School of Mathematics and Statistics
MAST30028 Numerical Methods & Scientific Computing
2020

Assignment 2: Root-finding, linear systems and least squares fitting.
Due: 17:00 October 15.

This assignment is worth 20% of the total assessment in MAST30028. When you submit the assignment, you should submit two files: one pdf file and one zip file. The pdf file contains the answer you write, the numerical results you generated including figures or tables(0 mark if numerical results are not included in the pdf file), the comment or the explanation of the results, and the printout of your code (you may use Matlab `publish` or Matlab live scripts). The zip file should only contains all the m-files.

1 Root-finding [10 marks]

a. Use Newton's method to find roots of the following functions with the given initial guesses x_0 and the absolute and relative tolerance being 10^{-12}. In each case report and explain the results.

   (i)
   $$f(x) = \ln(x)\exp(-x), \qquad x_0 = 2$$

   (ii)
   $$f(x) = x^3 - x - 3, \qquad x_0 = 0$$

   (iii)
   $$f(x) = 1 - (1 + 3x)\exp(-3x), \qquad x_0 = 1$$

   In the case of (iii), investigate the *order of convergence*.

b. MATLAB has only one built-in function for finding roots : `fzero`. (The Optimization Toolbox has more). To find out about it, start up Matlab and type

   ```
   help fzero
   ```

   Try out `fzero` on the examples above. Interrogate the structure `output` (given by `fzero`) to find out what happened. Use the same tolerance as in the Newton's method.

2 Operation counts [7 marks]

Derive the operation count (multiplications and divisions only) for

a. Choleski factorization (without pivoting)

   The detailed algorithm for Choleski factorization is that implemented in `CholScalar`.

b. Gauss elimination (equivalent to LU factorization) without pivoting of an Upper Hessenberg matrix.

   An *Upper Hessenberg matrix* is one with zeroes below the subdiagonal. Just modify the derivation for Gauss elimination in lectures.

c. solution of a tridiagonal system by LU factorization without pivoting (also called the *Thomas algorithm*) This is the algorithm implemented in the code `tridisolve` in the `Asst2` folder.

3 Sparse matrices. [7 marks]

MATLAB has support for *sparse matrices*. *Banded matrices* are an important class of sparse matrices. Here you explore the sparse matrix facilities by concentrating on the tridiagonal matrix from Moler Exercise 2.19 but with n arbitrary instead of 100. Let's call it A. It has a diagonal full of 2s and a sub- and superdiagonal full of -1s.

Form the matrix A in 2 ways:

a. using 3 calls to `diag` to form A_1

b. using `spdiags` instead, to form A_2

For each of these forms, use `tic,toc` to benchmark, for $n = 200 : 200 : 2000$ the time to solve $\mathsf{Ax} = \mathsf{b}$ for some `b` using `\`. I suggest you time a loop of, say, 100 solves in order to get sensible results. If you have a fast machine, you may need to increase n.

Also try the tridiagonal solver `tridisolve` from the `Asst2` folder to solve the same problem. It will be slower because it's an M-file, not a built-in function.

How does the time taken by each of these 3 solvers scale with n? On the basis of Q2, what do you think Matlab's \ is doing? By using `whos`, describe another benefit of using `sparse`.

4 Norm equivalence [6 marks]

Prove the following inequalities, for arbitrary m, n. Here $\mathbf{x}$ is an m-vector and $\mathbf{A}$ is an $m \times n$ matrix.

$$
\begin{aligned}
||\mathbf{x}||_2 &\leq ||\mathbf{x}||_1 \\
||\mathbf{x}||_1 &\leq \sqrt{m}||\mathbf{x}||_2 \\
||\mathbf{A}||_2 &\leq \sqrt{n}||\mathbf{A}||_1 \\
||\mathbf{A}||_1 &\leq \sqrt{m}||\mathbf{A}||_2
\end{aligned}
$$

In the first two cases, give an example of a vector $\mathbf{x}$ where equality holds.

5 Conditioning [4 marks]

Let X be the $n \times n$ matrix defined by

```
[I,J] = ndgrid(1:n);
X = min(I,J) + 2*eye(n,n) - 2;
```


Which, if any, of the triangular factorizations `chol(X)`, `lu(X)`, and `qr(X)` reveal the poor conditioning of X when $n = 20$? Explain, using results from the lectures.

6 Are all fits equally good? [9 marks]

Examine the M-file `polyfit.m` (`edit polyfit`) and understand what lines 59–67 do.
Do Moler Exercise 5.7 but use the the values $t_k = (k-1)/5, y_k = \mathrm{erf}(t_k), k = 1 : 11$.
For part c, plot the errors of the three fitted functions with 5 parameters (i.e. a degree 4 polynomial, a degree 9 odd polynomial and the function in part c) on a loglinear scale. Which form fits the data best?

Hint: You may want to write your own M-file to help generate the least squares fits for parts b and c using `polyfit/polyval` as a template (although it's not necessary to do so).

7 Nonlinear transformations to get linear fits [7 marks]

The standard equation describing simple enzyme kinetics is

$$v_0 = \frac{V_m}{1 + K_m/s} \tag{1}$$

where v_0 is the initial reaction rate, V_m is the maximum reaction rate, s is the substrate concentration, and K_m is the Michaelis constant. In a typical experiment, v_0 is measured as s is varied and then V_m and K_m are determined from the resulting data.

To avoid a nonlinear fit, a number of researchers have re-arranged Equation (1) into a linear model in new variables. Here are two examples:

a.
$$\frac{1}{v_0} = \frac{1}{V_m} + \frac{K_m}{V_m}\frac{1}{s} \tag{2}$$

b.
$$v_0 = V_m - K_m \frac{v_0}{s} \tag{3}$$

For each of these linear models (in new variables) determine the best fit to the data

s	2.5	5.0	10.0	15.0	20
v_0	0.024	0.036	0.053	0.060	0.064

Compare their results for V_m and K_m with those obtained from a nonlinear fit:
$V_m = 0.08586, K_m = 6.562$.
On the same figure, plot the data and all 3 model fits to the data.