Table of Contents

Housekeeping

```
clear all; close all; clc;

%{

    CSCI HW7
    Author: Connor O'Reilly
    Last Edited: 10/22:2021
    Email: coor1752@colorado.edu

%}

%create loop to change theta value
```

Part 1: Generating training data

```
%initializing function
f = @(x, thet) 1./( 1 + exp(-thet .* x));
n = 7;

%vector with n evenly spaced points on [-5,5] and initializing theta
 to
% one
theta = 1;
x_train = linspace(-5, 5, n);
y_train1 = f(x_train, theta);
y_train1 = y_train1.';

theta = 10;
y_train10 = f(x_train, theta);
y_train10 = y_train10.';
```

Part 2: Training the model

```
%p(x) = c(1) * x^5 + c(2) *x ^
```

```
%create vandermonde matrix, theta = 1
%initialize
V1 = zeros(n,n);
for i = 1:n
    V1(i,:) = x_train(i);
end

%raise elements to power
for i = 0:n-1
    V1(: , i + 1) = V1(:, i + 1).^i;
end

%determine cond number
condition_V1 = cond(V1);
if(condition_V1 <= eps)
    error('Matrix is ill conditioned')
end

%create vandermonde matrix, theta = 10
%initialize
V10 = zeros(n,n);
for i = 1:n
    V10(i,:) = x_train(i);
end
%raise elements to power
for i = 0:n-1
    V10(: , i + 1) = V10(:, i + 1).^i;
end

%determine cond number
condition_V10 = cond(V10);
if(condition_V10 <= eps)
    error('Matrix is ill conditioned')
end

%use code from HW5 to solve for coefficients, theta = 1
theta = 1;
c_theta1 = LU_or_Chol(V1, y_train1);
c_theta1 = flip(c_theta1);

%initialize new grid space for results
x_part2 = linspace(-5, 5, 1000);
y_f1 = f(x_part2, theta);
y_p1 = polyval(c_theta1, x_part2);

%use code from HW5 to solve for coefficients, theta = 10
theta = 10;
c_theta10 = LU_or_Chol(V10, y_train10);
c_theta10 = flip(c_theta10);

y_f10 = f(x_part2, theta);
y_p10 = polyval(c_theta10, x_part2);
```

Part 3: Generating Testing Data

```
%new vector with 101 evenly spacedd points in [-5, 5], x'
x_part3 = linspace(-5, 5, 101);

%compute y_i' = f_theta(x_i'), theta = 1
theta = 1;
y_part31 = f(x_part3, theta);

%compute the mean of y_i'
mean_theta1 = mean(y_part31, 'all');

%compute the standard deviation of y_i'
STD_theta1 = std(y_part31);

%compute y_i' = f_theta(x_i'), theta = 10
theta = 10;
y_part310 = f(x_part3, theta);

%compute the mean of y_i'
mean_theta10 = mean(y_part310, 'all');

%compute the standard deviation of y_i'
STD_theta10 = std(y_part310);
```

Part 4: Computing testing error

```
%evaluate polynomial, theta = 1
p_error1 = polyval(c_theta1, x_part3);

abs_error1 = max( abs(y_part31 - p_error1)./abs(y_part31) );

%evaluate polynomial, theta = 10
p_error10 = polyval(c_theta10, x_part3);

abs_error10 = max( abs(y_part310 - p_error10)./abs(y_part310) );
```

Extra Credit

```
for n = 8:15
```

Part 3: Generating Testing Data

```
    %new vector with 101 evenly spacedd points in [-5, 5], x'
    x_part3extra = linspace(-5, 5, n);

    %compute y_i' = f_theta(x_i'), theta = 1
    theta = 1;
    y_part31extra = f(x_part3extra, theta);
```

```
    %compute y_i' = f_theta(x_i'), theta = 10
    theta = 10;
    y_part310extra = f(x_part3extra, theta);
```

Part 4: Computing testing error

```
    %evaluate polynomial, theta = 1
    p_errorextra = polyval(c_theta1, x_part3extra);

    abs_error1extra(n - 7) = max( abs(y_part31extra - p_errorextra)./
abs(y_part31extra) );

    %evaluate polynomnal, theta = 10
    p_error10extra = polyval(c_theta10, x_part3extra);

    abs_error10extra(n - 7) = max( abs(y_part310extra -
 p_error10extra)./abs(y_part310extra) );

end
```

