Queston 1

```
clc;
clear all;
close all;
% 1c
```

```
function x = fast_pentadiag(a,b,c,d,e,f)
% STEP 1
n = length(f);
x = zeros(n,1);
B = zeros(n-2,1);
K = zeros(n-2,1);
for i = 1:n-2
    B(i) = b(i)/a(i);
    K(i) = d(i)/a(i);
    a(i+1) = a(i+1) - B(i)*c(i);
    c(i+1) = c(i+1) - B(i)*e(i);
    b(i+1) = b(i+1) - K(i)*c(i);
    a(i+2) = a(i+2) - K(i)*e(i);
    f(i+1) = f(i+1) - B(i)*f(i);
    f(i+2) = f(i+2) - K(i)*f(i);
end
a(n) = a(n) - (b(n - 1)/a(n - 1))*c(n-1);
f(n) = f(n) - (b(n - 1)/a(n - 1))*f(n - 1);
x(n) = f(n)/a(n);
x(n - 1) = (f(n - 1) - x(n)*c(n - 1))/a(n - 1);
%STEP 2
for i = n-2:-1:1
    x(i) = (f(i) - x(i+1)*c(i) - x(i+2)*e(i))/a(i);
end
end
```

```
%1d
```

```
N = [100, 1000]';
error_norm = zeros(length(N),1);
for n = 1:length(N)
    a = 1:1:N(n);
    b = -1/3 .*((1:1:N(n)-1) + 1);
    c = b;
    d = -1/6 .*((1:1:N(n)-2) +2);
    e = d;
    f = zeros(N(n) -4,1);
    f = vertcat([1/2, 1/6]',f);
    f(end+1) = 1/6;
    f(end+1) = 1/2;
    x_f =fast_pentadiag(a,b,c,d,e,f);
    A = diag(d,-2)+diag(b,-1)+diag(a)+diag(c,1)+diag(e,2);
    x_ge = A\f;
    error = abs(x_ge - x_f);
    error_norm(n) = norm(error,2);
```

```
end
```

```
T= table(N,error_norm,'VariableNames',{'N', 'Error norm'})
```

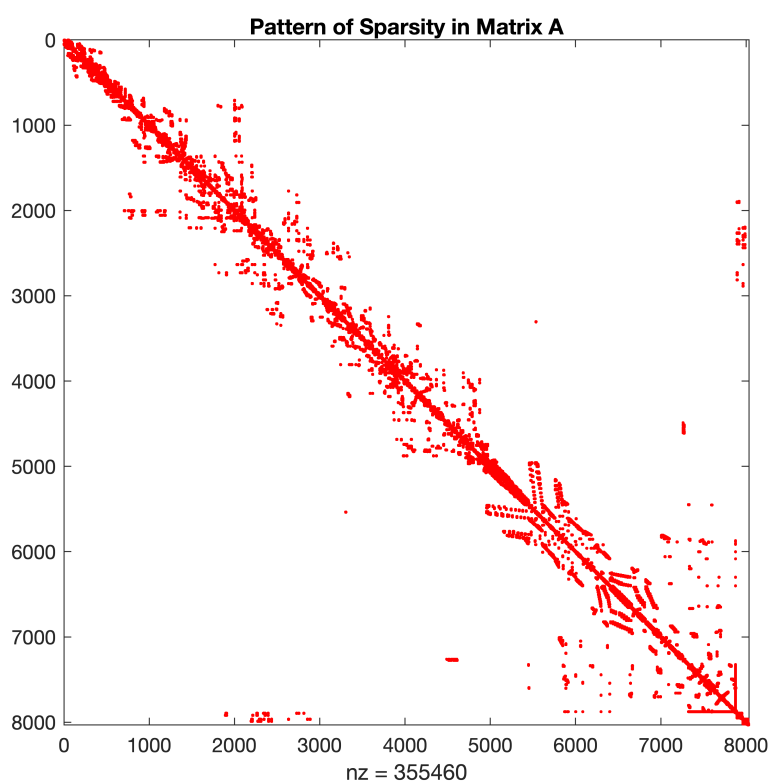
T = 2×2 table

	N	Error norm
1	100	1.2218e-13
2	1000	1.3501e-12

Queston 2

```
%2a
```

```
load bcsstk38;
A = Problem.A;
spy(A,'r')
title("Pattern of Sparsity in Matrix A")
```

