Homework 3: Sparse Operators and Iterative Methods

Due Saturday, November 9th at 1 am.

Problem 1. Differential Operators. By this point, we have seen how to leverage central differences to create differential operators with and without periodic boundary conditions for 1D and 2D problems. In this problem you will simply generate these operators, to make sure you understand the basic building blocks. To make this a useful exercise for later problems, make sure to use the `spdiags` command when you generate the matrices, so that you can generate sparse operators of any dimension.

For grading, you cannot write out sparse matrices to ascii files, so to submit your answers please apply the `A=full(A)` in MATLAB to turn your sparse operators in to full matrices for grading.

(a) In one dimension, using span $[-1, 1]$ and mesh wiith $n = 8$, generate D_x and D_{xx} operators (first and second derivative), with and without periodic boundary conditions. This will give you four matrices:

- A_1: D_x without periodic boundary
- A_2: D_x with periodic boundary
- A_3: D_{xx} without periodic boundary
- A_4: D_{xx} with periodic boundary

Store A_i in Ai.dat. Remember you have to convert to full matrices before you write them out into ascii files. Using `imagesc` may be helpful as you build and check your operators. Try to think of simple unit tests to build to check your work.

(b) In two dimensions, using span $[-1, 1] \times [-1, 1]$ and mesh with $n = 8$ in each dimension, generate D_x, D_y, D_{xx}, D_{yy}, and Laplacian, with and without periodic boundary conditions. This will give you ten matrices:

- A_5: D_x without periodic boundary
- A_6: D_x with periodic boundary
- A_7: D_y without periodic boundary
- A_8: D_y with periodic boundary
- A_9: D_{xx} without periodic boundary
- A_{10}: D_{xx} with periodic boundary
- A_{11}: D_{yy} without periodic boundary
- A_{12}: D_{yy} with periodic boundary
- A_{13}: Laplacian without periodic boundary
- A_{14}: Laplacian with periodic boundary

Store A_i in Ai.dat. Remember you have to convert to full matrices before you write them out into ascii files.

Problem 2. In this problem you will implement two common basic iterative methods, the Jacobi Iteration and Gauss-Seidel, and try them out on difficult matrix. Remember that both methods arise out of the iteration

$$x_{k+1} = x_k - M^{-1}(b - Ax_k)$$

for different preconditioners M.

For each method, you will use it to solve a linear system $Ax = b$ where A is given by the (asymmetric) tridiagonal matrix

$$A = \begin{bmatrix} 1 & 0.16 & 0 & \dots & 0 \\ -1.16 & 1 & 0.16 & \ddots & \vdots \\ 0 & -1.16 & 1 & \ddots & 0 \\ \vdots & & \ddots & \ddots & 0.16 \\ 0 & \dots & & -1.16 & 1 \end{bmatrix}$$

and b is the vector of all ones, $b = [1, \dots, 1]^T$.

(a) Implement the Jacobi method, where M is the diagonal of A. Use the method to solve $Ax = b$ for dimensions $n = 10$ and $n = 30$.

Start with the vector x_1 of all zeros.

In each case, exit on the condition

$$\|b - Ax_k\| < 1 \times 10^{-8}.$$

Save the vector of errors in $A15$.dat for $n = 10$ and $A16.dat$ for $n = 30$. Before you submit, check that you implemented the errors correctly, i.e. that the first error in your vector is the norm of b, and the last error is less than 10^{-8}

(b) Implement the Gauss-Seidel method, where M is the full lower triangle of A. Use the method to solve $Ax = b$ for dimensions $n = 10$ and $n = 30$.

In each case, exit on the condition

$$\|b - Ax_k\| < 1 \times 10^{-8}.$$

Save the vector of errors in $A17$.dat for $n = 10$ and $A18$.dat for $n = 30$. Before you submit, check that you implemented the errors correctly, i.e. that the first error in your vector is the norm of b, and the last error is less than 10^{-8}.

(c) Make a plot of the errors $\|Ax_k - b\|_2$ as a function of iteration, so you can compare the performance of Jacobi and Gauss-Seidel for both problem sizes. You do not need to turn in the plot.