NUCLEAR REACTOR KINETICS

Lecture 3

STEADY STATE DIFFUSION
NUMERICAL SOLUTIONS IN 1-D, Introduction to 2-D

Massachusetts
Institute of
Technology

Errata: slide 35 of Lecture 2: Thanks Lulu and Vlad
Numerical Solution of Inverse Kinetics Equations

integrating both sides with respect to t:

$$[e^{At}][C(t)] = [A]^{-1}[e^{At}][Y(t)] + [c]$$

using the particular solution that $C_o = C(t=0)$

$$[C_0] = [A]^{-1}[Y_0] + [c] \Rightarrow [c] = [C_0] - [A]^{-1}[Y_0]$$

so

$$[e^{At}][C(t)] = [A]^{-1}\{[e^{At}][Y(t)] - [Y_0]\} + [C_0]$$

and finally,

$$[C(t)] = [e^{-At}][C_0] + [e^{-At}][A]^{-1}\{[e^{At}][Y] - [Y_0]\}$$

Not diag [beta]

$$where \quad Y(t) \equiv [\beta_i]\frac{T(t)}{\Lambda}$$

- We can solve for time-varying concentrations of precursors for any desired reactor power shape in time by applying this equation successively for discrete steps.

- From the PKEs, we can solve for reactivity in terms of precursor concentration/reactor power vs. time by making finite-difference approximation for derivative term:

$$\frac{d}{dt}T(t) = \frac{\rho(t)-\beta}{\Lambda}T(t) + \sum_i \lambda_i C_i(t) \qquad \Rightarrow \rho(t) = \frac{\Lambda}{T(t)}\frac{d}{dt}T(t) + \beta - \frac{\Lambda}{T(t)}\sum_i \lambda_i C_i(t)$$

$$\Rightarrow \rho_n = \frac{\Lambda}{T_n}\frac{\left(T_n - T_{n-1}\right)}{\left(t_n - t_{n-1}\right)} + \beta - \frac{\Lambda}{T_n}\sum_i \lambda_i C_{i,n}$$

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 with Direct Solutions	Smith		
6	24-Sep	Higher-order Time Integration and Runga-Kutta	Smith		PSET # 3: 2-D Fully-Implicit Diffusion
7	26-Sep	Time Stepping for Automatic Error Control	Smith		
8	1-Oct	PKE with Feedback: Operator Splitting/Exact Integration	Smith		PSET # 4: PKE from 2-D Diffusion
9	3-Oct	Quasi-Static Time-Integration and Synthesis Methods	Smith		
	8-Oct	Columbus Holiday (8th and 9th)			
10	10-Oct	2D F-I 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)			

Today's Lecture: Goals

- Make sure your understand PSet 1

- Review fine-mesh and nodal approach to reactor modeling
- Refresh memories on transport to diffusion approximations
- Refresh memories on multi-group cross section derivations
- Refresh memories on two-group diffusion equations

- Understand simple finite-difference method to solve diffusion equations
  - Derivation of neutron balance equations
  - Derivation of diffusion theory divergence terms
  - Derivation of boundary conditions

- Matrix representation/numerical solutions of 1-D slab diffusion problems
  - Fixed-source problem solutions
  - Direct MATLAB eigenvalue solutions
  - Fission source iterations
  - Flux iterations

- Matrix representation of 2-D and 3-D diffusion equations

PSET # 1
POINT KINETICS

Due: Sept 17, 2012

Massachusetts
Institute of
Technology

PSet 1

PART A: Write a computational tool for solving PKEs

Following the MATLAB example used in class to solve the PKEs, write your own solver (in any language you choose) and incorporate the following features:

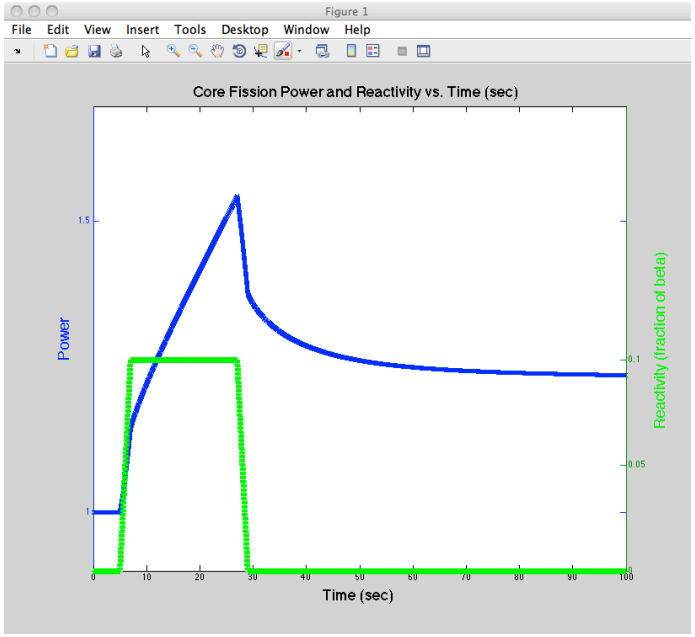
1. 8-group delayed neutron data for U^{235} fission
2. neutron velocity corresponding to 0.1 eV, sigma-fission cross section of 0.05 cm-1, and a nu-bar value of 2.45 for modeling prompt neutron lifetime
3. Initialize for steady-state operation at the start of each transient to a normalized power of 1.0
4. Incorporate ramp inputs of reactivity (as fractions of beta) vs. time so your code simulates reactor response to input reactivity changes
5. Produce plots of reactivity and power versus time

PSet 1

PART B: Testing your PKE Tool

Produce the following results to demonstrate that your tool functions correctly:

1. The value of total delayed neutron fraction (beta)
2. The value of prompt neutron lifetime (pnl) in seconds
3. Produce plot of relative power and reactivity vs. time for 100 seconds following a 2 second ramp insertion of +.1 beta, a hold for 20 seconds, and a 2 second ramp back to zero reactivity. (see Ex 6)
4. Make sure time-step size is small enough that results are converged.

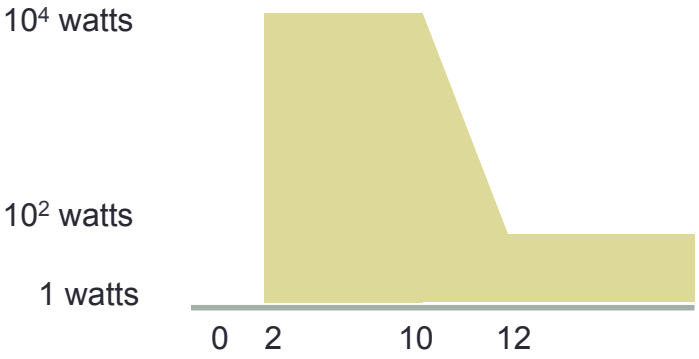


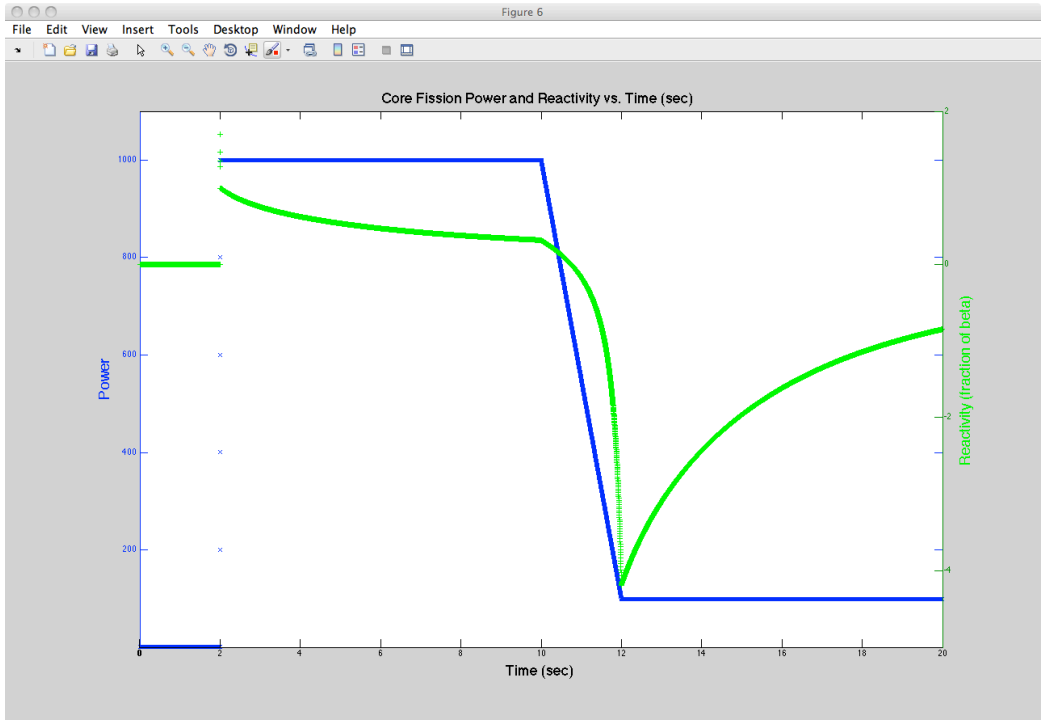
PSet 1

PART C: Write Inverse Kinetics Tool

- Solve IKE for reactivity vs. time that will match desired power shape

- Run PKE with your reactivity shape and prove it reproduces desired power shape





PWR Fine-Mesh Deterministic Transport Analysis Tools

3D Heterogeneous Approach

- Solve unique pins or lattice in great detail in space and energy

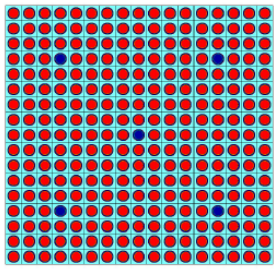
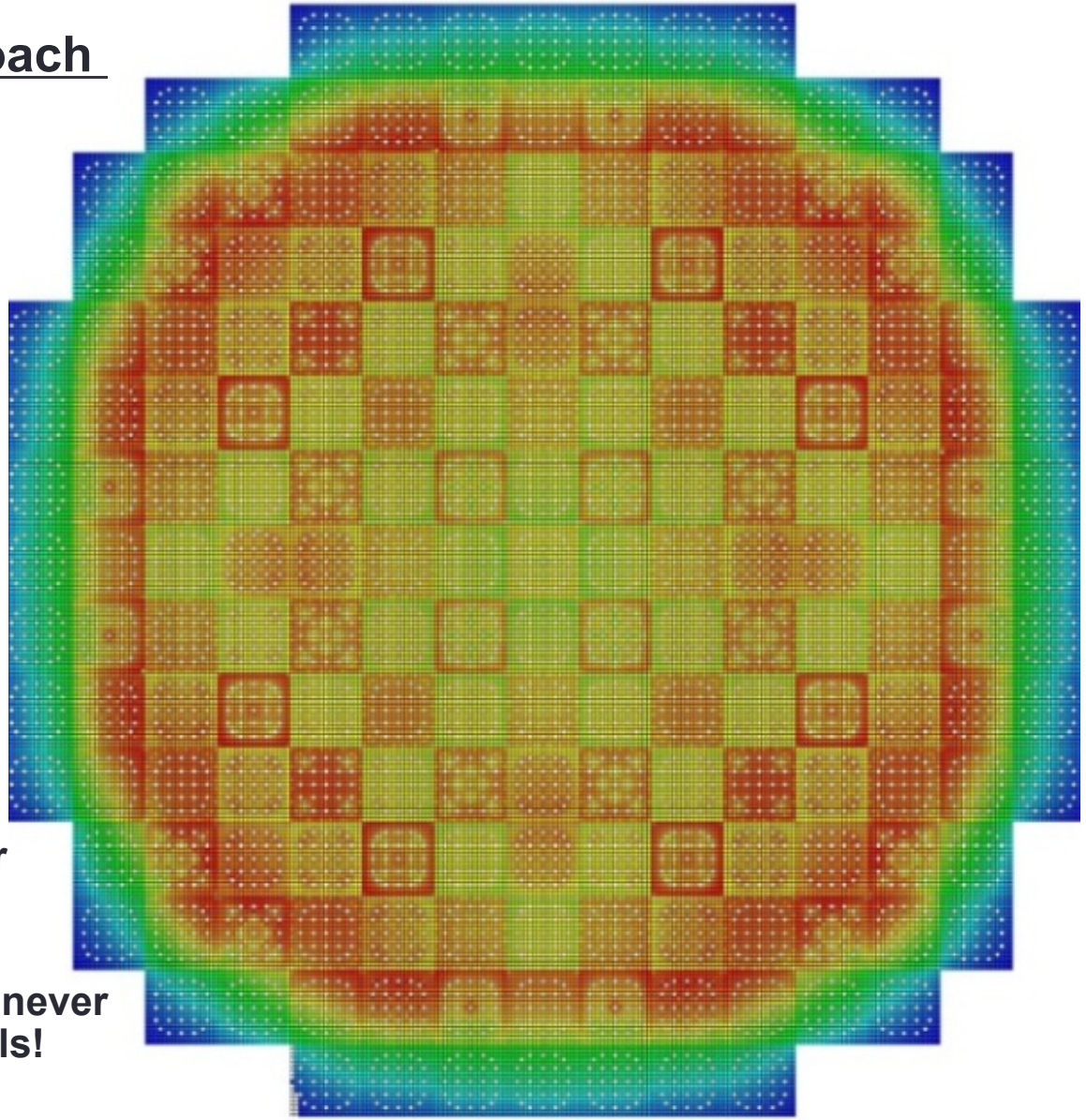


Figure 1. Fuel Assembly containing 5 Gd-UOX pins

- Define fine-energy multi-group cross sections for each material
- Note: cross sections may be different for the same material in different spatial regions (e.g., fuel next to a water rod, fuel at assembly periphery)

- Note: This problem has actually never been solved with realistic models! (it's the analyst's dream)



