Question 1

a)
$$\begin{bmatrix} I & A \\ A^T & 0 \end{bmatrix}\begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix} \quad -① $$

We apply Gaussian elimination on ① and obtain

$$\begin{bmatrix} I & A \\ 0 & A^T A \end{bmatrix}\begin{bmatrix} r \\ x \end{bmatrix} = \begin{bmatrix} b \\ A^T b \end{bmatrix} \quad -② $$

From (2) we obtain the following results;

$$r + Ax = b$$
$$A^T A x = A^T b$$
$$x = (A^T A)^{-1} A^T b$$

Thus x minimizes $\|Ax - b\|_2$.

$r = b - Ax$, which is the residual.

(b)

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

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

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

Expressing $\tilde{A}$ as blocks of 2×2 matrices have;

$\tilde{A}_1 = \begin{bmatrix} 1 & \sigma_1 \\ \sigma_1 & 0 \end{bmatrix}$, $\tilde{A}_2 = \begin{bmatrix} 1 & \sigma_2 \\ \sigma_2 & 0 \end{bmatrix}$, $\ldots$, $\tilde{A}_n = \begin{bmatrix} 1 & \sigma_n \\ \sigma_n & 0 \end{bmatrix}$

If $A \in \mathbb{R}^{m\times n}$ and $m \geq n$, we shall have some extra form expressed as

$\tilde{A}_{m-n} = I_{m-n}$.

In general, we can express $\tilde{A}_i$, $i = 1, 2, \ldots, n$ in a compact form as:

$$\tilde{A}_i = \begin{bmatrix} 1 & \sigma_i \\ \sigma_i & 0 \end{bmatrix}$$

we now compute the determinant of $\tilde{A}_i$

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

$$\lambda_i^2 - \lambda_i - \sigma_i^2 = 0 \Rightarrow \lambda_i = \frac{1 \pm \sqrt{1+4\sigma_i^2}}{2}, \; i = 1, 2, \ldots, n$$

$$\tilde{A}_{m-n} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

determinant of $\tilde{A}_{m-n}$:

$$\begin{bmatrix} 1-\lambda & 0 \\ 0 & 1-\lambda \end{bmatrix} = 0 \quad \Rightarrow \lambda = 1$$

The 2-norm condition number of A is given by

$$\kappa(A) = \frac{|\text{eigenvalue max } (A)|}{|\text{eigenvalue min } (A)|}$$

maximum eigenvalue of $A = \lambda_1 = \frac{1 + \sqrt{1+4\sigma_1^2}}{2}$

minimum eigen value of $A = \lambda_n = \frac{1 - \sqrt{1+4\sigma_n^2}}{2}$

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

since $1 - \sqrt{1+4\sigma_n^2} < 1$, we can re-arrange terms to incorporate the absolute sign.

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

1c

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

$$\kappa(A^TA) = \frac{\sigma_1^2}{\sigma_n^2} \quad - (2)$$

By comparing (1) and (2), we can observe that (2) will always be bigger for a given σ_1 and σ_n.

Question 3

We want to show mathematically that;

$$\begin{bmatrix} 1 & & & & \\ 0 & 1 & & & \\ \vdots & 0 & \ddots & & \\ \vdots & \vdots & 1 & & \\ \vdots & \vdots & l_{i+1,i} & 1 & \\ & & l_{i+2,i} & 0 & \\ 0 & 0 & \vdots & \vdots & 1 \\ & & l_{n,i} & 0 & \end{bmatrix}^{-1} = \begin{bmatrix} 1 & & & & \\ 0 & 1 & & & \\ \vdots & 0 & \ddots & & \\ \vdots & \vdots & & 1 & & \\ & & & -l_{i+1,i} & 1 & \\ 0 & 0 & & -l_{i+2,i} & 0 & 1 \\ & & & \vdots & \vdots & \\ & & & -l_{n,i} & 0 & \end{bmatrix} \quad — (1)$$

Let $L_K = \begin{bmatrix} 1 & & & & \\ & \ddots & & & \\ & & 1 & & \\ & & -l_{K+1,K} & 1 & \\ & & \vdots & & \ddots & \\ & & -l_{n,K} & & & 1 \end{bmatrix}$, $K = 1, 2, \cdots, n$