Plotting

```
n = 7;
%Part 1, theta = 1
fprintf('\n-----------------------------------------------------------------
\n')
fprintf('Part 1: Generating training data')
fprintf('\n-----------------------------------------------------------------
\n\n')
fprintf('theta = 1\n')
fprintf('      xi        |     f_theta(xi)\n')
fprintf('-----------------------------\n')
for i = 1:n
    fprintf('  %0.4f   |   %0.4f  \n', x_train(i), y_train1(i));
end
fprintf('\n');

%Part 1, theta = 10
fprintf('theta = 10\n')
fprintf('      xi        |     f_theta(xi) \n')
fprintf('-----------------------------\n')
for i = 1:n
    fprintf('  %0.4f   |   %0.4e  \n', x_train(i), y_train10(i));
end


%Part 2: plot of f_theta and p_p
%theta = 1
figure(1)
scatter(x_train, y_train1,'Linewidth',1.25);
hold on;
plot(x_part2, y_p1, 'r', 'Linewidth', 1.5)
```

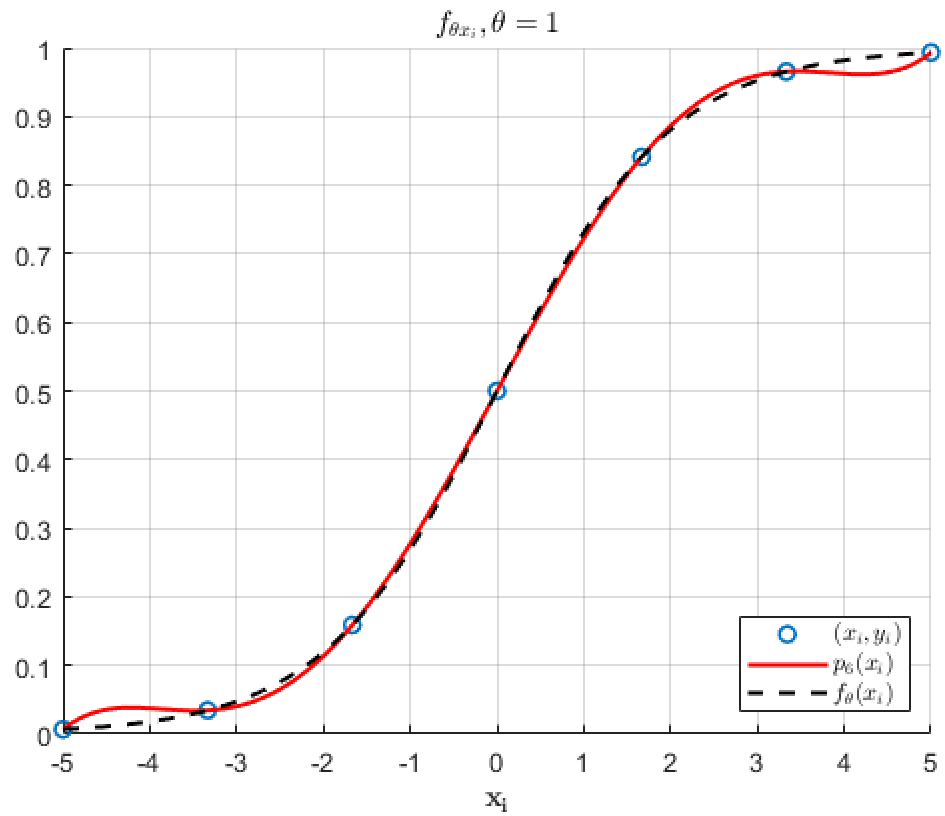
```
plot(x_part2, y_f1, 'k--', 'Linewidth', 1.5);
grid on;
xlabel('$\mathbf{x_i}$','Interpreter','latex');
legend('$(x_i,y_i)$', '$p_6(x_i)$', '$f_{\theta}(x_i)$'
 ,'Interpreter','latex', 'Location', 'southeast');
title(' $f_{\theta x_i},\theta = 1$','Interpreter','latex')
hold off;

%theta = 10
figure(2)
scatter(x_train, y_train10,'Linewidth',1.25);
hold on;
plot(x_part2, y_p10, 'r', 'Linewidth', 1.5)
plot(x_part2, y_f10, 'k--', 'Linewidth', 1.5);
grid on;
xlabel('$\mathbf{x_i}$','Interpreter','latex');
legend('$(x_i,y_i)$', '$p_6(x_i)$', '$f_{\theta}(x_i)$'
 ,'Interpreter','latex', 'Location', 'southeast');
title(' $f_{\theta x_i},\theta = 10$','Interpreter','latex')
hold off;

%Part 2: table of coefficients
fprintf('\n-----------------------------------------------------------------
\n')
fprintf('Part 2: Training the model')
fprintf('\n-----------------------------------------------------------------
\n\n')

%theta = 1
fprintf('theta = 1\n')
fprintf('   Table of ci coefficients \n')
fprintf('-----------------------------\n')
for i = 1:n
    fprintf(' c%i = %0.3e \n', i-1 , c_theta1(i));
end

%theta = 10
fprintf('theta = 10\n')
fprintf('   Table of ci coefficients \n')
fprintf('-----------------------------\n')
for i = 1:n
    fprintf(' c%i = %0.3e \n', i - 1, c_theta10(i));
end


%Part 2: Assessment
fprintf('\nPart 2: Assessment: Although the vandermonde system is
 ill-condditioned (When Theta = 1),\n the approximation is adequate
 and i would consider accurate. The vandermonde system accurately
 approximates the function on the interval of -3.3333 to 3.3333 \n but
 loses accuracy towards the endpoints shown as a minor blow up but due
 to its shallow slope adding more equally points would increase the
 accuray of the approximation.  \n')
```

