Computational Project 2

Due Friday Feb. 12th at noon.

For this project you will write a Matlab program that solves $\mathbf{Ax} = \mathbf{b}$ using the Jacobi iterative method for a few different linear systems that are defined in part 5. Below are the required steps to take towards building this program. I suggest first reading the entire assignment all the way through and then drawing a "code map" that helps you think about how all the different functions are interacting. Important formatting requirements are provided at the end.

1. Write a function that generates an $n \times n$ matrix $\mathbf{A}$ and an $n \times 1$ vector $\mathbf{b}$. The function inputs should be n and m, and the outputs should be $\mathbf{A}$ and $\mathbf{b}$, where the latter are calculated within the function as

   $\mathbf{A} = \text{magic}(n) + \text{eye}(n) * m;$

   (magic generates a matrix—a rather special one. What relevant impact will the addition of m*eye(n) have? Recall eye(n) generates an $n \times n$ identity matrix.)

   $\mathbf{b} = [1/1, 1/2, 1/3, \ldots, 1/n]^{\mathrm{T}};$

   where T indicates the transpose of this vector.

2. Write a function to evaluate the condition number and diagonal dominance of the matrix $\mathbf{A}$. It should take as input the matrix $\mathbf{A}$ and output a status variable. The status variable being equal to 0 means that $\mathbf{A}$ is well conditioned and diagonally dominant. If this is not the case, the function should return a non-zero value and display the reason (ill-conditioned or not diagonally dominant). Within the function, use cond($\mathbf{A}$) to check that the condition number isn't too large. Use equation 4.52 from the textbook (also in the notes) to determine if all rows are diagonally dominant.

3.Write a function to solve **Ax** = **b** using the Jacobi iterative method. The function should take in the following as inputs:

- a matrix A of size $n \times n$
- a vector b of size $n \times 1$
- a variable k_{max} that specifies the maximum number of iterations.

The outputs of the function should be the solution vector **x**. The function should iterate until either k_{max} is reached or the total estimated error summed across all elements is less than a relative error tolerance ε_t = 1E-10 (whichever is first), where relative error is calculated as

$$e_t = \sum_{i=1}^{n} \left| \frac{x_i^k - x_i^{k-1}}{x_i^k} \right|$$

Use the fprintf command to print the error at each iteration to the command window. The function should only iterate until a converged solution has been found. This happens when the relative error e_t is less than ε_t. If this happens before k_{max} iterations are performed, no more iterations should be performed. The function should also display a warning message that the system did not converge if a solution is not found within k_{max} iterations. In this case, return an answer of x = [–999].

4. Write a main function with the following features:

- Inputs are n, m, and k_{max}.
- Output is **x**.
- Call the function from (1) to build **A** and **b**.
- Call the function from (2) to check **A**.

- If the system is well conditioned and solvable iteratively, call the function from part (3) to find **x**. Otherwise, display an error message and set **x** = –999.

5. Write a function (called cp2_MEID) that calls the function from part (4) with the following values:

(a) n=5, m=1E+2, k_{max} =10

(b) n=5, m=1E+2, k_{max} =50

(c) n=5, m=1E+6, k_{max} =50

(d) n=20, m=1, k_{max} =50

(e) n=20, m=0, k_{max} =50

Formatting: Your main function should be called **cp2_MEID** and have **no inputs or outputs** (just hard code input parameters n, m, and k_{max} at the top of cp2_MEID). Publish as a pdf and print this function to turn in a hard copy in class.

Also print out your **final outputs** (suppress the outputs for each iteration where needed) for each case in part 5 and a **brief write-up** explaining any cases where the iterative approach failed. Submit your Matlab file to the dropbox on D2L. Please label parts of your code with the corresponding section number above and make sure to comment throughout. Remember to **submit only one m file to D2L** that contains your main function and all other local ones. Please **do not compress** your file.