Vera Xie
Ronald Nap
Andrew Rocco
3L = F 3:30 – 4:20 PM

Problem Set 4
Math 146 Spring 2023
Due Thursday, March 9, 11:59 PM

Note: Number 4(a-d) is a "pen and paper" problem for which you should show work. Numbers 1 - 3 & 4e are Matlab problems that require code submissions. All answers should be submitted in a write-up.

1. (a) (20 points) Create a function to solve the system $AX = B$, for A, X, and B, $n \times n$ matrices. The function should take as input matrices A and B and give the output X. Validate your code. Explain how you validated and provide any relevant output. Note: This only requires a slight modification to the code you created in Problem Set 3, Number 3b.

   (b) (20 points) Create a function to compute the inverse of a square $n \times n$ matrix A. The function should take as input the matrix A and give the output A^{-1}. Validate your code. Explain how you validated and provide any relevant output.

2. Included in the PS 4 Folder is a function that uses the Thomas Algorithm to solve the tridiagonal system of equations given below. Notice that the algorithm has been altered from the pseudocode provided in class in order to eliminate one of the for loops.

$$\begin{bmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & & \\ & & & & c_{n-1} \\ 0 & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x1 \\ \vdots \\ \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ \vdots \\ \\ d_n \end{bmatrix} \tag{1}$$

   (a) (20 points) The function takes as input $n \times 1$ vectors (a), $\mathbf{b}$, $\mathbf{c}$, and $\mathbf{d}$. Modify the algorithm to make a new function, Thomas_2, which takes as input the tridiagonal matrix A and vector $\mathbf{d}$ instead. Include in your function errors if A is not a square tridiagonal matrix. (Hint: Use the command `isbanded`.) Validate your code by creating a sparse 1000×1000 tridiagonal matrix made up of random numbers from 0 to 50 and a similarly formed vector for $\mathbf{d}$. Compare your solution to the solution found by using Matlab backslash, and report the relative difference between the two solutions, with respect to the max norm.

   (b) (30 BONUS points) Modify the algorithm to solve problems of the form

$$\begin{bmatrix} b_1 & c_1 & & & a_1 \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & & \\ & & & & c_{n-1} \\ 0 & & & a_n & b_n \end{bmatrix} \begin{bmatrix} x1 \\ \vdots \\ \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ \vdots \\ \\ d_n \end{bmatrix}, \tag{2}$$

   where all terms not on the tridiagonal or in the upper right corner are zeros. Show any work you did or explain how you made your modification. (You may want to explore

the original algorithm a little first.) Validate your code by solving the system

$$\begin{bmatrix} 1 & -2 & 0 & 0 & 0 & -1 \\ 2 & -2 & 3 & 0 & 0 & 0 \\ 0 & -2 & 3 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 & 1 & 0 \\ 0 & 0 & 0 & -2 & -4 & -1 \\ 0 & 0 & 0 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} x1 \\ \\ \vdots \\ \\ x_6 \end{bmatrix} = \begin{bmatrix} -9 \\ 7 \\ 9 \\ 22 \\ -34 \\ 17 \end{bmatrix}. \tag{3}$$

The exact solution is $\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}$. Report your relative error, with respect to the 1-norm, 2-norm, and max norm.

3. (a) (32 points) Create a function to implement the Cholesky factorization pseudocode from class. Your function should take as input an $n \times n$ positive definite symmetric matrix A and give output $R^\intercal$. Validate your code by factoring the symmetric positive definite matrix $A = \begin{bmatrix} 10 & 5 & 2 \\ 5 & 3 & 2 \\ 2 & 2 & 3 \end{bmatrix}$. Explain how you then verify that your solution is correct.

   (b) (18 points) Create a function that takes as input an $n \times n$ positive definite symmetric matrix A and a vector $\mathbf{b}$ and uses your Cholesky factorization from part (a) to solve the linear system $A\mathbf{x} = \mathbf{b}$. Validate your code. Explain how you validated the code and provide any relevant output.

4. You would like to fit the following data with a second order polynomial (a parabola).

x	y
-2	9
-1	5
0	3
1	4
2	8
3	12

   (a) (12 points) Let the polynomial be $p(x) = c_0 + c_1 x + c_2 x^2$. Write by hand the 6 equations given to us by this data. Also give the system of equations in matrix form.

   (b) (17 points) Find the normal equations to this system.

   (c) (22 points) Solve the normal equations by hand. What polynomial gives the Least Squares best fit polynomial for this data?

   (d) (17 points) Find the residual and its magnitude.

   (e) (22 points) Confirm your work in Matlab. Note: You can solve the normal equations using function from Problem (3b) since $A^\intercal A$ is a symmetric positive definite matrix. Plot the data and your Least Squares polynomial.

1. (a) (20 points) Create a function to solve the system $AX = B$, for A, X, and B, $n \times n$ matrices. The function should take as input matrices A and B and give the output X. Validate your code. Explain how you validated and provide any relevant output. Note: This only requires a slight modification to the code you created in Problem Set 3, Number 3b.

   (b) (20 points) Create a function to compute the inverse of a square $n \times n$ matrix A. The function should take as input the matrix A and give the output A^{-1}. Validate your code. Explain how you validated and provide any relevant output.