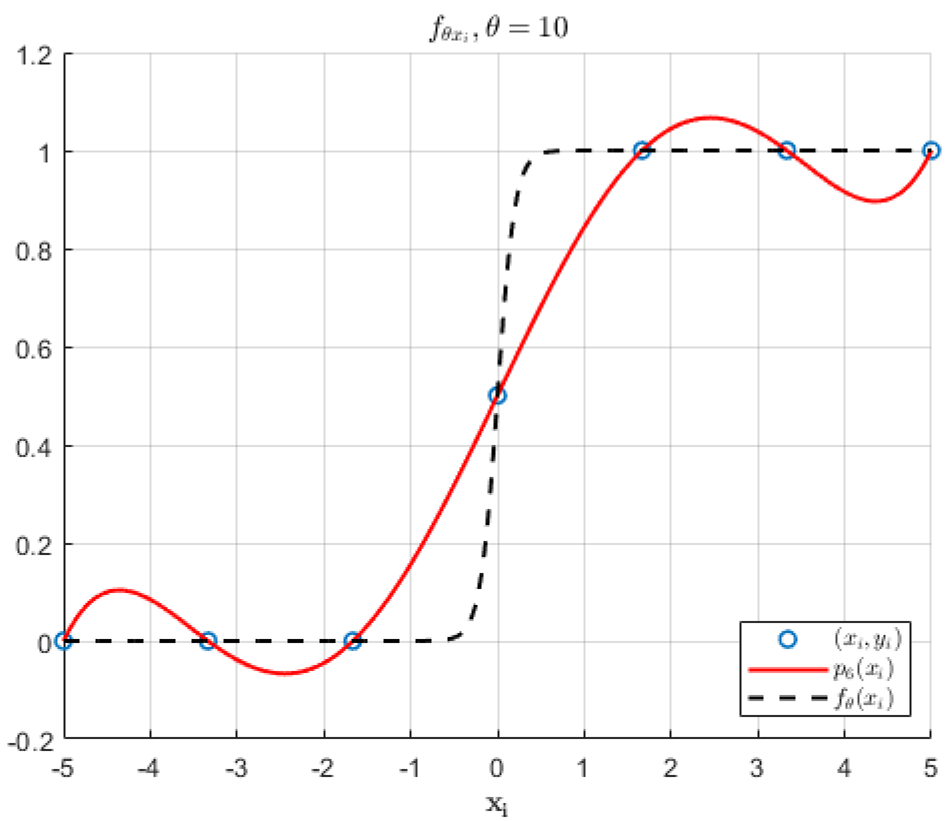
```
%Part 3:
fprintf('\n-----------------------------------------------------------------
\n')
fprintf('Part 3: Generating testing data')
fprintf('\n-----------------------------------------------------------------
\n\n')

%theta = 1
fprintf('theta = 1\n')
fprintf(' Mean from the set of points y_i'', mean = %0.04f\n ',
 mean_theta1);
fprintf(' STD from the set of points y_i'', mean = %0.04f\n ',
 STD_theta1);

%theta = 10
fprintf('\ntheta = 10\n')
fprintf(' Mean from the set of points y_i'', mean = %0.04f\n ',
 mean_theta10);
fprintf(' STD from the set of points y_i'', mean = %0.04f\n ',
 STD_theta10);

%Part 4:
fprintf('\n-----------------------------------------------------------------
\n')
fprintf('Part 4: Computing the testing error')
fprintf('\n-----------------------------------------------------------------
\n\n')

%theta = 1
fprintf('theta = 1\n')
fprintf('Absolute testing error = %0.4e\n\n',abs_error1)

%theta = 10
fprintf('theta = 10\n')
fprintf('Absolute testing error = %0.4e\n\n',abs_error10)

%Part 4: approximation
fprintf('Part 4 Assessment: as stated in lecture, if xi are equally
 spaced the vandermonde system is ill conditioned. \n\n It is not
 a problem for the smooth function when theta = 1, but the error
 explodes when theta = 10 shown by the blown upn oscillations towards
 the end points which is a good example of the Runge''s Phenomenom,
 \n to fix problem for when theta = 10 we can use chebyshev points to
 increase the density of points towards the endpoints.')
fprintf(' to accurately depict the steep slope of the theta = 10
 function, i cant explain it well. but as theta increases, the harder
 it is to fit a higher degree polynomial to the function due to the
 increase of oscillations towards the endpoints');

%extra credit
```

```
-----------------------------------------------------------------
Part 1: Generating training data
```

```
-----------------------------------------------------------------

theta = 1
      xi        |     f_theta(xi)
-----------------------------
  -5.0000   |   0.0067
  -3.3333   |   0.0344
  -1.6667   |   0.1589
  0.0000   |   0.5000
  1.6667   |   0.8411
  3.3333   |   0.9656
  5.0000   |   0.9933

theta = 10
      xi        |     f_theta(xi)
-----------------------------
  -5.0000   |   1.9287e-22
  -3.3333   |   3.3382e-15
  -1.6667   |   5.7777e-08
