```
format long e
```

## Q1

```
% Define the range and create column vector t
t = -5:0.5:6;
t = t';  % Convert row vector to column vector

% Define the function handle for f
f = @(x) sin(pi * x / 5) + x / 5;

% Initialize matrix A and vector b
A = zeros(23, 18);

% Populate matrix A with powers of t
for i = 1:18
    A(:, i) = t.^(i - 1);
end

% Compute vector b using function f
b = f(t);

% Part (a): Solve the linear system A * p1 = b
p1 = A \ b;
disp('Solution p1:');
disp(p1);

% Part (b): Solve the system using Cholesky factorization
% Compute the Cholesky factorization of A'*A
G = mychol(A' * A);

% Perform forward substitution to solve G' * y = A'*b
y = rowforward(G', A' * b);

% Perform backward substitution to solve G * p2 = y
p2 = colbackward(G, y);
disp('Solution p2:');
disp(p2);

% Part (c): Solve the augmented system with least squares approach
% Create the augmented matrix and vector
newmat = [[eye(23), A]; [A', zeros(18)]];
b_new = [b; zeros(18, 1)];

% Solve the augmented system
x_new = newmat \ b_new;
p3 = x_new(24:end);  % Extract solution p3
disp('Solution p3:');
disp(p3);
```

```
% Display condition numbers
disp('Condition number of A:');
disp(cond(A));
disp('Condition number of A''*A:');
disp(cond(A' * A));
disp('Condition number of augmented matrix:');
disp(cond(newmat));
```

```
     1.124562331640989e-13
     8.283185307171449e-01
    -8.120420253470189e-13
    -4.134170223851116e-02
     9.218642809402032e-13
     8.160524914802353e-04
    -3.855139206017004e-13
    -7.670585583580261e-06
     7.694828345808158e-14
     4.205862980588951e-08
    -8.135095745559988e-15
    -1.509403656724197e-10
     4.659225606904884e-16
     3.816578580951814e-13
    -1.361442059430574e-17
    -7.073469316781525e-16
     1.585922965557028e-19
     8.629096489726066e-19
     5.375166923376386e-10
     8.283185312847947e-01
    -3.225759472038396e-09
    -4.134170318031805e-02
     3.165298229469409e-09
     8.160528787676712e-04
    -1.178356766618016e-09
    -7.670632458975229e-06
     2.143087214646983e-10
     4.205604848856084e-08
    -2.103152317838180e-11
    -1.499195229677033e-10
     1.134831939438158e-12
     2.956671745928838e-13
    -3.161360216018132e-14
     2.351111930140153e-15
     3.544236118483370e-16
    -3.925185447240720e-17
     1.123172582907393e-13
     8.283185307171437e-01
    -8.109103666331663e-13
    -4.134170223850995e-02
     9.212208921876219e-13
     8.160524914800106e-04
    -3.854173924455750e-13
    -7.670585583579876e-06
     7.695103235172418e-14
     4.205862980837579e-08
    -8.136903423185821e-15
    -1.509403658152458e-10
     4.660840085122966e-16
     3.816578552860090e-13
    -1.362029392775221e-17
    -7.073465463298883e-16
```

```
1.586701324511556e-19
8.629021971662068e-19
5.634850960284134e+13
7.374974988736028e+28
1.961760623274147e+14
```

## Q2

```
% Compute the ratios for each method
ratio1 = computeRatio(A, p1, b);
ratio2 = computeRatio(A, p2, b);
ratio3 = norm(-x_new(1:23)) / norm(b);

% Display the results
disp(ratio1);
disp(ratio2);
disp(ratio3);
```

observation for Q2?

```
1.462306065958882e-13
3.313855804653104e-10
1.461844449958713e-13
```

