Math 566 **Homework 4** **Due Mar. 3, 2023**
Grady Wright

- Problems marked NLA (Numerical Linear Algebra) are from the book.

1. (**Data fitting with polynomials**, 30 pts) Consider the following problem of finding a polynomial of degree $n-1$, $p_{n-1}(t) = c_1 + c_2 t + c_3 t^2 + \cdots + c_n t^{n-1}$, that fits the data $f_1, f_2, \ldots, f_m$ at the points $t_1, t_2, \ldots, t_m$. We can write this problem as the following Vandermonde linear system of equations for determining a set of coefficients $c_1, c_2, \ldots, c_n$:

$$\underbrace{\begin{bmatrix} 1 & t_1 & t_1^2 & \cdots & t_1^{n-1} \\ 1 & t_2 & t_2^2 & \cdots & t_2^{n-1} \\ 1 & t_3 & t_3^2 & \cdots & t_3^{n-1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & t_m & t_m^2 & \cdots & t_m^{n-1} \end{bmatrix}}_{A} \underbrace{\begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ \vdots \\ \vdots \\ c_n \end{bmatrix}}_{\mathbf{x}} = \underbrace{\begin{bmatrix} f_1 \\ f_2 \\ f_3 \\ \vdots \\ \vdots \\ \vdots \\ f_m \end{bmatrix}}_{\mathbf{b}}. \tag{1}$$

   For this problem we will take $t_j = (j-1)/(m-1)$ for $j = 1, \ldots, m$ (i.e. m equally spaced points over $[0,1]$), $f_j = \cos(4t_j)$, and set $m = 50$, $n = 12$.

   Your task is to calculate the least squares solution to this problem using the six methods listed below:

   (a) Use the normal equations $A^T A\mathbf{x} = A^T\mathbf{b}$. You can use a builtin Gaussain elimination, or Cholesky, function to solve this system (e.g. \ in MATLAB).

   (b) QR decomposition computed using the classical Gram-Schmidt method you implemented on the previous homework (if you were unable to solve this problem, send me an email and I'll send you the code).

   (c) QR decomposition computed using the modified Gram-Schmidt method you implemented on the previous homework (if you were unable to solve this problem, send me an email and I'll send you the code).

   (d) QR decomposition computed using Householder matrices you implemented on the previous homework (if you were unable to solve this problem, send me an email and I'll send you the code).

   (e) QR decomposition using a builtin function, such as the MATLAB or NumPy `qr` functions (which are also based on Householder matrices).

   (f) SVD decomposition using a builtin function, such as the MATLAB or NumPy `svd` functions.

   You should obtain 12 coefficients for each of the six tests. Arrange these coefficients (to 15-16 decimal digits) in a table and highlight in red for each method the digits that appear to be wrong (due to rounding errors). What differences and similarities do you observe? For the normal equations method (a) and for the QR method (e), plot the difference between the polynomial approximant with those coefficients and the true solution $f(t) = \cos(4t)$.

2. (**Weighted least squares**, 20 pts) Often in least squares problems $A\mathbf{x} = \mathbf{b}$, where $A \in \mathbb{R}^{m\times n}$, we may trust some data more than others and thus may want to *weight* it more in the final approximation. One way to do this is to consider minimizing the residual $\mathbf{r} = A\mathbf{x} - \mathbf{b}$ using a *weighted 2-norm*. If $W \in \mathbb{R}^{m\times m}$ is a positive definite matrix then we can define the weighted 2-norm of a vector $\mathbf{y}$ as $\|\mathbf{y}\|_W = \sqrt{\mathbf{y}^T W \mathbf{y}}$.

   (a) Derive the normal equations for computing the solution $\mathbf{x}$ that minimizes $\|A\mathbf{x} - \mathbf{b}\|_W$.

   Hint: Consider minimizing $\|A\mathbf{x} - \mathbf{b}\|_W$ and follow the steps used to derive the standard, non-weighted, normal equations in class or from Lecture 11 of NLA.

   (b) In many problems the weighting matrix is a diagonal matrix with positive entries, $W = \text{diag}(w_1, w_2, \ldots, w_m)$, $w_j > 0$. In this case, $\|A\mathbf{x} - \mathbf{b}\|_W^2 = \|\mathbf{r}\|_W^2$ reduces to

$$\|\mathbf{r}\|_W^2 = \sum_{j=1}^{m} w_j r_j^2,$$

   i.e. we are weighting the square of each residual by a possibly different number. When using least squares to do spatial or temporal approximations as in problem 1, a common choice for the weight is a Gaussian function of the form $w_j = \exp(-(|t - t_j|/\delta)^2)$, where t is the point where you want the approximation and $\delta > 0$ is a parameter that controls how much weight is applied.[1]

   Using the same polynomial approximation problem from 1, compute the weighted least squares solution using the diagonal Gaussian weight with $t = 1/23$. Report the polynomial coefficients of the weighted least squares solution and report the value of the polynomial with these coefficients at $t = 1/23$ (i.e. report $p_{11}(1/23)$). Compare these coefficients and the value at $t = 1/23$ to the coefficients and value of the the non-weighted least squares solution. Which method provides a better approximation to $f(t) = \cos(4t)$ at $t = 1/23$.