  0.0000   |   5.0000e-01
  1.6667   |   1.0000e+00
  3.3333   |   1.0000e+00
  5.0000   |   1.0000e+00

-----------------------------------------------------------------
Part 2: Training the model
-----------------------------------------------------------------

theta = 1
   Table of ci coefficients
-----------------------------
 c0 = -9.344e-20
 c1 = 2.182e-04
 c2 = 4.514e-18
 c3 = -1.083e-02
 c4 = -5.224e-17
 c5 = 2.331e-01
 c6 = 5.000e-01
theta = 10
   Table of ci coefficients
-----------------------------
 c0 = -3.453e-19
 c1 = 6.480e-04
 c2 = 1.464e-17
 c3 = -2.700e-02
 c4 = -1.500e-16
 c5 = 3.700e-01
 c6 = 5.000e-01

Part 2: Assessment: Although the vandermonde system is ill-
condditioned (When Theta = 1),
 the approximation is adequate and i would consider accurate. The
 vandermonde system accurately approximates the function on the
 interval of -3.3333 to 3.3333
```

```
 but loses accuracy towards the endpoints shown as a minor blow up but
 due to its shallow slope adding more equally points would increase
 the accuray of the approximation.

------------------------------------------------------------------
Part 3: Generating testing data
------------------------------------------------------------------

theta = 1
 Mean from the set of points y_i', mean = 0.5000
  STD from the set of points y_i', mean = 0.3921

theta = 10
 Mean from the set of points y_i', mean = 0.5000
  STD from the set of points y_i', mean = 0.4924

------------------------------------------------------------------
Part 4: Computing the testing error
------------------------------------------------------------------

theta = 1
Absolute testing error = 2.2898e+00

theta = 10
Absolute testing error = 6.3101e+19

Part 4 Assessment: as stated in lecture, if xi are equally spaced the
 vandermonde system is ill conditioned.

 It is not a problem for the smooth function when theta = 1, but the
 error explodes when theta = 10 shown by the blown upn oscillations
 towards the end points which is a good example of the Runge's
 Phenomenom,
 to fix problem for when theta = 10 we can use chebyshev points to
 increase the density of points towards the endpoints. to accurately
 depict the steep slope of the theta = 10 function, i cant explain
 it well. but as theta increases, the harder it is to fit a higher
 degree polynomial to the function due to the increase of oscillations
 towards the endpoints
```

