Brian KYANJO

Home work 4

Math 566

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 $b \in \mathbb{R}^m$

a) Show that the x that minimizes $\|Ax - b\|_2$ is given by the solution to the square linear system.

$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix} \begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}$$

Given $$\mathcal{A} = \begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix}$$

The 2-by-2 block Gaussian Elimination of $\mathcal{A}$ will be

$$\begin{matrix} R_1 \\ R_2 \end{matrix}\left[\begin{array}{cc|c} I & A & b \\ A^T & 0 & 0 \end{array}\right] \xrightarrow[A^T R_1 - R_2 \to R_2]{} \left[\begin{array}{cc|c} I & A & b \\ 0 & A^TA & A^Tb \end{array}\right]$$

thus

$$\begin{bmatrix} I & A \\ 0 & A^TA \end{bmatrix} \begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ A^Tb \end{bmatrix}$$

$$Ir + Ax = b$$
$$A^TAx = A^Tb$$

$x = (A^TA)^{-1} A^Tb$ is the solution that minimizes $\|Ax - b\|_2$.

then $Ir + Ax = b$

$r + Ax = b \Rightarrow r = b - Ax$, which is the residual of $Ax = b$.

b) Determine the 2-norm condition number of A in terms of the singular values of A

Let $A = U\Sigma V^T$ be full SVD of A,

Consider

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

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

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

Computing Eigenvalues of $\tilde{A}$ and relate those to singular values of A

$$|\tilde{A} - \lambda I| = 0$$

$$\begin{vmatrix} I - \lambda I & \Sigma \\ \Sigma^T & -\lambda I \end{vmatrix} = 0$$

Given a 2X2 block matrix $M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$ where $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times m}$, $C \in \mathbb{R}^{m \times n}$ and, $D \in \mathbb{R}^{m \times m}$

If D is invertible then

$$\det(M) = \det(A - BD^{-1}C)\det(D)$$

Since

$$(1-\lambda)I \in \mathbb{R}^{n \times n}$$

$$\Sigma \in \mathbb{R}^{n \times m}, \quad \Sigma^T \in \mathbb{R}^{m \times n}$$

$$-\lambda I \in \mathbb{R}^{m \times m} \text{ and invertible}$$

then

$$\begin{vmatrix} I-\lambda I & \Sigma \\ \Sigma^T & -\lambda I \end{vmatrix} = \det\left((1-\lambda)I - \Sigma(-\lambda I)^{-1}\Sigma^T\right)\det(-\lambda I)$$

Since $\Sigma = \Sigma^T$

$$\det\left((1-\lambda)I + \lambda^{-1}\Sigma^2\right)\det(-\lambda I) = 0$$

$$\det\left(\lambda^{-1}(\lambda(1-\lambda)I + \Sigma^2)\right)\det(-\lambda I) = 0$$

$$\det(-\lambda I) = (-\lambda)^m$$

$$(-\lambda)^m(\lambda)^{-m}\det\left(\lambda(1-\lambda)I + \Sigma^2\right) = 0$$

$$(-1)^m \det\left(\lambda(1-\lambda)I + \Sigma^2\right) = 0$$

$$\det\left(\lambda(\lambda-1)I - \Sigma^2\right) = 0$$

$$\Sigma^2 = \begin{bmatrix} \sigma_1^2 & & & & \\ & \sigma_2^2 & & & \\ & & \ddots & & \\ & & & \sigma_n^2 & \\ & & & & 0 \\ & & & & & \ddots \end{bmatrix}$$

$$\det\left(\lambda(\lambda-1) I - \Sigma^2\right) = 0$$

$$\left| \begin{bmatrix} \lambda(\lambda-1) & & & \\ & \lambda(\lambda-1) & & 0 \\ & & \ddots & \\ & 0 & & \lambda(\lambda-1) \end{bmatrix} - \begin{bmatrix} \sigma_1^2 & & & & \\ & \sigma_2^2 & & & 0 \\ & & \ddots & & \\ & & & \sigma_n^2 & \\ & 0 & & & 0 \\ & & & & & \ddots \end{bmatrix} \right| = 0$$

$$\left| \begin{bmatrix} \lambda(\lambda-1)-\sigma_1^2 & & & & \\ & \lambda(\lambda-1)-\sigma_2^2 & & & \\ & & \ddots & & \\ & & & \lambda(\lambda-1)-\sigma_n^2 & \\ & & & & \lambda(\lambda-1) \\ & & & & & \ddots \end{bmatrix} \right| = 0$$

$$\lambda(\lambda-1) - \sigma_1^2 = \cdots = \lambda(\lambda-1) - \sigma_n^2 = 0$$

and

$$\lambda(\lambda-1) = 0$$

Hence solving for λ,

$$\lambda^2 - \lambda - \sigma_i^2 = 0, \quad i = 1, 2, 3, \cdots, n$$

$$\lambda = \frac{1 \pm \sqrt{1 + 4\sigma_i^2}}{2}$$

$\lambda^2 - \lambda = 0 \Rightarrow \lambda = 1$ or $\lambda = 0$

The 2-norm Condition number $K(A)$

$$K(A) = \frac{\| \text{maximum Eigen Value} \|}{\| \text{minimum Eigen Value} \|}$$

$$K(A) = \left| \frac{1 + \sqrt{1 + 4\sigma_1^2}}{1 - \sqrt{1 + 4\sigma_n^2}} \right|$$

$$K(A) = \frac{1 + \sqrt{1 + 4\sigma_1^2}}{-1 + \sqrt{1 + 4\sigma_n^2}}$$

c) How does the Condition number of A Compare to the Condition number of the normal Equation Matrix A^TA?

For the Equation Matrix, the Condition

$K(A^TA)$

$$K(A^TA) = \frac{\sigma_1^2}{\sigma_n^2}$$

where is different

$$K = \frac{1 + \sqrt{1 + 4\sigma_1^2}}{-1 + \sqrt{1 + 4\sigma_n^2}}$$

d) Compute the polynomial coefficients from problem 1 of the homework

4(c) Show that growth for the matrix A (but for the general n-by-n case) is exactly

$$g_n(A) = 2^{n-1}$$

The growth factor of the nxn matrix A is given by

$$\rho = \frac{\max_{i,j} |u_{ij}|}{\max_{i,j} |a_{ij}|}$$

with U the upper triangular Matrix obtained from A by performing LU decomposition of A

u_{ij} and a_{ij} are respective Entries of the upper triangular matrix U and Matrix A.