PWR Fine-Mesh Diffusion Reactor Design Tools

3D Homogenized Pin Approach

- Solve unique pins or lattice in great detail in space and energy

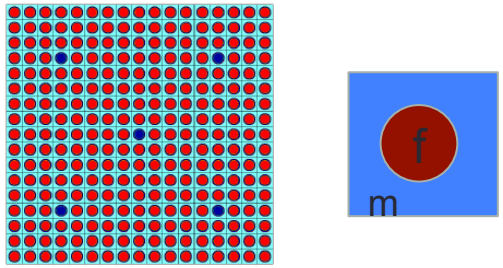
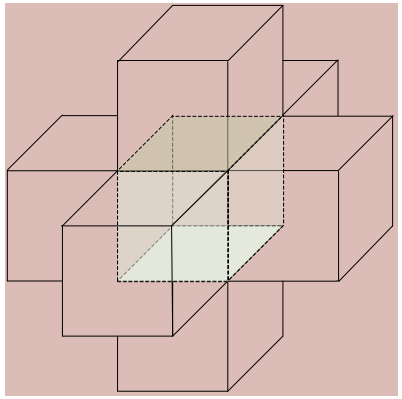
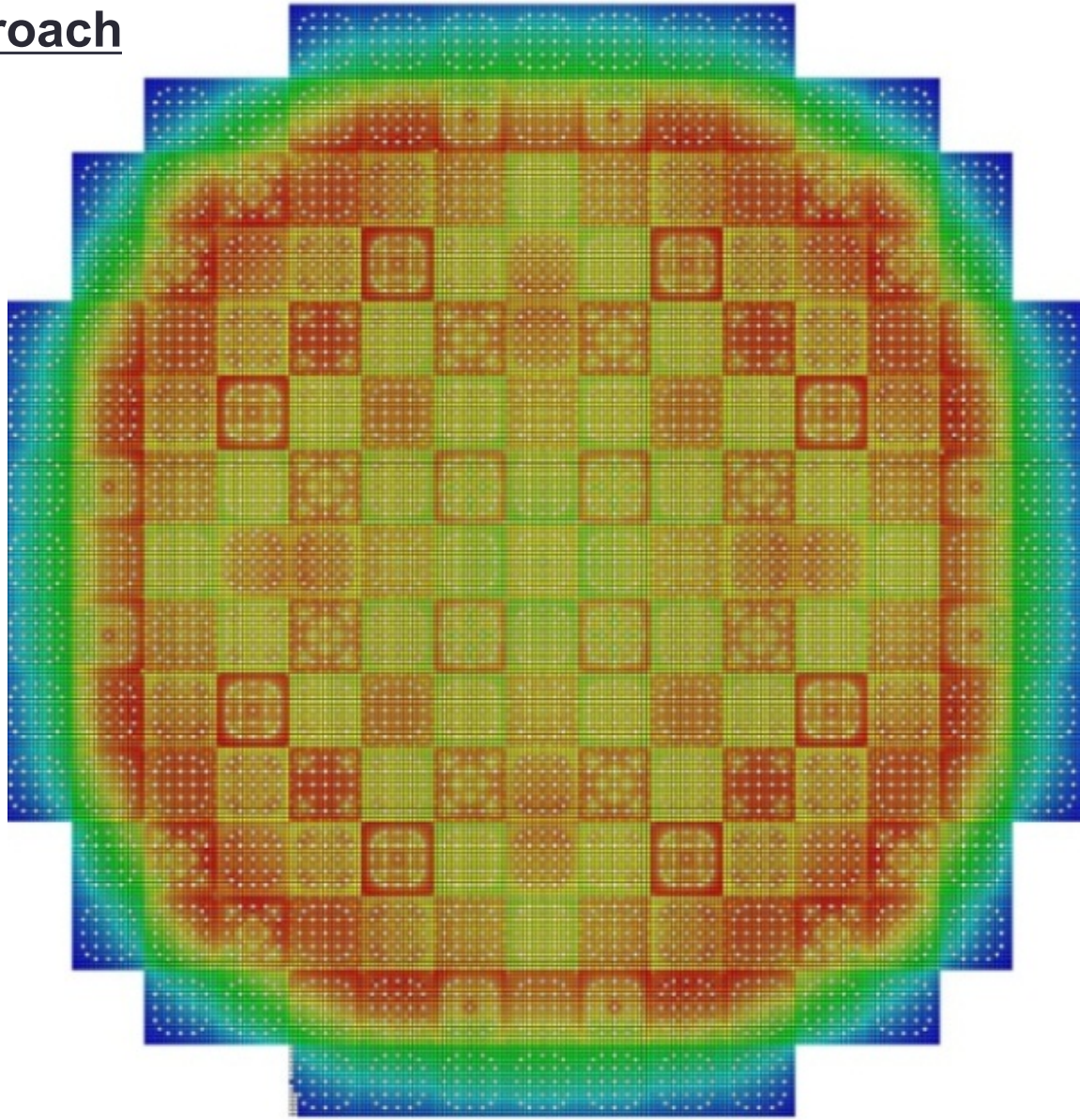


Figure 1. Fuel Assembly containing 5 Gd-UOX pins

- Treat each pin as "homogenized"
- Solve 3D homogenized-pin diffusion problem

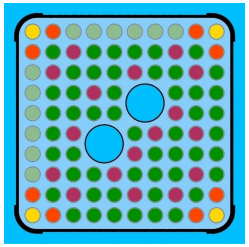




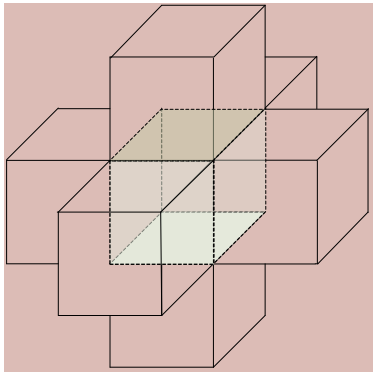
Neutronics Reactor Design Tools: Nodal Diffusion Tools

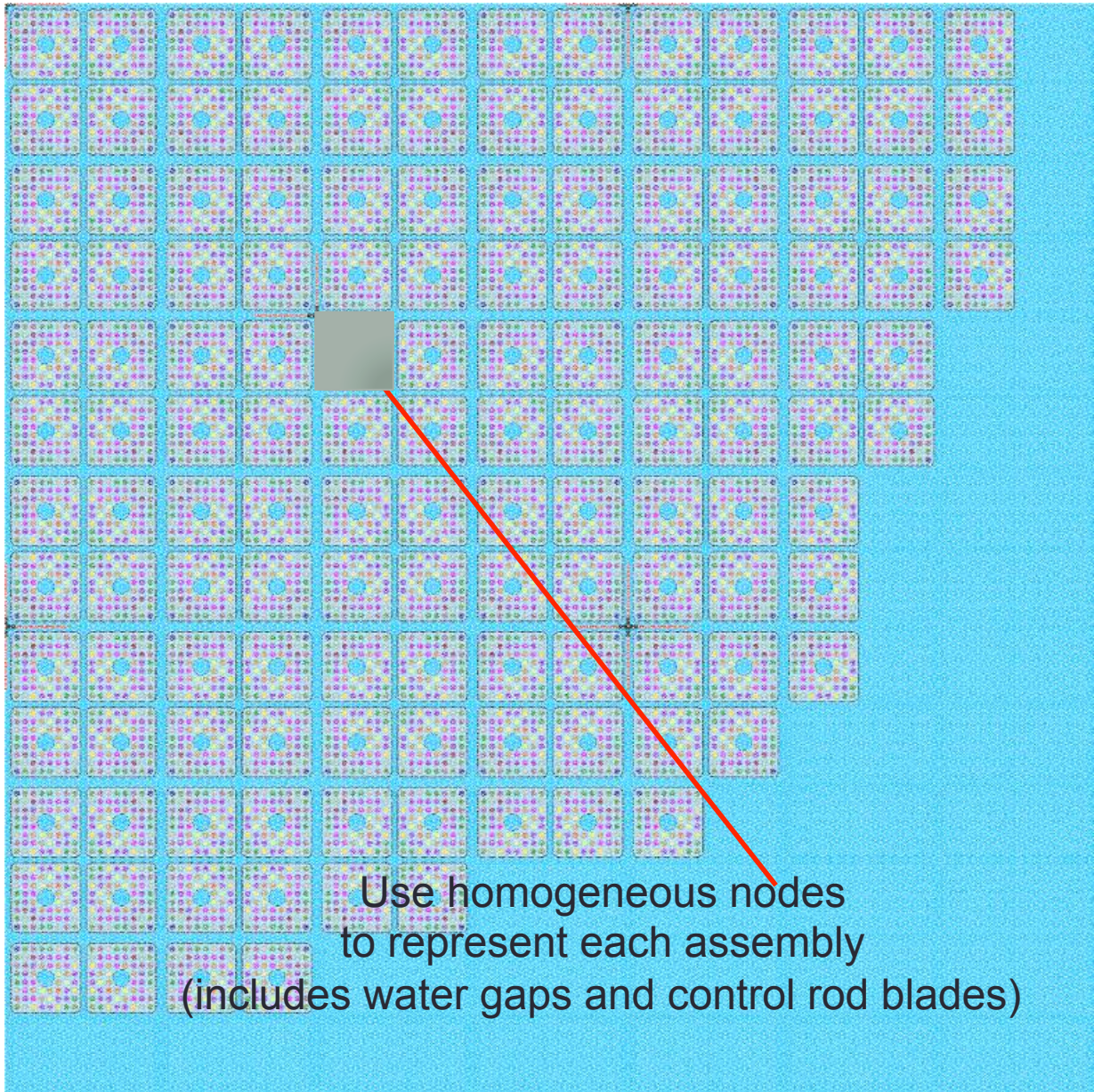
3D Nodal Approach

- Solve unique lattice in detail

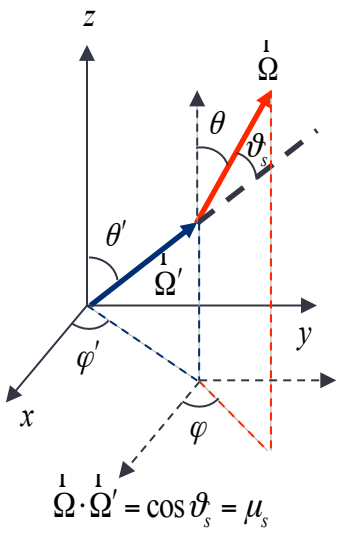


- Treat each assembly as "homogenized"
- Solve 3D homogenized-assembly few-group diffusion problem





Transport Theory vs. Diffusion Theory



- **The Steady-State Continuous Energy Transport Equation:**

$$\vec{\Omega}\cdot\nabla\psi(\vec{r},E,\vec{\Omega})+\Sigma_t(\vec{r},E)\psi(\vec{r},E,\vec{\Omega}) = \int dE'\int d\Omega'\Sigma_s(\vec{r},E'\to E,\vec{\Omega}'\to\vec{\Omega})\psi(\vec{r},E',\vec{\Omega}')+S(\vec{r},E,\vec{\Omega})$$

- **The Steady-State Continuous Diffusion Equation:**

$$-\nabla\cdot D(\vec{r},E)\nabla\phi(\vec{r},E)+\Sigma_t(\vec{r},E)\phi(\vec{r},E)=\int_{E'}dE'\Sigma_s^i(\vec{r},E'\to E)\phi(\vec{r},E')+S_0(\vec{r},E)$$

- **Related quantities: scalar flux and net current**

$$\int_{4\pi}d\Omega\psi(\vec{r},E,\vec{\Omega})=\phi(\vec{r},E) \qquad \left[\int_{4\pi}d\Omega\vec{\Omega}\psi(\vec{r},E,\vec{\Omega})\right]=\vec{J}(\vec{r},E)$$

Diffusion Theory Continuity Conditions

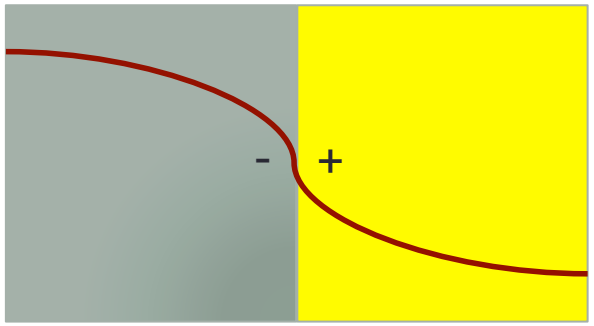
We know from the transport equation that

$$\psi(\vec{r}_i^{\,-},E,\vec{\Omega}) = \psi(\vec{r}_i^{\,+},E,\vec{\Omega}) \quad \text{(continuity of angular flux)}$$

- **Which integrating over angle leads to continuity of scalar flux:**

$$\int_{4\pi} \psi(\vec{r}_i^{\,-},E,\vec{\Omega})d\Omega = \int_{4\pi} \psi(\vec{r}_i^{\,+},E,\vec{\Omega})d\Omega$$

$$\phi(\vec{r}_i^{\,-},E) = \phi(\vec{r}_i^{\,+},E) \quad \text{(continuity of scalar flux)}$$



- **Multiplying by omega and integrating over angle:**

$$\int_{4\pi} \vec{n}\cdot\vec{\Omega}\psi(\vec{r}_i^{\,-},E,\vec{\Omega})d\Omega = \int_{4\pi} \vec{n}\cdot\vec{\Omega}\psi(\vec{r}_i^{\,+},E,\vec{\Omega})d\Omega$$

$$J_n(\vec{r}_i^{\,-},E) = J_n(\vec{r}_i^{\,+},E) \quad \text{(continuity of normal component of current)}$$

$$\vec{n}\cdot D(\vec{r}_i^{\,-},E)\nabla\phi(\vec{r}_i^{\,-},E) = \vec{n}\cdot D(\vec{r}_i^{\,+},E)\nabla\phi(\vec{r}_i^{\,+},E)$$

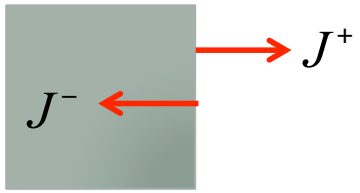
$$\frac{\nabla\phi(\vec{r}_i^{\,-},E)}{\nabla\phi(\vec{r}_i^{\,+},E)} = \frac{D(\vec{r}_i^{\,-},E)}{D(\vec{r}_i^{\,+},E)}$$

Partial Currents: Derived from Transport Theory

- **Related quantities: partial currents in diffusion approximation**

$$J^+(\vec{r},E)=\int_{\vec{n}\cdot\vec{\Omega}>0} d\Omega \vec{n}\cdot\vec{\Omega}\psi(\vec{r},E,\vec{\Omega})=\frac{1}{4\pi}\int_{\vec{n}\cdot\vec{\Omega}>0} d\Omega \vec{n}\cdot\vec{\Omega}[\phi(\vec{r},E)+3\vec{\Omega}\cdot\vec{J}(\vec{r},E)]$$

$$=\frac{1}{4\pi}\int_0^{2\pi} d\varphi\int_0^1 d\mu[\mu\phi(\vec{r},E)+3\mu^2 J(\vec{r},E)]=\frac{1}{4}\phi(\vec{r},E)+\frac{1}{2}J(\vec{r},E)$$



- **Partial Currents in diffusion theory:**

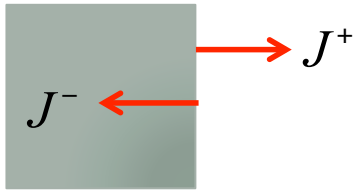
$$J^-(\vec{r},E)=\int_{\vec{n}\cdot\vec{\Omega}<0} d\Omega\,|\vec{n}\cdot\vec{\Omega}|\,\psi(\vec{r},E,\vec{\Omega})=-\frac{1}{4\pi}\int_0^{2\pi} d\varphi\int_{-1}^0 d\mu[\mu\phi(\vec{r},E)+3\mu^2 J(\vec{r},E)]=\frac{1}{4}\phi(\vec{r},E)-\frac{1}{2}J(\vec{r},E)$$

