MACM 316 – Computing Assignment #4

Due Date: Friday November 6 at 11:00pm

Submission Instructions: You must upload one .pdf file to **Crowdmark** that consists of 2 pages ONLY: page 1 is your report which should fit all of your results, discussion, data and figures into a single page; and page 2 is a listing of your code. The deadline is **11:00pm** on the due date. The actual due time is set to 11:05pm – if Crowdmark indicates that you submitted late then you will be assigned a grade of 0 on this assignment. Your TA has emailed you a Crowdmark link that you will use for the entire semester to upload your assignment solutions.

- Review the **Guidelines for Computing Assignments** carefully.
- Acknowledge any collaborations or assistance from colleagues/TAs/instructor.
- If you have questions about this assignment or Matlab programming, you can obtain help by posting your questions to the "Computing Assignment" discussion board in Canvas. This discussion board will be checked regularly, and also monitored continuously during computational workshop hours.

Computing Assignment – Ill-conditioned linear systems

The $n \times n$ Hilbert matrix $H^{(n)}$ is a matrix with entries defined by

$$h_{ij} = \int_0^1 x^{i+j-2} dx = \frac{1}{i+j-1} \quad \text{for } 1 \le i, j \le n.$$

As I discussed in class, the Hilbert matrix is a "classic" example of an ill-conditioned matrix. The Matlab command `hilb(n)` generates such a Hilbert matrix with the dimension `n` passed as an input parameter[†]. It is easy to show that $H^{(n)}$ is invertible for any n, and so the linear system $H^{(n)}x = b$ is always guaranteed to be solvable for unknowns $x = (x_1, x_2, \ldots, x_n)^T$. However, the solution is not easy to compute accurately in floating point arithmetic. This is what you will investigate in this computing assignment.

Your report must include the following:

(a) First, investigate how the norm and condition number of the Hilbert matrix grow with n. For values of $n = 2, 3, 4, \ldots, 30$:

- set up the Hilbert matrix $H^{(n)}$ using `hilb`
- compute the 1-norm and 2-norm of $H^{(n)}$ using `norm`
- compute the condition number of $H^{(n)}$ in the same two norms using `cond`

On the same set of axes plot versus n your four results, $\|H^{(n)}\|_{1,2}$ and $\text{cond}_{1,2}(H^{(n)})$. Make sure to use an appropriate axis scaling[‡].

[†]Use `type hilb` to see how Matlab sets up the Hilbert matrix in just 2 lines of code. This is a great example of how Matlab's "vectorized" operations can be used to write clean and simple code that is free of loops.

[‡]In other words, think about which plotting function to use: `plot`, `semilogx`, `semilogy` or `loglog`.

(b) It's possible to show theoretically that the condition number of the Hilbert matrix grows as $O\left(\frac{(1+\sqrt{2})^{4n}}{\sqrt{n}}\right)$ as $n \to \infty$[§] (don't try to show this, just use it). Add this function to your plot from part (a) and discuss whether or not this rate of growth seems consistent with your results. Can you now explain why your choice of axis scaling (and plotting function) from part (a) is the right one?

(c) Next, you'll investigate the performance of three different methods for solving the linear system $H^{(n)}x = b$, again using values of $n \in [2, 30]$:

- Generate the $n \times n$ Hilbert matrix $H^{(n)}$, initialize the exact solution $x = (1, 1, \ldots, 1)^T$, and compute the corresponding right hand side vector $b = H^{(n)}x$.
- Compute floating-point approximations $\hat{x}$ to the linear system $H^{(n)}\hat{x} = b$ using three methods: the PLU decomposition (followed by forward/backward substitution), Matlab's backslash command, and the Jacobi iteration. For Jacobi, choose an initial guess $x^{(0)} =$ `rand(n,1)`, which is an n-vector containing random real numbers chosen from the interval $[0, 1]$.
- Compute the absolute solution error $E_x = \|x - \hat{x}\|_2$ in the 2-norm for each of your three approximations.
- Plot your three errors versus the problem dimension $n \in [2, 30]$.

What do you observe? Compare and contrast the performance and cost of methods.

(d) Finally, recall that in lectures the condition number of a square matrix A is defined in terms of the matrix norm as $\text{cond}_p(A) = \|A\|_p \cdot \|A^{-1}\|_p$ (for $p = 1, 2, \infty$). Investigate how the inverse Hilbert matrix $(H^{(n)})^{-1}$ behaves for large n by doing the following.

Use Matlab's `inv` function to determine the inverse of $H^{(n)}$ for the same $n \in [2, 30]$, and then compute the matrix product $M = H^{(n)} \cdot (H^{(n)})^{-1}$. Of course, this M should be equal to the $n \times n$ identity matrix I, but is it? To measure how close/far M actually is from I, determine the maximum entry of M (in absolute value) and plot this number versus the matrix size n. Discuss your results.

Partial code for this assignment can be obtained by downloading the file `macm316ca4SampleCode.m` from Canvas. You may also use the code `jacobi2.m` discussed in class to perform your Jacobi iterations.

Final note: If you're intrigued by the Hilbert matrix, you can learn more about it by reading the recent blog post by Nick Higham at

`http://nhigham.com/2020/06/30/what-is-the-hilbert-matrix`

It's interesting how he argues that "the Hilbert matrix is not a good test matrix, for several reasons." Read the post to find out the reasons why ...

[§]See `http://en.wikipedia.org/wiki/Hilbert_matrix`