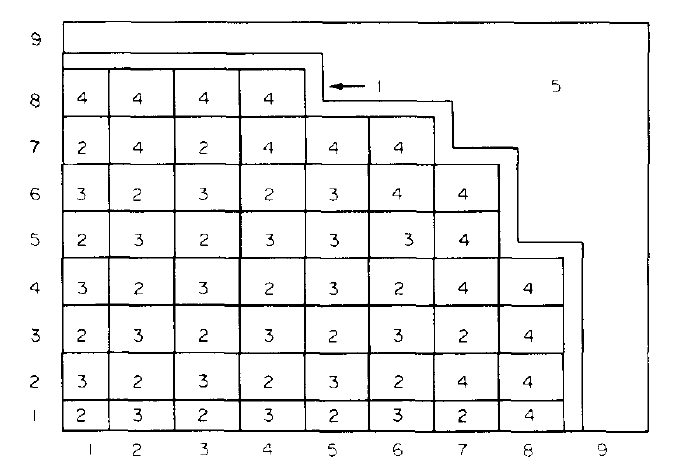
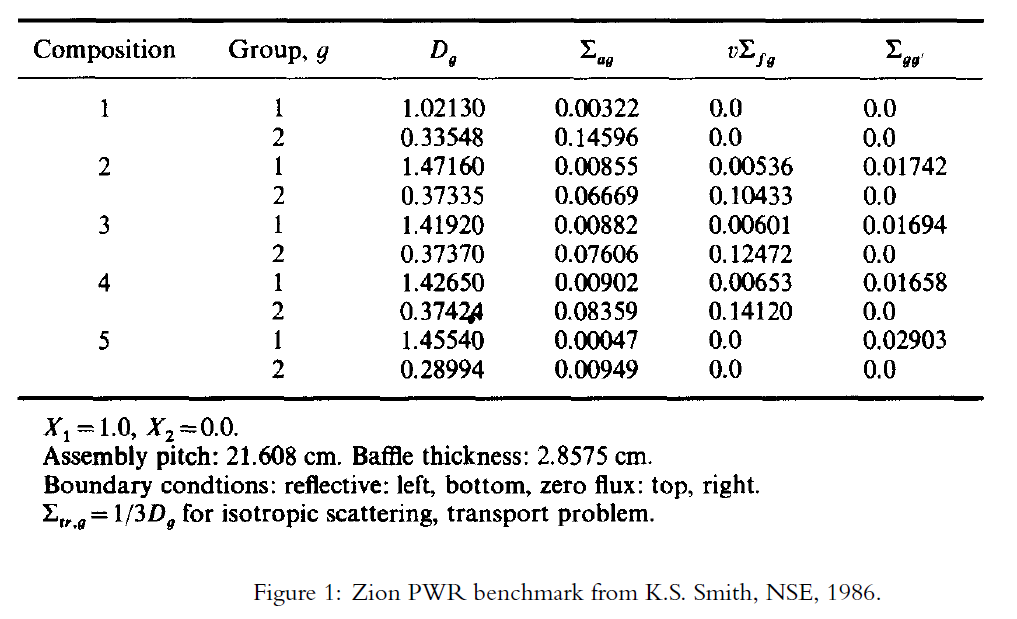
1. The Zion PWR Benchmark (K.S. Smith, NSE, 1986), is a 2-D, two-group, reactor benchmark calculation where a quarter reactor is specified by the following figure:





A quarter of the reactor is a square that is divided in the figure into a 9 x 9 grid of 21.608 x 21.608 cm squares (i.e., the quarter reactor size is 194.472 x 194.472 cm). At the midline there are 8 assemblies in the x and y directions. In the problem specification there is only downscattering and what we have called Σrg is called Σag. Also, note that all fission neutrons are born fast.

Solve the problem using finite difference, nodal and CMFD acceleration.

Starting with the time-dependent multigroup diffusion equation:

I believe that the different methods argue about , so let’s lump and define the other terms first (with multiple regions and forbearing the time term):

Fission Source Time Inscattering

n indices added for time (would need to add to previous equations)

Absorb Outscatter Time

For a 1 group problem the fission term should be coupled with the removal term (no summation just )

This simplifies our problem to:

This turns into a matrix solve:

Notes to self:

Iterate till convergence

I do not believe A changes from iteration to iteration.