- **Relationship of partial currents to scalar flux and net current:**

$$\phi(\vec{r},E)=2[J^+(\vec{r},E)+J^-(\vec{r},E)] \qquad J_n(\vec{r},E)=J^+(\vec{r},E)-J^-(\vec{r},E)$$

- **Albedo boundary conditions:**

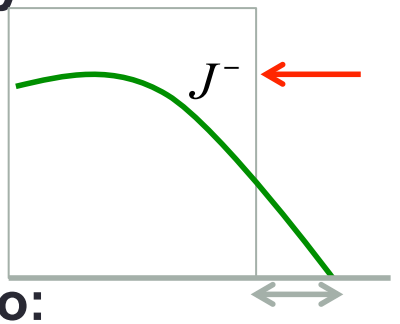
$$\alpha=\frac{J^-(\vec{r}_i,E)}{J^+(\vec{r}_i,E)}$$



Extrapolated Flux Boundary Conditions for Diffusion Theory

- **Outer surface boundary condition in Transport Theory:**

$$\psi(\vec{r},E,\vec{\Omega})\Big|_{\vec{n}\cdot\vec{\Omega}<0} = 0$$



- **Integrating over all angles in half space:**

$$J^-(\vec{r}_i,E) = 0$$

- **In diffusion theory, the incoming partial current is zero:**

$$J^-(\vec{r},E) = \frac{1}{4}\phi(\vec{r},E) - \frac{1}{2}J_n(\vec{r},E)$$

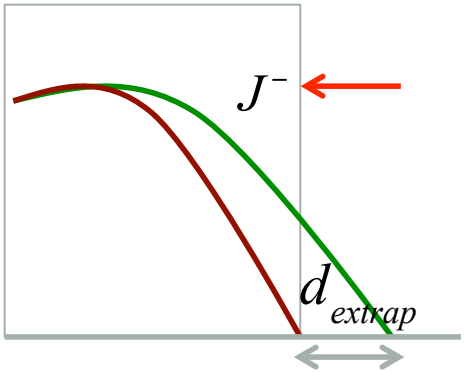
which implies

$\frac{J}{\phi} = \frac{1}{2}$, can be used as a boundary condition, or

$$J^- = \frac{1}{4}\phi + \frac{D}{2}\nabla\phi_n$$

$$\frac{\nabla\phi}{\phi} = -\frac{1}{2D} = -\frac{3\Sigma_{tr}}{2} = -\frac{1}{d_{extrap}}$$

$$d_{extrap} = \frac{2}{3}\lambda_{tr}$$

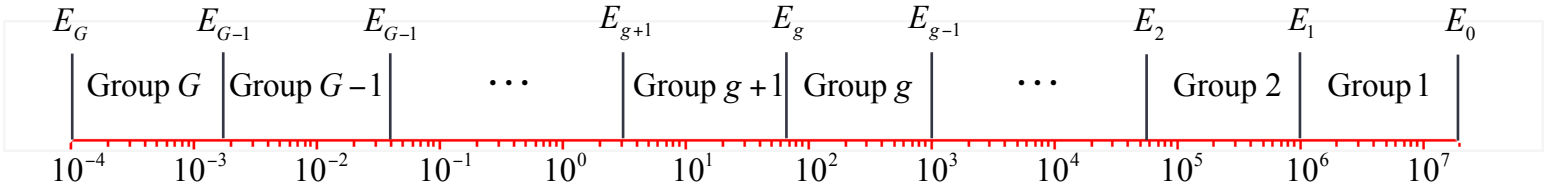


$\phi(\vec{r}_s) = 0$ $\phi(\vec{r}_s + d_{extrap}) = 0$

- **More accurate transport-derived b.c.** $d_{extrap} = 0.7104\lambda_{tr}(\vec{n},E)$

Multi-group Neutron Diffusion Equations

- Start from steady-state continuous energy diffusion equation and divide the energy range into G intervals (groups):



- Define as yet unknown group fluxes by the integrals of the flux over individual groups

$$\phi_g(\vec{r}) = \int_{E_g}^{E_{g-1}} \phi(\vec{r},E)\,dE$$

- Average cross sections (or group constants) in such way that the reaction rates in each interval are preserved

$$\sigma^i_{xg}(\vec{r}) = \frac{1}{\phi_g(\vec{r})}\int_{E_g}^{E_{g-1}} \sigma^i_x(\vec{r},E)\phi(\vec{r},E)\,dE = \frac{\int_{E_g}^{E_{g-1}} \sigma^i_x(\vec{r},E)\phi(\vec{r},E)\,dE}{\int_{E_g}^{E_{g-1}} \phi(\vec{r},E)\,dE}$$

- Group constants are typically determined using neutron spectra obtained by solving local problems with approximate boundary conditions (e.g. lattice physics code)

Multi-group Neutron Diffusion Equation (2)

- Integrate over each of the g groups:

$$-\int_{E_g}^{E_{g-1}} dE \nabla \cdot D(\vec{r},E)\nabla\phi(\vec{r},E) + \int_{E_g}^{E_{g-1}} dE \Sigma_t(\vec{r},E)\phi(\vec{r},E) = \int_{E_g}^{E_{g-1}} dE S(\vec{r},E)$$

$$+\int_{E_g}^{E_{g-1}} dE \chi(E)\int_{E'} dE' \nu\Sigma_f(\vec{r},E')\phi(\vec{r},E') + \int_{E_g}^{E_{g-1}} dE \int_{E'} dE' \Sigma_s(\vec{r},E' \to E)\phi(\vec{r},E')$$

- For total interaction term:

$$\int_{E_g}^{E_{g-1}} dE \Sigma_t(\vec{r},E)\phi(\vec{r},E) = \Sigma_{tg}(\vec{r})\int_{E_g}^{E_{g-1}} dE\phi(\vec{r},E) = \Sigma_{tg}(\vec{r})\phi_g(\vec{r})$$

- For the source term:

$$\int_{E_g}^{E_{g-1}} dE S(\vec{r},E) = S_g(\vec{r})$$

- Fission source term:

$$\int_{E_g}^{E_{g-1}} dE \chi(E)\int_{E'} dE' \nu\Sigma_f(\vec{r},E')\phi(\vec{r},E') = \chi_g \sum_{g'=1}^{G} \nu\Sigma_{fg}(\vec{r})\phi_g(\vec{r}) \qquad \chi_g = \int_{E_g}^{E_{g-1}} dE \chi(E)$$

- For scattering source term:

$$\int_{E_g}^{E_{g-1}} dE \int_{E'} dE' \Sigma_s(\vec{r},E' \to E)\phi(\vec{r},E')\,dE = \int_{E_g}^{E_{g-1}} dE \sum_{g'=1}^{G} \int_{E_{g'}}^{E_{g'-1}} dE' \Sigma_s(\vec{r},E' \to E)\phi(\vec{r},E')$$

$$= \sum_{g'=1}^{G} \int_{E_g}^{E_{g-1}} dE \int_{E_{g'}}^{E_{g'-1}} dE' \Sigma_s(\vec{r},E' \to E)\phi(\vec{r},E') = \sum_{g'=1}^{G} \Sigma_{sg'g}(\vec{r})\phi_{g'}(\vec{r})$$

Multi-group Neutron Diffusion Equation (3)

- Leakage (diffusion) term:

$$\int_{E_g}^{E_{g-1}} dE \nabla \cdot D(\vec{r},E)\nabla\phi(\vec{r},E) = \nabla \cdot D_g(\vec{r}) \int_{E_g}^{E_{g-1}} dE \nabla\phi(\vec{r},E) = \nabla \cdot D_g(\vec{r})\nabla\phi_g(\vec{r})$$

$$\mathbf{D}_g(\vec{r})\nabla\phi_g(\vec{r}) = \int_{E_g}^{E_{g-1}} dE\ D(\vec{r},E)\nabla\phi(\vec{r},E)$$

- The group diffusion coefficients $\mathbf{D}_g$ should be a tensor (3×3 symmetric matrix, called the Eddington tensor)

$$\mathbf{D}_g(\vec{r}) = \begin{bmatrix} D_g^{xx} & D_g^{xy} & D_g^{xz} \\ D_g^{xy} & D_g^{yy} & D_g^{yz} \\ D_g^{xz} & D_g^{xy} & D_g^{zz} \end{bmatrix}$$

- Usually the tensor is approximated by a diagonal matrix:

$$D_g^u(\vec{r}) = \frac{\int_{E_g}^{E_{g-1}} dE\ D(\vec{r},E)\frac{\partial}{\partial u}\phi(\vec{r},E)}{\frac{\partial}{\partial u}\phi_g}, \quad u = x,y,z$$

Multi-group Neutron Diffusion Equation (4)

- Multi-group diffusion equation

$$-\nabla \cdot D_g(\vec{r})\nabla\phi_g(\vec{r}) + \Sigma_{tg}(\vec{r})\phi_g(\vec{r}) = \chi_g \sum_{g'=1}^{G} \nu\Sigma_{fg'}(\vec{r})\phi_{g'}(\vec{r}) + \sum_{g'=1}^{G} \Sigma_{sg'g}(\vec{r})\phi_{g'}(\vec{r}) + S_g(\vec{r}),$$

$$g = 1,2,\cdots,G$$

- Canceling the within group scattering cross section from both sides, and defining the group-wise net removal cross section

$$\Sigma_{tg} = \Sigma_{ag} + \Sigma_{sg} = \Sigma_{ag} + \sum_{g'=1}^{G} \Sigma_{sgg'}$$

$$\Sigma_{rg} = \Sigma_{tg} - \Sigma_{sgg} = \Sigma_{a,g} + \sum_{\substack{g'=1 \\ g' \neq g}}^{G} \Sigma_{sgg'}$$

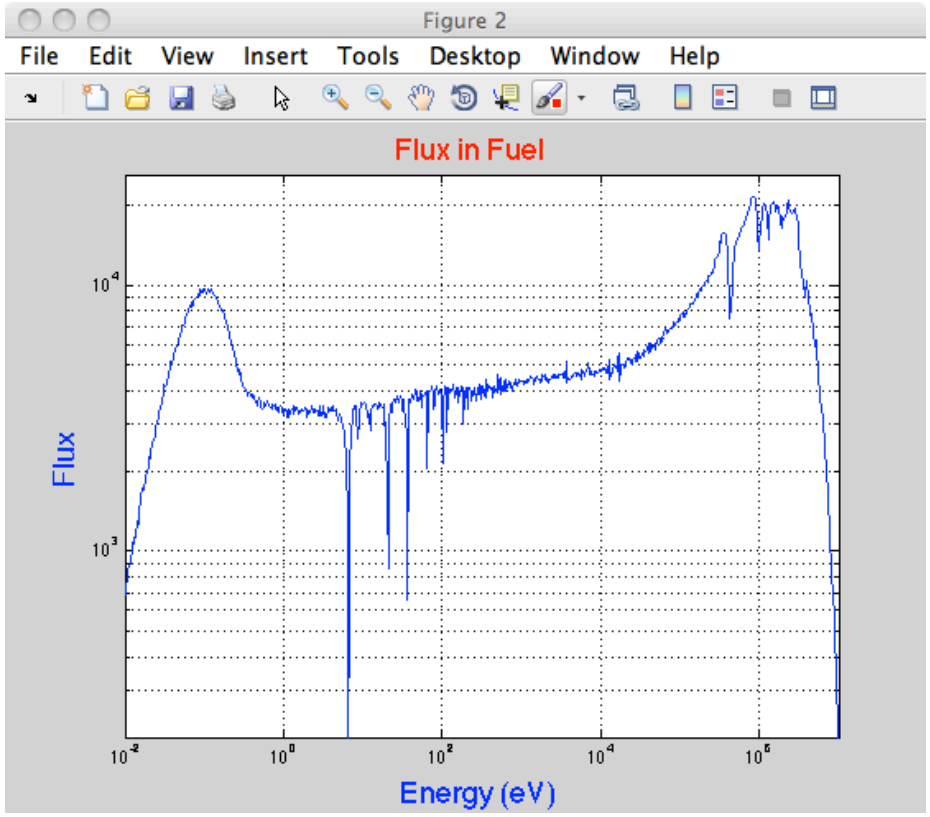
- We obtain the final multi-group form of the neutron diffusion equation

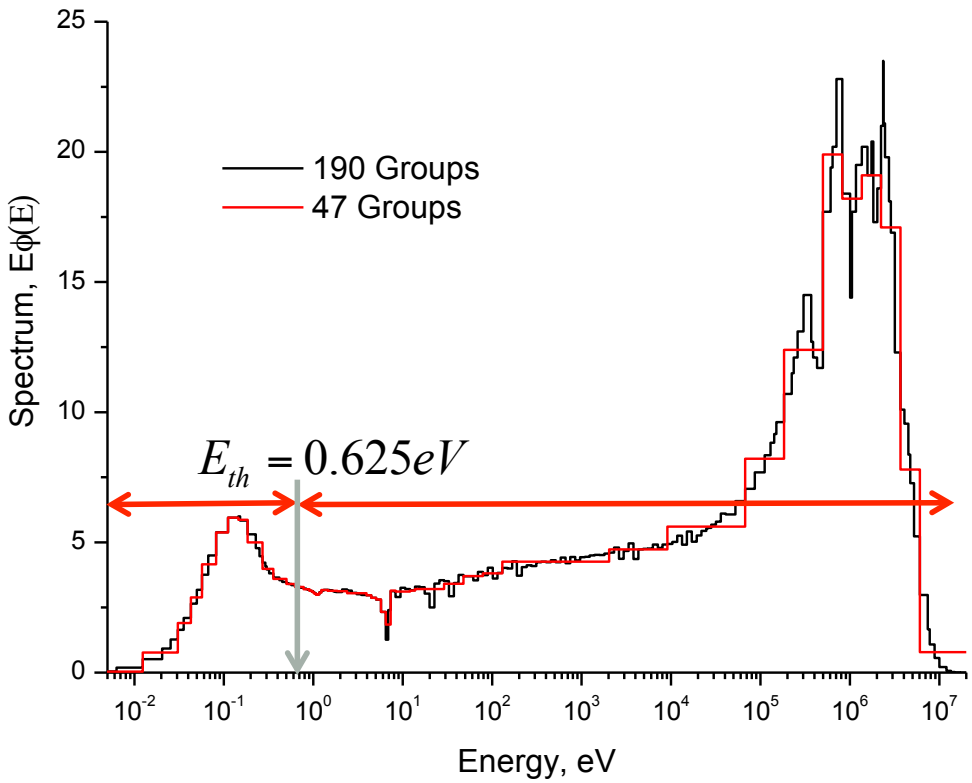
$$-\nabla \cdot D_g(\vec{r})\nabla\phi_g(\vec{r}) + \Sigma_{rg}(\vec{r})\phi_g(\vec{r}) = \chi_g \sum_{g'=1}^{G} \nu\Sigma_{fg'}(\vec{r})\phi_{g'}(\vec{r}) + \sum_{\substack{g'=1 \\ g' \neq g}}^{G} \Sigma_{sg'g}(\vec{r})\phi_{g'}(\vec{r}) + S_g(\vec{r}),$$

$$g = 1,2,\cdots,G$$

Computing Neutron Spectra for Cross Section Collapsing

- Remember that spectral calculations (lattice calculations) require detailed resonance models, unless you use >10,000 energy groups
- Few group data should depend on local spectra and assuming invariance of cross sections may not be a good approximation





