity Checks: Control Rod Withdrawal and Fractional Insertion



100% re-insertion



80% re-insertion

Next week's assignment:

- Spatial and temporal convergence of transient equations

Following weeks' assignments:

- Compare PKE, Quasi-static, etc. to reference solutions
- Better time integrators than fully-implicit
- Automatic time step selection

Next Few Lectures/Assignments

- Compare true time-dependent solution to:
  - PKE solution using unity-weighted initial steady-state fluxes
  - PKE solution using unity-weighted final steady-state fluxes
  - PKE solution using adjoint-weighted initial steady-state fluxes
  - PKE solution using adjoint-weighted final steady-state fluxes
  - 2-group PKE solution using unity-weighted initial fluxes
  - 2-group PKE solution using unity-weighted final fluxes
  - Geometrically-interpolated shape- and weight-function PKEs
  - Quasi-Static solution
  - Improved Quasi-Static solution

Assignment for Next Class

- Solve PSet 2: steady-state 1-D, 2-group finite-difference problems
- Think about how you will compute generalized PKE parameters from your 1-D, 2-group solutions.
- Study the kinetics review paper by Sutton and Aviles (KAPL)
- Start working on time-dependent solutions to 1-D, 2-group finite-difference problems.
- Start thinking about time-dependent solutions to 2-D, 2-group finite-difference problems.