ENME 631 Numerical Methods

Name: Henry Stoldt
ID: 10127324
Date: Thursday Oct. 10th, 2019
Language: Julia

The file "Assignment 2.jl" runs all of code requested in Q1-3. Explanation and comments below.

Question 1:

The Solvers:

I've coded the Gauss-Elimination and Gauss-Seidel/SOR methods in the file "LinearSolvers.jl". The Gauss-Seidel method simply calls the SOR method with an omega value of 1. The Gauss Elimination solver includes scaled partial pivoting to minimize round-off error and avoid zeros on the diagonals.
To run the unit tests I've set up for the linear solvers (including examples from the textbook and the assignment questions) run the "test_LinearSolvers.jl" file.
To try the solvers on an arbitrary matrix you can run "interactive_LinearSolvers.jl".

Solving the sample matrix:

The Gauss Elimination method works fine for this matrix, but the iterative solvers diverge because the matrix is not diagonally dominant. I could not see any obvious way to make the matrix diagonally-dominant, so I assume this is the expected result.

Question 2:

Question interpretation:

Several things were odd about this question:

1. According to wikipedia: https://en.wikipedia.org/wiki/Pentadiagonal_matrix and the textbook (pg. 456-457) this matrix is not actually pentadiagonal, but 9-diagonal, since the nonzero elements furthest from the diagonal are at positions diag+4 and diag-4.
2. This gives a bandwidth of 9, which doesn't match the "b" dimension in the photo.
3. How can a matrix be both pentadiagonal (bandwith = 5) and have a variable bandwidth/band "b"?
4. If, instead of trying to solve a pentadiagonal matrix, we are trying to solve arbitrary banded matrices, it seems like Stone's strongly implicit method is the preferred approach: https://en.wikipedia.org/wiki/Stone_method. This was also pointed out in class, but it was stated that we won't be convering it in this course.

I may be missing something (or several things) in my interpretation of the question, but based on my current understanding of the question there is much room for interpretation. I've chosen to interpret the question as: "Develop an algorithm for a b-bandwidth diagonal matrix". This lends itself to an extension of the Thomas algorithm which we did cover in class and which is the suggested method for solving pentadiagonal systems in the textbook (pg. 467).

The Solver:

The n-diagonal solver I developed is called "Solve_Diag!" and is in the file "LinearSolvers.jl". As in Q1, test suite is "test_LinearSolvers.jl" and the solver can be used interactively by running "interactive_LinearSolvers.jl" and selecting solver 4. Matrices need to be passed in as an nRows x bandwidth matrix containing only the diagonal entries.

Solving the sample matrix:

It is easy to see that while this solver would work well for a tri- or pentadiagonal matrix of this size, it is not good for the example matrix provided in Q2, because that one is a 9-diagonal matrix, which is actually larger in diagonal form than in its regular form (6 rows x 9 diagonals vs. 6 rows x 6 entries).

Question 3:

The Solver:

The Newton-Raphson solver I've developed is in "NonLinearSolvers.jl". The arguments for the solver are an array of functions and an array of initial x-guesses. Each function must accept all x-values as inputs and return the residual of the equation it's representing. The matrix of partial derivative is evaluated numerically using the second order central method by the function "ddx". I've hardcoded the functions f1, f2, and f3 to represent those given in the assignment.

(Not) Solving the sample equations:

Unfortunately that system of equations does not seem to have a non-imaginary solution. This can be shown as follows:
Using eqn 3: x3 = sin(x1) -> sub into eqn 2
Eqn 2: x2 - (sin(x1))^2 - 1 = 0
Eqn 2: x2 = (sin(x1))^2 + 1 -> sub into eqn 1
Eqn 1: exp(2*x1) - (sin(x1))^2 + 3 = 0
By inspection, the LHS of this function has a minimum value of 2 when x1 is very negative. Therefore the system has no real-valued solution.
This causes the Newton-Raphson method to Test cases for the solver are in "test_nonLinearSolver.jl" and it can be used interactively by running "interactive_NonLinearSolvers.jl".