## Q3

```
t= -5:0.5:6;
t=t';
f= @(x) sin(pi*x/5) + x/5;
A= zeros(23,18);

for i= 1: 18
    A(:,i)=((t-2)./6).^(i-1);
end
b= f(t);

%a
p1= A\b;
disp(p1)
```

```
 1.351056516294944e+00
 2.364966623230611e+00
-6.758317136761566e+00
-2.759462330349890e+00
 8.004229984846443e+00
 1.960905697905998e+00
-3.791931985776404e+00
-6.635437004242898e-01
 9.623537451938028e-01
 1.309770482723623e-01
-1.519697550240153e-01
-1.691881924160751e-02
 1.636800960924975e-02
 1.538042337241782e-03
-1.290160713892709e-03
```

```
   -1.094828848482195e-04
    8.321478757576571e-05
    1.460590301297570e-05
```

```
%b
G= mychol(A'*A);
y= rowforward(G',A'*b);
p2 =colbackward(G,y);
disp(p2)
```

```
    1.351056570513610e+00
    2.364965899903410e+00
   -6.758330069463549e+00
   -2.759415436190012e+00
    8.004747960564133e+00
    1.960232599696222e+00
   -3.799780163543956e+00
   -6.625204098293666e-01
    1.017837128827519e+00
    1.649595409425667e-01
   -3.393529089598992e-01
   -2.452967826566502e-01
    2.605439229212110e-01
    5.369564274012127e-01
    7.743756471119281e-02
   -3.965352592327956e-01
   -3.116527808201618e-01
   -7.305411088989688e-02
```

```
%c
newmat =[[eye(23), A] ; [A', zeros(18)]];
b_new =[b; zeros(18,1)];
x_new =newmat\b_new;
p3=x_new(24:end);
disp(p3)
```

```
    1.351056517418814e+00
    2.364966608658973e+00
   -6.758317397578230e+00
   -2.759461419552630e+00
    8.004240205606372e+00
    1.960893053445651e+00
   -3.792084202362179e+00
   -6.635283777371361e-01
    9.634155057297451e-01
    1.316434644649746e-01
   -1.555173030096729e-01
   -2.127939236925824e-02
    2.094029215781557e-02
    1.164920703780273e-02
    2.346529563008737e-04
   -7.542914439757961e-03
   -5.770710803171587e-03
   -1.357417234750816e-03
```

```
disp(cond(A))
```

```
    1.219027935493970e+08
```

```
disp(cond(A'*A))
```

```
    1.004736786068950e+16
```

```
disp(cond(newmat))
```

```
    5.358382905257431e+14
```

```
r1 = b-A*p1;
norm1= norm(r1);
normb= norm(b);
ratio1= norm1/normb;

r2= b-A*p2;
norm2=norm(r2);
ratio2= norm2/normb;

r3= -x_new(1:23);
norm3=norm(r3);
ratio3= norm3/normb;

disp(ratio1)
```

```
    1.461985118596680e-13
```

```
disp(ratio2)
```

```
    3.524987042100492e-08
```

```
disp(ratio3)
```

```
    6.691315795447028e-10
```

When transitioning from basis 1 to basis 2:

1. **Condition Number Behavior**:

- For the first and second methods, the condition number of the matrix decreases.
- For the third method, the condition number remains approximately the same.

1. **Ratio of Norms**:

- The ratio of the norm of the residual norm(r)\text{norm}(r)norm(r) to the norm of the right-hand side norm(b)\text{norm}(b)norm(b) remains unchanged for the first method.
- For the second and third methods, this ratio increases.

# Functions

```
function x = rowforward(L,b)
    [n,~]= size(L);
    for k =1:n
        for j = 1:k-1
            b(k) = b(k) - L(k,j)*b(j);
        end
        if L(k,k) == 0
            disp('Singular matrix')
            break;
        else
            b(k)=b(k)/L(k,k);
        end
    end
    x=b;
end

function x = colbackward(U, b)
    [n, ~] = size(U);
    for i = n: -1: 1
        if U(i, i) ~= 0
            b(i) = b(i) / U(i, i);
        else
            disp('Matrix is Singular');
            break;
        end

        for j = i - 1: -1: 1
            b(j) = b(j) - U(j, i) * b(i);
        end
    end

    x = b;
end

function G = mychol(A)
    [n,~] = size(A);
    if A(1,1) <= 0
        error('Diagonal element is non-positive');
    end

    G = zeros(n,n);
    G(1,1) = sqrt(A(1,1));

    if n == 1
        return;
    end

    b = A(1,2:n).';
```

```
    A_reduced = A(2:n,2:n);
    g = G(1,2:n).';
    g = b / G(1,1);
    G(1,2:n) = g.';
    G(2:n,2:n) = mychol(A_reduced - g * g.');

    return;
end
% Define a function to compute the residual norm ratio
function ratio = computeRatio(A, p, b)
    residual = b - A * p;
    ratio = norm(residual) / norm(b);
end
```