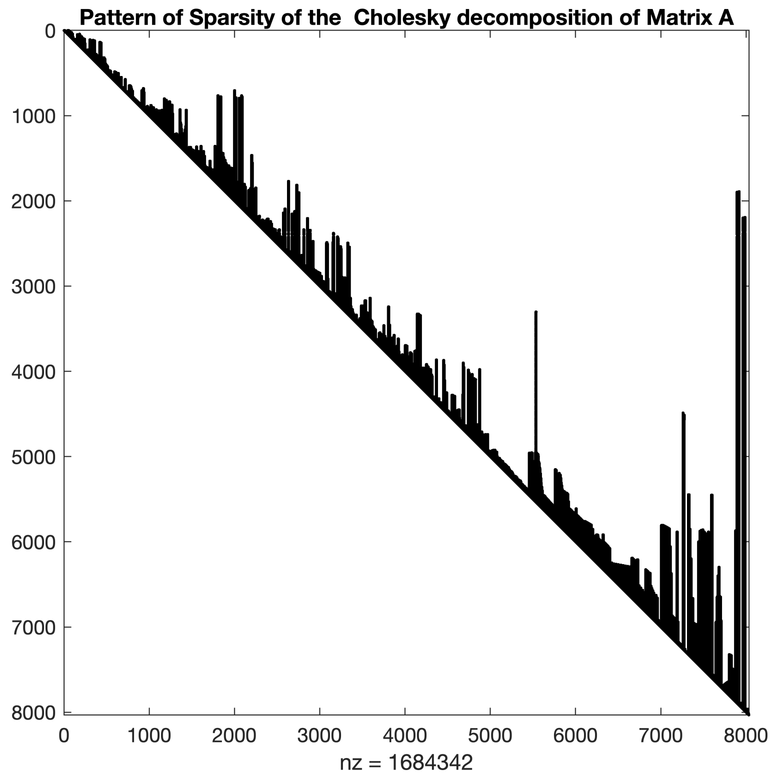



```
sparsity_ratio = 1 - (nnz(A)/numel(A))
```

sparsity_ratio = 0.9945

```
%2b
```

```
% Compute the Cholesky factorization of the matrix
R = chol(A);
spy(R,'k')
title("Pattern of Sparsity of the  Cholesky decomposition of Matrix A")
```




```
num_fillin = nnz(R)/nnz(A)
```

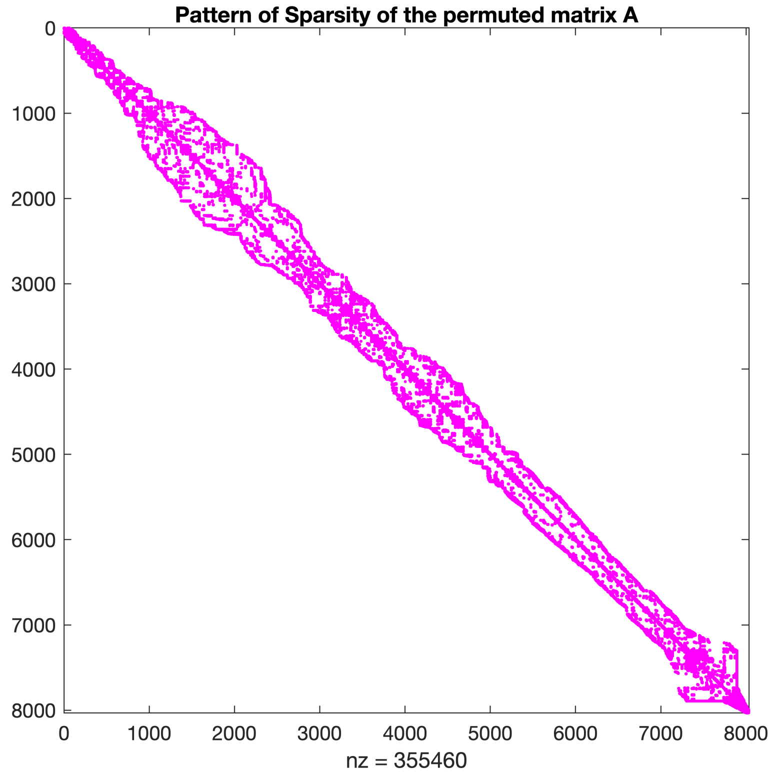
```
num_fillin = 4.7385
```

```
%2c
```