The multipliers in L_K can be expressed as: $l_{jK} = \frac{a_{jK}}{a_{KK}}$, $j = K+1, \cdots, n$

Let $l_K = \begin{bmatrix} 0 \\ \vdots \\ l_{K+1,K} \\ \vdots \\ l_{n,K} \end{bmatrix}$

We can then express L_K as: $\quad — (2)$

$L_K = I - l_K e_K$

where $e_K = \begin{bmatrix} 0 & \cdots & 0 & & \\ \vdots & & 1 & \ddots & \\ \vdots & & \vdots & & \ddots \\ 0 & & 1 & & 0 \end{bmatrix}$

from (2), It follows that $L_k^{-1} = (I + l_k e_k)$.

For (1) to hold, then $L_k L_k^{-1} = I$.

$$
\begin{aligned}
L_k L_k^{-1} &= (I - l_k e_k)(I + l_k e_k) \\
&= I + l_k e_k - l_k e_k - l_k \underbrace{e_k l_k}_{=0} e_k \\
&= I - l_k e_k l_k e_k
\end{aligned}
$$

But the inner product $e_k l_k = 0$ because the non-zero term in e_k get multiplied with a zero term in l_k. The non-zero terms of l_k begins from index $k+1$. Hence $e_k l_k = 0$.

Therefore:

$$L_k L_k^{-1} = I - 0 = I$$

Question 4

We apply induction to show that the growth factor of A is exactly : $g_n(A) = 2^{n-1}$ i.e for $A \in \mathbb{R}^{n\times n}$. $g_n(A) = \frac{\max_{i,j}|u_{ij}|}{\max_{i,j}|a_{ij}|}$.

Let us consider a few base cases that are easy to visualize

When $n=2$:

$$A = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \xrightarrow{\text{LU Decomposition}} U = \begin{bmatrix} 1 & 1 \\ 0 & 2 \end{bmatrix}, L = \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix}.$$

$$g_2(A) = \frac{2^1}{1} = 2.$$

When $n = 3$.

$$A = \begin{bmatrix} 1 & 0 & 1 \\ -1 & 1 & 1 \\ -1 & -1 & 1 \end{bmatrix} \xrightarrow{\text{LU Decomposition}} U = \begin{bmatrix} 1 & 0 & 2^0 \\ -1 & 1 & 2^1 \\ -1 & -1 & 2^2 \end{bmatrix}, L = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ -1 & -1 & 1 \end{bmatrix}$$

$$g_3(A) = \frac{2^2}{1} = 2^2$$

Let us assume that the growth factor holds for $A \in \mathbb{R}^{n\times n}$ where $n = K$

That is, $g_K(A) = 2^{K-1}$

Now we shall show that it is true for $n = K+1$.

let $A \in \mathbb{R}^{n\times n}$ for $n = K+1$.

$$A = \begin{bmatrix} 1 & & 0 & & 1 \\ -1 & 1 & & & \vdots \\ \vdots & & \ddots & & \vdots \\ \vdots & & & \ddots & \vdots \\ -1 & \cdots & \cdots & -1 & 1 \end{bmatrix} \xrightarrow{\text{LU Decomposition}} U = \begin{bmatrix} 1 & & 0 & 2^0 \\ -1 & \ddots & & 2^1 \\ \vdots & & \ddots & 2^2 \\ \vdots & & & \vdots \\ -1 & \cdots & -1 & 2^{(K+1)-1} \end{bmatrix}$$

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

$$g_{k+1}(A) = \frac{2^{(k+1)-1}}{1} = 2^{(k-1)+1} = g_k(A) \cdot 2 = 2^k.$$

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

```
clear all; close all; format long;
```

Question 1

```
%%%%%%%%% (d) %%%%%%%%%%%%%%%
```

```
% setup vandermonde matrix

m=50; n=12;
t=linspace(0,1,m)';
A=t.^(0:n-1);
b=cos(4*t);
```

```
A_block = [eye(m) A; A' zeros(n)];
[Q_block,R_block]=qr(A_block);
b_block = [b;zeros(n,1)];
x = R_block\(Q_block'*b_block);
x = x(m+1:end);
```

```
Z = A'*A;
R = chol(Z);
```

```
c1=R\(R'\(A'*b));
```

```
[Q4,R4]=qr(A);
c5=R4\(Q4'*b);
```

```
T= table(x,c1,c5,'VariableNames',{'x','Cholesky','qr'})
```