2-Group LWR Neutron Diffusion Equation

Group structure for LWR analysis:

- Group 1: E ≥ 0.625 eV → fast group
- Group 2: E < 0.625 eV → thermal group

Total Fission source:

$$S_f(\vec{r}) = \nu\Sigma_{f1}(\vec{r})\phi_1(\vec{r}) + \nu\Sigma_{f2}(\vec{r})\phi_2(\vec{r})$$

- $\chi_1 = 1.0$ and $\chi_2 = 0.0$ which implies that fission source in thermal group = 0

Scattering source:

- Define effective down-scatter, so up-scattering is zero → $\Sigma_{21}(\vec{r}) = 0$

Final 2-group diffusion equations:

$$-\nabla \cdot D_1(\vec{r})\nabla\phi_1(\vec{r}) + [\Sigma_{a1}(\vec{r}) + \Sigma_{s12}(\vec{r})]\phi_1(\vec{r}) = \nu\Sigma_{f1}(\vec{r})\phi_1(\vec{r}) + \nu\Sigma_{f2}(\vec{r})\phi_2(\vec{r}) + s_1(\vec{r})$$

$$-\nabla \cdot D_2(\vec{r})\nabla\phi_2(\vec{r}) + \Sigma_{a2}(\vec{r})\phi_2(\vec{r}) = \Sigma_{s12}(\vec{r})\phi_1(\vec{r}) + s_2(\vec{r})$$

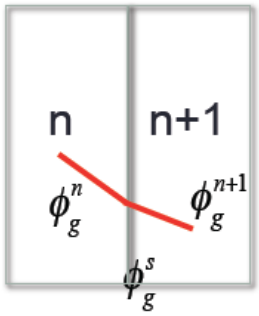
Be able To Write This In Your Sleep!!!

2-Group Finite-Difference Diffusion Equations

Consider two neighboring cells having the same width, and make a linear flux approximation between the cell centers and cell edges:

$$J_n^+ = -D_g^n \frac{d}{dx}\phi_g^n(x)\bigg|_{n+} = -D_g^n \frac{\phi_g^s - \phi_g^n}{\Delta/2}$$

$$J_n^- = -D_g^{n+1} \frac{d}{dx}\phi_g^{n+1}(x)\bigg|_{n+} = -D_g^{n+1} \frac{\phi_g^{n+1} - \phi_g^s}{\Delta/2}$$



and by continuity of net current one solves for interface flux

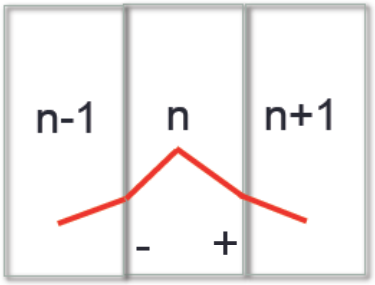
$$-D_g^n \frac{\phi_g^s - \phi_g^n}{\Delta/2} = -D_g^{n+1} \frac{\phi_g^{n+1} - \phi_g^s}{\Delta/2} \Rightarrow \phi_g^s = \frac{D_g^n \phi_g^n + D_g^{n+1}\phi_g^{n+1}}{(D_g^n + D_g^{n+1})}$$

one gets an approximation for the net current in terms of mesh-averaged fluxes:

$$J_n^+ = -\frac{2D_g^n}{\Delta}\left[\frac{D_g^n \phi_g^n + D_g^{n+1}\phi_g^{n+1}}{(D_g^n + D_g^{n+1})} - \phi_g^n\right] = \frac{2D_g^n D_g^{n+1}}{\Delta(D_g^n + D_g^{n+1})}\phi_g^{n+1} - \frac{2D_g^n D_g^{n+1}}{\Delta(D_g^n + D_g^{n+1})}\phi_g^n$$

2-Group Finite-Difference Diffusion Equations

Consider two neighboring cells having the same width, and make a linear flux approximation between the cell centers and cell edges:



$$J_n^+ = -\hat{D}_g^{n,n+1}\left[\phi_g^{n+1} - \phi_g^n\right] \qquad J_n^- = -\hat{D}_g^{n-1,n}\left[\phi_g^n - \phi_g^{n-1}\right] \quad where \quad \hat{D}_g^{n,n+1} \equiv \frac{2D_g^n D_g^{n+1}}{\Delta(D_g^n + D_g^{n+1})}$$

and one obtains two group balance equations:

$$\hat{D}_1^{n-1.n}\left[\phi_1^n - \phi_1^{n-1}\right] - \hat{D}_1^{n,n+1}\left[\phi_1^{n+1} - \phi_1^n\right] + \Sigma_{r1}^n \Delta\phi_1^n = \Sigma_{f1}^n \Delta\phi_1^n + \nu\Sigma_{f2}^n \Delta\phi_2^n + s_1^n \Delta$$

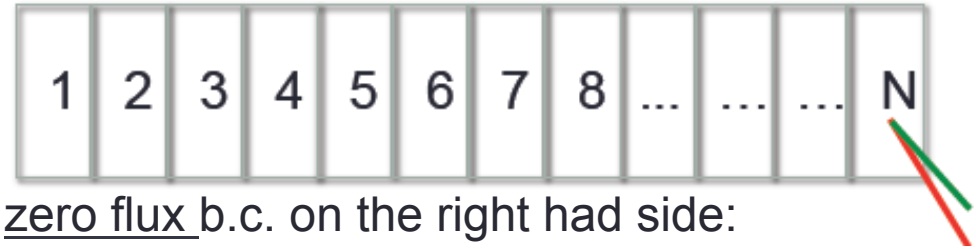
$$\hat{D}_2^{n-1.n}\left[\phi_2^n - \phi_2^{n-1}\right] - \hat{D}_2^{n,n+1}\left[\phi_2^{n+1} - \phi_2^n\right] + \Sigma_{a2}^n \Delta\phi_1^n = \Sigma_{s12}^n \Delta\phi_1^n + s_2^n \Delta$$

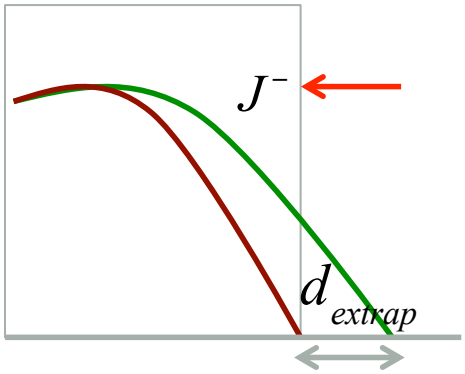
and simplifying one obtains the linear finite-difference diffusion equations:

$$-\hat{D}_1^{n-1.n}\phi_1^{n-1} + \left[\Sigma_{r1}^n \Delta + \hat{D}_1^{n-1.n} + \hat{D}_1^{n,n+1}\right]\phi_1^n - \hat{D}_1^{n,n+1}\phi_1^{n+1} = \Sigma_{f1}^n \Delta\phi_1^n + \nu\Sigma_{f2}^n \Delta\phi_2^n + s_1^n \Delta$$

$$-\hat{D}_2^{n-1.n}\phi_2^{n-1} + \left[\Sigma_{a2}^n \Delta + \hat{D}_2^{n-1.n} + \hat{D}_2^{n,n+1}\right]\phi_2^n - \hat{D}_2^{n,n+1}\phi_2^{n+1} = \Sigma_{s12}^n \Delta\phi_1^n + s_2^n \Delta$$

Boundary Conditions for Finite-Difference Equations





$\phi(\vec{r}_s) = 0$ $\phi(\vec{r}_s + d_{extrap}) = 0$

For zero flux b.c. on the right had side:

$$J_g^N = -D_g^N \frac{\left[-\phi_g^{N+1}\right] - \phi_g^N}{\Delta} \Rightarrow J_g^N = \left[\frac{2D_g^N}{\Delta}\right]\phi_g^n$$

note: net current is positive on right surface

For zero incoming flux b.c.

$$J^-(\vec{r},E) = \frac{1}{4}\phi(\vec{r},E) - \frac{1}{2}J_n(\vec{r},E)$$

which implies

$$J_g^N = -D_g^N \frac{\phi_g^s - \phi_g^N}{\Delta/2} = \frac{\phi_g^s}{2} \Rightarrow \phi_g^s\left(\frac{1}{2} + \frac{2D_g^N}{\Delta}\right) = \frac{2D_g^N}{\Delta}\phi_g^N \Rightarrow\Rightarrow J_g^N = -\frac{2D_g^N}{\Delta}\left[\frac{\frac{2D_g^N}{\Delta}}{\left(\frac{1}{2} + \frac{2D_g^N}{\Delta}\right)} - 1\right]\phi_g^N$$

$$J_g^N = \left[\frac{1}{1 + \frac{4D_g^N}{\Delta}}\right]\left[\frac{2D_g^N}{\Delta}\right]\phi_g^N$$

2-Group 1-D Finite-Difference Matrix Equations

If we number our cells consecutively,

We can express the finite difference equations in matrix form as:

$$\begin{aligned}\left[L+D+U\right]_1\left[\phi_1\right] &= \left[M\right]_1\left[\phi_1\right]+\left[M\right]_2\left[\phi_2\right]+\left[S_1\right] \\ \left[L+D+U\right]_2\left[\phi_2\right] &= \left[T\right]_2\left[\phi_1\right] \qquad\qquad +\left[S_2\right]\end{aligned}$$

Defining a vector of group fluxes of group one followed by group two fluxes:

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

Or in compressed matrix notation:

$$\left[A\right]\left[\phi\right]=\left[M\right]\left[\phi\right]+\left[S\right]$$

Note: vector length is 2N

Five Ways to Solve the 2-Group 1-D Diffusion Problems in MATLAB

```
function oneD_diffusion
%
% define two group cross sections for all potential materials
%
  nmat=4;
  xs(1,1)=1.3; xs(2,1)=0.5; 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)=1.3; xs(2,2)=0.5; 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)=1.5; xs(2,3)=0.5; xs(3,3)=0.0002; xs(4,3)=0.014; xs(5,3)=0.032; xs(6,3)=0.000; xs(7,3)=0.0000;  % water
  xs(1,4)=1.3; xs(2,4)=0.5; 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
  for n=1:nmat
      kinf=(xs(6,n)+xs(7,n)*xs(5,n)/xs(4,n))/(xs(3,n)+xs(5,n))
  end
%
% define problem geometry and assign materials:
%
% #zones-NZONE w(nzone),n(nzone), mat(nzone)
%
%  bc | slab 1| slab 2| slab 3| ..... slab(NZONE) | bc
%
  ido=1;
  if ido == 1
    NZONE=1;
    w(1)=45;                          % width    per zone
    m(1)=1;                           % material per zone
    n(1)=1;                           % mesh     per zone
    n=n*60;                           % mesh refinement factor
  else
    NZONE=3;
    w(1)= 40; w(2)=280; w(3)=40;    % width    per zone
    m(1)= 2;  m(2)=1;   m(3)=2;     % material per zone
    n(1)= 20; n(2)=140; n(3)=20;    % mesh     per zone
    n=n*2;                            % mesh refinement factor
  end

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

  [A,M,x,NP,L] = Diffusion_setup (xs,w,n,m,NZONE,bc);

  OneD_source    (A,M,x,NP,L)                                % solve as a series of fixed source problems

  OneD_eigen     (A,M,x,NP,L)                                % solve as with MATLAB eigensolver package

     eps=1.e-6; maxouter=1000;                                % set convergence criteria for fission source L2 norm and max iterations

     OneD_matlab     (A,M,x,NP,L,eps,maxouter)               % solve by fission source iteration with Matlab direct flux inversion

     OneD_gmres      (A,M,x,NP,L,eps,maxouter)               % solve by fission source iteration with Matlab GMRES flux inversion

       maxinner=50;                                          % set maximum number of fission source iterations

       OneD_iterative (A,M,x,NP,L,eps,maxouter,maxinner)     % solve by fission source iteration with Matlab GMRES flux inversion

end
```