3. (**Unstable algorithm**, 10 pts) Consider the function $f(x) = \log(x + 1)/x$. This function is perfectly smooth for all $x > -1$, even though it may appear that $x = 0$ could be problematic.

   (a) Compute the condition number of this function for $x = 0$ and some values very close to zero. What does the condition number tell you about the stability of evaluating $f(x)$ near zero?

   (b) Evaluate the function $f(x) = \log(x+1)/x$ using the expression as given for x values $x_j = 2^{-52+j/10}$, $j = 0, 1, \ldots, 520$. Make a plot of f at these values of x using a logarithmic scale on the x-axis (`semilogx` for the MATLAB users). Does your "algorithm" for evaluating $f(x)$ look to be stable near $x = 0$?

   (c) Now evaluate $f(x)$ at the same x_j values as part (b) using the following algorithm (do not simplify any computations mathematically):

```
z = 1 + x
y = log(z)/(z-1)
```

   Make a plot of these values on the same plot as part (b). What do you notice about the results of this algorithm compared to the one in part (b)?

4. (**Stability of linear systems**, 10 pts) Consider the linear system of equation $A\mathbf{x} = b$, where

$$A = \begin{bmatrix} 1/2 & 1/3 & 1/4 & 1/5 & 1/6 \\ 1/3 & 1/4 & 1/5 & 1/6 & 1/7 \\ 1/4 & 1/5 & 1/6 & 1/7 & 1/8 \\ 1/5 & 1/6 & 1/7 & 1/8 & 1/9 \\ 1/6 & 1/7 & 1/8 & 1/9 & 1/10 \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} 0.882 \\ 0.744 \\ 0.618 \\ 0.521 \\ 0.447 \end{bmatrix}.$$

[1] This idea forms the foundation of the moving least squares (MLS) approximation scheme for fitting data.

Suppose you have in some way obtained the approximate solution vector to this system:

$$\hat{\mathbf{x}} = \begin{bmatrix} -2.1333 \\ 0.6258 \\ 17.4552 \\ -11.8692 \\ -1.4994 \end{bmatrix}.$$

It is then easy to show that the residual becomes *exactly*

$$\hat{\mathbf{r}} = \mathbf{b} - A\hat{\mathbf{x}} = \begin{bmatrix} -0.00001 \\ 0.00001 \\ -0.00001 \\ 0.00001 \\ -0.00001 \end{bmatrix}.$$

(a) Does this imply that $\hat{\mathbf{x}}$ is close to the exact solution $\mathbf{x}$?

(b) Use some computational software (e.g. MATLAB) to obtain an accurate solution to the given system.

(c) Obtain a condition number for A using this same software again. Use the appropriate result on perturbations of the right hand side (RHS) of a linear system to confirm that this very small residual indeed is big enough to allow for the solution to be as far away from the correct one as occurs in this example.

Hint: The A matrix can be constructed easily in MATLAB using the function `hankel` (use MATLAB `help` to see why). The following code constructs A:

```
A = 1./hankel(2:6,6:10).
```

The vector b can be entered into MATLAB manually:

```
b = [0.882 0.744 0.618 0.521 0.447]'.
```

The linear system $A\mathbf{x} = \mathbf{b}$ can be solved in MATLAB using the `mldivide` command or simply the backslash operator "\":

```
x = A\b
```

The condition number (with respect to the two-norm) can be computed using the MATLAB function `cond`:

`cond(A)` Make sure you use the same norm for all computations.

5. (**Condition of the Vandermonde system**, 10 pts) The Vandermonde matrix A from problem 1 (see (1)) is a classic example of an ill-conditioned system. Using the same setup as problem 1 for the points t_j, and setting $m = n$, make a plot of the two-norm condition number of A, as a function of n, for $n = 1, 2, \ldots, 30$. Use a logarithmic scale on y-axis and linear scale on the x-axis for your plot (i.e. `semilogy` in MATLAB). Repeat this experiment, but now set $m = 2n - 1$ so that the system is overdetermined. Does this have any effect on the condition numbers from the first experiment?

6. NLA 13.3 (5 pts)

7. (Skeel condition number, 10 pts) The *Skeel condition number* (or *componentwise relative condition number*) can sometimes be used to give a sharper bound on the relative error when solving a perturbed linear system. This is especially true when different columns of a matrix have widely varying scales. The Skeel condition number is defined as [1]

$$\kappa_{CR}(A) = \| \, |A^{-1}| \, |A| \, \|,$$

where absolute value of a matrix means the absolute value of its entries.

To test the claim, consider the matrix

$$P = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

This is a permutation matrix, and thus an orthogonal matrix, so that its condition number is 1. Verify that this is true for both the standard condition number and the Skeel condition number using the two norm[2]

Now consider scaling the third column of P by 10^{-10} and leaving all the other columns unchanged. Compare the standard condition number for the modified P to the Skeel condition number, again using the two norm. Use words and numbers when doing your comparison.

While the large condition number using the standard definition for the modified P looks scary, it's really not. There will be no issues with Gaussian elimination when using the modified P to solve a linear system.

References

[1] Nicholas J. Higham. *Accuracy and Stability of Numerical Algorithms*. Society for Industrial and Applied Mathematics, January 2002.

[2] I know I said in class that you should never compute the inverse of a matrix, but for this problem it is okay. There are ways of estimating the Skeel condition number that do not involve computing an inverse, but we will not use these here.