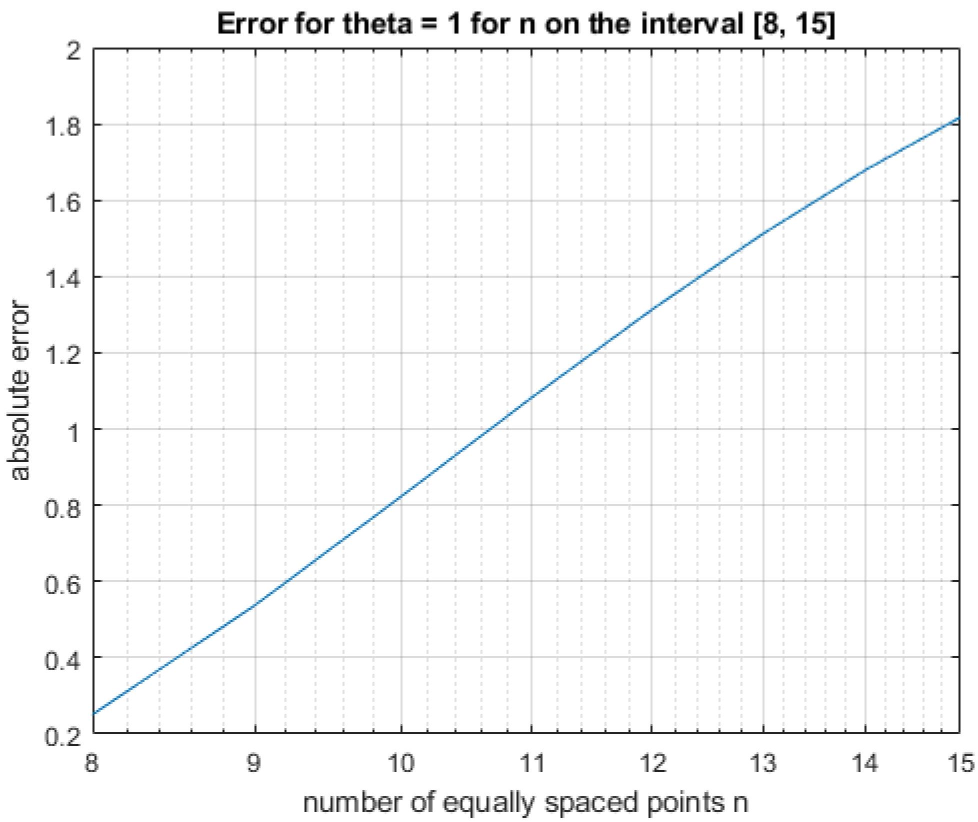

$f_{\theta x_i}, \theta = 1$
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
-5
-4
-3
-2
-1
0
1
2
3
4
5
$\mathbf{x_i}$
(x_i, y_i)
$p_6(x_i)$
$f_\theta(x_i)$