Constructing Destruction and Production Matrix

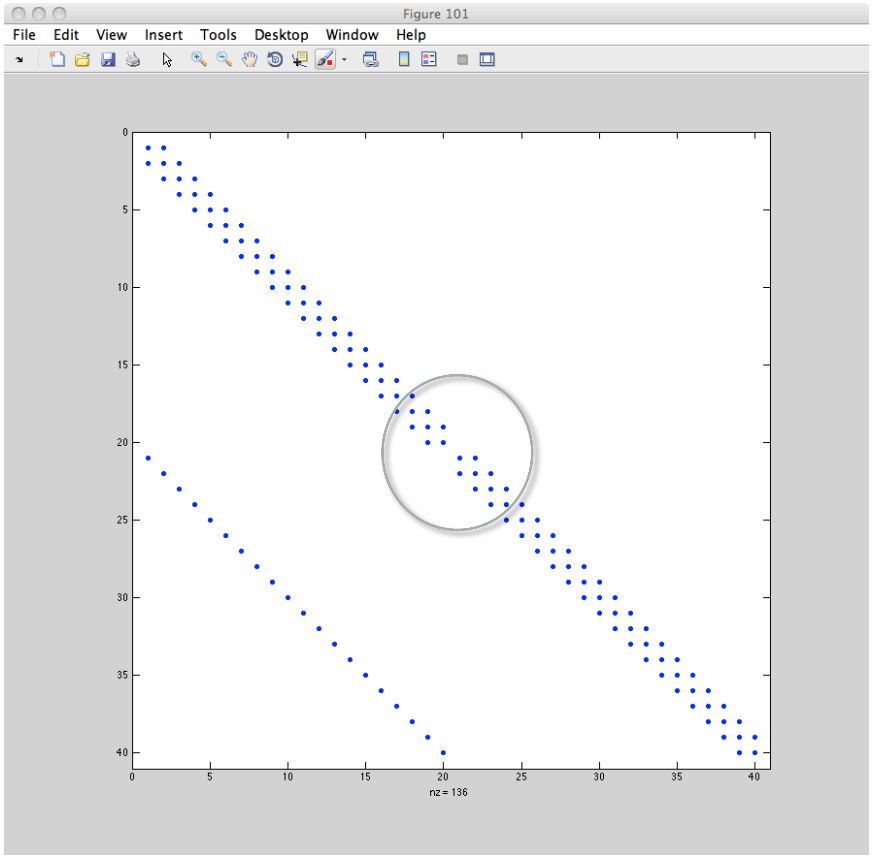
```
function [A,M,x,NP,L] = Diffusion_setup (xs,w,n,m,NZONE,bc)
%
% build mesh
%
  NP=0; L=0;
  for j=1:NZONE; dx=w(j)/n(j); L=L-
%
% build removal/fission matrix
%
  A([1:2*NP],[1:2*NP])=0.0; M([1:2*NP],[1:2*NP])=0.0; D([1:2*NP])=0.0; np=0;
  for j=1:NZONE
    mat=m(j);
    d1=xs(1,mat); d2=xs(2,mat); siga1= xs(3,mat); siga2= xs(4,mat);
                  sig12=xs(5,mat); nsigf1=xs(6,mat); nsigf2=xs(7,mat);
    for mm=1:n(j)
      np=np+1;
      A(np,np)       =h(np)*(siga1+sig12);     % group 1 removal
      A(NP+np,NP+np)=h(np)*(siga2       );     % group 2 removal
      A(NP+np,np)    =h(np)*(-sig12     );     % group 1 scattering source
      M(np,    np)   =h(np)*(nsigf1     );     % group 1 fission
      M(np,NP+np)    =h(np)*(nsigf2     );     % group 2 fission
      D(np)    =d1/h(np);                      % group 1 Ds
      D(NP+np)=d2/h(np);                       % group 2 Ds
    end
  end
  for NPP=0:NP:NP
    for n=1:NP-1; np=NPP+n;          % add net curent term for upper surface
      dhm=D(np);dhp=D(np+1);dsum=dhp+dhm;
      coeffp=2.*dhm*(1-dhm/dsum); coeffm=2.*dhp*dhm/dsum;
      A(np,np+1)=A(np,np+1)-coeffp; A(np,np)=A(np,np)+coeffm;
    end
    for n=2:NP; np=NPP+n;            % add net curent term for lower surface
      dhm=D(np-1);dhp=D(np);dsum=dhp+dhm;
      coeffp=2.*dhm*(1-dhm/dsum); coeffm=2.*dhp*dhm/dsum;
      A(np,np-1)=A(np,np-1)-coeffm; A(np,np)=A(np,np)+coeffp;
    end
  end
%
%  add bc terms: 0 is flux=0, 4.0 is Jin=0, infinity is zero current
%
  if bc == 0 factor=0; elseif bc == 1 factor=4.0; else factor=1.e20; end

  A(1    ,    1)=A(1    ,    1) + D(    1)*2./(1.+factor*D(1    ));
  A(NP   ,   NP)=A(NP   ,   NP) + D(   NP)*2./(1.+factor*D(NP   ));
  A(NP+1,NP+1)=A(NP+1,NP+1) + D(NP+1)*2./(1.+factor*D(NP+1));
  A(2*NP,2*NP)=A(2*NP,2*NP) + D(2*NP)*2./(1.+factor*D(2*NP));

 figure (101); spy(A)
 figure (102); spy(M)
end
```

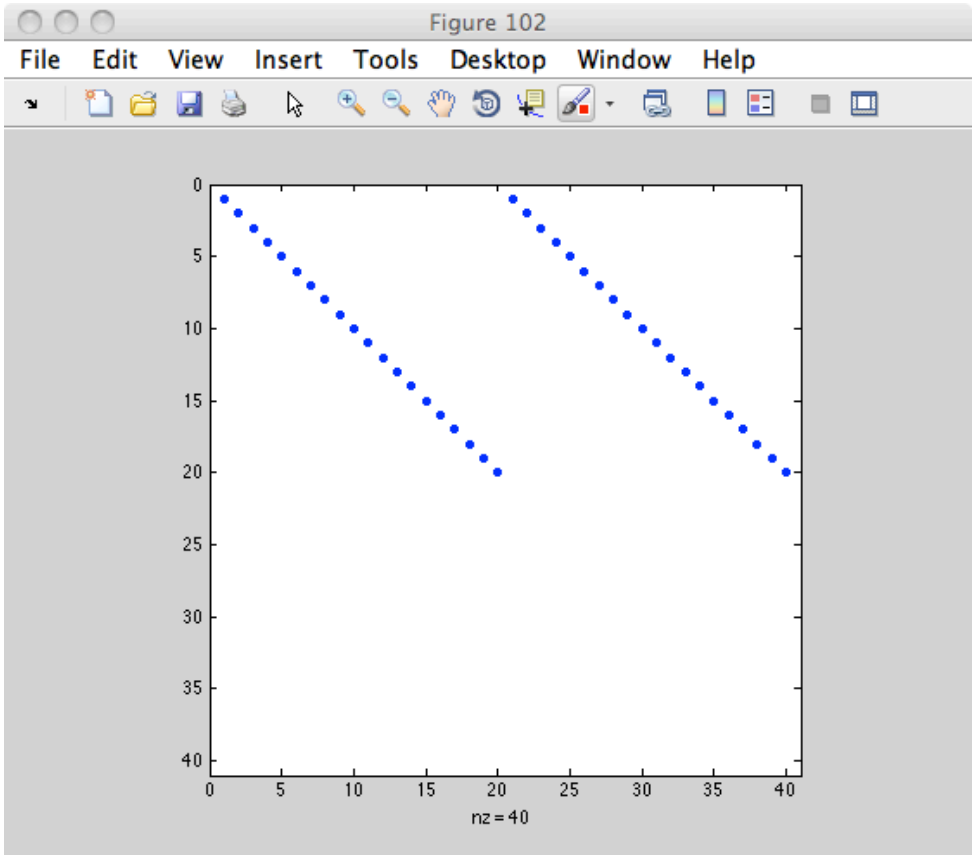
$$-\hat{D}_1^{n-1.n}\phi_1^{n-1}+\left[\Sigma_{r1}^n\Delta+\hat{D}_1^{n-1.n}+\hat{D}_1^{n,n+1}\right]\phi_1^n-\hat{D}_1^{n,n+1}\phi_1^{n+1}=\Sigma_{f1}^n\Delta\phi_1^n+\nu\Sigma_{f2}^n\Delta\phi_2^n+s_1^n\Delta$$

$$-\hat{D}_2^{n-1.n}\phi_2^{n-1}+\left[\Sigma_{a2}^n\Delta+\hat{D}_2^{n-1.n}+\hat{D}_2^{n,n+1}\right]\phi_2^n-\hat{D}_2^{n,n+1}\phi_2^{n+1}=\Sigma_{s12}^n\Delta\phi_1^n+s_2^n\Delta$$

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

Constructing Destruction and Production Matrix





$[A]$ $[M]$

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

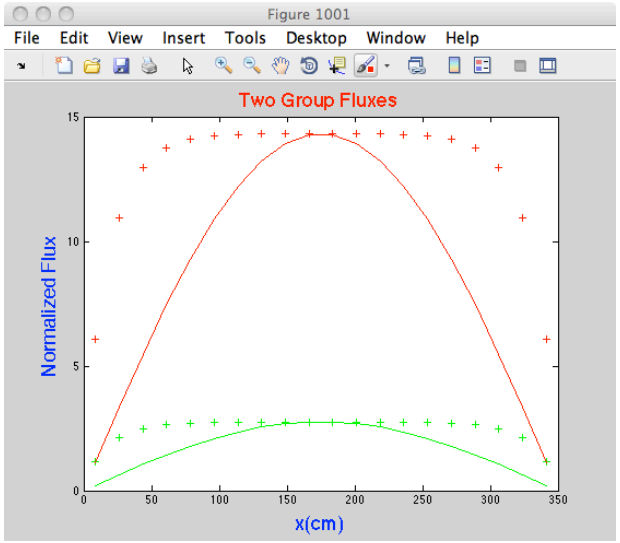
Using Sequential Source Solutions

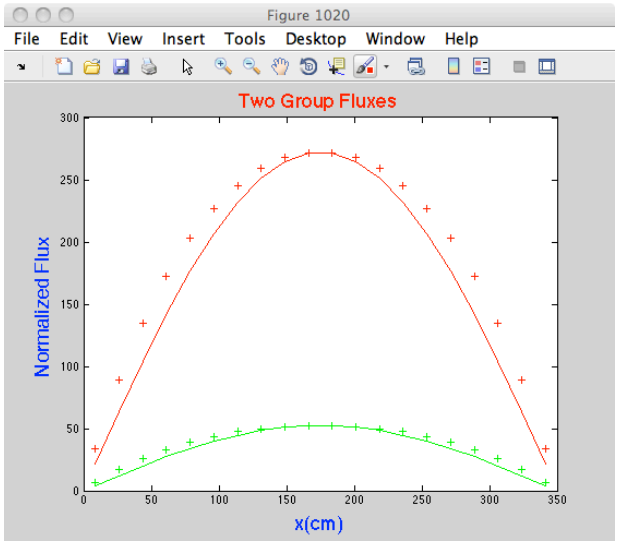
$$[A][\phi] = \frac{1}{k_{eff}}[M][\phi] + [S]$$

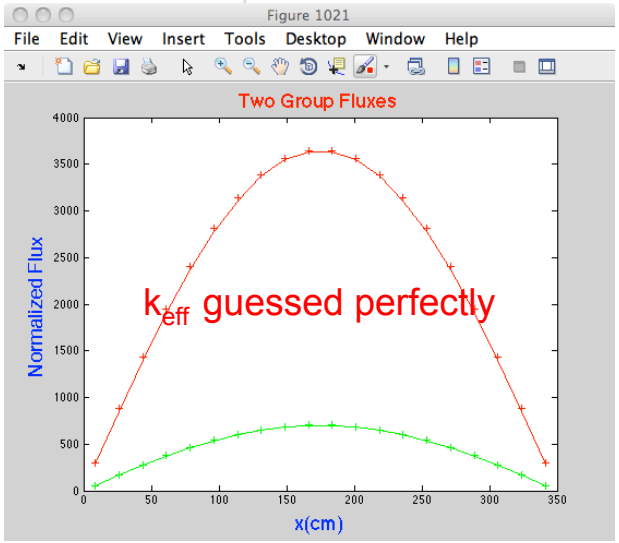
$$\left[A - \frac{1}{k_{eff}}M\right][\phi] = [S]$$

$$[\phi] = \left[A - \frac{1}{k_{guess}}M\right]^{-1}[S]$$

```
function OneD_source(A,M,x,NP,L)
%
% solve for two group fluxes using sequential source approximation
%
fprintf ('\n\n%s',' Fixed Source Solution');
  m=1000;
  for keff=1.569 :-.01: 1.3574    % guess keff
    m=m+1; MM=M/keff; s([1:2*NP])=0; s(1:NP)=1.;
    AA=A-MM; phi=(AA^-1)*s'; flux1=phi(1:NP); flux2=phi(NP+1:2*NP);
    max1=max(flux1); max2=max(flux2); y3=max1*sin(pi*x/L); y4=max2*sin(pi*x/L);
    mylinplot (m,x,flux1,flux2,y3,y4,'Two Group Fluxes','x(cm)','Normalized Flux',0,L,0,0);
  end
end
```







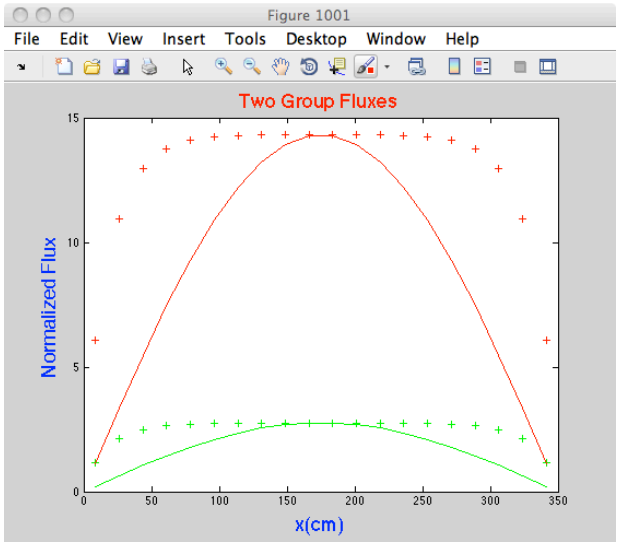
Using Sequential Source Solutions

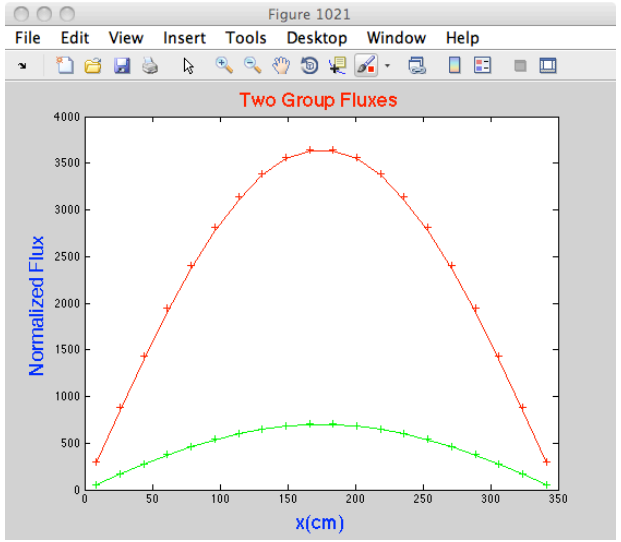
$$[A][\phi] = \frac{1}{k_{eff}}[M][\phi] + [S]$$

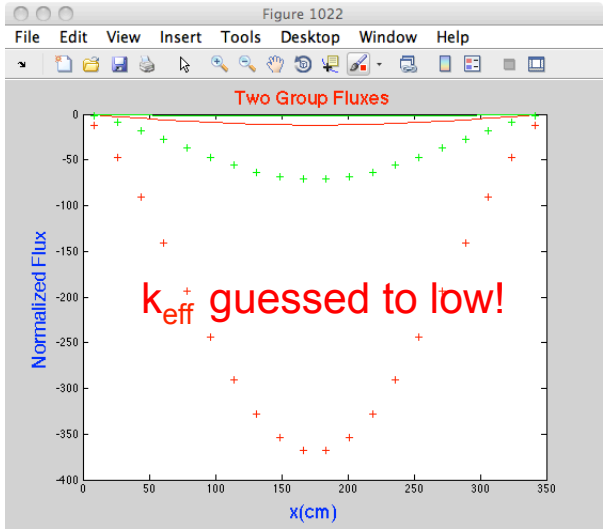
$$\left[A - \frac{1}{k_{eff}}M\right][\phi] = [S]$$

$$[\phi] = \left[A - \frac{1}{k_{guess}}M\right]^{-1}[S]$$

```
function OneD_source(A,M,x,NP,L)
%
% solve for two group fluxes using sequential source approximation
%
fprintf ('\n\n%s',' Fixed Source Solution');
  m=1000;
  for keff=1.569 :-.01: 1.3574    % guess keff
    m=m+1; MM=M/keff; s([1:2*NP])=0; s(1:NP)=1.;
    AA=A-MM; phi=(AA^-1)*s'; flux1=phi(1:NP); flux2=phi(NP+1:2*NP);
    max1=max(flux1); max2=max(flux2); y3=max1*sin(pi*x/L); y4=max2*sin(pi*x/L);
    mylinplot (m,x,flux1,flux2,y3,y4,'Two Group Fluxes','x(cm)','Normalized Flux',0,L,0,0);
  end
end
```