① MATLAB : submitted **Math146_PS4_Q1.m** file

a.) function to solve the system $AX = B$ for A, X, B : $n \times n$ matrices
- INPUTS : A , $n \times n$ matrix
  B , $n \times n$ matrix
- OUTPUT : X , $n \times n$ matrix

→ Explain how you validated & provide any relevant output
- NOTE : only requires a slight modification to the code you created in PS 3 #3b

⟹ submitted **GEPivoting_Matrices.m** file ⟹ PART of the file :
- submitted : **LUDecomp_Pivot.m** file
  forward_sub.m file
  backward_sub.m file

→ $\text{Rel. Error}_\infty = 9.7135 \times 10^{-15} \approx 10 \times 10^{-15} = 1 \times 10^{-14}$ ⟹ $\text{Rel. Error}_\infty = O(10^{-14})$

→ $K(A)_\infty = 94.1370 \approx 100 = 1 \times 10^{2}$ ⟹ $K(A)_\infty = O(10^{2})$

∴ lose 2 DIGITS

To validate my code, we generated a 10×10 matrix of random integers between 0 to 50 for both the given matrices A and B, and compared the solution matrix X to MATLAB'S calculated solution to determine the relative "error" (or relative difference) between both solutions. The relative difference with respect to max norm is $O(10^{-14})$. As the size of the relative difference is relatively close to ε_{mach}, the function we created is working properly, such that it outputs a good approximation to the solution. Thus, this indicates we have about 13 or 14 digits of accuracy for each component of the solution matrix X. The condition number of matrix A with respect to max norm is $O(10^2)$. This confirms we lose only 2 digits of accuracy, which indicates the function we created is well-conditioned. Thus, we can only "trust" 14 digits of any component of the solution matrix X.

b.) function to compute the INVERSE of a square $n \times n$ matrix A.

— INPUT : A, $n \times n$ matrix

— OUTPUT : A^{-1}, $n \times n$ matrix

└ Validate your code. Explain how you validated & provide any relevant output

⟹ submitted **inv_Matrix.m** file ⟹ PART of the file :

— submitted : **LU_Decomp.m** file
forward_sub.m file
backward_sub.m file

→ Error $= I - (A \cdot A^{-1})$

└ Max Error $= 4.9738 \times 10^{-14}$ ⟹ Max Error $= O(10^{-14})$

→ Rel. Error$_\infty$ $= 2.8535 \times 10^{-13}$ ⟹ Rel. Error$_\infty$ $= O(10^{-13})$

∴ lose 3 DIGITS

→ $K(A)_\infty = 1.1562 \times 10^{3}$ ⟹ $K(A)_\infty = O(10^{3})$

For a given matrix A, the INVERSE of the matrix (A^{-1}) can be determined using $A \cdot A^{-1} = A^{-1} \cdot A = I$, where I is the identity matrix. To validate my code, we generated a 10×10 matrix of random integers between 0 to 50 for the given matrix A, and compared the solution (i.e, inverse of matrix A) to MATLAB'S calculated solution. We checked if $I - (A \cdot A^{-1})$ returns $\underline{0}$, then there is no error. We determined the largest component in magnitude of the error and got $O(10^{-14})$. The size of the error is relatively close to ε_{mach}. This confirms the function we created is working properly, such that it outputs a good approximation for the inverse of the matrix. We also calculated the relative error and the condition number to check for the digits of accuracy. As $A \times A^{-1} = I$, we have an exact solution, which means we can compute the relative error. The relative error with respect to max norm is $O(10^{-13})$. Thus, this indicates we have about 12 or 13 digits of accuracy for each component of the output. The condition number of matrix A with respect to max norm is $O(10^{3})$. This confirms we lose only 3 digits of accuracy, which indicates the function we created is well-conditioned. Thus, we can "trust" 13 digits of any component of the calculated inverse of the matrix.

2. Included in the PS 4 Folder is a function that uses the Thomas Algorithm to solve the tridiagonal system of equations given below. Notice that the algorithm has been altered from the pseudocode provided in class in order to eliminate one of the for loops.

$$\begin{bmatrix} b_1 & c_1 & & & 0 \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & & \\ & & & c_{n-1} & \\ 0 & & a_n & b_n \end{bmatrix} \begin{bmatrix} x1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ \vdots \\ d_n \end{bmatrix} \tag{1}$$

(a) (20 points) The function takes as input $n \times 1$ vectors **a**, **b**, **c**, and **d**. Modify the algorithm to make a new function, Thomas_2, which takes as input the tridiagonal matrix A and vector **d** instead. Include in your function errors if A is not a square tridiagonal matrix. (Hint: Use the command `isbanded`.) Validate your code by creating a sparse 1000×1000 tridiagonal matrix made up of random numbers from 0 to 50 and a similarly formed vector for **d**. Compare your solution to the solution found by using Matlab backslash, and report the relative difference between the two solutions, with respect to the max norm.

