Homework 3

Due: Friday 11:59pm, Oct 18, 2024, on Gradescope

Notes: Please read the syllabus for information on programming expectations, the late homework policy, and the collaboration policy. Please submit both the written and coding portions of your homework on Gradescope. The written portion should be submitted as a PDF (prepared however you like, but LaTeX is excellent), with plots and tables in the appropriate places. Results and plots requested in the problems should be placed in your write-up—you should not simply refer to your code. Please mark the location of each problem in your Gradescope submission—this makes it much easier for us to grade.

Plotting and formatting: Put thought into presenting your data and results in the clearest way possible. Think about what information you want to convey, and how to convey it most effectively. Sometimes, using a table can be useful, but at other times data is best presented using a plot. All tables and plots should be accompanied by some discussion of what we are intended to learn from them.

Choose proper ranges and scales (e.g., semilogx, semilogy, loglog), always label axes, and give meaningful titles. Make sure your axis numbering and labels are large enough to be readable—getting this right will require some exploration in MATLAB, or whatever language you are using. If you do print numbers, in MATLAB for example, use `fprintf` to format the output nicely. Use `format compact` and other format commands to control how MATLAB prints things. When you create figures using MATLAB (or Python or Julia), please try to export them in a vector graphics format (.eps, .pdf, .dxf) rather than raster graphics or bitmaps (.jpg, .png, .gif, .tif). Vector graphics-based plots avoid pixelation and thus look much cleaner.

Code: Try to write clean, concise, and easy-to-read code, with meaningful variable names. Your code must be well-commented. Every line of code should be explained by comments inside the code, unless it is absolutely self-explanatory. Commenting your code is like "showing your work", and grading will take this into account in a similar manner.

You should submit two code files for this assignment: `gausselim.m` and `prob3.m`.

---

1. (a) **[10 pts]** Write a function `gausselim` (in a file `gausselim.m`) implementing Gaussian elimination with partial pivoting, followed by backward substitution, to solve a linear system $Ax = b$. The function should take as input an $n \times n$ matrix A and a vector b, and return the solution x. You can use the provided function `gausselim_nopivot` as a starting point. However, note that unlike that function, which does not perform the backward substitution step itself and therefore returns an upper-triangular matrix and a modified right hand side, yours should perform backward substitution (you can use the `backsub` function you wrote in the previous homework for this) and return the solution x. Also remember to handle the special case in which all possible pivots in a given column are zero: in this case,

the matrix is singular, and your function should terminate and return a message indicating this.

(b) **[10 pts]** Refer to Problems 4d and 4e from Homework 2. Use your new pivoted Gaussian elimination code to solve the linear systems described there, and measure the errors. Compare these errors with those you obtained in Homework 2, and discuss your results. Note: It is not important that there, you did LU factorization and then forward/backward substitution and here, you are doing Gaussian elimination and backward substitution directly, since Gaussian elimination is equivalent to LU factorization. The important difference here is that you are using partial pivoting.

(c) **[10 pts]** Consider the $n \times n$ Hilbert matrix H_n, with entries given by

$$H_{ij} = \frac{1}{i+j-1}, \quad i,j = 1,2,\ldots,n.$$

This is a famous example of a family of matrices with rapidly-growing condition numbers. Write a script which loops through $n = 2,\ldots,15$, forms the Hilbert matrix H_n, and (i) computes the condition number $\kappa(H_n)$ of H_n using MATLAB's `cond` function (see the documentation for more information), and (ii) solves the linear system $H_n x = b$ for $b_i = \sum_{j=1}^{n} \frac{1}{i+j-1}$ using your `gausselim` function. The solution in this case is $x = (1,1,\ldots,1)^T$. Compute the relative error in the solution for each n in the Euclidean norm $||\cdot||$ (use the `norm` function in MATLAB), and plot it as a function of n (use `semilogy`). On the same plot, also plot the upper bound on the relative error in terms of the condition number given by the estimate $\kappa(H_n)\varepsilon$, where ε is the machine precision (use MATLAB's `eps` to obtain this). This is obtained from the bound derived in class by noting that in floating point arithmetic, the relative error $||b-\hat{b}||/||b||$ in the right hand side should be close to ε. Finally, on the same plot, plot the relative residual error $||H_n x - b||/||b||$. Compare the relative error with the condition number bound and explain your results. Compare the relative error with the relative residual and explain what you find.