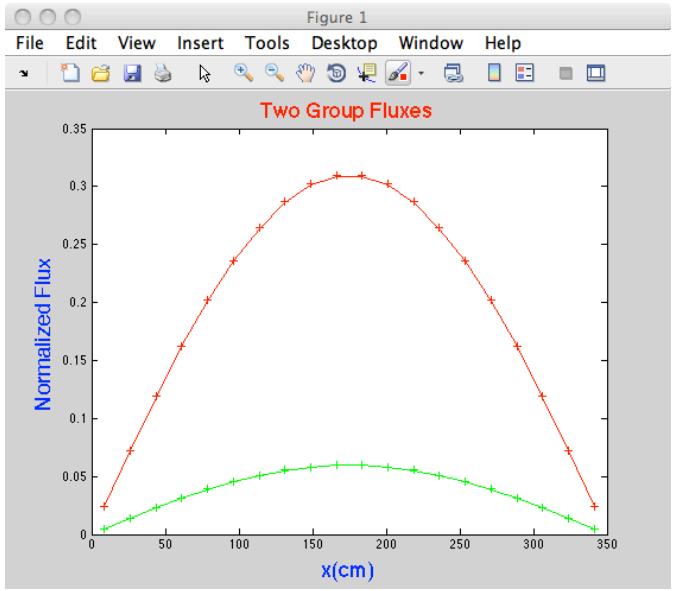




Using MATLAB's Direct Eigenvalue Solver

$$[A][\phi] = \frac{1}{k_{eff}}[M][\phi] \qquad [A]^{-1}[M][\phi] = k_{eff}[\phi]$$

```
function OneD_eigen(A,M,x,NP,L)
%
% Direct MATLAB Eigenfunctions
%
   fprintf ('\n\n%s',' MATLAB Direct Solution');
   [V,BB]=eigs((A^-1)*M); flux1=abs(V(1:NP,1)); flux2=abs(V(NP+1:2*NP,1));
   figure(333); plot (V(:,1));
   figure(334); plot (V(:,2));
   figure(335); plot(V)
   diag(diag(BB))
   fprintf ('%s','                                                    k-eff = ');
   max1=max(flux1); max2=max(flux2); y3=max1*sin(pi*x/L); y4=max2*sin(pi*x/L);
   mylinplot (1,x,flux1,flux2,y3,y4,'Two Group Fluxes','x(cm)','Normalized Flux',0,L,0,0);
end
```



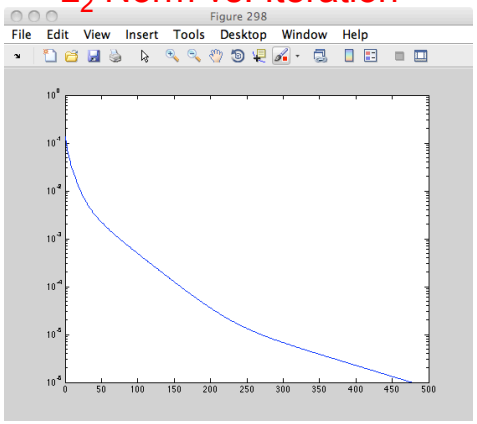

```
 MATLAB Direct Solution
ans =

    1.3672         0         0         0         0         0
         0    1.3527         0         0         0         0
         0         0    1.3297         0         0         0
         0         0         0    1.2994         0         0
         0         0         0         0    1.2634         0
         0         0         0         0         0    1.2236

                                                  k-eff =    1.3671514
```

Power Iteration with MATLAB's Direct Matrix Inverter

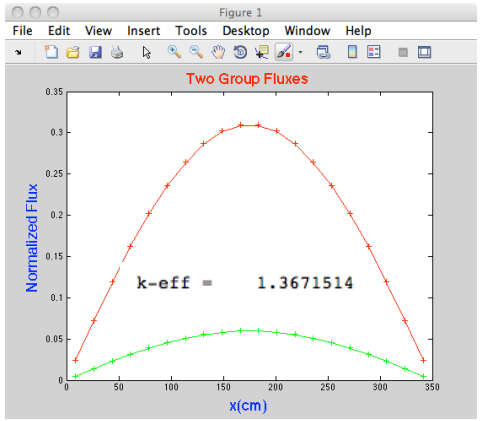
$$[A][\phi]^{i+1} = \frac{1}{k_{eff}}[M][\phi]^{i} \qquad [\phi]^{i+1} = \frac{1}{k_{eff}}[A]^{-1}[M][\phi]^{i} \qquad k_{eff} = \frac{[M][\phi]^{i+1}}{[M][\phi]^{i}}$$

```
function OneD_matlab(A,M,x,NP,L,criteria,maxouter)
%
% 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','                          iter, keff, eps, dr'); fprintf ('%12.7f',iter, keff,eps,approxdr);
   flux1=phi(1:NP); flux2=phi(NP+1:2*NP);
   max1=max(flux1); max2=max(flux2); y3=max1*sin(pi*x/L); y4=max2*sin(pi*x/L);
   mylinplot (2,x,flux1,flux2,y3,y4,'Two Group Fluxes','x(cm)','Normalized Flux',0,L,0,0);
   figure (298); semilogy(epssave(2:iter-1)); figure (299); plot (drsave(2:iter-1)); ylim([0.9,1.0]);
end
```

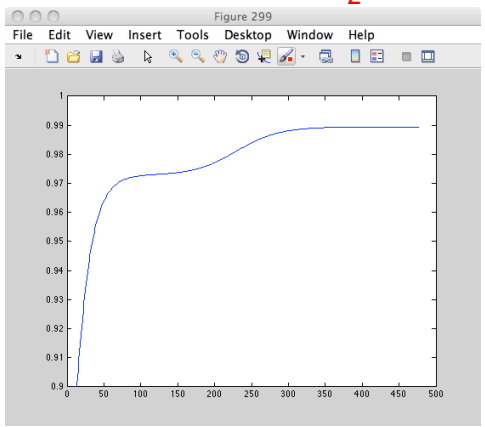
L_2 Norm vs. Iteration



Final Flux Solution



Ratio of Successive L_2 Norms

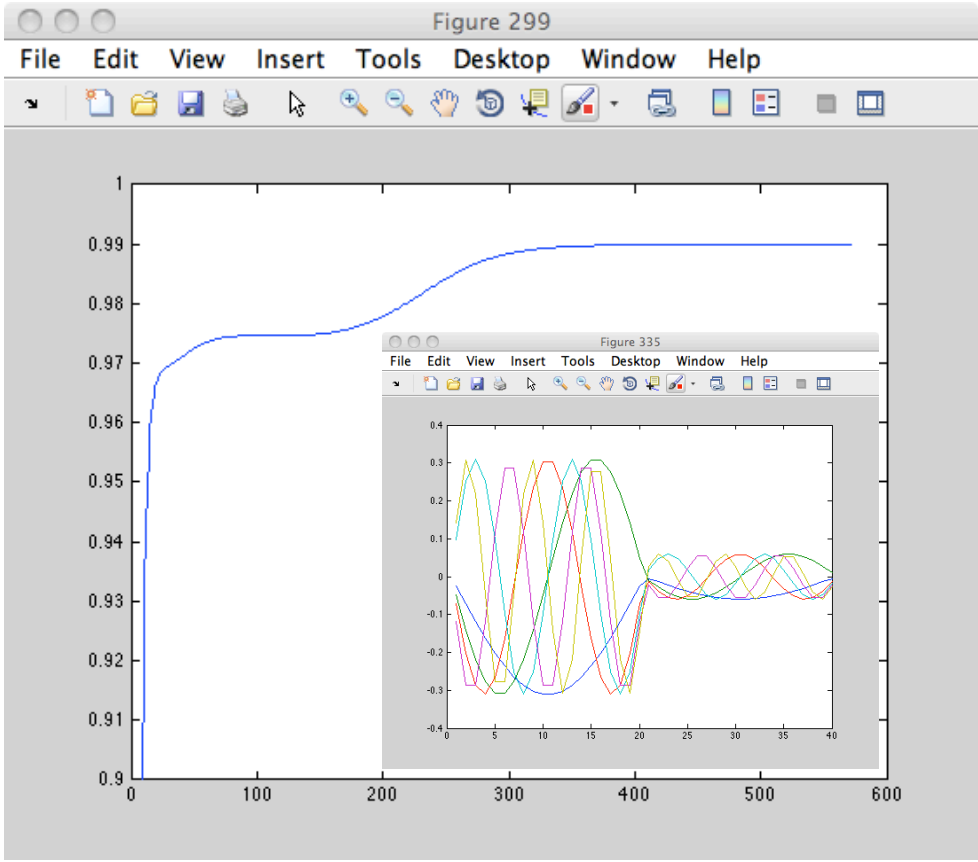


Power Iteration and Dominance Ratio

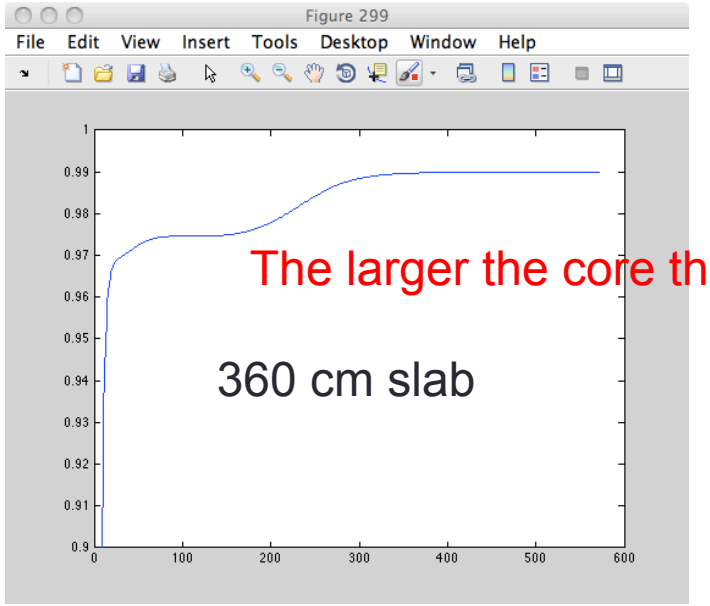
- Each mode has a dominance ratio, $dr_n = \frac{\lambda_n}{\lambda_1}$
- Power iteration kills off the lowest dominance ratio modes first
- Last remaining mode is the fundamental mode
- If starting guess is symmetric, asymmetric modes do not exist, (and do not need to be killed off)
- Random guess excites all modes

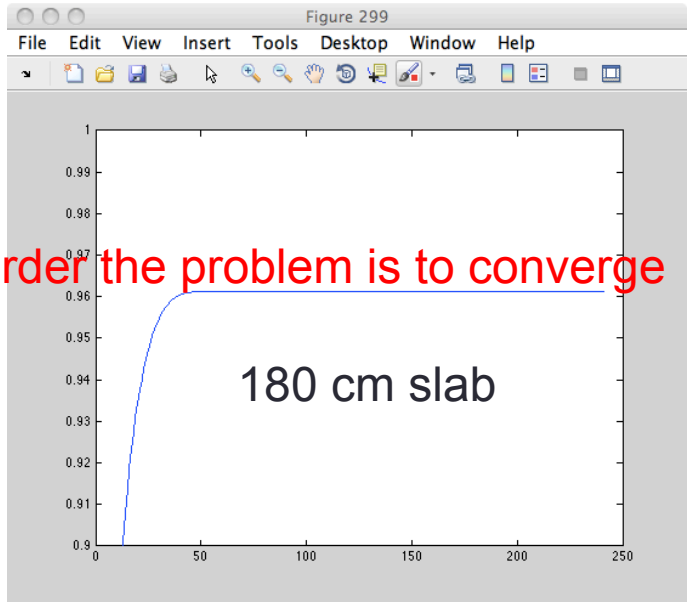
```
 MATLAB Direct Solution
ans =

    1.3672         0         0         0
         0    1.3527         0         0
         0         0    1.3297         0
         0         0         0    1.2994
         0         0         0         0
         0         0         0         0
```