Consider for $n=3$, $A \in \mathbb{R}^{3\times 3}$

$$A = \begin{bmatrix} 1 & 0 & 1 \\ -1 & 1 & 1 \\ -1 & -1 & 1 \end{bmatrix}$$

Performing Gaussian Elimination on A, we obtain

$$A = LU = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 4 \end{bmatrix}$$

$$U = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 2 \\ 0 & 0 & 4 \end{bmatrix}$$

The maximum Entry in U is 4, and in A is 1 so the growth factor.

$$\rho = \frac{4}{1} = 4 = 2^{3-1} = 2^{n-1}$$

for $n=4$, $A \in \mathbb{R}^{4\times 4}$

$$A = \begin{bmatrix} 1 & 0 & 0 & 1 \\ -1 & 1 & 0 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & -1 & -1 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & -1 & 1 & 0 \\ -1 & -1 & -1 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 2 \\ 0 & 0 & 1 & 4 \\ 0 & 0 & 0 & 8 \end{bmatrix}$$

L $\qquad$ U

$$\max |u_{ij}| = 8 \quad , \max |a_{ij}| = 1$$

$$\rho = \frac{8}{1} = 2^{4-1} = 2^{n-1}$$

Since it's true for the base case, Assume it's true for $n=k-1$, $A \in \mathbb{R}^{(k-1)\times(k-1)}$

$$A = \begin{bmatrix} 1 & 0 & \cdots & 1 \\ -1 & 1 & 0 & \cdots 1 \\ \vdots & & \ddots & \vdots \\ -1 & \cdots & -1 & 1 \end{bmatrix} \rightarrow \begin{bmatrix} 1 & 0 & \cdots & 0 \\ -1 & 1 & 0 & \cdots 0 \\ \vdots & & \ddots & \vdots \\ -1 & \cdots & -1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & \cdots & 0 & 1 \\ & 1 & & & 2^1 \\ & & \ddots & & 2^2 \\ & & & 1 & \vdots \\ & & & & 2^{k-2} \end{bmatrix}$$

$$\max |u_{ij}| = 2^{k-2}$$

$$\max |a_{ij}| = 1$$

$$\rho = \frac{2^{k-2}}{1} = 2^{k-2} = 2^{(k-1)-1}$$

If it true for $n=k-1$, then it true for $n=k$, then the maximum entry, ~~Max~~ $|u_{ij}|$

The maximum $|u_{ij}| = 2^{k-1}$, and $\text{Max}|a_{ij}| = 1$

$$\rho = \frac{\text{Max}|u_{ij}|}{\text{Max}|a_{ij}|} = 2^{k-1}$$

Therefore the growth factor for nxn Matrix A is exactly 2^{n-1}

Have proved by induction.

3. (Unit Lower triangular Matrix)

Given a matrix $A \in \mathbb{C}^{m \times n}$

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2n} \\ \vdots & & & & \vdots \\ a_{m1} & & & & a_{mn} \end{bmatrix}$$

The LU decomposition of A is given by $LU = A$ where:

L is a unit Lower triangular matrix with ones on the diagonal.

U is the Upper triangular Matrix.

- The multipliers, L_{ji}, accumulated into the Lower triangular part with a Change of sign

$$L_{ji} = \frac{a_{ji}}{a_{ii}}, \quad j = i+1, \cdots, n$$

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

Let

$$l_i = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ l_{i+1,i} \\ \vdots \\ l_{n,i} \end{bmatrix}$$

$$L_i = \underbrace{\begin{bmatrix} 1 & & & & \\ & 1 & & 0 & \\ & & \ddots & & \\ & & & 1 & \\ & 0 & & & \ddots \\ & & & & & 1 \end{bmatrix}}_{I} - \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ l_{i+1,i} \\ \vdots \\ l_{n,i} \end{bmatrix}}_{l_i} \underbrace{\begin{bmatrix} 0 & 0 & \cdots & 0 \\ & & \ddots & \\ & & & 0 \\ & & & 1 \\ & & & 1 \\ & & & \vdots \\ 0 & 1 & & 0 \end{bmatrix}}_{e_i'}$$

Elementwise multiplication

$$L_i = I - l_i e_i'$$

Since L_i is Unitary then $L_i L_i^{-1} = I$

$$(I - L_i e_i')(I - l_i e_i')^{-1} = (I - l_i e_i')(I + l_i e_i')$$

$$= I - I + I l_i e_i' - l_i e_i' I - l_i e_i' l_i e_i'$$

$$= I - l_i e_i' l_i e_i = I$$

The element wise multiplication of $e_i' l_i = 0$, since the only zero entry in e_i', which is i^{th} entry multiplies with zero entries in l_i,

Since non zero entries in L_i, start at $i+1$, first
This implies that for any i,

$$L_i^{-1} = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & l_{i+1,i} & 1 & \\ & & \vdots & & \ddots \\ & & l_{n,i} & & 1 \end{bmatrix}$$

Therefore mathematically the result is true.