T = 12×3 table

	x	Cholesky	qr
1	0.99999999855...	0.99999998099...	1.00000000099...
2	0.00000029543...	0.00000527619...	-0.0000004227...
3	-8.0000082095...	-8.0001921291...	-7.9999812356...
4	0.00007893423...	0.00275327159...	-0.0003187631...
5	10.6663835056...	10.6461350381...	10.6694307954...
6	-0.0000547303...	0.09046194227...	-0.0138202860...
7	-5.6861450817...	-5.9406827542...	-5.6470756325...
8	-0.0036916343...	0.45910836422...	-0.0753160151...
9	1.60889281356...	1.06554610228...	1.69360695304...

	x	Cholesky	qr
10	0.06845242444...	0.46615017180...	0.00603211622...
11	-0.4002989990...	-0.5653188824...	-0.3742417064...
12	0.09274706534...	0.12239000974...	0.08804057660...

```
max(abs(x-c5))
```

```
ans =
   0.084714139482691
```

It can be observed that most of the polynomial coefficients of x are close to those produce by qr. And the maximum absolute difference between them is approximately 0.085.

Question 2

```
%%%%%%%%% (a) %%%%%%%%%%%%%%%
```

```
k = @(t)1/(0.1*sqrt(2*pi))*exp((-2+2*cos(t))/(2*0.1^2));
```

```
N = 256;
h = 2*pi/N;
j=0:1:N-1;
t = h*j;
M = h*k(t);
A = toeplitz(M, M([1 end:-1:2]));
```

```
cond(A)
```

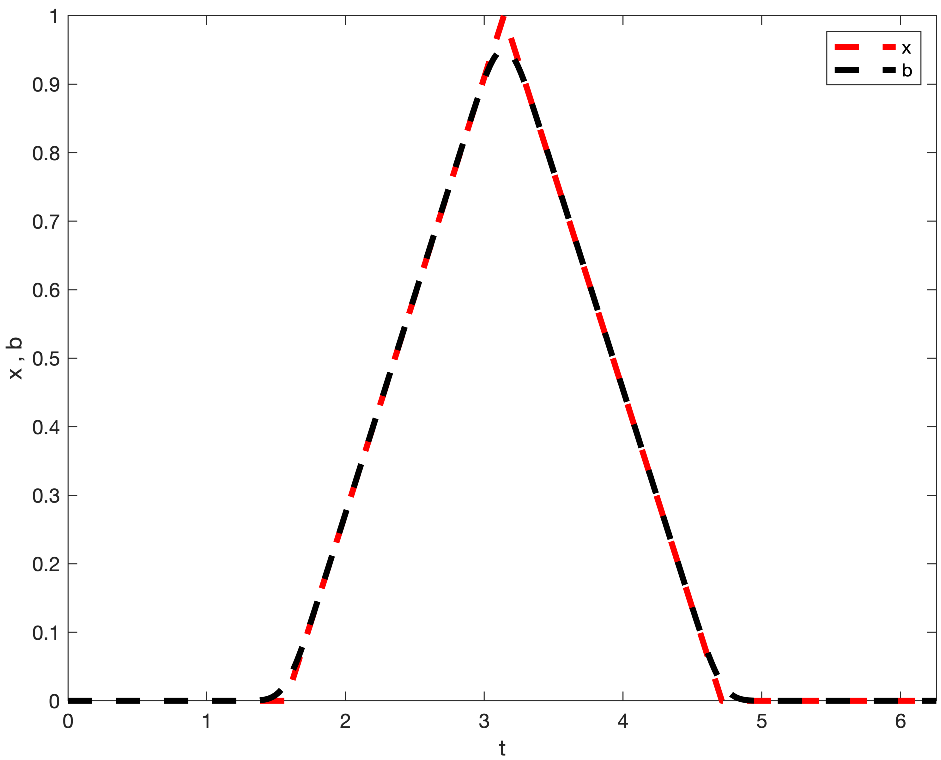
```
ans =
     1.046984691950261e+16
```

```
%%%%%%%%% (b) %%%%%%%%%%%%%%%
```

```
x = zeros(N,1);
for i = 1:N
    if (abs(t(i) - pi) < pi/2)
        x(i)= 1 - 2/pi*abs(t(i)-pi);
    else
        x(i) = 0;
    end
end
```

```
b = A*x;
```

```
figure(2)
plot(t,x,'--r', LineWidth=3), hold on;
plot(t,b,'--k', LineWidth=3)
axis tight;
ylabel("x , b")
xlabel("t")
legend("x","b")
```



