Exercise 9. Neutron Transport.

1. Consider a one-group representation of neutron transport in a square two-dimensional reactor of side length $2L$. The reactor has uniform material properties; so that the steady diffusion equation becomes

$$-D\nabla^2 F + (\Sigma_t - S)F - \frac{1}{k}GF = 0$$

where the diffusion coefficient (divided by speed) D, the total attenuation "macroscopic cross-section" Σ_t, the scattering and fission source terms S, G, are simply scalar constants. For convenience, write $\Sigma_t - S = \Sigma$. The eigenvalue k must be found for this equation.

The boundary conditions at $x, y = \pm L$ are that the neutron density, F, equals zero. That's only an approximate representation of actual physics.

Formulate the finite-difference diffusion equation on a uniform mesh of $N_x = N_y$ nodes; node spacing $h = 2L/(N_x + 1)$; so that $x_i = L[2i - (N_x + 1)]/(N_x + 1)$ for $i = 1, N_x$, $y_i = L[2i - (N_y + 1)]/(N_y + 1)$ for $j = 1, N_y$. That means the mesh indexes 0 and $N_x + 1$ correspond to the boundaries, but we don't actually include them in the representation, because their F-values are zero. Exhibit the difference equation in the form of a $N_xN_y \times N_xN_y$ matrix equation

$$[\mathbf{M} - \frac{1}{k}\mathbf{G}]\boldsymbol{F} = 0$$

writing out the matrix $\mathbf{M}$ explicitly for the (minimal) case $N_x = 3$ (so $\mathbf{M}$ is 9×9), carefully considering the incorporation of the boundary condition. [The column vector $\boldsymbol{F}$ should be arranged in column-major order; i.e.

$$\boldsymbol{F} = (F_{11}, F_{21}, F_{31}, F_{12}, F_{22}, F_{32}, F_{13}, F_{23}, F_{33})^T$$

(where the suffixes of F_{ij} refer to the x and y nodal numbers), and the matrices ordered correspondingly.]

2. Implement this finite difference scheme and (using some library function) find the eigenvalue, k, when $D = 1$, $\Sigma = 1$, $G = 1$, and $L = 2$ or $L = 10$. Use large enough N_x in your code that the solution is reasonably converged.

[Octave/MATLAB® hint. There are (in Octave) two routines for calculating eigenvalues: eig() and eigs(). Calling `eigs(M,K)` returns the largest `K` eigenvalues of the matrix `M`. Don't forget that the eigenvalue returned is λ solving $[\mathbf{M} - \lambda]F = 0$, in other words it is the inverse of k. We want just the *smallest* λ, which corresponds to the largest k. We can trick this routine into giving it by using the generalized eigenvalue form $\mathbf{I} - k\mathbf{M} = 0$, so calling `eigs(eye(Nx*Ny),M,1)`. The eigs() routine uses an iterative technique. The eig() routine returns all the eigenvalues. It uses a direct solution technique. One then has to find the smallest, and invert it to give k. For $N_x = 50$, which is beginning to stress these routines, eigs() takes about 8 seconds and eig() takes 70s. This shows some of the benefit of an iterative technique.]

22.15 Essential Numerical Methods

Fall 2014