N0.1d

```
clear all;
close all;
%m equally spaced points over [0,1]
m = 50; n=12;

% Vandermonde matrix t
t = zeros(m,n);
for i = 1:n
    for j = 1:m
        t(j,i) = ((j-1)/(m-1))^(n-i);
    end
end

%fliping the vandermonde matrix t to form A
A = fliplr(t);

%fuction f
tj = zeros(m,1);
for j = 1:m
    tj(j) = (j-1)/(m-1);
end

f = cos(4*tj);

format long
%block matrix Ablock
Ablock = [eye(m) A; A' zeros(n)];
b = [f;zeros(n,1)];
[qb,rb] = qr(Ablock); xb = rb\(qb'*b);

%solution x, extracted from xb
xblock = xb(m+1:end)

%solution r
fprintf('Solution of r\n\n');
r = xb(1:m);

%(a). normal equations
x = (A'*A)\(A'*f);


%(e). QR decomposition using inbuilt Householder
[q,r] = qr(A); xh = r\(q'*f);

Table = table(x,xh,xblock, 'VariableNames',{'Normal equation','Builtin function','Block Matrix'})

residual = norm((xh - xblock),2);

fprintf('The solution, xblock, solution is almost similar to the computed solutions although some little values are off. \n And the residual %f is sma
```

```
xblock =

   1.000000003176009
  -0.000001053361969
  -7.999957773466207
  -0.000661681286866
  10.672037092264638
  -0.025505465801884
  -5.614140276352710
  -0.135309892771110
   1.764148110435273
  -0.045663530262449
  -0.352769283842230
   0.084180130494221

Solution of r

Warning: Matrix is close to singular or badly scaled. Results may be inaccurate.
RCOND =  2.800825e-17.

Table =

  12×3 table

     Normal equation         Builtin function          Block Matrix
    ___________________    ____________________    ___________________

       0.999999989587329         1.00000000099661         1.00000000317601
    2.84029396133336e-06    -4.22742915903599e-07    -1.05336196876486e-06
       -8.00010214560535        -7.99998123568936        -7.99995777346621
      0.00144399781030622    -0.000318763182123933    -0.000661681286865863
         10.6560552366266         10.6694307955344         10.6720370922646
        0.0460832179149034      -0.0138202863975713      -0.0255054658018845
        -5.81579888120301        -5.64707563175731        -5.61414027635271
        0.231892082482534       -0.0753160164200601       -0.13530989277111
         1.33247545357655         1.69360695433438         1.76414811043527
        0.270659249202885       0.00603211536110353      -0.0456635302624494
```

```
       -0.484158233866101       -0.374241706147935        -0.35276928384223
        0.107803579981325       0.0880405765490434        0.084180130494221

 The solution, xblock, solution is almost similar to the computed solutions although some little values are off.
  And the residual 0.113807 is small.
```

Published with MATLAB® R2020b

```
clear all
close all


%Rank-deficient problems and regularization
%Consider a 2pi periodic signal x(t) sampled at the N equally spaced points
%tj = hj

N = 256;
delta = 0.1;

j = [0:N-1]';
h = 2*pi/N;
tj = h*j;

%periodic version of the Gaussian function
k = @(t,delta) (1/(delta*sqrt(2*pi)))*exp(-(2-(2*cos(t)))/(2*delta^2));

aj = h*k(tj,delta);

fprintf('2a). Construct A and compute its condition number. \n\n');
```

```
2a). Construct A and compute its condition number.
```

```
%matrix A
A = circulant([aj]);

%condition number
kapa = cond(A);
fprintf('The condition number of A is: %e \n\n',kapa);
```

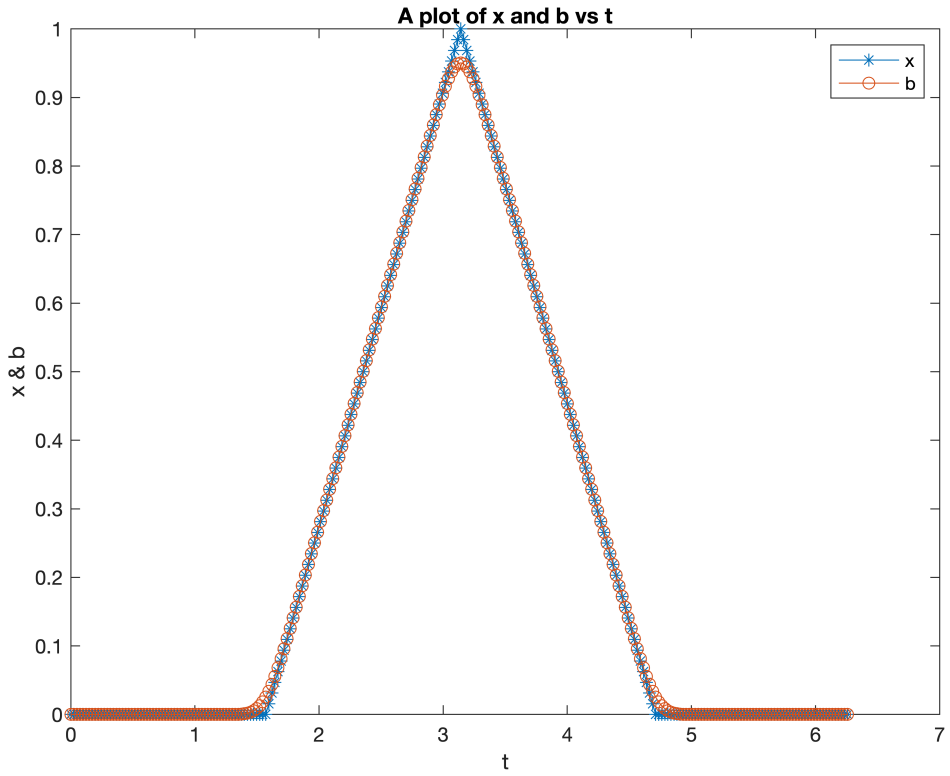
```
The condition number of A is: 1.144935e+16
```

```

fprintf('2b). Construct vector x. \n\n');
```

```
2b). Construct vector x.
```

```
%vector x
X = zeros(N,1);
for j = 1:N
    X(j) = signal(tj(j));
end

%Compute the blurred signal b
b = A*X;

%Make a plot of x vs t and b vs t
figure(1)
plot(tj,X,'-*')
hold on
plot(tj,b,'-o')
legend('x','b');
xlabel('t'); ylabel('x & b');
```

```
title('A plot of x and b vs t');
```