(b) (30 BONUS points) Modify the algorithm to solve problems of the form

$$\begin{bmatrix} b_1 & c_1 & & & a_1 \\ a_2 & b_2 & c_2 & & \\ & \ddots & \ddots & & \\ & & & c_{n-1} & \\ 0 & & a_n & b_n \end{bmatrix} \begin{bmatrix} x1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ \vdots \\ d_n \end{bmatrix}, \tag{2}$$

where all terms not on the tridiagonal or in the upper right corner are zeros. Show any work you did or explain how you made your modification. (You may want to explore the original algorithm a little first.) Validate your code by solving the system

$$\begin{bmatrix} 1 & -2 & 0 & 0 & 0 & -1 \\ 2 & -2 & 3 & 0 & 0 & 0 \\ 0 & -2 & 3 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 & 1 & 0 \\ 0 & 0 & 0 & -2 & -4 & -1 \\ 0 & 0 & 0 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} x1 \\ \vdots \\ x_6 \end{bmatrix} = \begin{bmatrix} -9 \\ 7 \\ 9 \\ 22 \\ -34 \\ 17 \end{bmatrix}. \tag{3}$$

The exact solution is $\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}$. Report your relative error, with respect to the 1-norm, 2-norm, and max norm.

② MATLAB : submitted **Math146_PS4_Q2a.m** file

a.) The function [Thomas.m] takes as input : n × 1 vectors **a**, **b**, **c**, & **d**
Modify the algorithm to make a new function [Thomas_2.m]
— INPUTS : A, n×n TRIDIAGONAL matrix
d, n × 1 vector
— OUTPUT : **x**, n × 1 vector
└ include in your function ERRORS if A is NOT a square tridiagonal matrix
(HINT : use "isbanded" command)
└ validate your code by : A = SPARSE 1000 × 1000 tridiagonal matrix of random #'s b/w [0,50]
d = 1000 × 1 vector of random #'s b/w [0,50]

compare your solution to MATLAB's backslash solution &
report the relative difference b/w the 2 solutions w.r.t. max norm

⟹ **THOMAS Algorithm**

$$\begin{bmatrix} b_1 & c_1 & & 0 \\ a_2 & b_2 & \ddots & \\ & \ddots & \ddots & c_{n-1} \\ 0 & & a_n & b_n \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ \vdots \\ d_n \end{bmatrix}$$

PSEUDOCODE

$$c_1' = \frac{c_1}{b_1}$$

for $i = 2 : n-1$

$$d_i' = \frac{c_i}{b_i - a_i c_{i-1}'}$$

use PREVIOUS found c_i'

end

$$d_1' = \frac{d_1}{b_1}$$

$d_i =$

for $i = 2 : n$

$$d_i' = \frac{d_i - a_i d_{i-1}'}{b_i - a_i c_{i-1}'}$$

end

$$x_n = d_n'$$

count BACKWARDS from $n-1$ TO 1

for $i = n-1 : 1 : 1$

$$x_i = d_i' - c_i' x_{i+1}$$

end

⟶ Rel. Diff$_\infty$ = $1.14529806 \times 10^{-13}$ $\Longrightarrow$ Rel. Diff$_\infty$ = $O(10^{-13})$

⟶ $K(A)_\infty$ = 6.5907408×10^4 $\approx$ 10×10^4 $= 10^5$ $\Longrightarrow$ $K(A)_\infty = O(10^5)$

Both my solution and MATLAB's solution of $\underline{x}$ was validated using the "isequal" command. The output was "logical 1", which confirms that both solutions are equivalent. To further validate the code, we determined the relative difference between my function and MATLAB's backslash command with respect to max−norm. The relative difference with respect to max−norm is $O(10^{-13})$, which validates my code as the relative difference is extremely small (relatively close to ϵ_{mach}). This indicates we have about 12 or 13 digits of accuracy, where 3 or 4 digits of accuracy are lost. The condition number of this matrix A with respect to max−norm is $O(10^5)$, which indicates we can lose 5 or 6 digits of accuracy. Thus, we expect the true relative error to be $O(10^{-11})$ or $O(10^{-10})$. Most likely both my algorithm and MATLAB's algorithm are computing the solution $\underline{x}$ in a similar manner, which results in errors of similar magnitude. Thus, the relative difference is smaller than the true relative error

b.) BONUS : submitted **Math146_PS4_Q2b_BONUS.m** file

Modify the algorithm to solve problems of the form, where all terms NOT on the tridiagonal or in the upper right corner are 0's.

$$\begin{bmatrix} b_1 & c_1 & & & a_1 \\ a_2 & b_2 & \ddots & & \\ & \ddots & \ddots & c_{n-1} & \\ 0 & & a_n & b_n & \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} d_1 \\ \vdots \\ d_n \end{bmatrix}$$

→ Show any work you did OR explain how you made your modification.

→ Validate your code by solving the system

$$\begin{bmatrix} 1 & -2 & 0 & 0 & 0 & -1 \\ 2 & -2 & 3 & 0 & 0 & 0 \\ 0 & -2 & 3 & 1 & 0 & 0 \\ 0 & 0 & 3 & 2 & 1 & 0 \\ 0 & 0 & 0 & -2 & -4 & -1 \\ 0 & 0 & 0 & 0 & 1 & 2 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_6 \end{bmatrix} = \begin{bmatrix} -9 \\ 7 \\ 9 \\ 22 \\ -34 \\ 17 \end{bmatrix} \Longrightarrow \quad \text{EXACT solution : } \underline{x} = \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{bmatrix}$$