$$dr = \frac{\lambda_2}{\lambda_1} \approx 0.99$$

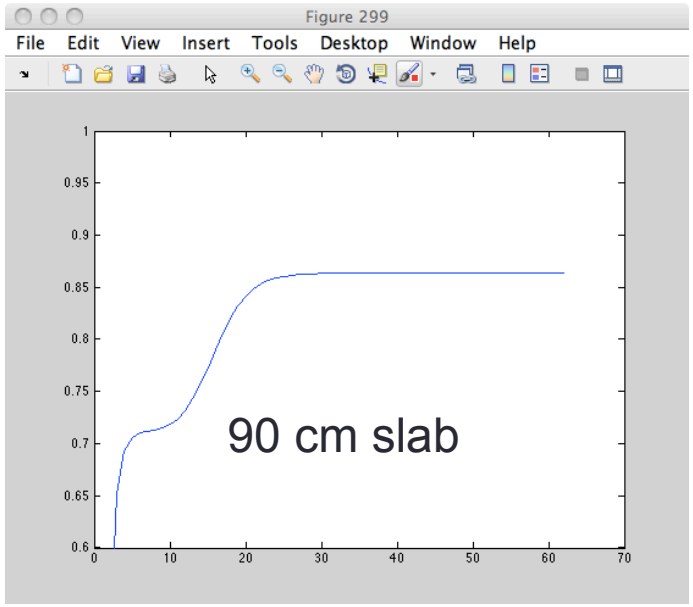


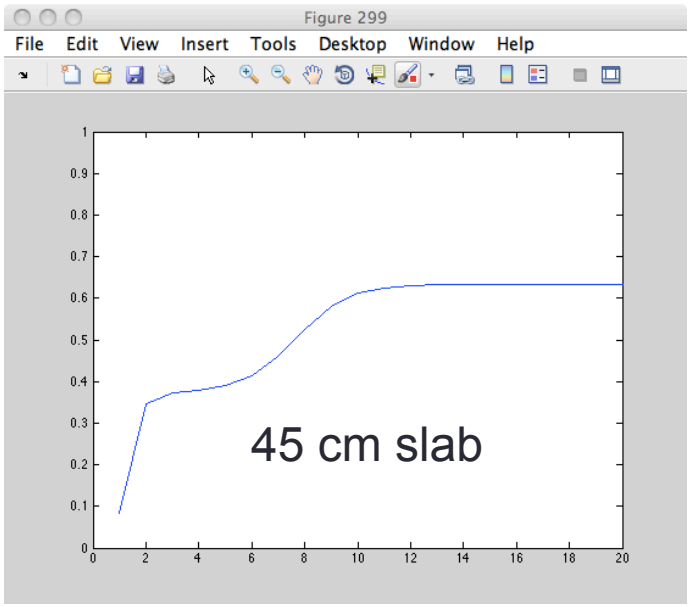
Dominance Ratio vs. Core Size





The larger the core the harder the problem is to converge

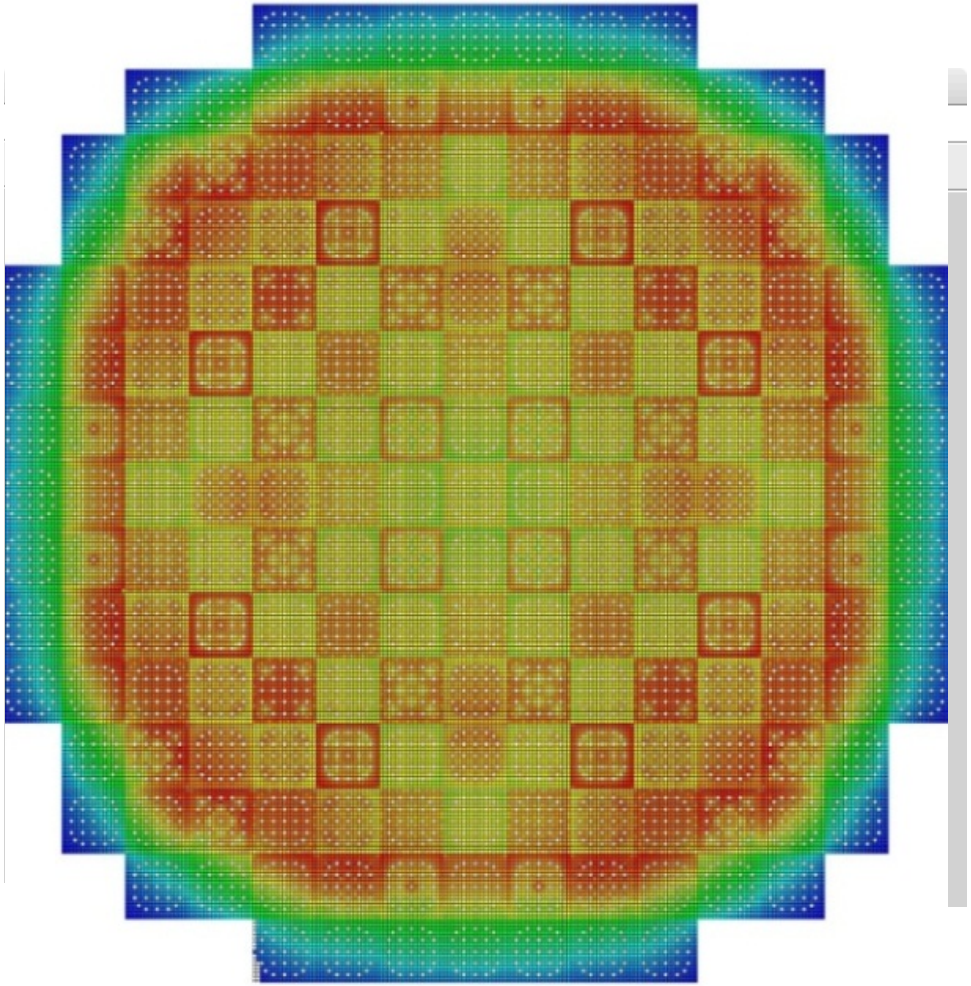




Dominance Ratio Sensitivities

Dominance ratio is effected by:

- Core size
- Decoupling of radial zones:
  - Control rod insertion
  - Asymmetric core loadings
  - Xenon distributions
- Axial zones:
  - Partially inserted control rods
  - Axial fuel enrichment zoning
  - Axial burnable absorber loading



- Dominance ratio (spatial decoupling) is very important characteristic for the design and control of reactors – not just a convergence issue.

Power Iteration with MATLAB's GMRES Numerical Invertion

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

```
function OneD_gmres(A,M,x,NP,L,criteria,maxouter)
%
% solve eignevalue problem iteratively, GMRES inversion of flux matrix
%
  fprintf ('\n\n%s',' Iterative Fission Source Solution by GMRES');
  epsold=10.; phi(1:2*NP,1)=1.0; 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=gmres(A,sold,25,1.e-6);
      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','              iter, keff, eps, dr'); fprintf ('%12.7f',iter, keff,eps,approxdr);
  flux1=phi(1:NP); flux2=phi(NP+1:2*NP);
  max1=max(flux1); max2=max(flux2); y3=max1*sin(pi*x/L); y4=max2*sin(pi*x/L);
  mylinplot (3,x,flux1,flux2,y3,y4,'Two Group Fluxes','x(cm)','Normalized Flux',0,L,0,0);
  figure (398); semilogy(epssave(2:iter-1)); figure (399); plot (drsave(2:iter-1)); ylim([0.9,1.0]);
end
```

```
 Iterative Fission Source Solution by GMRESgmres(25) converged at outer iteration 1 (inner iteration 10) to a solution with relative residual 4e-07.
gmres(25) converged at outer iteration 1 (inner iteration 10) to a solution with relative residual 3.3e-07.
gmres(25) converged at outer iteration 1 (inner iteration 10) to a solution with relative residual 2.7e-07.
gmres(25) converged at outer iteration 1 (inner iteration 9) to a solution with relative residual 8.9e-07.
gmres(25) converged at outer iteration 1 (inner iteration 9) to a solution with relative residual 7.1e-07.
gmres(25) converged at outer iteration 1 (inner iteration 9) to a solution with relative residual 5.6e-07.
gmres(25) converged at outer iteration 1 (inner iteration 9) to a solution with relative residual 4.5e-07.
gmres(25) converged at outer iteration 1 (inner iteration 9) to a solution with relative residual 3.6e-07.
gmres(25) converged at outer iteration 1 (inner iteration 9) to a solution with relative residual 2.8e-07.
gmres(25) converged at outer iteration 1 (inner iteration 9) to a solution with relative residual 2.2e-07.
gmres(25) converged at outer iteration 1 (inner iteration 9) to a solution with relative residual 1.7e-07.
gmres(25) converged at outer iteration 1 (inner iteration 8) to a solution with relative residual 9.5e-07.
gmres(25) converged at outer iteration 1 (inner iteration 8) to a solution with relative residual 7.4e-07.
gmres(25) converged at outer iteration 1 (inner iteration 8) to a solution with relative residual 5.8e-07.
gmres(25) converged at outer iteration 1 (inner iteration 8) to a solution with relative residual 4.5e-07.
gmres(25) converged at outer iteration 1 (inner iteration 7) to a solution with relative residual 8.2e-07.
gmres(25) converged at outer iteration 1 (inner iteration 7) to a solution with relative residual 6.4e-07.
gmres(25) converged at outer iteration 1 (inner iteration 7) to a solution with relative residual 4.9e-07.
gmres(25) converged at outer iteration 1 (inner iteration 7) to a solution with relative residual 3.8e-07.
gmres(25) converged at outer iteration 1 (inner iteration 7) to a solution with relative residual 3e-07.
```

Power Iteration with Point-Jacobi Iterative Flux Matrix Inversion

$$[A][\phi]^{i+1} = \frac{1}{k_{eff}}[M][\phi]^{i} \qquad [A]=[D]+[O] \quad \Rightarrow [D][\phi]^{m+1} = \frac{1}{k_{eff}}[M][\phi]^{i} - [O][\phi]^{m}$$

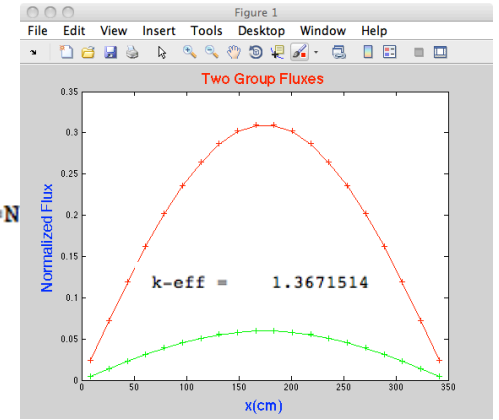
$$[\phi]^{m+1} = [D]^{-1}\left\langle \frac{1}{k_{eff}}[M][\phi]^{i} - [O][\phi]^{m} \right\rangle \qquad k_{eff} = \frac{[M][\phi]^{i+1}}{[M][\phi]^{i}}$$

```
function OneD_iterative(A,M,x,NP,L,criteria,maxouter,maxinner)
%
% solve eignevalue problem with both fission source and flux iteration
%
  fprintf ('\n\n%s',' Iterative Fission Source and Iterative Flux Inversion');
  DIAG=diag(diag(A)); O=A-DIAG;
  phi(1:2*NP,1)=1.0; sold=(M*phi); sumold=sum(sold); sold=sold/(sumold/NP); sumold=N
  for outer=1:maxouter
      for inner=1:maxinner
        phi=sold-O*phi;
        for np=1:2*NP
            phi(np)=phi(np)/DIAG(np,np) ;
        end
      end
      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);
      if outer > 50 && eps < criteria;  break; end
  end
  fprintf ('%s','   iter, keff, eps,    '); fprintf ('%12.7f',outer,keff,eps);
  flux1=phi(1:NP); flux2=phi(NP+1:2*NP);
  max1=max(flux1); max2=max(flux2); y3=max1*sin(pi*x/L); y4=max2*sin(pi*x/L);
  mylinplot (4,x,flux1,flux2,y3,y4,'Two Group Fluxes','x(cm)','Normalized Flux',0,L,0,0);
end
```

Outer : Fission Source Iteration

Inner : Flux Iteration

Final Flux Solution



Power Iteration with Gauss-Seidel Iterative Flux Matrix Inversion

Point-Jacobi

$$[A][\phi]^{i+1} = \frac{1}{k_{eff}}[M][\phi]^{i} \qquad [A]=[D]+[O] \Rightarrow [D][\phi]^{m+1} = \frac{1}{k_{eff}}[M][\phi]^{i} - [O][\phi]^{m}$$

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

Gauss-Seidel

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

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

example

$$\begin{bmatrix} a_{11} & & & & \\ a_{21} & a_{22} & & & \\ a_{31} & a_{32} & a_{33} & & \\ a_{41} & a_{42} & a_{43} & a_{44} & \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{bmatrix} \begin{bmatrix} \phi_1^{m+1} \\ \phi_2^{m+1} \\ \phi_3^{m+1} \\ \phi_4^{m+1} \\ \phi_5^{m+1} \end{bmatrix} = \begin{bmatrix} FS_1^{i} \\ FS_2^{i} \\ FS_3^{i} \\ FS_4^{i} \\ FS_5^{i} \end{bmatrix} - \begin{bmatrix} & a_{12} & a_{13} & a_{14} & a_{15} \\ & & a_{23} & a_{24} & a_{25} \\ & & & a_{34} & a_{35} \\ & & & & a_{45} \\ & & & & \end{bmatrix} \begin{bmatrix} \phi_1^{m} \\ \phi_2^{m} \\ \phi_3^{m} \\ \phi_4^{m} \\ \phi_5^{m} \end{bmatrix}$$

Same number of operations as P-J:

Better Convergence

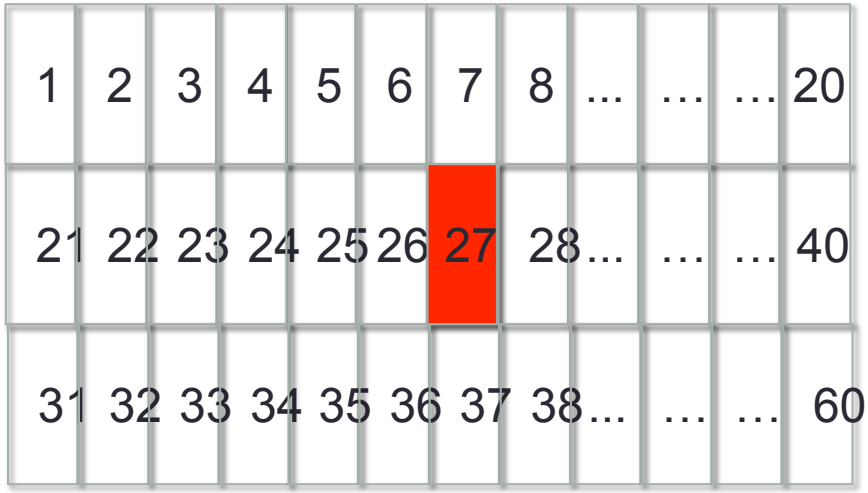
Numerical Solutions

Remember for real reactors and large problems:

- Matrix inversion is of order N^3
- Finding all eigenvalues is at least order N^2
- Iterative inversion must be order N [or at least N log(N)] to be practical for large problems
- Multi-level iteration is a practical necessity
- Iterative methods must work for 1D, 2D, or 3D to be practical

2-Group Finite-Difference Matrix Equations in 2D

We number our cells consecutively,



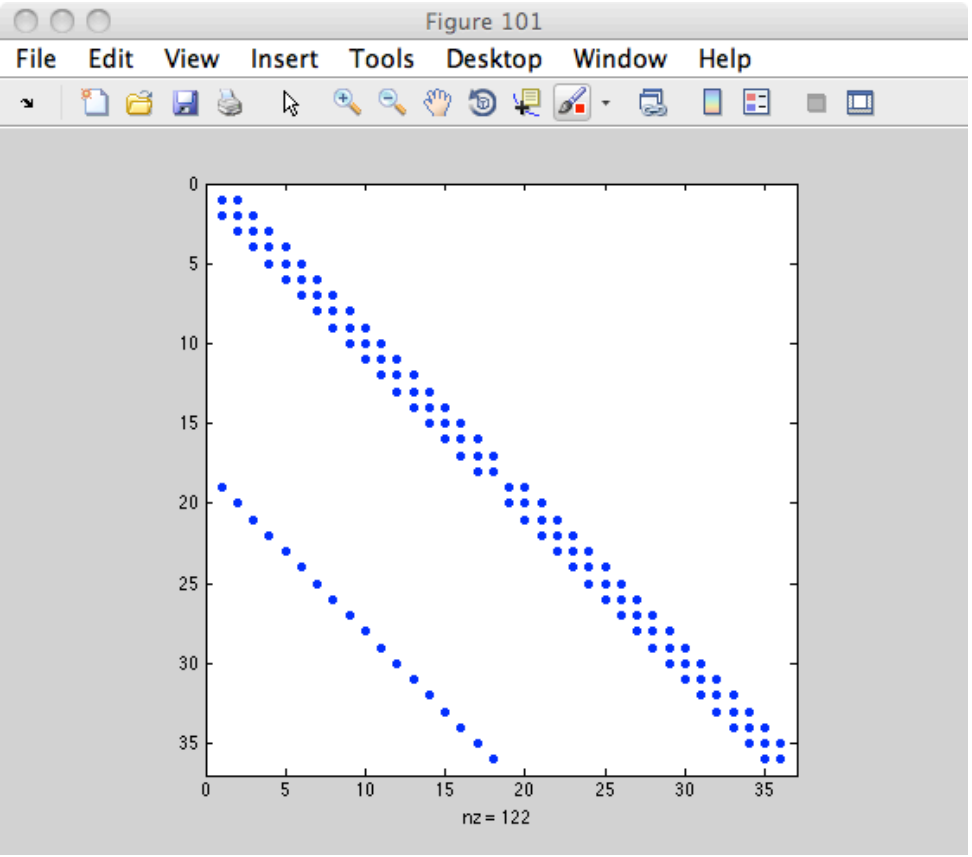
Defining a vector of group fluxes of group one followed by group two fluxes:

