EP 501 Homework 1: Numerical Linear Algebra

September 19, 2019

Instructions:

- Complete all problems.
- Submit all source code and publish Matlab results via Canvas.
- Discussing the assignment with others is fine, but you must not copy anyone's code.
- I must be able to run your code and produce all results by executing a single top-level Matlab script, e.g. assignment1.m or similar.
- You may use any of the example codes from our course repository: `https://github.com/mattzett/EP501/`.
- Do not copy verbatim any other codes (i.e. any source codes other than from our course repository). You may use other examples as a reference but you must write you own programs (except for those I give you).
- For demonstrating that your code is correct when you turn in the assignment, you must use the test problem in the course repository found in `linear_algebra/testproblem.mat` (elimination methods) and `linear_algebra/iterative_testproblem.mat` (iterative methods requiring diagonal dominance).

Purpose of this assignment:

- Demonstrate competency with existing numerical linear algebra tools in Matlab
- Refresh basic Matlab and coding skills
- Learn principles behind numerical linear algebraic techniques like elimination, iteration, and LU factorization.
- Develop good coding and documentation practices, such that your programs are easily understood by others.
- Hone skills of developing, debugging, and testing your own software
- Learn how to build programs on top of existing codes

1. Develop some basic elimination and substitution tools for linear equations:

   (a) Write a Matlab function that uses simple forward elimination (without pivoting or scaling). Do not use any vectorized Matlab operations - write out all loops explicitly. You will need to rewrite the version in the repository to get rid of such operations.

   (b) Create a test problem and solve the problem using your function and the back-substitution function in the course repository. Check that your method gives the same solution as the Matlab built-in solutions and the same solution as with the Gaussian elimination function from the repository. demonstrate that your solution gives the same results

   (c) Write and a forward substitution function for a lower-triangular system.

   (d) Create a test problem and verify that your function gives the same results as the Matlab built-in solution routines (e.g. `A\b` or `inv(A)*b`).

2. Elimination methods for computing matrix inverses:

   (a) Alter (i.e. create a new version of) your simple elimination function to work for multiple right-hand sides (RHS).

   (b) Create a function that implements Gauss-Jordan elimination; you should start from your forward elimination function for multiple RHS and add the backward eliminations needed.

   (c) Find the inverse of a test matrix using your Gauss-Jordan elimination function

   (d) Compare your results with Matlab built-in inverse operation(s) and show that they agree.

3. LU factorization and its application to solve linear systems

   (a) Alter (i.e. create a new version of) your simple forward elimination function that performs Doolittle LU factorization. Please read the book section 1.4, which explains how this can be implemented in an efficient way (the book also gives an example fortran code you can look over).

   (b) Using just the output of the factorization and a back-substitution function (provided in the repository), solve a test linear system of equations.

   (c) Use your LU factorization on the same test system to set up a solution for the this system with a different RHS using only your forward- and back-substitution functions.

   (e) Use your LU factorization function to find a matrix inverse for your test problem.

4. Iterative methods for solving linear systems:

   (a) Using the Jacobi function from the repository, create a new function that implements successive over-relaxation.

   (b) Try this solver on a test problem and show that it gives the same results as the built-in Matlab utilities. Your test problem must be diagonally dominant (cf. Section 1.2.1 of the textbook) or else iterative approaches will not work.

5. Numerical approaches for computing determinants:

   (a) Create a new version of the Gaussian elimination function (from the course repo) that also outputs the determinant of the system (cf. section 1.3.6 of the textbook).

   (b) Demonstrate that your software gives the same results as the Matlab built-in `det()`