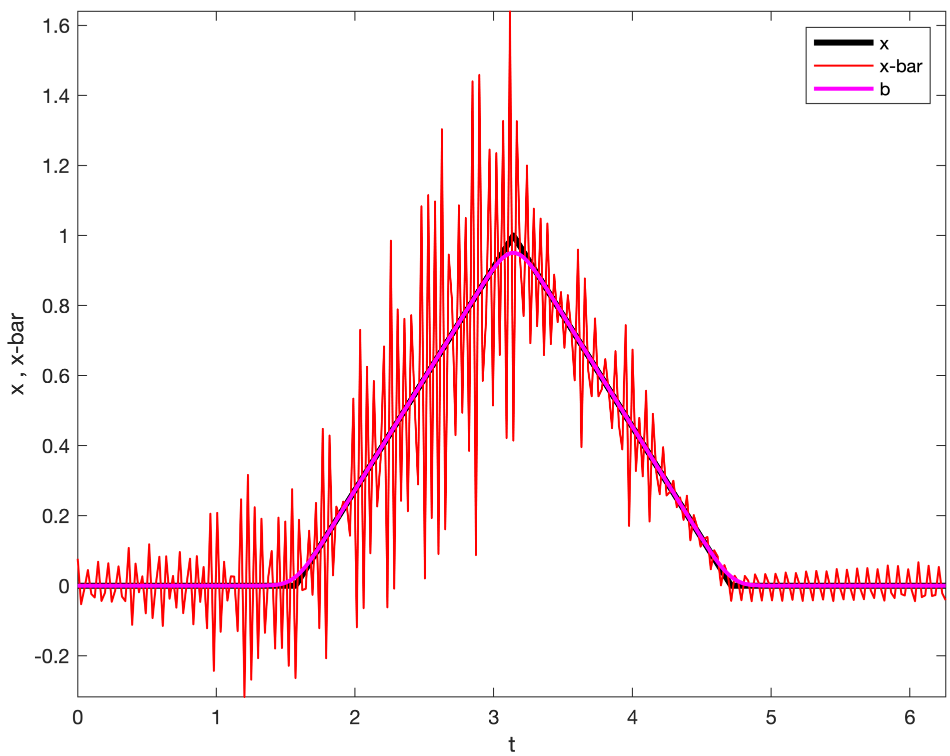

```
%%%%%%%%% (c) %%%%%%%%%%%%%%
```

```
x_bar = A\b
```

```
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
6.632924e-17.
x_bar = 256×1
   0.077031751177694
  -0.052629260653967
  -0.007467077764012
   0.044717880290056
  -0.023568499443814
  -0.033696424976526
   0.068510349432990
  -0.043410189661611
```

```
-0.020641257857149
 0.063869545997445
   ⋮
```

```
figure(3)
plot(t,x,'k', LineWidth=3), hold on;
plot(t,x_bar,'r', LineWidth=1), hold on
plot(t,b,'m', LineWidth=2)
axis tight;
ylabel("x , x-bar")
xlabel("t")
legend("x","x-bar","b")
```



x_bar appears to be very noisy. It looks substantially different from x.