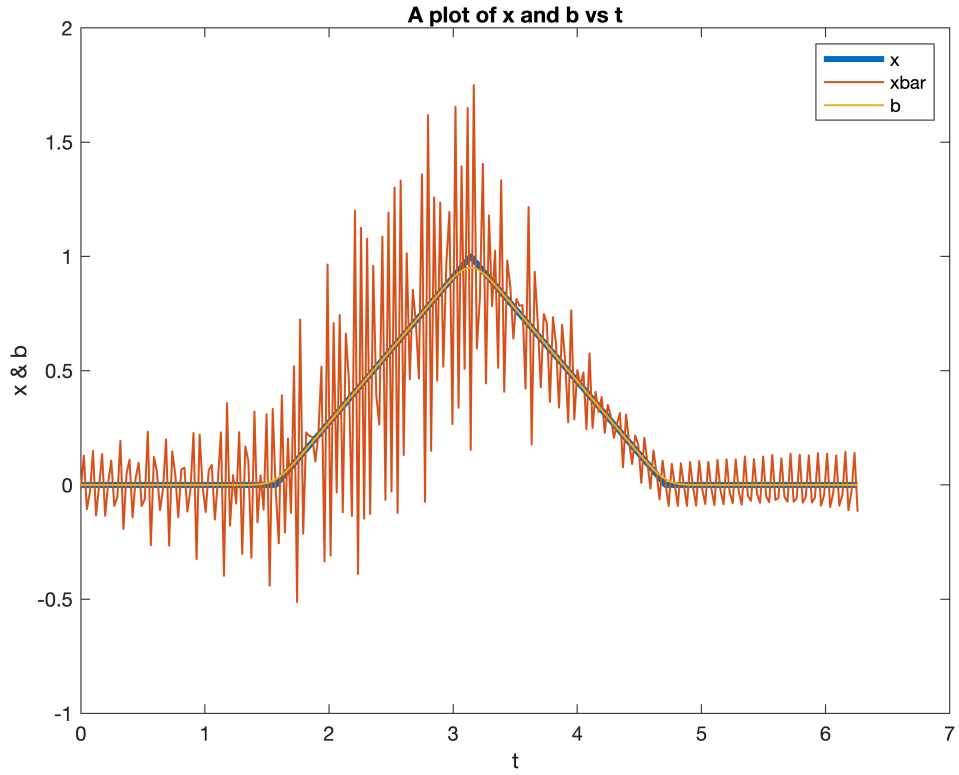

```
fprintf('2c). Solve Axbar = b  \n\n');
```

```
2c). Solve Axbar = b
```

```
xbar = A\b;
```

```
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
 6.559665e-17.
```

```
figure(2)
plot(tj,X,'linewidth',3)
hold on
plot(tj,xbar,'linewidth',1)
hold on
plot(tj,b,'linewidth',1)
legend('x','xbar','b');
xlabel('t'); ylabel('x & b');
title('A plot of x and b vs t');
```




```
fprintf('xbar doesnot look anything close to x, since A is ill conditioned, xbar has al
```

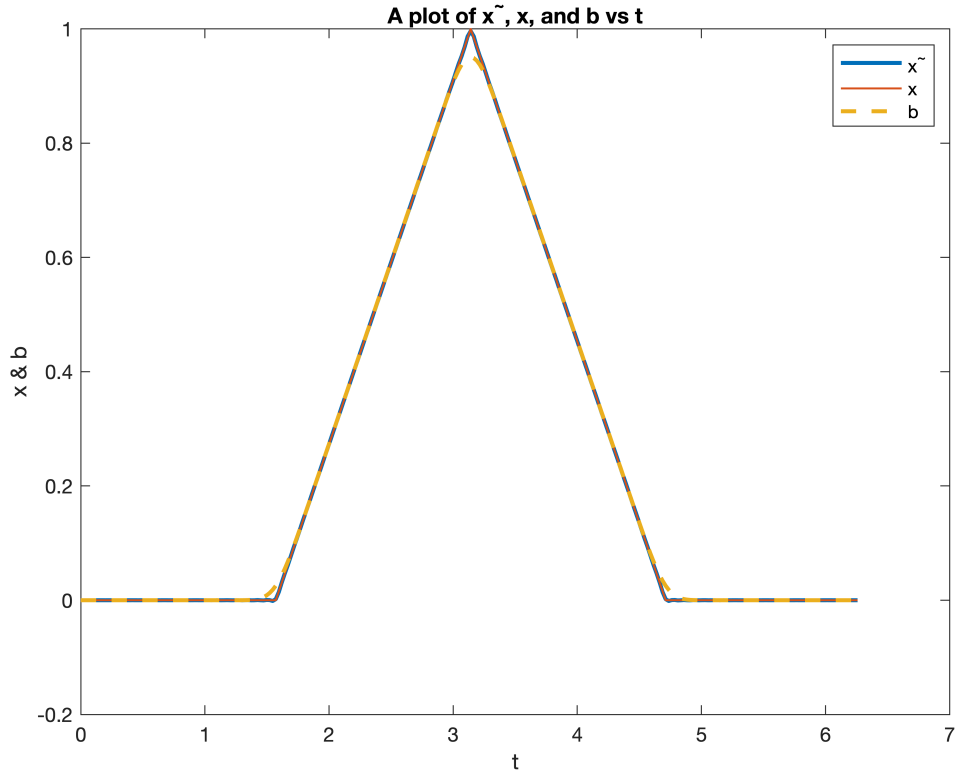
```
xbar doesnot look anything close to x, since A is ill conditioned, xbar has alot of noise.
```

```
fprintf('2d). Compute a reduced rank least squares solution  \n\n');
```

```
2d). Compute a reduced rank least squares solution
```

```
[u,s,v] = svd(A);

%sigular values
sigma = [];
U = [];
V = [];
for i = 1:N
   sig = s(i,i);
   if sig >= 1e-12
      sigma = [sigma,sig];
      U = [U,u(:,i)];
      V = [V,v(:,i)];
      continue
   end
end

%compute xtilda
```

```
xtilda = rls(sigma,U,V,b);

%plot
figure(3)
plot(tj,xtilda,'linewidth',2)
hold on
plot(tj,X,'linewidth',1)
hold on
plot(tj,b,'--','linewidth',2)
legend('x^{~}','x','b');
xlabel('t'); ylabel('x & b');
title('A plot of x^{~}, x, and b vs t');
```




```
fprintf('The reduced rank least squares solution ,xhat, fits the data much better than
```