→ Report your relative error w.r.t. 1-norm, 2-norm, & Max-norm

2b) This almost tridiagonal matrix can be solved by first reducing our matrix to exclude the last row, column. This is because we want to use the Thomas algorithm but cannot because of the x_n term. By reducing our matrix into n-1 we can then solve for our system in terms of x_n. We can later return to these system of equations to solve for $x_1 - x_{n-1}$ once we solve for x_n. To solve x_n, we utilize the solutions to our reduced matrix. These solutions act as solution vector for our reduced system. Next, we need to compute the coefficients of our x_n using the entries a_1 and c_{n-1} from the last column that we ignored. Using the solution vector and the coefficients of x_n we can generate the following

(1) $x_{n-1} = v_{n-1} - t_{n-1} x_n$ where v is the solution vector, t is the coefficient of x_n

Now that we have a relationship for x_{n-1}, we can solve for x_n using the last row that we ignored. Using matrix multiplication we find that $d_n = c_n x_{n-1} + a_n x_{n-1} + b_n x_n$

→ if we solve for x_n and substitute (1) so $x_n = \dfrac{d_n - c_n v_1 - a_n v_{n-1}}{b_n - c_n t_1 - a_n t_{n-1}}$

After solving for x_n we can plug in x_n into our remaining equations to solve for $x_1 - x_{n-1}$, our remaining solutions. The algorithm implemented is called Ahlberg-Nilson-Walsh algorithm and "The Theory of Splines and Their Applications" was used as a reference to complete the modification.

→ Rel. Error$_1$ = $3.172065785 \times 10^{-16}$ ⟹ Rel. Error$_1$ = $O(10^{-16})$

→ Rel. Error$_2$ = $3.6359210 \times 10^{-16}$ ⟹ Rel. Error$_2$ = $O(10^{-16})$

→ Rel. Error$_\infty$ = $4.4408920985 \times 10^{-16}$ ⟹ Rel. Error$_\infty$ = $O(10^{-16})$

These relative errors with respect to 1-norm, 2-norm, and max-norm are all extremely small (essentially the same magnitude as ϵ_{mach}). This confirms the modified Thomas algorithm we created is working, where we have about 15 or 16 digits of accuracy.

3. (a) (32 points) Create a function to implement the Cholesky factorization pseudocode from class. Your function should take as input an $n \times n$ positive definite symmetric matrix A and give output R^T. Validate your code by factoring the symmetric positive definite matrix $A = \begin{bmatrix} 10 & 5 & 2 \\ 5 & 3 & 2 \\ 2 & 2 & 3 \end{bmatrix}$. Explain how you then verify that your solution is correct.

(b) (18 points) Create a function that takes as input an $n \times n$ positive definite symmetric matrix A and a vector $\mathbf{b}$ and uses your Cholesky factorization from part (a) to solve the linear system $A\mathbf{x} = \mathbf{b}$. Validate your code. Explain how you validated the code and provide any relevant output.

③ MATLAB : submitted Math146_PS4_Q3.m file

a.) function to implement CHOLESKY FACTORIZATION pseudocode
- INPUT : A , n×n symmetric (+) definite Matrix
- OUTPUT : R^T

↳ validate your code by factoring the symmetric (+) definite matrix A : $\begin{bmatrix} 10 & 5 & 2 \\ 5 & 3 & 2 \\ 2 & 2 & 3 \end{bmatrix}$

↳ Explain how you then verify that your solution is correct.

⟹ CHOLESKY FACTORIZATION Pseudocode

- INPUT : A
- OUTPUT : R^T , LOWER Δr s.t. $A = R^T R$

```
for j = 1 : n-1
    a_jj = sqrt(a_jj)
    for i = j+1 : n
        a_ij = a_ij / a_jj          } w / sqrt(a_11)
    end
    for k = j+1 : n        % COLUMNS
        for i = j+1 : n    % ROWS
            a_ik = a_ik - a_ij a_kj   } K - w w^T
        end
    end
end
a_nn = sqrt(a_nn)
R^T = tril(A)
```

Cholesky Factorization is the decomposition of a given symmetric positive definite matrix into the product of a lower triangular matrix (R^T) and its transpose (R), where the decomposition is of the form : $A = R^T R$. The given matrix A is $\begin{bmatrix} 10 & 5 & 2 \\ 5 & 3 & 2 \\ 2 & 2 & 3 \end{bmatrix}$, which is a 3×3 symmetric positive definite matrix. Using my function we created for the Cholesky Factorization, the output R^T is $\begin{bmatrix} 3.1623 & 0 & 0 \\ 1.5811 & 0.7071 & 0 \\ 0.6325 & 1.4142 & 0.7746 \end{bmatrix}$, which is a lower triangular matrix. Its transpose (R) is $\begin{bmatrix} 3.1623 & 1.5811 & 0.6325 \\ 0 & 0.7071 & 1.4142 \\ 0 & 0 & 0.7746 \end{bmatrix}$, which is an upper triangular matrix. These matrices are the same as MATLAB's function [chol(A)], which factorizes the given symmetric positive definite matrix into an upper triangular matrix (R). To validate my code, we checked that the product of my calculated lower triangular matrix (R^T) and its transpose (R) outputs the same exact matrix A given. The output A is $\begin{bmatrix} 10.0000 & 5.000 & 2.000 \\ 5.000 & 3.000 & 2.000 \\ 2.000 & 2.000 & 3.000 \end{bmatrix}$, which is consistent with the given matrix A. We also checked the relative difference between my solution and MATLAB's solution of the upper triangular matrix (R) with respect to max norm and got 3.6516×10^{-15}. Similarly, we also checked the relative difference between both solutions of the lower triangular matrix (R^T) with respect to max norm and got 1.1070×10^{-14}. As the size of these relative differences are relatively close to ε_{mach}, the function we created is working properly, such that it outputs a good approximation to the solution. The condition number of matrix A with respect to max norm is 266.3333, which is approximately 300 or 3×10^2. Thus, this indicates we lose 2 digits of accuracy, which confirms the function we created is well-conditioned. Thus, we can "trust" 14 or 15 digits of accuracy for each component of the solution.