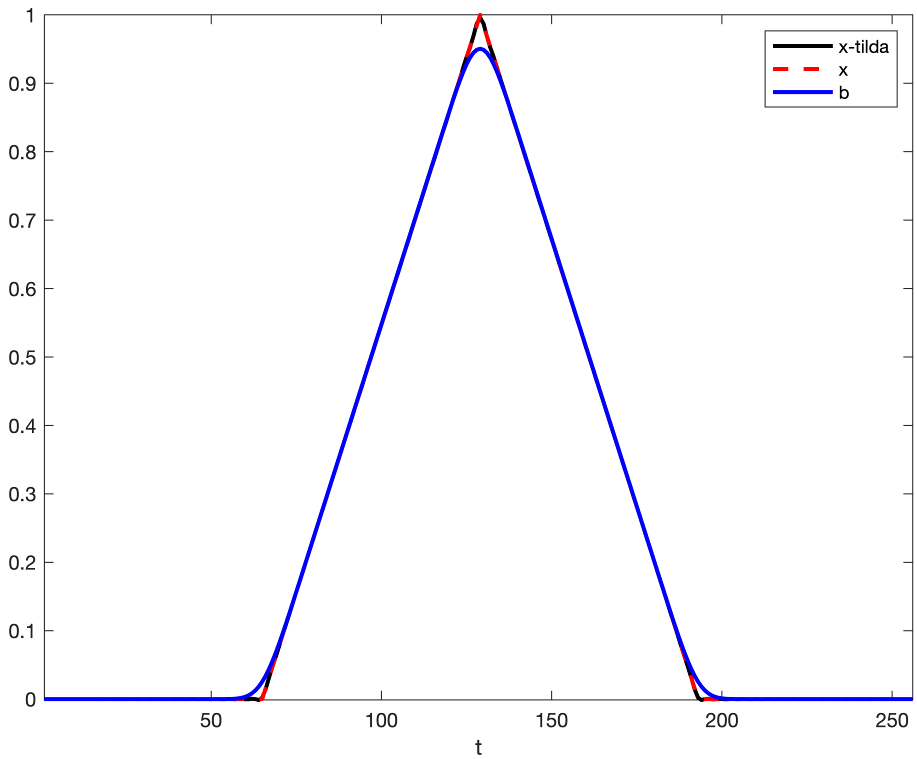
```
%%%%%%%%% (d) %%%%%%%%%%%%%%%
```

```
[U,S,V] = svd(A);
```

```
nz = sum(S(1:size(S,1)+1:end)>10e-12);
```

```
v= [];u=[];
for i = 1:N
    if (S(i,i) > 10e-12)
        v = [v,V(:,i)];
        u = [u,U(:,i)];
    end
end
sigma =S(S>10e-12);
x_hat = lsq(u,v,b,sigma);
```

```
figure(4)
plot(1:N, x_hat,'k', LineWidth=2), hold on
plot(1:N, x,'--r', LineWidth=2)
plot(1:N,b,'b', LineWidth=2)
xlabel("t")
legend("x-tilda","x","b")
axis tight
```