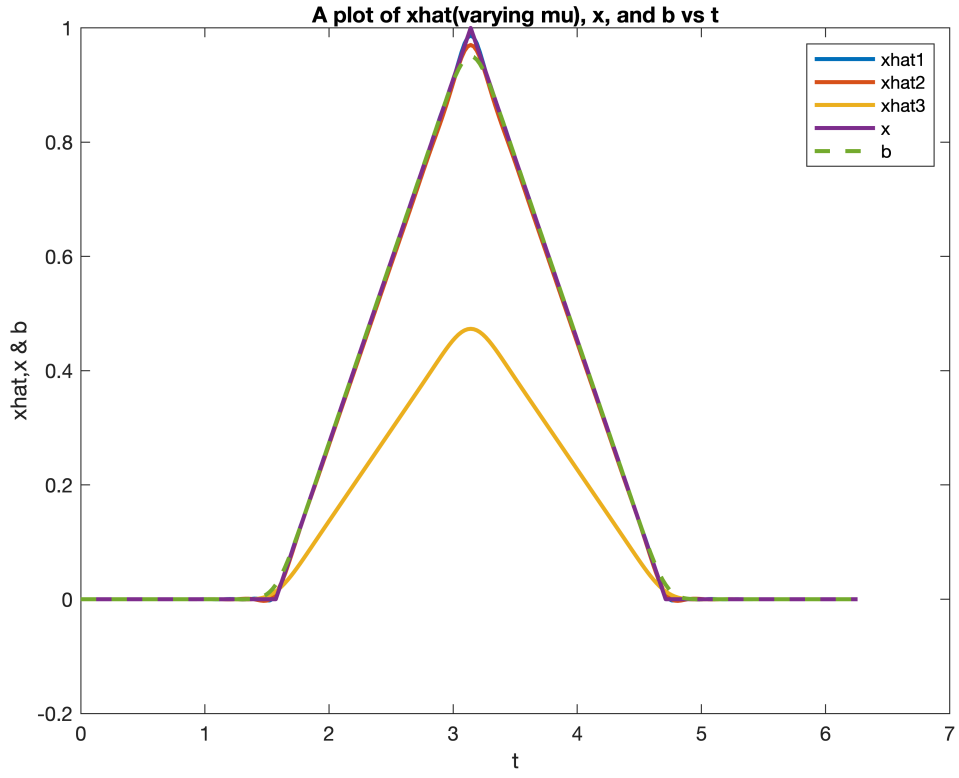
```
The reduced rank least squares solution ,xhat, fits the data much better than xbar.
```

```
fprintf('2e).Use ridge regression \n\n');
```

```
2e).Use ridge regression
```

```
%compute xhat
mu1 = 1e-4; mu2 = 1e-2; mu3 = 1;
xhat1 = ridge(mu1,s,u,v,b,N);
xhat2 = ridge(mu2,s,u,v,b,N);
xhat3 = ridge(mu3,s,u,v,b,N);
```

```
%plot
figure(4)
plot(tj,xhat1,'linewidth',2)
hold on
plot(tj,xhat2,'linewidth',2)
hold on
plot(tj,xhat3,'linewidth',2)
hold on
plot(tj,X,'linewidth',2)
hold on
plot(tj,b,'--','linewidth',2)
legend('xhat1','xhat2','xhat3','x','b');
xlabel('t'); ylabel('xhat,x & b');
title('A plot of xhat(varying mu), x, and b vs t');
```




```
fprintf('According to the plot above, the small the value of the regularization paramet
```

```
According to the plot above, the small the value of the regularization parameter the better the
 approximation, as its seen for mu = 1, the solution is completely off.
```

```
fprintf('Xhat perfomance inturns of approximation depends on parameter, mu, so if we se
```

```
Xhat perfomance inturns of approximation depends on parameter, mu, so if we select a good parameter,
 then it approximates better than the rest
```

```
fprintf('2f). Repeat (c) - (e) by perturbing each entry of b \n\n');
```

2f). Repeat (c) - (e) by perturbing each entry of b

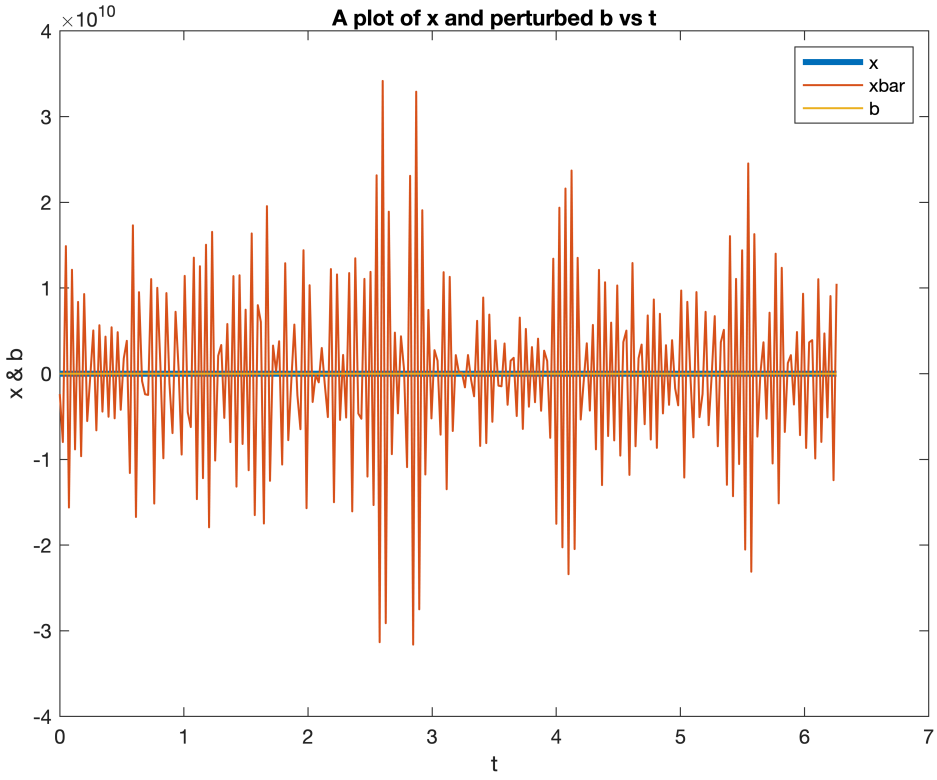
```
b = b + 1e-5*randn(N,1);
fprintf('Repeated 2c). \n\n');
```

```
Repeated 2c).
```

```
xbar = A\b;
```

```
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
 6.559665e-17.
```

```
figure(5)
plot(tj,X,'linewidth',3)
hold on
plot(tj,xbar,'linewidth',1)
hold on
plot(tj,b,'linewidth',1)
legend('x','xbar','b');
xlabel('t'); ylabel('x & b');
title('A plot of x and perturbed b vs t');
```