b.) function that takes : — INPUTS : A, $n \times n$ symmetric (+) definite matrix
$\underline{b}$, $n \times 1$ vector

& uses your CHOLESKY FACTORIZATION from part (a) to solve the linear system $A\underline{x} = \underline{b}$

→ validate your code. Explain how you validated the code & provide any relevant output.

→ Rel. $\text{Diff}_\infty = 1.374257 \times 10^{-13} \implies$ **Rel. $\text{Diff}_\infty = O(10^{-13})$**

→ $K(A)_\infty = 1.20702 \times 10^5 \implies$ **$K(A)_\infty = O(10^5)$**

First, we generated a 10×10 matrix G made up of random numbers between 0 and 50. Then, we formed a symmetric positive definite matrix using the relationship $A = G^T G$. We also generated a 10×1 vector $\underline{b}$ made up of random numbers between 0 and 50. As matrix A is now symmetric positive definite, we can solve the linear system $A\underline{x} = \underline{b}$ using CHOLESKY FACTORIZATION. By using Cholesky Factorization, we can decompose matrix A into a lower triangular matrix (R^T), which can be transposed to find the upper triangular matrix (R). The solution $\underline{x}$ can be determined using forward substitution on R^T with $\underline{b}$ to solve for $\underline{y}$ $[R^T\underline{y} = \underline{b}]$, followed by using backward substitution on R with $\underline{y}$ to solve for $\underline{x}$ $[R\underline{x} = \underline{y}]$. To validate my code, we determined the relative difference between both solutions with respect to max-norm, using MATLAB's backslash function as the "exact" solution. The size of the relative difference is $O(10^{-13})$, which is relatively close to ϵ_{mach}. This indicates we have about 12 or 13 digits of accuracy, where we lose about 3 or 4 digits of accuracy. I also checked the condition number of the symmetric positive definite matrix A with respect to max-norm and got $O(10^5)$. This indicates we expect to lose about 5 digits of accuracy for the worst case scenario, which indicates the size of the TRUE relative error is about $O(10^{-11})$. Both my function and MATLAB's function likely have relatively the same size error as they both compute the solution $\underline{x}$ in a similar manner, so the relative difference is expected to be smaller than the true relative error.

submitted **Cholesky_Factor.m** file

$\Longrightarrow$ PART of file :

submitted : **forward_sub.m** file
backward_sub.m file

4. You would like to fit the following data with a second order polynomial (a parabola).

x	y
-2	9
-1	5
0	3
1	4
2	8
3	12

(a) (12 points) Let the polynomial be $p(x) = c_0 + c_1 x + c_2 x^2$. Write by hand the 6 equations given to us by this data. Also give the system of equations in matrix form.

(b) (17 points) Find the normal equations to this system.

(c) (22 points) Solve the normal equations by hand. What polynomial gives the Least Squares best fit polynomial for this data?

(d) (17 points) Find the residual and its magnitude.

(e) (22 points) Confirm your work in Matlab. Note: You can solve the normal equations using function from Problem (3b) since $A^T A$ is a symmetric positive definite matrix. Plot the data and your Least Squares polynomial.

④

x	y
-2	9
-1	5
0	3
1	4
2	8
3	12

a.) 2^{nd} – order polynomial : $p(x) = c_0 + c_1 x + c_2 x^2$

write the 6 EQNS given by this data & give the system of eqns. in MATRIX FORM

→ $p(x) = c_0 + c_1 x + c_2 x^2 \implies p(x_i) = y_i \quad ; \; i = 0, 1, \ldots, n$

↳ $p(-2) = c_0 + c_1(-2) + c_2(-2)^2$
$= c_0 - 2c_1 + 4c_2 \implies p(-2) = c_0 - 2c_1 + 4c_2 = 9$

↳ $p(-1) = c_0 + c_1(-1) + c_2(-1)^2$
$= c_0 - c_1 + c_2 \implies p(-1) = c_0 - c_1 + c_2 = 5$

↳ $p(0) = c_0 + c_1(0) + c_2(0)^2$
$= c_0 \implies p(0) = c_0 = 3$

↳ $p(1) = c_0 + c_1(1) + c_2(1)^2$
$= c_0 + c_1 + c_2 \implies p(1) = c_0 + c_1 + c_2 = 4$

↳ $p(2) = c_0 + c_1(2) + c_2(2)^2$
$= c_0 + 2c_1 + 4c_2 \implies p(2) = c_0 + 2c_1 + 4c_2 = 8$