(d) **[10 pts]** Gaussian elimination is backward stable in almost all cases, and is therefore used a workhorse of computational science across fields. However, there are rare cases in which backwards stability fails. Consider the example due to Wilkinson: a matrix with entries -1 on the lower triangle, 1 on the diagonal and last column, and 0 elsewhere:

$$A = \begin{pmatrix} 1 & 0 & 0 & \cdots & 1 \\ -1 & 1 & 0 & \cdots & 1 \\ -1 & -1 & 1 & \cdots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & -1 & \cdots & 1 \end{pmatrix}.$$

You can generate the $n \times n$ version of this matrix in MATLAB using the following concise command:

```
A = tril(-ones(n)) + 2*eye(n); A(:,end) = 1;
```

(To see the suprising result below, you might need to add a small perturbation to the last column of the matrix in order to make sure some rounding errors are made during Gaussian elimination, rather than doing exact arithmetic with integers. You can do this with the following command: `A(:,end) = A(:,end) + 1e-15*randn(n,1);`) Take $n = 100$, and solve a linear system $Ax = b$ using your `gausselim` function. Generate b by taking a random vector x, and computing $b = Ax$; this way, you know the true solution x (`xtrue = randn(n,1); b= A*xtrue;`). Compute the condition number of A. Approximately what relative error do you expect in the solution based on this? Now compute the relative error in the solution: what do you find? Finally, compute the relative residual $||Ax - b||/||b||$: what do you find? Based on this evidence, discuss whether or not Gaussian elimination is backward stable for this example. Why do these results not violate the relative error bound on the solution based on the condition number which we derived in class?

(e) **[10 pts]** Provide evidence that your Gaussian elimination code has computational complexity $\mathcal{O}(n^3)$, in the form of a plot showing wall clock timings vs. n on a log-log scale, for values of n ranging from 2^5 to 2^{10} (this might take a bit of time to run). The linear systems themselves are unimportant, since the same operations are performed regardless of the entries of A and b—you can use random matrices A and random vectors b. To demonstrate $\mathcal{O}(n^3)$ scaling, plot a curve $y = Cn^3$, for some constant C (which you can adjust to make the plot look reasonable), on the same plot, and compare slopes. You can use the script `time_matmat.m` as an example starting point.

2. In this problem, we will derive a bound on the relative error of the solution of a linear system $Ax = b$ given a perturbation both in the right hand side $b \in \mathbb{R}^n$, and in the matrix $A \in \mathbb{R}^{n\times n}$, assumed to be non-singular. Let $\hat{b}$ be a perturbation of b, and $A + \delta$ a perturbation of A, with $\delta \in \mathbb{R}^{n\times n}$. Suppose $Ax = b$, and $(A + \delta)\hat{x} = \hat{b}$. Let $\|\cdot\|$ be any norm on $\mathbb{R}^n$.

(a) **[10 pts]** Show that the relative error in the solution x is bounded by

$$\frac{\|x - \hat{x}\|}{\|x\|} \leq \left\|(A + \delta)^{-1}\right\| \|A\| \left(\frac{\left\|b - \hat{b}\right\|}{\|b\|} + \frac{\|\delta\|}{\|A\|}\right),$$

where $\|A\|$ is the matrix norm of A induced by $\|\cdot\|$. Hint: Show that $(A + \delta)x = b + \delta x$, and take the difference between this and $(A + \delta)\hat{x} = \hat{b}$. Solve for $x - \hat{x}$. Then bound the norm of the result. Finally, divide both sides by $\|x\|$, use the trick $\|x\| = \|x\| \, \|b\| \, / \, \|b\|$, use that $b = Ax$, and that $\|Ax\| \, / \, \|x\| \leq \|A\|$.