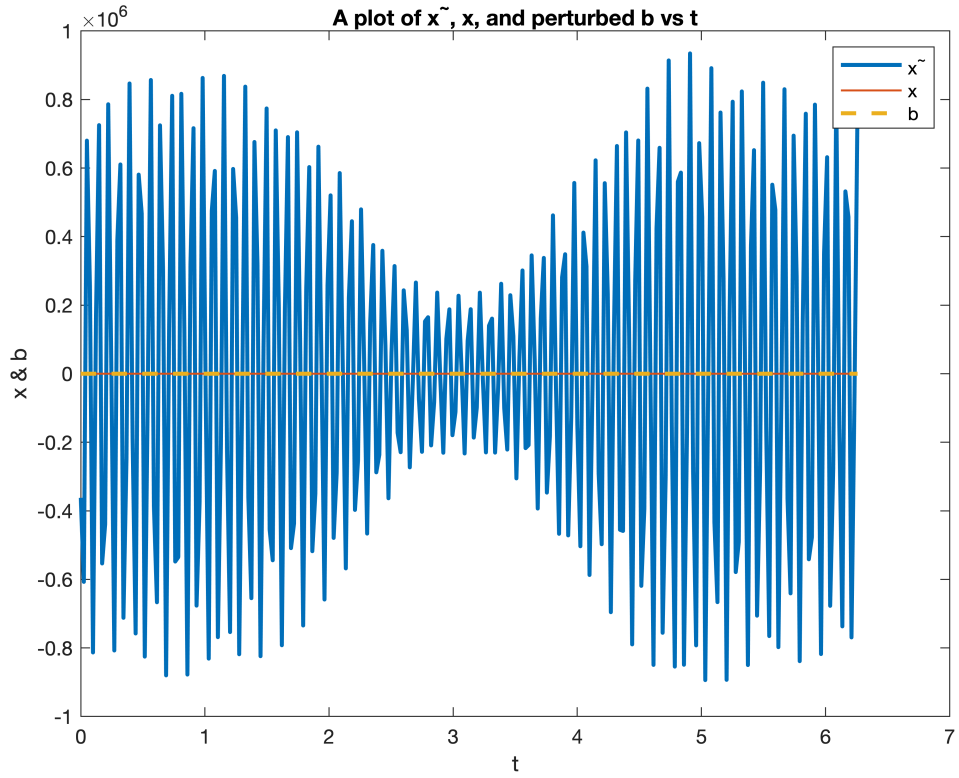

```
fprintf('xbar depicts alot of noise compared to x and b, this is due to the pertubation
```

```
xbar depicts alot of noise compared to x and b, this is due to the pertubation caused at b
```

```
fprintf('Repeated 2d). Compute a reduced rank least squares solution  \n\n');
```

```
Repeated 2d). Compute a reduced rank least squares solution
```

```
b = A*X;
b = b + 1e-5*randn(N,1);
%compute xtilda
xtilda = rls(sigma,U,V,b);

%plot
figure(6)
plot(tj,xtilda,'linewidth',2)
hold on
plot(tj,X,'linewidth',1)
hold on
plot(tj,b,'--','linewidth',2)
legend('x^{~}','x','b');
xlabel('t'); ylabel('x & b');
title('A plot of x^{~}, x, and perturbed b vs t');
```