Solve group by group (fast to slow) using solutions from previous groups to update solution.

If you wanted to add time dependence maybe should add delayed neutrons and solve for , after the flux converges then move onto next time step.

1. **Finite Difference:**

Discrete equations for a zone away from the boundary (2-D):

Where:

For left boundary:

Where:

If B =0

Some algebra is needed to figure what goes where in A, but it is the hope of the student that Dr. McClarren’s code already helps with this.

Notes to self about the blessings of code (James 1:17):

**def** diffusion\_steady\_fixed\_source(Dims,Lengths,BCs,D,Sigma,Q, tolerance=1.0e-12, LOUD=False):

*"""Solve a steady state, single group diffusion problem with a fixed source*

*Inputs:*

*Dims [1,3]: number of zones (I,J,K)*

*Lengths [1,3]: size in each dimension (Nx,Ny,Nz): Total size of problem*

*BCs [6,3]: A, B, and C for each boundary, there are 6 of these*

*D,Sigma,Q [I,J,K]: Properties at each cell center. Each is an array of size (I,J,K) con taining the quantity. Need to determine before hand these according to the size in each dimension*

*Outputs:*

*x,y,z: Vectors containing the cell centers in each dimension*

*phi: A vector containing the solution*

*"""*

Needs while loop to converge upon a solution (added later to multigroup). Q needs to be updated through the iteration(added later to multigroup), lattice doesn’t necessarily need to be called. Somehow Q needs to be modified with the flux. Sigma doesn’t need to be modified (but is every iteration within multigroup diffusion).

**def** lattice(Lengths,Dims):

“Defines D,Sigma, and Q

…thank you

**def** time\_dependent\_diffusion(phi0, v, dt, T, Dims,Lengths,BCs,D,Sigma,Q, tolerance=1.0e-12):

Make sure to change the diffusion\_steady\_fixed\_source Sigma to Sighat and Q to Qhat.

**def** lattice2G(Lengths,Dims):

Needs to be modified for the geometry. Q needs to be updated through the iteration, lattice doesn’t necessarily need to be called. Somehow Q needs to be modified with the flux. Sigma doesn’t need to be modified.

**def** steady\_multigroup\_diffusion(G,Dims,Lengths,BCGs,

Sigmatg,Sigmasgg,nuSigmafg,

nug,chig,D,Q,

lintol=1.0e-8,grouptol=1.0e-6,maxits = 12,

LOUD=False):

Need to remove a convergence criteria? I am not sure which, I guess the maxits. This will do the solving I need for the finite element problem. It will also solve many groups. The lattice 2G is 2group…but I am ok with that.

**def** time\_dependent\_mg\_diffusion(phi0, v, dt, T, G,Dims,Lengths,BCGs,

Sigmatg,Sigmasgg,nuSigmafg,nug,chig,D,Q,lintol=1.0e-8,grouptol=1.0e-6,maxits = 12):

This updates Qhat and Sighat. I will assume this is solved correctly.

**def** inner\_iteration(G,Dims,Lengths,BCGs,Sigmar,Sigmasgg,

D,Q,lintol=1.0e-8,grouptol=1.0e-6,maxits = 20,LOUD=False):

Is exactly like steady\_multigroup\_diffusion, except it doesn’t recalculate Sigmar…which is good.

**def** kproblem\_mg\_diffusion(G,Dims,Lengths,BCGs,Sigmarg,Sigmasgg,nuSigmafg,

chig,D,lintol=1.0e-8,grouptol=1.0e-6,tol=1.0e-8,maxits = 12, k = 1, LOUD=False):

solves a keigen value problem…what I need. Maybe its this one that needs to have convergence criteria removed.

Maybe do a first iteration with steady\_multigroup

Then do subsequent iterations with kproblem\_mg\_diffusion

I will not go beyond this to the alphas…

My “Pseudo Code” (once these functions are in proper order)

Sigmaag,Sigmasgg,nuSigmafg,nug,chig,D,Q = lattice2G((Nx,Nx,1),(I,J,K))

1. **Nodal Method:**