↳ $p(3) = c_0 + c_1(3) + c_2(3)^2$
$= c_0 + 3c_1 + 9c_2 \implies p(3) = c_0 + 3c_1 + 9c_2 = 12$

$$
\begin{aligned}
p(-2) &= C_0 - 2C_1 + 4C_2 = 9 \\
p(-1) &= C_0 - C_1 + C_2 = 5 \\
p(0) &= C_0 = 3 \\
p(1) &= C_0 + C_1 + C_2 = 4 \\
p(2) &= C_0 + 2C_1 + 4C_2 = 8 \\
p(3) &= C_0 + 3C_1 + 9C_2 = 12
\end{aligned}
\quad \Longrightarrow \quad
\begin{bmatrix} 1 & -2 & 4 \\ 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix}
\begin{bmatrix} C_0 \\ C_1 \\ C_2 \end{bmatrix}
= \begin{bmatrix} 9 \\ 5 \\ 3 \\ 4 \\ 8 \\ 12 \end{bmatrix}
$$

[SYSTEM OF EQUATIONS] [in MATRIX FORM]

b.) find the NORMAL EQNS to this system

$\Longrightarrow$ Normal Eqns. : system of n linear equations in n unknowns

$$[A^T A]\underline{x} = [A^T \underline{b}] \quad ; \quad B = A^T A, \quad \underline{y} = A^T \underline{b}$$

B : symmetric (+) definite, given A has full column rank

$$\longrightarrow A = \begin{bmatrix} 1 & -2 & 4 \\ 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix} \Longrightarrow A^T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ -2 & -1 & 0 & 1 & 2 & 3 \\ 4 & 1 & 0 & 1 & 4 & 9 \end{bmatrix}$$

$$\longrightarrow A^T A = \underset{3\times 6}{\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ -2 & -1 & 0 & 1 & 2 & 3 \\ 4 & 1 & 0 & 1 & 4 & 9 \end{bmatrix}} \underset{6\times 3}{\begin{bmatrix} 1 & -2 & 4 \\ 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix}} = \underset{3\times 3}{\begin{bmatrix} 6 & 3 & 19 \\ 3 & 19 & 27 \\ 19 & 27 & 115 \end{bmatrix}} \} B$$

$$\longrightarrow A^T \underline{b} = \underset{3\times 6}{\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 \\ -2 & -1 & 0 & 1 & 2 & 3 \\ 4 & 1 & 0 & 1 & 4 & 9 \end{bmatrix}} \underset{6\times 1}{\begin{bmatrix} 9 \\ 5 \\ 3 \\ 4 \\ 8 \\ 12 \end{bmatrix}} = \underset{3\times 1}{\begin{bmatrix} 41 \\ 33 \\ 185 \end{bmatrix}} \} \underline{y}$$

$$\longrightarrow [A^TA]\underline{x} = [A^T\underline{b}] \Longrightarrow \boxed{\begin{bmatrix} 6 & 3 & 19 \\ 3 & 19 & 27 \\ 19 & 27 & 115 \end{bmatrix}\begin{bmatrix} C_0 \\ C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 41 \\ 33 \\ 185 \end{bmatrix}} \quad \begin{bmatrix} \text{NORMAL} \\ \text{EQNS.} \end{bmatrix}$$

$$B \qquad \underline{C} \qquad \underline{y}$$

c.) solve the normal equations.

what polynomial gives you the LEAST SQUARES BEST FIT POLYNOMIAL for this data?

$$\begin{bmatrix} 6 & 3 & 19 \\ 3 & 19 & 27 \\ 19 & 27 & 115 \end{bmatrix}\begin{bmatrix} C_0 \\ C_1 \\ C_2 \end{bmatrix} = \begin{bmatrix} 41 \\ 33 \\ 185 \end{bmatrix}$$

$$\longrightarrow 6C_0 + 3C_1 + 19C_2 = 41 \Longrightarrow C_0 = \frac{41 - 3C_1 - 19C_2}{6}$$

$$\longrightarrow 3C_0 + 19C_1 + 27C_2 = 33 \longrightarrow 3\left[\frac{41 - 3C_1 - 19C_2}{2\cancel{6}}\right] + 19C_1 + 27C_2 = 33$$

$$= \frac{41 - 3C_1 - 19C_2}{2} + 19C_1 + 27C_2 = 33$$

$$= \frac{41 - 3C_1 - 19C_2 + 38C_1 + 54C_2}{2} = 33$$

$$= \frac{41 + 35C_1 + 35C_2}{2} = 33$$

$$= 41 + 35C_1 + 35C_2 = 66 \Longrightarrow C_1 = \frac{66 - 41 - 35C_2}{35}$$

$$= \frac{25 - 35C_2}{35}$$

$$= \frac{5 - 7C_2}{7} = \frac{5}{7} - C_2$$

$$\longrightarrow 19C_0 + 27C_1 + 115C_2 = 185 \longrightarrow 19\left[\frac{41 - 3C_1 - 19C_2}{6}\right] + 27\left[\frac{5 - 7C_2}{7}\right] + 115C_2 = 185$$