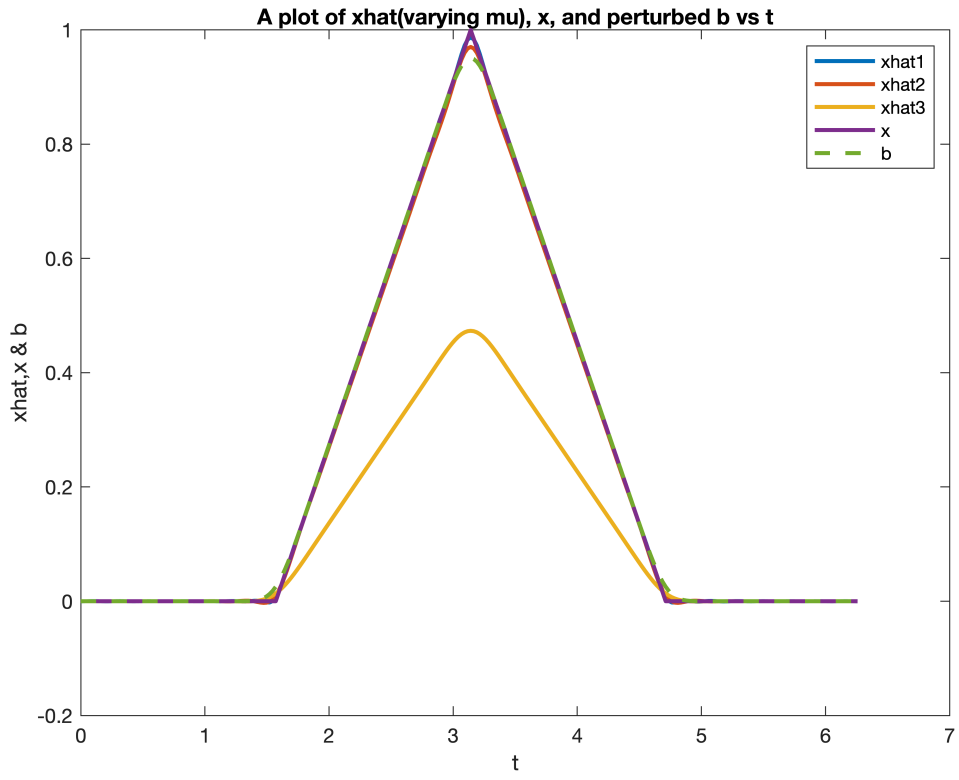

```
fprintf('The reduced rank least squares solution ,xhat, also exibits noise as xbar, but
```

```
The reduced rank least squares solution ,xhat, also exibits noise as xbar, but in large amplitudes.
```

```
fprintf('Repeated 2e).Use ridge regression \n\n');
```

```
Repeated 2e).Use ridge regression
```

```
b = A*X;
b = b + 1e-5*randn(N,1);

%compute xhat
mu1 = 1e-4; mu2 = 1e-2; mu3 = 1;
xhat1 = ridge(mu1,s,u,v,b,N);
xhat2 = ridge(mu2,s,u,v,b,N);
xhat3 = ridge(mu3,s,u,v,b,N);
%plot
figure(7)
plot(tj,xhat1,'linewidth',2)
hold on
plot(tj,xhat2,'linewidth',2)
hold on
plot(tj,xhat3,'linewidth',2)
hold on
plot(tj,X,'linewidth',2)
hold on
plot(tj,b,'--','linewidth',2)
legend('xhat1','xhat2','xhat3','x','b');
xlabel('t'); ylabel('xhat,x & b');
title('A plot of xhat(varying mu), x, and perturbed b vs t');
```




```
fprintf('Pertubing b, does not affect xhat, due to obtaining the same plots, before and
```

```
Pertubing b, does not affect xhat, due to obtaining the same plots, before and after perturbing b
```

```
fprintf('According to the plot above, the small the value of the regularization paramet
```

```
According to the plot above, the small the value of the regularization parameter the better the
 approximation, as its seen for mu = 1, the solution is completely off.
```

```
fprintf('Xhat perfomance inturns of approximation depends on parameter, mu, so if we se
```

```
Xhat perfomance inturns of approximation depends on parameter, mu, so if we select a good parameter,
 then it approximates better than the rest
```

```
%ridge regression solution
```

```
function xhat = ridge(mu,s,u,v,b,N)

    xhat = 0;
    s = diag(s);
    for j = 1:N
        xhat = xhat + ((s(j)./((s(j).^2)+mu)).*(u(:,j)'*b)).*v(:,j);
    end

end

%Reduced rank least squares solution
function xtilda = rls(sigma,U,V,b)
    r = length(sigma);
    xtilda = 0;
    for j = 1:r
        xtilda = xtilda + ((U(:,j)'*b)./sigma(j)).*V(:,j);
    end
end


%periodic signal
function [x] = signal(t)

    if abs(t-pi) < pi/2
        x = 1 - (2/pi)*abs(t-pi);
    else
        x = 0;
    end

end
```

N0.4a

```
function [L,U,P,g] = lupp(Ao)
%lupp  Computes the LU decomposition of A with partial pivoting
%
%   [A,p] = lupp(A) Computes the LU decomposition of A with partial
%   pivoting using vectorization. The factors L and U are returned in the
%   output A, and the permutation of the rows from partial pivoting are
%   recorded in the vector p. Here L is assumed to be unit lower
%   triangular, meaning it has ones on its diagonal. The resulting
%   decomposition can be extracted from A and p as follows:
%       L = eye(length(LU))+tril(LU,-1);     % L with ones on diagonal
%       U = triu(LU);                        % U
%       P = p*ones(1,n) == ones(n,1)*(1:n);  % Permutation matrix
%   A is then given as L*U = P*A, or P'*L*U = A.
%
%   Use this function in conjuction with backsub and forsub to solve a
%   linear system Ax = b.
A = Ao;
n = size(A,1);
p = (1:n)';

for k=1:n-1
    % Find the row in column k that contains the largest entry in magnitude
    [~,pos] = max(abs(A(k:n,k)));
    row2swap = k-1+pos;
    % Swap the rows of A and p (perform partial pivoting)
    A([row2swap, k],:) = A([k, row2swap],:);
    p([row2swap, k]) = p([k, row2swap]);
    % Perform the kth step of Gaussian elimination
    J = k+1:n;
    A(J,k) = A(J,k)/A(k,k);
    A(J,J) = A(J,J) - A(J,k)*A(k,J);
end

%Permutation matrix
P = p*ones(1,n) == ones(n,1)*(1:n);

LU = A;

%unit lower triangular matrix L
L = eye(length(LU))+tril(LU,-1);

%upper triangular matrix U
U = triu(LU);

%growth factor
u = abs(U);
%A = P'*L*U;
a = abs(Ao);
u1 = max(u,[],'all');
a1 = max(a,[],'all');
g = u1/a1;

end
```

```
Not enough input arguments.
```

N0.4b

```
clear all
close all

fprintf('4b). Use your code to compute the growth factor of the 21 by 21 matrix. \n\n')

m = 21;
A = zeros(m,m);
for i = 1:m
   for j = 1:m
       if i == j
            A(i,j) = 1;
       elseif i > j
           A(i,j) = -1;
       else
           A(i,m) = 1;
       end
   end
end

[L,U,P,g] = lupp(A);

fprintf('The growth factor for A is: %d\n',g);
```

```
4b). Use your code to compute the growth factor of the 21 by 21 matrix.

The growth factor for A is: 1048576
```

