Math 566 **Homework 4** **Due March 12, 2021**
Grady Wright

1. (**Another way to solve the least squares problem**) Let A be a real m-by-n matrix with $m \geq n$ and $\text{rank}(A) = n$, and let $\mathbf{b} \in \mathbb{R}^m$.

   (a) Show that the $\mathbf{x}$ that minimizes $\|A\mathbf{x} - \mathbf{b}\|_2$ is given by the solution to the square linear system

   $$\underbrace{\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix}}_{\mathcal{A}} \begin{bmatrix} \mathbf{r} \\ \mathbf{x} \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ 0 \end{bmatrix}, \tag{1}$$

   where I is the m-by-m identity matrix. What does $\mathbf{r}$ represent here?

   Hint: Try doing 2-by-2 block Gaussian elimination on $\mathcal{A}$ to zero-out the (2,1) block entry of the matrix. This should give you a familiar equation for determining $\mathbf{x}$.s

   (b) Determine the 2-norm condition number of $\mathcal{A}$ in terms of the singular values of A.

   Hint: One way to proceed is to let $A = U\Sigma V^T$ be the ecconomized SVD of A and then consider $\mathcal{A}$ written as

   $$\mathcal{A} = \begin{bmatrix} I & U\Sigma V^T \\ V\Sigma U^T & 0 \end{bmatrix}.$$

   Next, let

   $$\mathcal{P} = \begin{bmatrix} U & 0 \\ 0 & V \end{bmatrix},$$

   and note that $\mathcal{P}$ is an orthogonal matrix and that

   $$\mathcal{P}^T \mathcal{A} \mathcal{P} = \underbrace{\begin{bmatrix} I & \Sigma \\ \Sigma & 0 \end{bmatrix}}_{\widetilde{\mathcal{A}}}.$$

   Since this $\mathcal{P}^T\mathcal{A}\mathcal{P}$ is a similarity transform of $\mathcal{A}$ we know that it will have the same eigenvalues as $\widetilde{\mathcal{A}}$. The last steps involve brute force computing the eigenvalues of $\widetilde{\mathcal{A}}$ and relating these to the singular values of A. You may consider looking up results for determinants of block 2-by-2 matrices.

   (c) How does the condition number of $\mathcal{A}$ compare to the condition number of the normal equations matrix A^TA?

   (d) Compute the polynomial coefficients from problem 1 of homework 3 with this method and compare the results to the normal equation method (a) and the QR decomposition method (e). You can use any builtin Gaussian elimination solver (or Cholesky) to solve 1 for this part of the problem.

2. *Rank-deficient problems and regularization* One area where (near) rank deficient problems arise is in image processing. As an example, we will consider the ill-posed problem of deblurring a

signal (image), which leads to a (near) rank deficient system to solve. Consider a 2π-periodic signal $x(t)$ sampled at the N equally spaced points $t_j = hj$, $j = 0, 1, \ldots, N$, where $h = 2\pi/N$. A common type of blurring that occurs in a Gaussian blur, which can be modeled by the (circular) convolution of f with a Gaussian function as follows:

$$(\kappa * x)(t) = \int_0^{2\pi} \kappa(t-\tau)x(\tau)d\tau, \tag{2}$$

where

$$\kappa(t) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{2-2\cos(t)}{2\sigma^2}}.$$

Here $\kappa(t)$ is a periodic version of the Gaussian function. We can discretize (2) using trapezoidal rule to get the linear system

$$\underbrace{\begin{bmatrix} a_0 & a_{N-1} & a_{N-2} & \cdots & a_1 \\ a_1 & a_0 & a_{N-1} & \cdots & a_2 \\ a_2 & a_1 & a_0 & \cdots & a_3 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ a_{N-1} & a_{N-2} & a_{N-3} & \cdots & a_0 \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ \vdots \\ x_{N-1} \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ \vdots \\ b_{N-1} \end{bmatrix}}_{\mathbf{b}}, \tag{3}$$

where $a_j = h\kappa(t_j)$, $j = 0, 1, \ldots, N-1$ and $x_j = x(t_j)$. The forward blurring problem can then be approximated as $\mathbf{b} = A\mathbf{x}$, i.e. given the non-blurred signal $\mathbf{x}$, compute the blurred version $\mathbf{b}$. The inverse (and ill-conditioned) problem is to solve $A\mathbf{x} = \mathbf{b}$ for $\mathbf{x}$, i.e. given the blurred signal $\mathbf{b}$, recover the non-blurred signal $\mathbf{x}$.

Note that A is a circulant matrix and the tools of Fourier analysis can be used to do many operations involving A (via the discrete convolution theorem), but we will not make use of any of that theory here.

(a) Construct the matrix A with $N = 256$ and $\sigma = 0.1$ and compute its condition number. Report this value.