(b) **[5 pts]** If a matrix $M \in \mathbb{R}^{n\times n}$ has norm less than 1, $\|M\| < 1$, then the following *Neumann series expansion* holds:

$$(I + M)^{-1} = I - M + M^2 - M^3 + \cdots = \sum_{k=0}^{\infty} (-1)^k M^k.$$

This is analogous to the geometric series expansion $(1+x)^{-1} = \sum_{k=0}^{\infty}(-1)^k x^k$ for $x \in \mathbb{R}$, $|x| < 1$. Here M^k simply represents a matrix-matrix product of M with itself k times, and we define $M^0 = I$, the identity matrix. Assume $\|\delta\|$ is sufficiently small so that $\|A^{-1}\delta\| \leq \|A^{-1}\| \, \|\delta\| < 1$. Using $(A+\delta)^{-1} = (A(I + A^{-1}\delta))^{-1} = (I + A^{-1}\delta)^{-1}A^{-1}$, show that

$$(A+\delta)^{-1} = A^{-1} + \left(\sum_{k=1}^{\infty} (-1)^k (A^{-1}\delta)^k \right) A^{-1}.$$

Conclude that

$$\left\| (A+\delta)^{-1} \right\| \leq \left\| A^{-1} \right\| \left(1 + \sum_{k=1}^{\infty} \left\| A^{-1}\delta \right\|^k \right).$$

(c) **[5 pts]** Using the geometric series sum $x/(1-x) = \sum_{k=1}^{\infty} x^k$ for $x \in \mathbb{R}$, $|x| < 1$, show that

$$\left\| (A+\delta)^{-1} \right\| \leq \left\| A^{-1} \right\| \left(1 + \frac{\|A^{-1}\delta\|}{1 - \|A^{-1}\delta\|} \right) \leq \left\| A^{-1} \right\| \left(1 + \frac{\|A^{-1}\| \, \|\delta\|}{1 - \|A^{-1}\| \, \|\delta\|} \right).$$

(d) **[5 pts]** Using algebraic manipulations, show that the right hand side of the bound above can be written

$$\frac{\|A^{-1}\|}{1 - \kappa(A) \, \|\delta\| \, / \, \|A\|},$$

where $\kappa(A) = \|A\| \, \|A^{-1}\|$ is the condition number of A (Hint: use multiplication by $1 = \|A\| \, / \, \|A\|$ to help you). Using the bound derived in part (a), conclude that

$$\frac{\|x - \hat{x}\|}{\|x\|} \leq \frac{\kappa(A)}{1 - \kappa(A) \, \|\delta\| \, / \, \|A\|} \left(\frac{\left\| b - \hat{b} \right\|}{\|b\|} + \frac{\|\delta\|}{\|A\|} \right).$$

This is a bound on the relative error in the solution x given perturbations of A and b in terms of the condition number $\kappa(A)$.

3. Consider the function $f(x) = \cos^3(\sin(x))$.

(a) **[5 pts]** Show that f is even and 2π-periodic.

(b) **[20 pts]** A smooth, even, 2π-periodic function can be expanded as a cosine series,

$$f(x) = \sum_{k=0}^{\infty} a_k \cos(kx),$$

for some coefficients $a_k \in \mathbb{R}$. Inspired by this fact, we will fit noisy data generated using f by a cosine expansion. Generate data at 500 random points on $[0, 2\pi]$ by adding some small random noise to f:

```
ndata = 500;
f = @(x) cos(sin(x)).^3;
xdata = 2*pi*rand(ndata,1);
fdata = f(xdata) + 0.05*randn(ndata,1);
```

Use least squares fitting to fit the data to a cosine series of the form above truncated to the first 10 terms; i.e., find coefficients $a_0, \ldots, a_9$ which yield the best fit of an expansion $\hat{f}(x) = \sum_{k=0}^{9} a_k \cos(kx)$ to the data in the least squares sense. You will need to form a 500×10 overdetermined linear system, and solve it using a QR decomposition (use MATLAB's `qr` function) to obtain the coefficients. Using the coefficients, evaluate the cosine series fit $\hat{f}(x)$ on a fine grid of 1000 equispaced points on $[0, 2\pi]$. Plot the noisy data (using `plot(xdata,fdata,'o')`), your cosine series fit $\hat{f}(x)$ on the fine grid, and $f(x)$ itself. Describe what you see. Report $\max \left| f(x) - \hat{f}(x) \right|$ computed on the fine grid, and verify that it is comparable to the noise level of 0.05 (it will not be exactly the same, but should be a comparable order of magnitude). Put the code used to generate your results in a script, `prob3.m`, and submit this in addition to your write-up.