EP 501 Homework 1: Numerical Linear Algebra, part I

September 10, 2020

Instructions:

- Complete all listed steps to the problems.
- Submit all source code, which must be either MATLAB or Python, and output (results printed to the screen or plotted) via Canvas.
- Results must be compiled into a single .pdf file which contains descriptions of the calculations that you have done alongside the results.
- Source codes must run and produce the same essential output presented in your documents.
- Discussing the assignment with others is fine, but you must not copy the code of another student *verbatim*; this is considered an academic integrity violation.
- During submission on the canvas website compress all of your files for a given assignment into a single .zip file, e.g. `assignment1.zip` . I should be able to run your codes by unzipping the files and then opening them and running them in the appropriate developer environment (MATLAB or Spyder).
- You may use, *verbatim* or modified, any of the example codes from one of the course repositories contained on the GitHub organization website `https://github.com/Zettergren-Courses`.
- For demonstrating that your code is correct when you turn in the assignment, you must use the test problems in the course repository found in `linear_algebra/testproblem.mat` (elimination method tests, including multiple right-hand sides), `linear_algebra/lowertriang_testproblem.mat` (lower triangular tests). To load these data into your workspace use:

```
load iterative_testproblem.mat
```

  or double click on the .mat file in the Matlab file browser. To the load these files in Python see the example included in the Python basics section of the course repository (i.e. ./basic_python/load_matlab_file.py).

Purpose:

- Demonstrate competency with existing numerical linear algebra tools in MATLAB and Python
- Refresh basic MATLAB or Python coding skills
- Learn principles behind numerical linear algebraic techniques, particularly elimination-based methods.

- Develop good coding and documentation practices, so that your programs are easily run and understood by others.
- Hone skills of developing, debugging, and testing your own software
- Learn how to build more complicated programs on top of existing, simple codes that have been provided to you.

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

   (a) Write a MATLAB or Python function that uses simple forward elimination (without pivoting or scaling, viz. not Gaussian elimination). Note that this has already been implemented in script form in class; you simply need to convert it into a function so it may be easily used with later programs to be developed as part of future assignments.

   (b) Solve the test problem from the repository (viz. `./testproblem.mat`) using your function and the back-substitution function in the course repository; use the right-hand side given in the `b` array (`b2,b3` are for later use). Demonstrate that your method gives the same solution as the MATLAB or Python built-in solutions and the same solution as with the Gaussian elimination function from the repository - see examples in course repositories for how to use the built-in solvers for MATLAB and Python.

   (c) Write a forward substitution function for a lower-triangular system and run your code using the lower triangular test problem provided in this repo in `./lowertriang_testproblem.mat`.

   (d) Verify that your function gives the same results as the Matlab or Python built-in solution routines for the *lower triangular* test problem.

2. Elimination methods for computing matrix inverses:

   (a) Create a new version of your simple elimination function which will 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 the test problem matrix using your Gauss-Jordan elimination function

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

3. 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 or Python built-in determinant functions - use the test problem included with this assignment in `./testproblem.mat`.