(b) Construct the vector $\mathbf{x}$ from the signal

$$x(t) = \begin{cases} 1 - \frac{2}{\pi}|t-\pi| & |t-\pi| < \frac{\pi}{2} \\ 0 & |t-\pi| \geq \frac{\pi}{2} \end{cases}.$$

Compute the blurred signal $\mathbf{b}$ using $\mathbf{b} = A\mathbf{x}$. Make a plot of $\mathbf{x}$ vs. t and $\mathbf{b}$ vs. t on the same figure.

(c) Using $\mathbf{b}$ from part (b) solve $A\overline{\mathbf{x}} = \mathbf{b}$ for $\overline{\mathbf{x}}$ using the built-in Gaussian elimination solver from the software system you are working with. Make a plot of $\mathbf{x}$ vs. t, $\overline{\mathbf{x}}$ vs. t and $\mathbf{b}$ vs. t on the same figure. Does $\overline{\mathbf{x}}$ look anything close to $\mathbf{x}$?

(d) Compute a reduced rank least squares solution to the linear system $A\tilde{\mathbf{x}} = \mathbf{b}$ using the SVD by truncating all singular values of A that are less than 10^{-12}. If $\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_N$ are the left singular vectors of A, $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_N$ are the right singular vectors, and $\sigma_1, \sigma_2, \ldots, \sigma_N$ are the singular values, then the reduced rank least squares solution $\tilde{\mathbf{x}}$ is given by

$$\tilde{\mathbf{x}} = \sum_{j=1}^{r} \left(\frac{\mathbf{u}_j^T \mathbf{b}}{\sigma_j} \right) \mathbf{v}_j,$$

where r is the integer such that $\sigma_r \geq 10^{-12}$ and $\sigma_{r+1} < 10^{-12}$. Compute $\tilde{\mathbf{x}}$ and plot it together with $\mathbf{x}$ and $\mathbf{b}$ on the same figure. How does $\tilde{\mathbf{x}}$ compare to $\overline{\mathbf{x}}$? The reduced rank solution provides one way to regularize an ill-posed problem. The next part, considers another.

(e) Use ridge regression (Tikhonov regularization) to compute a regularized solution to the linear system $A\hat{\mathbf{x}} = \mathbf{b}$ using the SVD. In its simplest form, ridge regression replaces solving the linear system $A\hat{\mathbf{x}} = \mathbf{b}$ with the minimization problem

$$\min_{\hat{\mathbf{x}}}\{\|A\hat{\mathbf{x}} - \mathbf{b}\|_2^2 + \mu\|\hat{\mathbf{x}}\|_2^2\}$$

where $\mu > 0$ is the regularization parameter. Note that if $\mu = 0$ and A is square, then the solution to the minimization problem is just the solution to $A\hat{\mathbf{x}} = \mathbf{b}$. Using the same notation as part (c) for the SVD, the ridge regression solution is given as

$$\hat{\mathbf{x}} = \sum_{j=1}^{N} \left(\frac{\sigma_j}{\sigma_j^2 + \mu} \mathbf{u}_j^T \mathbf{b} \right) \mathbf{v}_j.$$

Compute $\hat{\mathbf{x}}$ for $\mu = 10^{-4}$, $\mu = 10^{-2}$, and $\mu = 1$. Plot all three of these solutions together with $\mathbf{x}$ and $\mathbf{b}$ on the same figure. How does μ effect the solution? How does $\hat{\mathbf{x}}$ compare to $\overline{\mathbf{x}}$ and $\tilde{\mathbf{x}}$?

One of the challenges with ridge regression is choosing an optimal μ. Classic techniques include the L-curve and generalized cross validation (GCV). However, this is still an area of active research, albeit with more sophisticated choices in the minimization problem.

(f) In real problems, the blurred signal will typically also be corrupted with noise. Repeat (c)–(e) by perturbing each entry of $\mathbf{b}$ by a normally distributed random number with mean zero and standard deviation 10^{-5}. In MATLAB this can be done with the `randn` function as follows: `b = b + 1e-5*randn(N,1)`

3. (**Unit lower triangular matrix**) In the derivation of the LU decomposition algorithm the following result is used:

$$\begin{bmatrix} 1 & & & & & & & & \\ 0 & 1 & & & & & & & \\ 0 & 0 & \ddots & & & & & & \\ \vdots & \vdots & \ddots & 1 & & & & & \\ 0 & 0 & 0 & \ell_{i+1,i} & 1 & & & & \\ 0 & 0 & 0 & \ell_{i+2,i} & 0 & \ddots & & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \\ 0 & 0 & 0 & \ell_{n,i} & 0 & \vdots & \vdots & \ddots & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & & & & & & & & \\ 0 & 1 & & & & & & & \\ 0 & 0 & \ddots & & & & & & \\ \vdots & \vdots & \ddots & 1 & & & & & \\ 0 & 0 & 0 & -\ell_{i+1,i} & 1 & & & & \\ 0 & 0 & 0 & -\ell_{i+2,i} & 0 & \ddots & & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \ddots & \\ 0 & 0 & 0 & -\ell_{n,i} & 0 & \vdots & \vdots & \ddots & 1 \end{bmatrix},$$

for $i = 1, 2, \ldots, n$. Show mathematically that this result is correct.