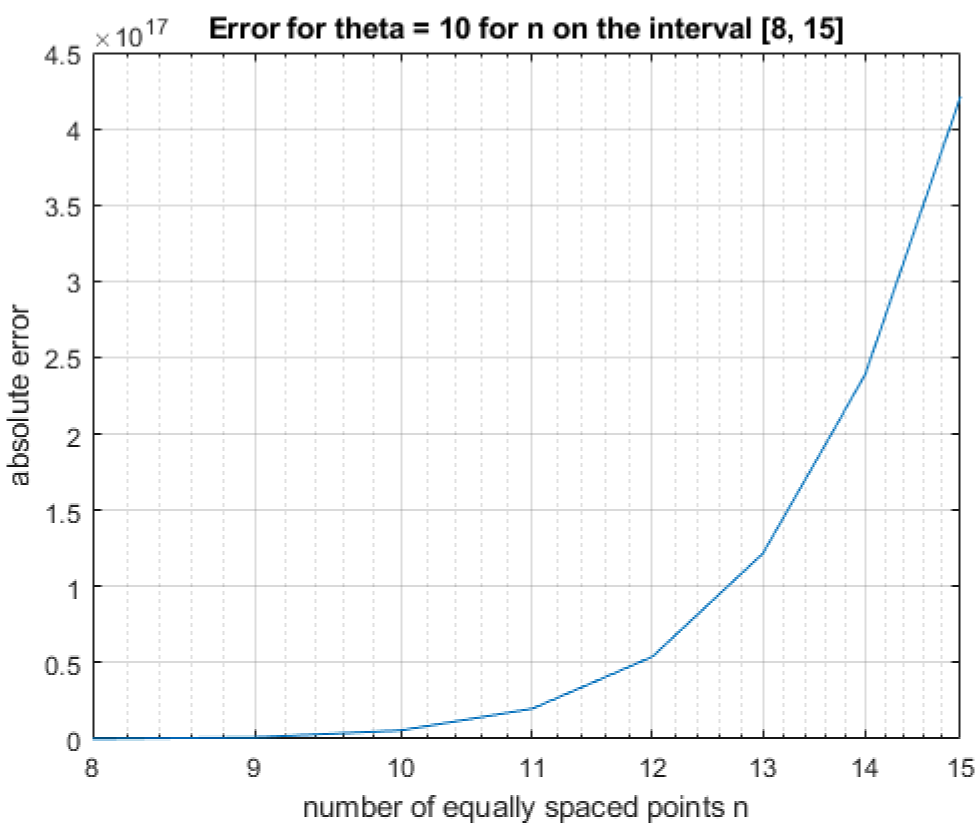

$f_{\theta x_i}, \theta = 10$
1.2
1
0.8
0.6
0.4
0.2
0
-0.2
-5
-4
-3
-2
-1
0
1
2
3
4
5
$\mathbf{x_i}$
(x_i, y_i)
$p_6(x_i)$
$f_\theta(x_i)$

this was rushed my bad

```
figure(3)
semilogx(8:15, abs_error1extra)
hold on;
grid on;
title('Error for theta = 1 for n on the interval [8, 15]')
xlabel('number of equally spaced points n')
ylabel('absolute error')
hold off;

figure(4)
semilogx(8:15, abs_error10extra)
hold on;
grid on;
title('Error for theta = 10 for n on the interval [8, 15]')
xlabel('number of equally spaced points n')
ylabel('absolute error')
hold off;

fprintf('\n-----------------------------------------------------------------
\n')
fprintf('Extra Credit: ')
fprintf('\n-----------------------------------------------------------------
\n\n')
fprintf('the assesment of this part was rushed last minute, but
 for the function when theta = 10  the plot does not shor the error
 converging but continuosly increasing exponentially as n increasse
 due to the runge''s phenonenom, but looking at the error plot for
 when theta = 1, the error converges quadratically towards 2 as n
 increases')
```

```
-----------------------------------------------------------------
Extra Credit:
-----------------------------------------------------------------

the assesment of this part was rushed last minute, but for the
 function when theta = 10  the plot does not shor the error converging
 but continuosly increasing exponentially as n increasse due to the
 runge's phenonenom, but looking at the error plot for when theta = 1,
 the error converges quadratically towards 2 as n increases
```


Error for theta = 1 for n on the interval [8, 15]
absolute error
number of equally spaced points n
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
8
9
10
11
12
13
14
15


Error for theta = 10 for n on the interval [8, 15]
×10^17
absolute error
number of equally spaced points n
4.5
4
3.5
3
2.5
2
1.5
1
0.5
0
8
9
10
11
12
13
14
15

Functions

```
%part 2

function [exe, chol_bool] = LU_or_Chol(mat, RHS)
%{
    Purpose: solves Ax = b using LU decomposition or cholesky
 factorization
    depending on whether the matrix is symmetric or not

    Inputs:
        mat: A  matrix
        RHS: right hand side

    Outputs:
        exe: matrix of computed solution x
        chol_cool: boolean determining if LU decomposition or Cholesky
        factorization was used
%}

    tf = issymmetric(mat);
    chol_bool = false;
    if(tf)
        d = eig(mat);
        chol_bool  = all(d > 0);
    end

    if(chol_bool)
        %if matrix is positive definite use Cholesky factorization to
 solve
        %Ax = b
        R = chol(mat);
        exe = R\(R'\RHS);

    else
        %if matrix is not positive definite use LU decomposition
        [L, U, P] = lu(mat);
        y = L\(P*RHS);
        exe = U\y;

    end

end
```

Published with MATLAB® R2021a