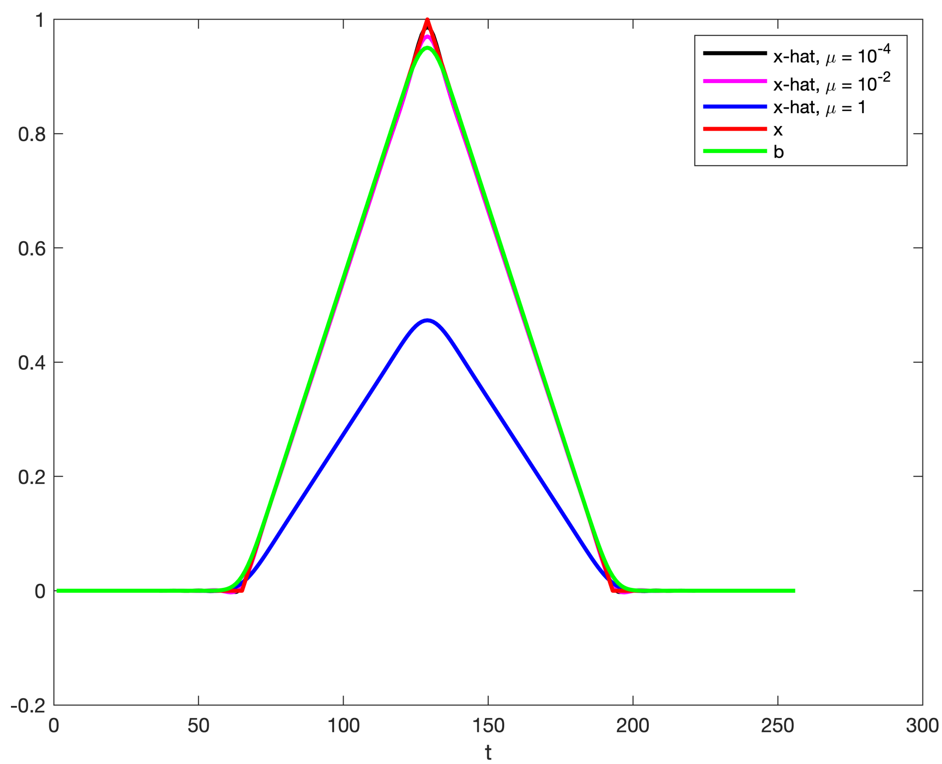


x_tilda look more like x unlike x_bar. It approximates x better than b does.

```
%%%%%%%%% (e) %%%%%%%%%%%%%%%
```

```
mu = [10^-4, 10^-2,1];
```

```
color = ['k','m','b'];
lgd = {'x-hat, \mu = 10^{-4}','x-hat, \mu = 10^{-2}','x-hat, \mu = 1'};
figure(5)
for j = 1:3
    x_hat = ridge_regression(u,v,b,sigma,mu(j));
    plot(1:N, x_hat,color(j), LineWidth=2), hold on
end
lgd{end + 1} = 'x';
lgd{end + 1} = 'b';
plot(1:N, x,'r', LineWidth=2)
plot(1:N, b,'g', LineWidth=2)
legend(lgd)
xlabel("t")
```



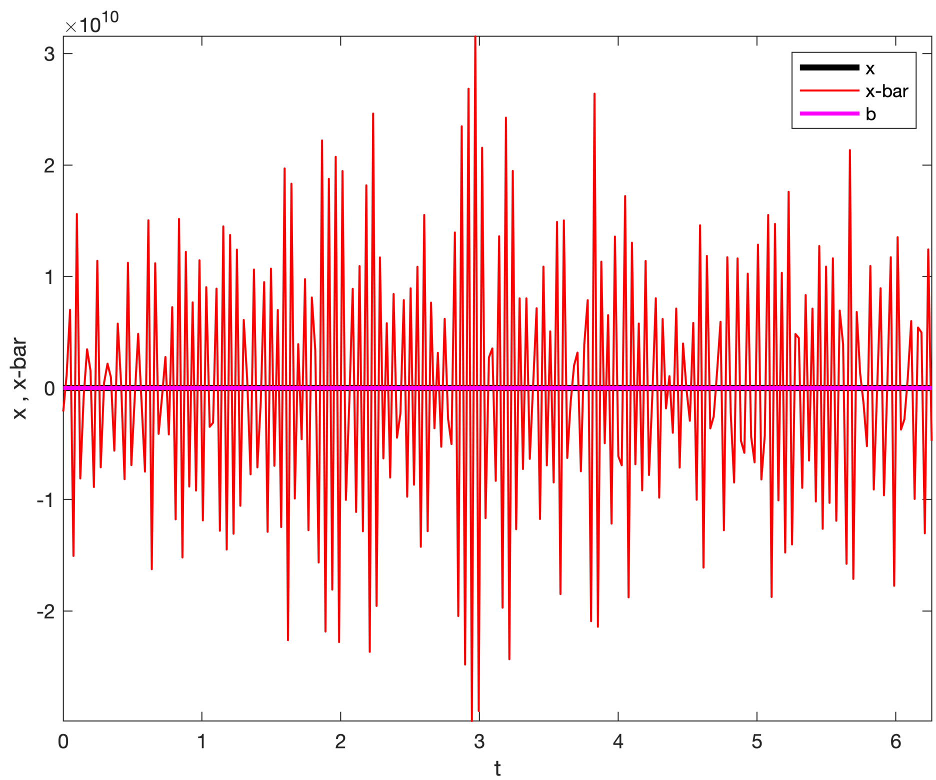
It can be obseerved that the x-hat approximates x better than b for small μ.

```
%%%%%%%%% (f) %%%%%%%%%%%%%%%
```