$$= \frac{779 - 57C_1 - 361C_2}{6} + \frac{135 - 189C_2}{7} + 115C_2 = 185$$

$$= \frac{7[779 - 57C_1 - 361C_2] + 6[135 - 189C_2] + 42[115C_2]}{42} = 185$$

$$= 5453 - 399C_1 - 2527C_2 + 810 - 1134C_2 + 4830C_2 = 7770$$

$$= 6263 - 399C_1 + 1169C_2 = 7770$$

$$\Longrightarrow C_2 = \frac{7770 - 6263 + 399C_1}{1169} = \frac{1507 + 399C_1}{1169}$$

$$\longrightarrow \quad C_2 = \frac{1507 + 399C_1}{1169}$$

$$= \frac{1507 + \overset{57}{\cancel{399}}\left[\frac{5 - 7C_2}{\cancel{7}}\right]}{1169} = \frac{1507 + 57[5 - 7C_2]}{1169}$$

$$= \frac{1507 + 285 - 399C_2}{1169}$$

$$C_2 = \frac{1792 - 399C_2}{1169} \longrightarrow 1169C_2 = 1792 - 399C_2$$

$$\therefore 1568C_2 = 1792 \Longrightarrow C_2 = \frac{1792}{1568}$$

$$\longrightarrow C_2 = \frac{1792}{1568} = 1.142857143 \Longrightarrow \boxed{C_2 = 1.142857143}$$

$$\longrightarrow C_1 = \frac{5}{7} - C_2$$

$$= \frac{5}{7} - [1.142857143] = -0.4285714287 \Longrightarrow \boxed{C_1 = -0.4285714287}$$

$$\longrightarrow C_0 = \frac{41 - 3C_1 - 19C_2}{6}$$

$$= \frac{41 - 3[-0.4285714287] - 19[1.142857143]}{6}$$

$$= \frac{41 - [-1.285714286] - [21.71428572]}{6}$$

$$= \frac{20.57142857}{6} = 3.428571428 \Longrightarrow \boxed{C_0 = 3.428571428}$$

$$\Longrightarrow p(x) = C_0 + C_1 x + C_2 x^2$$

$$= [3.428571428] + [-0.4285714287]x + [1.142857143]x^2$$

$$\boxed{p(x) = 3.428571428 - 0.4285714287x + 1.142857143x^2}$$

[LEAST SQUARES BEST - FIT Polynomial]

d.) Find the RESIDUAL & its MAGNITUDE

$\Longrightarrow$ Residual : $\hat{r} = \hat{b} - A\hat{x}$

$\longrightarrow \hat{r} = \hat{b} - A\hat{c}$; $\begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 3.428571428 \\ -0.4285714287 \\ 1.142857143 \end{bmatrix}$

$\longrightarrow A\hat{c} = \begin{bmatrix} 1 & -2 & 4 \\ 1 & -1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 2 & 4 \\ 1 & 3 & 9 \end{bmatrix} \begin{bmatrix} 3.428571428 \\ -0.4285714287 \\ 1.142857143 \end{bmatrix} = \begin{bmatrix} 8.857142857 \\ 5 \\ 3.428571428 \\ 4.142857142 \\ 7.142857143 \\ 12.42857143 \end{bmatrix}$

6×3 $\quad 3 \times 1$ $\quad 6 \times 1$

$\longrightarrow \hat{r} = \begin{bmatrix} 9 \\ 5 \\ 3 \\ 4 \\ 8 \\ 12 \end{bmatrix} - \begin{bmatrix} 8.857142857 \\ 5 \\ 3.428571428 \\ 4.142857142 \\ 7.142857143 \\ 12.42857143 \end{bmatrix} = \begin{bmatrix} 0.1428571143 \\ 0 \\ -0.428571428 \\ -0.142857142 \\ 0.857142857 \\ -0.42857143 \end{bmatrix}$

$\Longrightarrow \boxed{\hat{r} = \begin{bmatrix} 0.1428571143 \\ 0 \\ -0.428571428 \\ -0.142857142 \\ 0.857142857 \\ -0.42857143 \end{bmatrix}}$ [RESIDUAL] & $\boxed{\|\hat{r}\|_\infty = 0.857142857}$ [MAGNITUDE w.r.t. Max-norm]

$\longrightarrow \hat{r}^T = \begin{bmatrix} 0.1428571143 & 0 & -0.428571428 & -0.142857142 & 0.857142857 & -0.42857143 \end{bmatrix}$

$\hat{r}^T\hat{r} = \begin{bmatrix} 0.1428571143 & 0 & -0.428571428 & -0.142857142 & 0.857142857 & -0.42857143 \end{bmatrix} \begin{bmatrix} 0.1428571143 \\ 0 \\ -0.428571428 \\ -0.142857142 \\ 0.857142857 \\ -0.42857143 \end{bmatrix}$

$= 1.142857143 = \lambda$

$\|\hat{r}\|_2 = \sqrt{\lambda}$

$= \sqrt{1.142857143} = 1.069044968$ $\Longrightarrow \boxed{\|\hat{r}\|_2 = 1.069044968}$ [MAGNITUDE w.r.t. 2-norm]

e.) Confirm your work in MATLAB.

— NOTE: you can solve the NORMAL EQNS. using your function from #3b since A^TA is a symmetric positive definite matrix

→ Plot your data & your Least Squares polynomial.