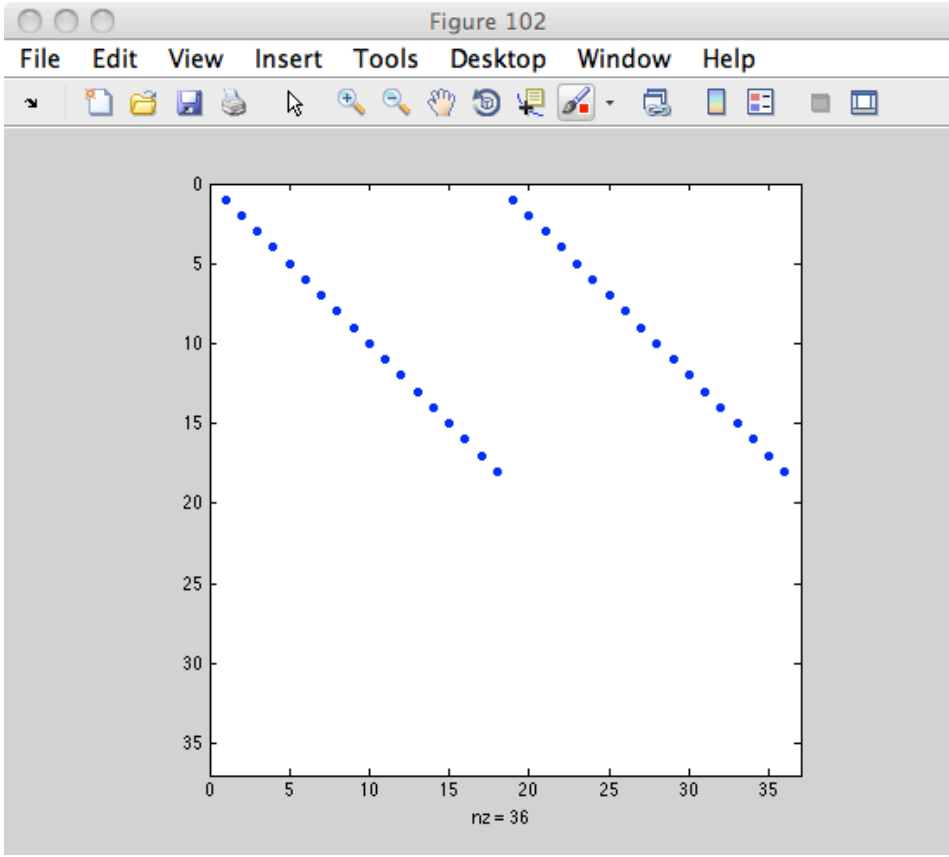
$$\begin{bmatrix} [L+D+U]_1 & [0] \\ -[T]_2 & [L+D+U]_1 \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} = \begin{bmatrix} [M]_1 & [M]_2 \\ [0] & [0] \end{bmatrix} \begin{bmatrix} \phi_1 \\ \phi_2 \end{bmatrix} + \begin{bmatrix} S_1 \\ S_2 \end{bmatrix}$$

Or in compressed matrix notation:

$$[A][\phi] = [M][\phi] + [S]$$

Constructing Destruction and Production Matrix in 1D



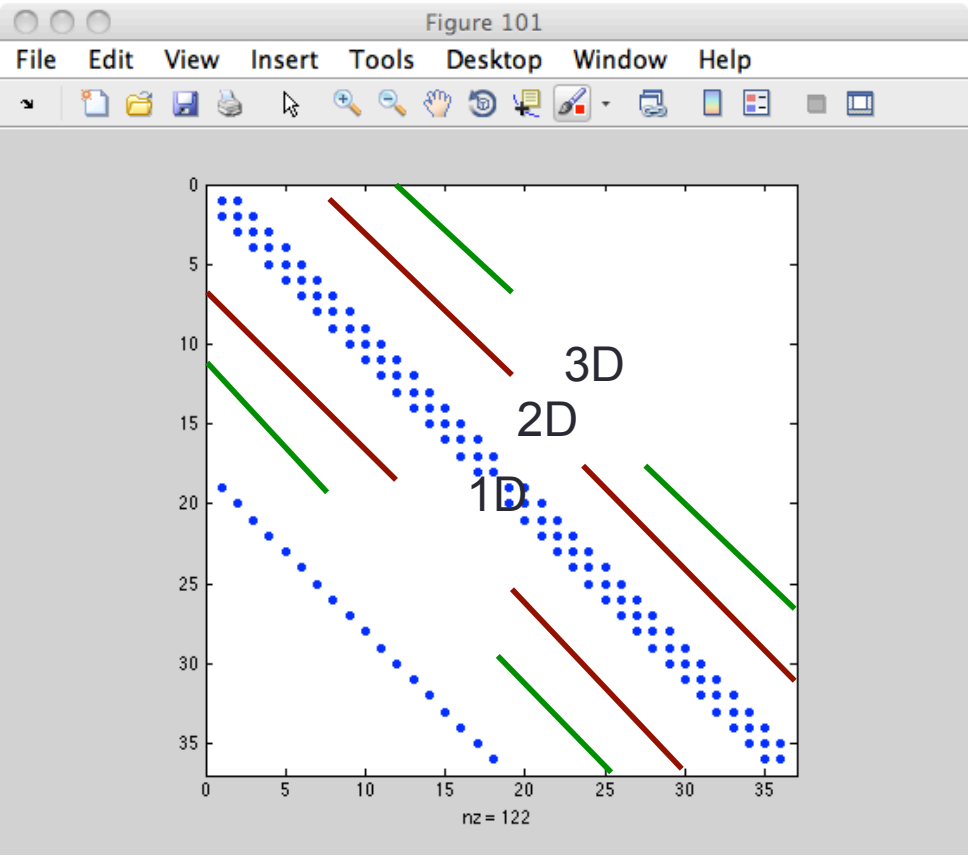


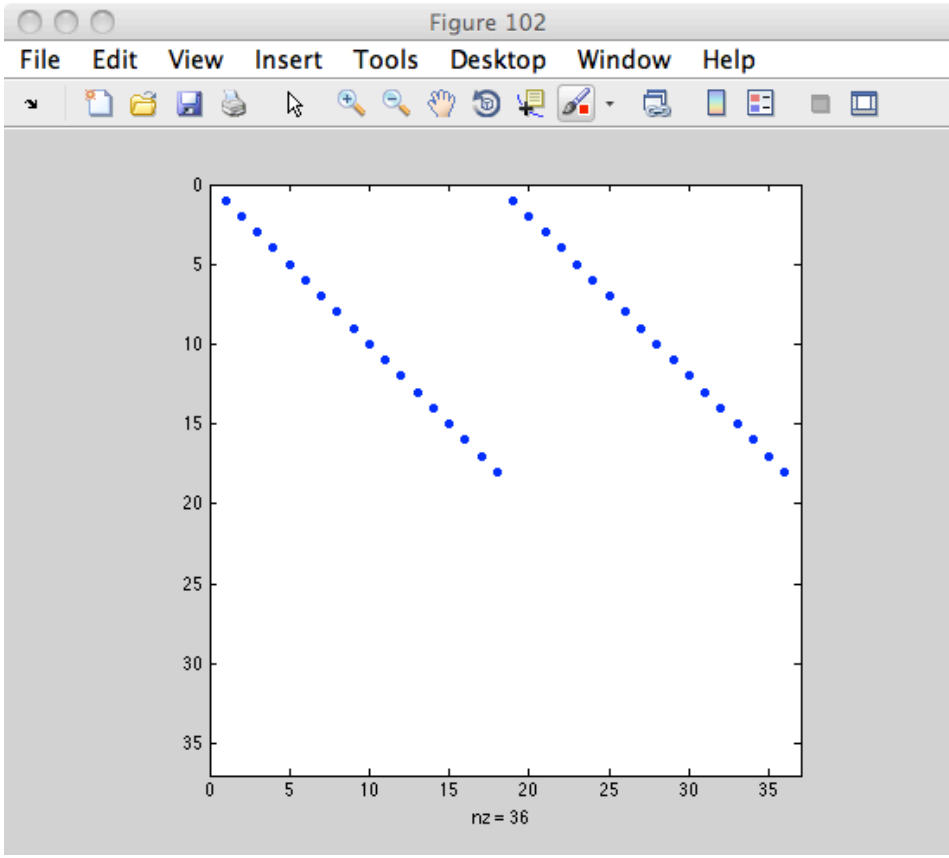
A

M

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

Constructing Destruction and Production Matrix in 2D





A

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

M

1-D

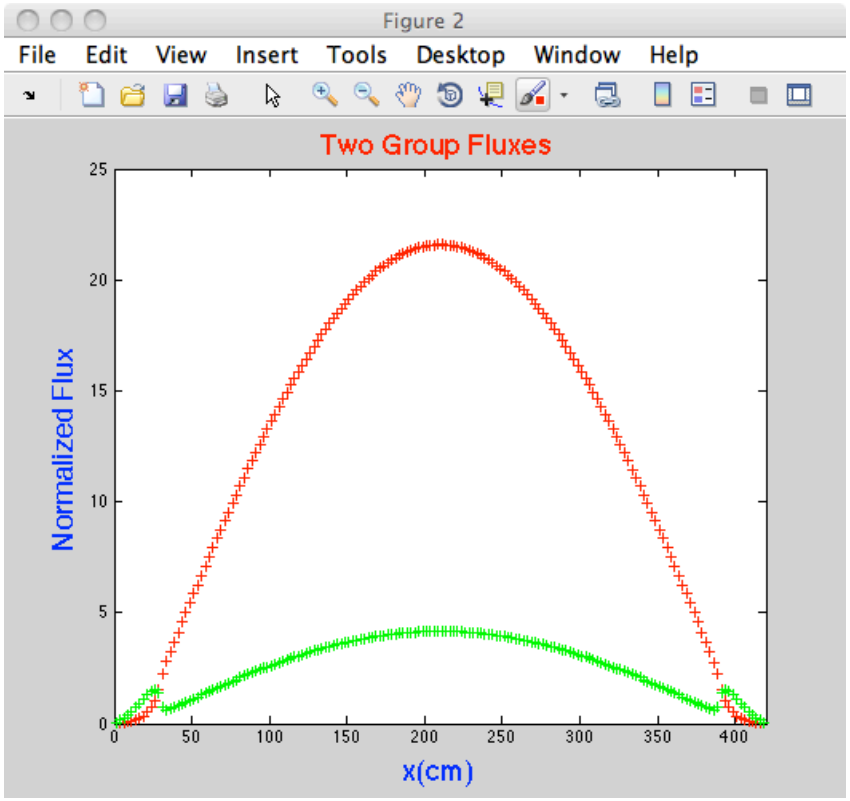
$$-\hat{D}_1^{n-1.n}\phi_1^{n-1} + \left[\Sigma_{r1}^n\Delta + \hat{D}_1^{n-1.n} + \hat{D}_1^{n,n+1}\right]\phi_1^n - \hat{D}_1^{n,n+1}\phi_1^{n+1} = \Sigma_{f1}^n\Delta\phi_1^n + \nu\Sigma_{f2}^n\Delta\phi_2^n + s_1^n\Delta$$

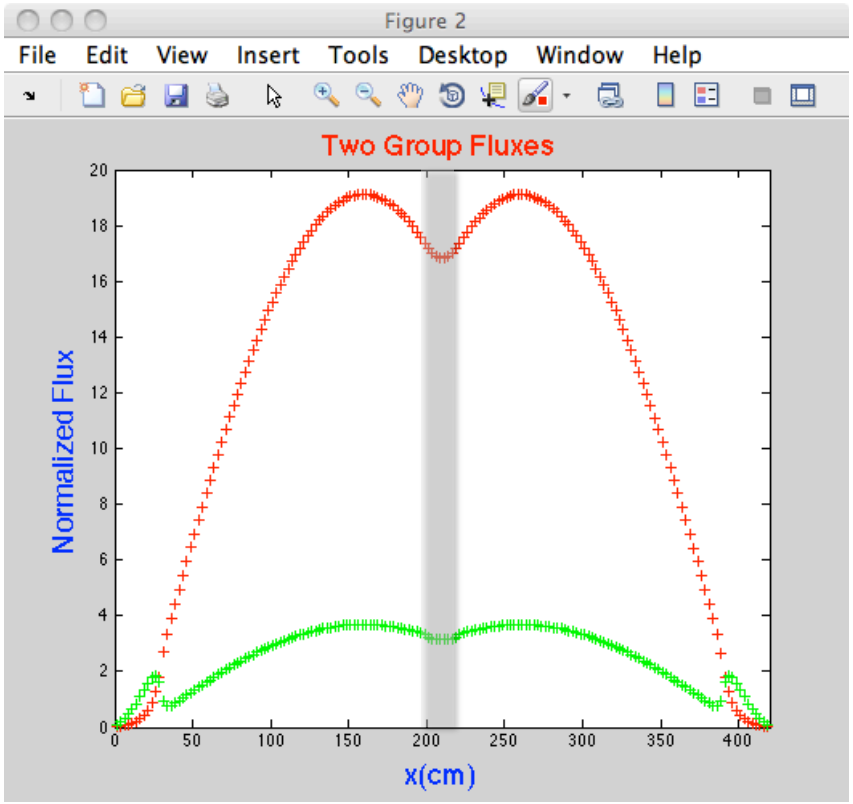
$$-\hat{D}_2^{n-1.n}\phi_2^{n-1} + \left[\Sigma_{a2}^n\Delta + \hat{D}_2^{n-1.n} + \hat{D}_2^{n,n+1}\right]\phi_2^n - \hat{D}_2^{n,n+1}\phi_2^{n+1} = \Sigma_{s12}^n\Delta\phi_1^n + s_2^n\Delta$$

2-D?
3-D?

Where are we headed next?

Control Rod Insertions in 1-D:





Simulations in 1-D:

Statics
Dynamics
Point-Kinetics
Generalized Point-Kinetics
Improved Quasi-Static Methods
Numerical Methods for Solutions

Assignment for Next Class

- Finish PSet 1
- Look at solution techniques for 1-D finite-difference diffusion equation
- Start writing a generalized solver for 1-D, 2-group diffusion equations
- If you get bored, write solver for 2-D, 2-group steady-state LRA problem