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

```
Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND =
6.632924e-17.
```

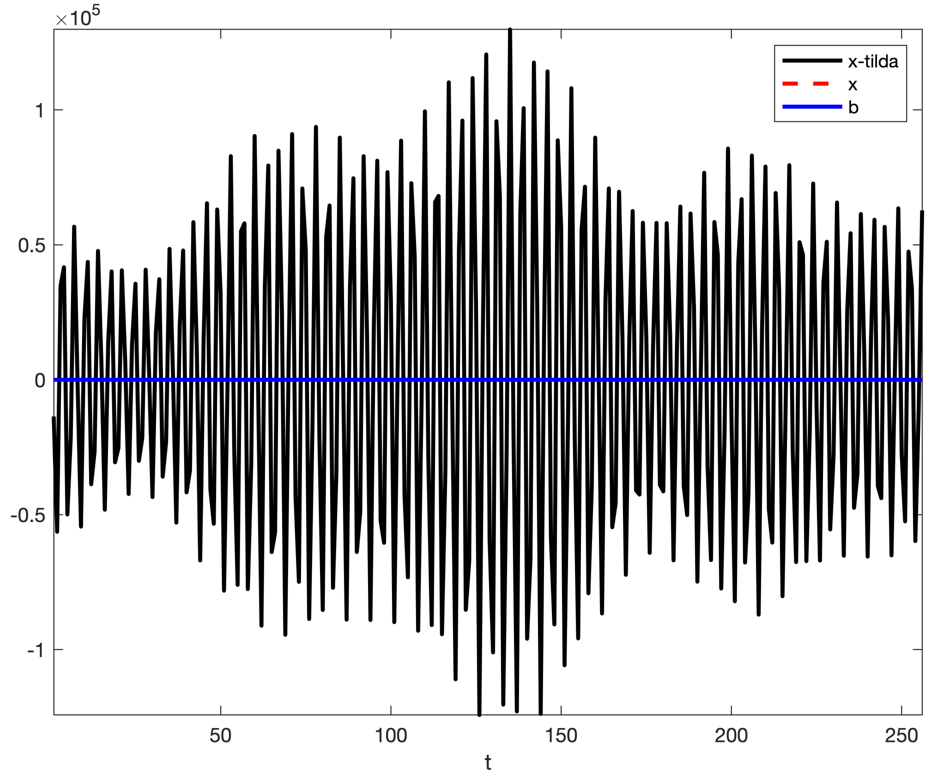
```
figure(6)
plot(t,x,'k', LineWidth=3), hold on;
plot(t,x_bar,'r', LineWidth=1), hold on
plot(t,b,'m', LineWidth=2)
axis tight;
ylabel("x , x-bar")
xlabel("t")
legend("x","x-bar","b")
```



x-bar is very noisy and its very different from x. Indeed the perturbation had a significant effect.

```
x_hat = lsq(u,v,b,sigma);
```