$\Longrightarrow$ From MATLAB :

$$\hat{r} = \begin{bmatrix} 0.142857142857142 \\ -0.000000000000001 \\ -0.428571428571429 \\ -0.142857142857143 \\ 0.857142857142857 \\ -0.428571428571427 \end{bmatrix}$$

$\longrightarrow \|\hat{r}\|_\infty = 0.857142857142857$

$\longrightarrow \|\hat{r}\|_2 = 1.069044967649697$

$\longrightarrow$ Rel. $\text{Diff}_\infty = 6.476300977 \times 10^{-16} \Longrightarrow$ Rel. $\text{Diff}_\infty = O(10^{-16})$

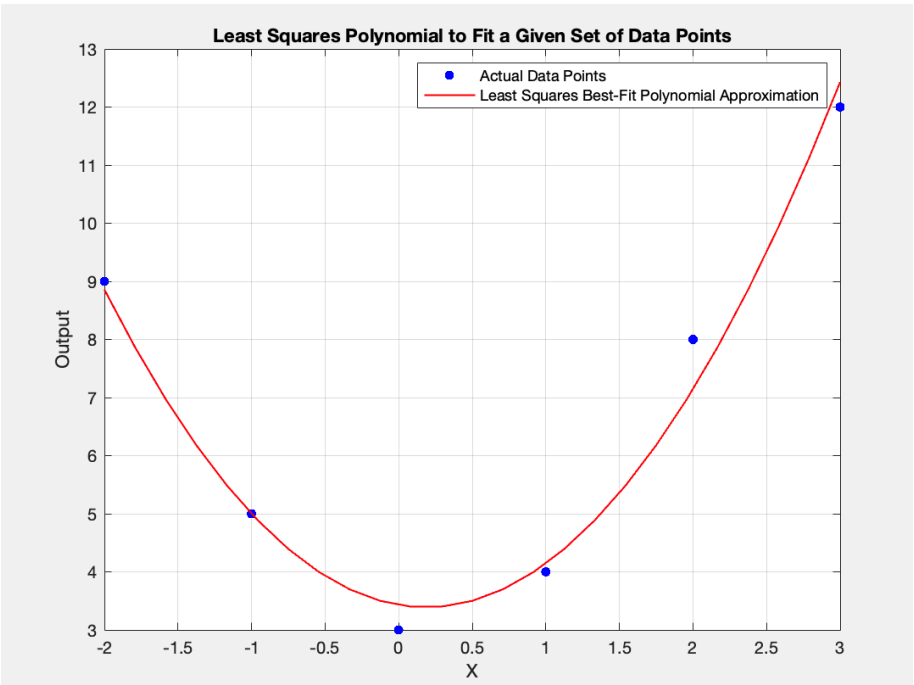
$\longrightarrow K(A)_\infty = 78.19999 \approx 80$

$= 8 \times 10^1 \approx 10 \times 10^1$

$= 10^2 \Longrightarrow K(A)_\infty = O(10^2)$

To confirm my work in MATLAB, the process is similar as how we solved the linear system $A\underline{x} = \underline{b}$ in #3b. The solution from MATLAB was determined using Cholesky Factorization. For Cholesky Factorization, it involves Gaussian Elimination without pivoting since the pivots of the lower and upper triangular matrices as a result of LU Decomposition are all positive. To use Cholesky Factorization, we formed a symmetric positive definite matrix M using the relationship $M = A^TA$, where matrix A was determined using the given set of data points and converting it to a 2^{nd} – order polynomial in matrix form. We also formed a similar vector $\underline{b}$ using the relationship $A^T\underline{y}$. This system $M\underline{c} = \underline{b}$ or $A^TA\underline{c} = A^T\underline{y}$ solves the normal equations, which determines the coefficients of the Least Squares best – fit polynomial ($\underline{c}$) using Cholesky Factorization followed by forward and backward substitution. The coefficients calculated using my function is nearly the same as MATLAB's calculated solution using the backslash command. The relative difference is $O(10^{-16})$, which is exactly ϵ_{mach}. This indicates both my algorithm and MATLAB's algorithm are essentially agreeing on all 16 digits. Since the algorithm of my function computes in a similar manner as MATLAB's function, the relative difference between both solutions is essentially down to ϵ_{mach}. I checked the condition number of the symmetric positive definite matrix M with respect to max – norm and got $O(10^2)$. This indicates that we can lose about 2 digits of accuracy, which indicates the size of the true relative error is $O(10^{-14})$. This is the size of the relative error we expect to get if we were able to

solve for the true exact relative error. Nonetheless, this proves the function we created is working properly, such that it outputs a good approximation of the coefficients for the Least Squares best-fit polynomial. We also computed the residual and its magnitude with respect to max-norm and 2-norm, which are shown above. Comparing both solutions, my pen-and-paper work matches the output from MATLAB.
The plot shows the data points in BLUE and the Least Squares Best-Fit polynomial in RED. From the plot, the curve of the Least Squares polynomial is a reasonable fit to the given set of data points as the curve passes through majority of the given data points and is relatively "close" to all the data points. Thus, the calculated Least Squares polynomial $[\, p(x) = 3.428571428 - 0.4285714287x + 1.142857143x^2 \,]$ is the best-fitted function to approximate the given set of data points.



→ MATLAB : submitted **Math146_PS4_Q4.m** file

→ submitted **Cholesky_Factor.m** file ⟹ PART of file :

submitted : **forward_sub.m** file
backward_sub.m file