Published with MATLAB® R2020b

N0.5a

```
function [LU,p,q,gf,L,U]= lucp(Ao)
%lupp  Computes the LU decomposition of A with partial pivoting
%
%   [LU,p,q,gf]= lucp(A) Computes the LU decomposition of A with complete
%   pivoting using vectorization. The factors L and U are returned in the
%   output A, and the permutation of the rows from partial pivoting are
%   recorded in the vector p. Here L is assumed to be unit lower
%   triangular, meaning it has ones on its diagonal. The resulting
%   decomposition can be extracted from A and p as follows:
%       L = eye(length(LU))+tril(LU,-1);     % L with ones on diagonal
%       U = triu(LU);                        % U
%       P = p*ones(1,n) == ones(n,1)*(1:n);  % Permutation matrix
%   A is then given as L*U = P*A, or P'*L*U = A.
%
%   output
%   ------
%   LU: matrix containing upper and unit lower triangular matrix
%   p: row permutation
%   q: column permutations
%   gf: growth factor of the matrix
%   Use this function in conjuction with backsub and forsub to solve a
%   linear system Ax = b.

A = Ao;
n = size(A,1);
p = (1:n);
q = (1:n);

for k=1:n-1
    % Find the row in column k that contains the largest entry in magnitude
    pos = max(max(abs(A(k:n,k:n))));
    [i,j] = find(abs(A(k:n,k:n)) == pos);

    row2swap = k-1+i(1);
    column2swap = k-1+j(1);

    %row swap
    A([k,row2swap],:) = A([row2swap,k],:);
    p([k,row2swap]) = p([row2swap,k]);
    %column swap
    A(:,[k,column2swap]) = A(:,[column2swap,k]);
    q([k,column2swap]) = q([column2swap,k]);
    % Perform the kth step of Gaussian elimination
    J = k+1:n;
    A(J,k) = A(J,k)/A(k,k);
    A(J,J) = A(J,J) - A(J,k)*A(k,J);
end

%unit lower triangular matrix L
L = eye(n);
L = L+tril(A,-1);

%upper triangular matrix U
U = triu(A,0);

LU = L*U;

%growth factor
```

```
u = abs(U);
%A = L*U;
a = abs(Ao);
u1 = max(u,[],'all');
a1 = max(a,[],'all');
gf = u1/a1;

end
```

```
Not enough input arguments.

Error in lucp (line 25)
A = Ao;
```

Published with MATLAB® R2020b

N0.5b

```
close all;
clear all;

%Complete Pivoting
fprintf('no4b.\n\n')
n = 12;
A = vandermonde(n);

b = A(:,n);

[LU,p,q,gf,L,U]= lucp(A);
P = eye(n); P=P(p,:);
Q = eye(n); Q = Q(:,q);
yc = forsub(L,P*b);
zc = backsub(U,yc);

fprintf('Compute pivoting solution \n')
xc = Q*zc

x = A\b
residual = norm(b - A*xc);
fprintf('The value of the max-norm of the residual = %f\n\n',residual);

fprintf('no4c.\n\n')
%Gaussian Elimination with Partial pivoting
[Lp,Up,Pl,g] = lupp(A);
yp = forsub(Lp,Pl*b);

fprintf('Partial pivoting solution \n')
xp = backsub(Up,yp)

residual_xp = norm(b - A*xp);
fprintf('The value of the max-norm of the residual = %f\n\n',residual_xp);

fprintf('Results from part (b), are exactly the same as the exact solution x, since even the residual\n is zero. But for part(c), the solution doesnot

function A = vandermonde(n)

    t = zeros(n,n);
    for i = 1:n
        for j = 1:n
            t(j,i) = j^(n-i);
        end
    end

    %fliping the vandermonde matrix t to form A
    A = fliplr(t);

end
```

```
no4b.

Compute pivoting solution

xc =

     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     1

Warning: Matrix is close to singular or badly scaled. Results may be inaccurate.
RCOND =  8.296438e-17.

x =

     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     0
     1
```

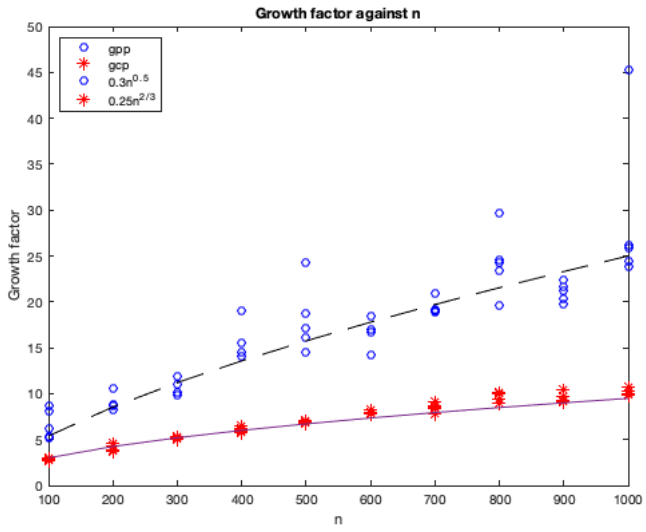
```
The value of the max-norm of the residual = 0.000000

no4c.

Partial pivoting solution

xp =

   -0.0080
    0.0233
   -0.0279
    0.0184
   -0.0075
    0.0020
   -0.0004
    0.0000
   -0.0000
    0.0000
   -0.0000
    1.0000

The value of the max-norm of the residual = 0.000236

Results from part (b), are exactly the same as the exact solution x, since even the residual
 is zero. But for part(c), the solution doesnot properly approximate the actual solution, according to the residual computed.
```

Published with MATLAB® R2020b

N0.6

```
clear all
close all

n = [100:100:1000]';

N = length(n);

gcp = zeros(N,5);
gpp = zeros(N,5);
for i = 1:N
    for j = 1:5
        A = rand(n(i));
        [LU,p,q,gf,L,U]= lucp(A);
        gcp(i,j) = gf;
        [L,U,P,g] = lupp(A);
        gpp(i,j) = g;
    end
end

for k = 1:5
    plot(n,gpp(:,k),'bo');
    hold on
    plot(n,gcp(:,k),'r*');
    hold on

end

plot(n,0.3*(n.^0.5))
hold on
plot(n,0.25*(n.^(2/3)),'k--')
hold on
legend('gpp','gcp','0.3n^0^.^5','0.25n^2^/^3','Location','northwest')
title('Growth factor against n')
xlabel('n'); ylabel('Growth factor')

fprintf('As n increases, the growth factor gradually increases, however, it doesnot shoot up with n, but instead fit the two constat functions. \n Eve
```

```
As n increases, the growth factor gradually increases, however, it doesnot shoot up with n, but instead fit the two constat functions.
 Even though, the growth factor computed with complete pivoting fits the constant curves better than, the one for partial pivoting.
 This becasue the upper triangular matrix obtained from complete pivorting is better than that obtained from partial pivoting
```



Published with MATLAB® R2020b