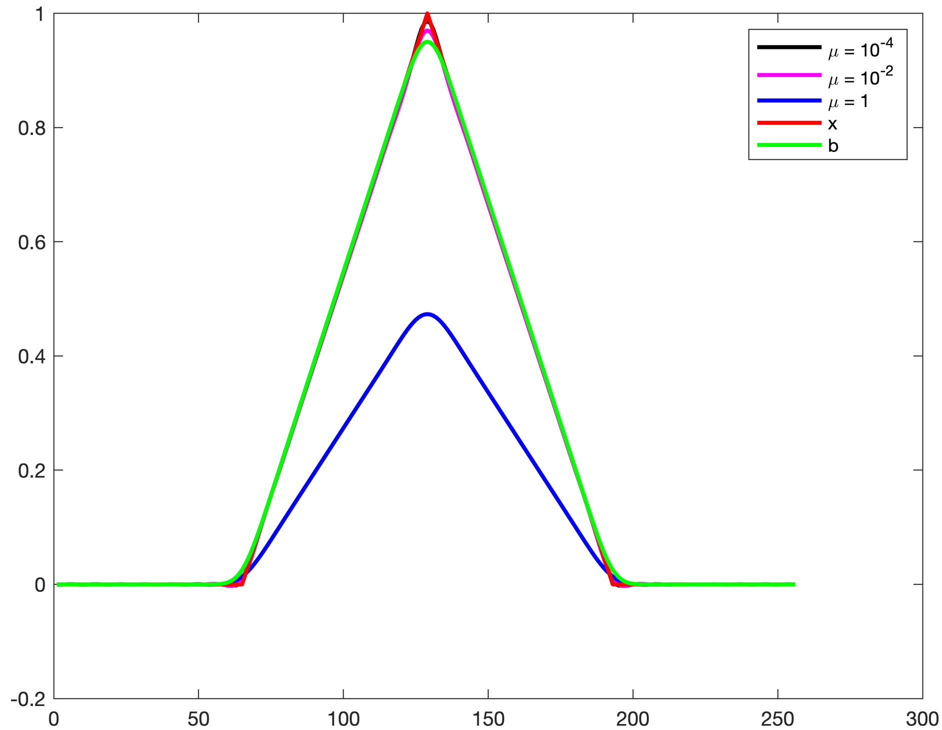
```
figure(7)
plot(1:N, x_hat,'k', LineWidth=2), hold on
plot(1:N, x,'--r', LineWidth=2)
plot(1:N,b,'b', LineWidth=2)
xlabel("t")
legend("x-tilda","x","b")
axis tight
```



x-tilda is also very noisy but with a smaller amplitude compared to x-bar.

```
mu = [10^-4, 10^-2,1];

color = ['k','m','b'];
lgd = {'\mu = 10^{-4}','\mu = 10^{-2}','\mu = 1'};
figure(8)
for j = 1:3
    x_hat = ridge_regression(u,v,b,sigma,mu(j));
    plot(1:N, x_hat,color(j), LineWidth=2), hold on
end
lgd{end + 1} = 'x';
lgd{end + 1} = 'b';
plot(1:N, x,'r', LineWidth=2)
plot(1:N, b,'g', LineWidth=2)
legend(lgd)
```



It appears that x-hat is invariant of perturbation, as no change is observed in the plot compared to the unperturbed case.

Question 4

```
%%%%%%%%% (a) %%%%%%%%%%%%%%%%
```

```
function [L,U,gf,p] = lupp(A)
%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.
A1 = A;
```

```
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
P = p*ones(1,n) == ones(n,1)*(1:n);
LU = P*A;
L = eye(length(LU))+tril(LU,-1);
U = triu(LU);
gf = max(abs(U),[],[1 2])/ max(abs(A1),[],[1 2]);
end
```

```
%%%%%%%%%%%%% (b) %%%%%%%%%%%%%%%%%%%%%%%%%
```

```
A1 = zeros(21,21);
N=21;
for i=1:N
    for j = 1 : N
        if (i ==j)
            A1(i,j) = 1;
       elseif (i> j)
            A1(i,j) = -1;

       elseif (j == N)
            A1(i,j) =1;
       end
    end
end
[L,U,gf,p] = lupp(A1);
gf
```

```
gf =
     1048576
```

```
function [L,U,gf,p] = lupp(A)
```

```
%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.
A1 = A;
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
P = p*ones(1,n) == ones(n,1)*(1:n);
LU = P*A;
L = eye(length(LU))+tril(LU,-1);
U = triu(LU);
gf = max(abs(U),[],[1 2])/ max(abs(A1),[],[1 2]);
end

function [x_hat] = ridge_regression(u,v,b,sigma,mu)
nz = length(sigma);
x_hat = 0;
for i=1:nz
      q=sigma(i)/(sigma(i)^2 + mu);
      x_hat =x_hat + (q*(u(:,i)'*b)).*v(:,i);
end
end


function [x_hat] = lsq(u,v,b,sigma)
nz = length(sigma);
```

```
x_hat = 0;
for i = 1: nz
    x_hat =x_hat + ((u(:,i)'*b)./sigma(i)).*v(:,i);

end
end
```