4. (**Growth factor**) The growth factor for Gaussian elimination applied to the matrix A is given by

$$g_n(A) = \frac{\max_{i,j} |u_{i,j}|}{\max_{i,j} |a_{i,j}|},$$

where $u_{i,j}$ is the entry (i, j) entry of the upper triangular matrix U in the LU factorization of A.

(a) Add code to the lupp.m MATLAB function on the course webpage (or an equivalent function in another language) to compute the growth factor and return it as an output.

(b) Use your code to compute the growth factor of the 21-by-21 matrix

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

Report the value you get and turn in your modified code. If you did things correctly, you should get $g_{21}(A) = 2^{20}$.

(c) Show that the growth for the matrix in A (but for the general n-by-n case) is exactly $g_n(A) = 2^{n-1}$.

5. (**Complete pivoting**)

(a) Using the lupp.m MATLAB code posted on the course webpage as a template, implement a function for doing Gaussian elimination with *complete pivoting* (you can use an alternative language if you like). Your function should return one matrix containing the upper and unit lower triangular matrices (similar to lupp.m), a vector p representing the row permutations that were performed, a vector q representing the column permutations that were performed, and the growth factor of the matrix. A possible function declaration in MATLAB would look like

```
[LU,p,q,gf] = lucp(A)
```

(b) Using the forsub.m and backsub.m codes in conjunction with your `lucp` code from part (a), solve the linear system

$$A = \begin{bmatrix} 1 & 1 & 1^2 & \cdots & 1^{n-1} \\ 1 & 2 & 2^2 & \cdots & 2^{n-1} \\ 1 & 3 & 3^2 & \cdots & 3^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & n-1 & (n-1)^2 & \cdots & (n-1)^{n-1} \\ 1 & n & n^2 & \cdots & n^{n-1} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_{n-1} \\ x_n \end{bmatrix} = \begin{bmatrix} 1^{n-1} \\ 2^{n-1} \\ 3^{n-1} \\ \vdots \\ (n-1)^{n-1} \\ n^{n-1} \end{bmatrix},$$

for $n = 12$. Report the max-norm of the residual in your computed solution and the max-norm of the error (you should easily be able to figure out exact solution).

(c) Compare your results from part (b) with the Gaussian elimination with partial pivoting, using, for example, the lupp.m code.

6. (**Growth factors in the wild**) Using the `lupp` and `lucp` codes from problems 4 and 5, respectively, compute the growth factors of random matrices of size $n = 100, 200, 300, \ldots, 1000$. For each n, generate 5 random matrices and compute the growth factor for each one. Plot all the results similar to what is shown in Figure 1. What do you observe in terms of how the growth factors increase with n?

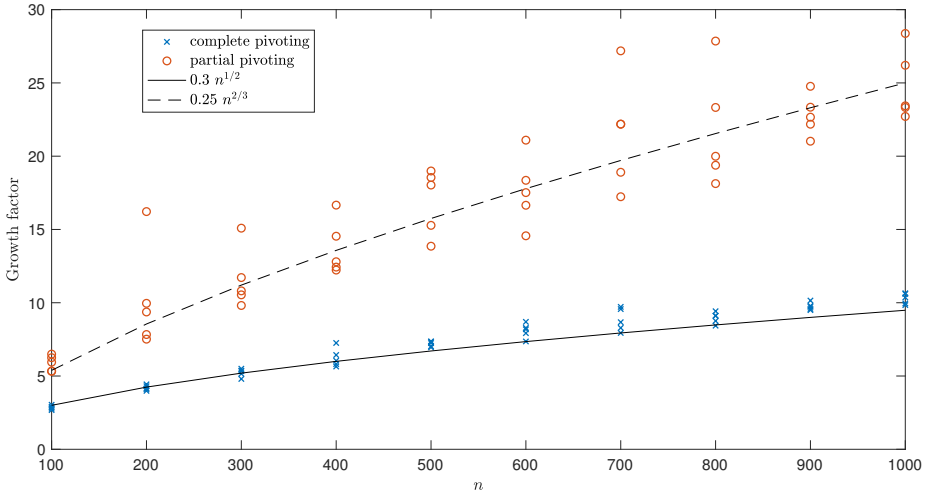

Figure 1: Comparison of the growth factors for complete and partial pivoting of some random matrices. The lines indicate the observed trend in how the growth factors are increasing with n.