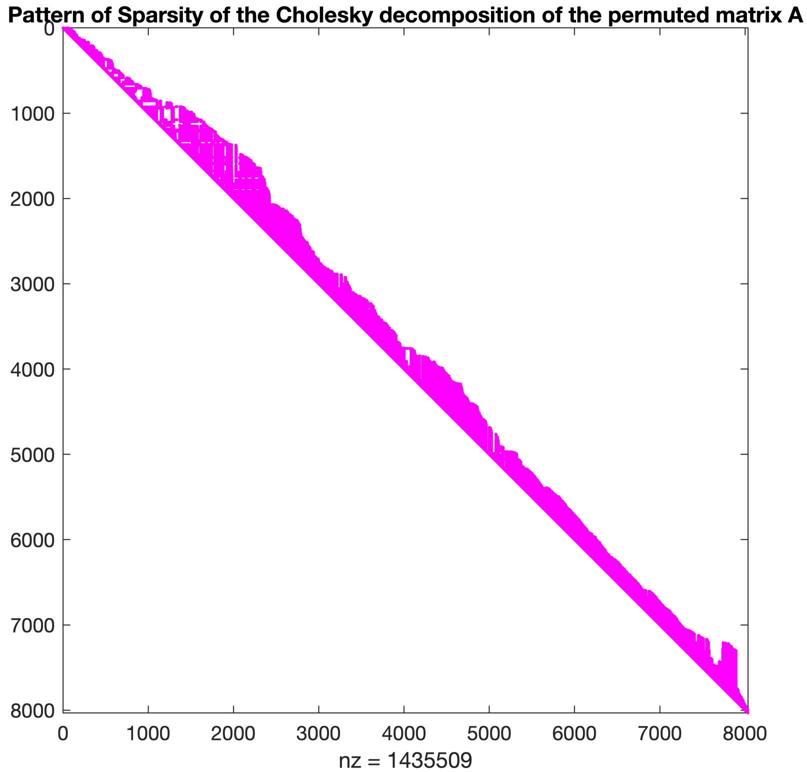
```
% Reorder the matrix using symrcm
p = symrcm(A);
B = A(p,p);

% Compute the Cholesky factorization of the reordered matrix
L = chol(B);


spy(B,'m')
title("Pattern of Sparsity of the permuted matrix A")
```




```
spy(L,'m')
title("Pattern of Sparsity of the Cholesky decomposition of the permuted matrix A")
```




```
num_fillin2  = nnz(L)/nnz(B)
```

```
num_fillin2 = 4.0385
```

Queston 3

```
%3c
```

```
n = 100;
beta = 0;

h = 1/(n+1);
a1 =-2* ones(n,1);
a2 = ones(n-1,1);

A = diag(a2,-1)+diag(a1)+diag(a2,1);
A = sparse(A);

u = zeros(n,1);
u(1) = 1;

v = 2*ones(1,n);
```

```
b = zeros(n,1);

gamma = 2/3;
b =-2* h^2 *(ones(n-2,1))';
b= [-2*h^2  - 2*gamma / h, b];

b= [b, -2*h^2 - beta]';
```

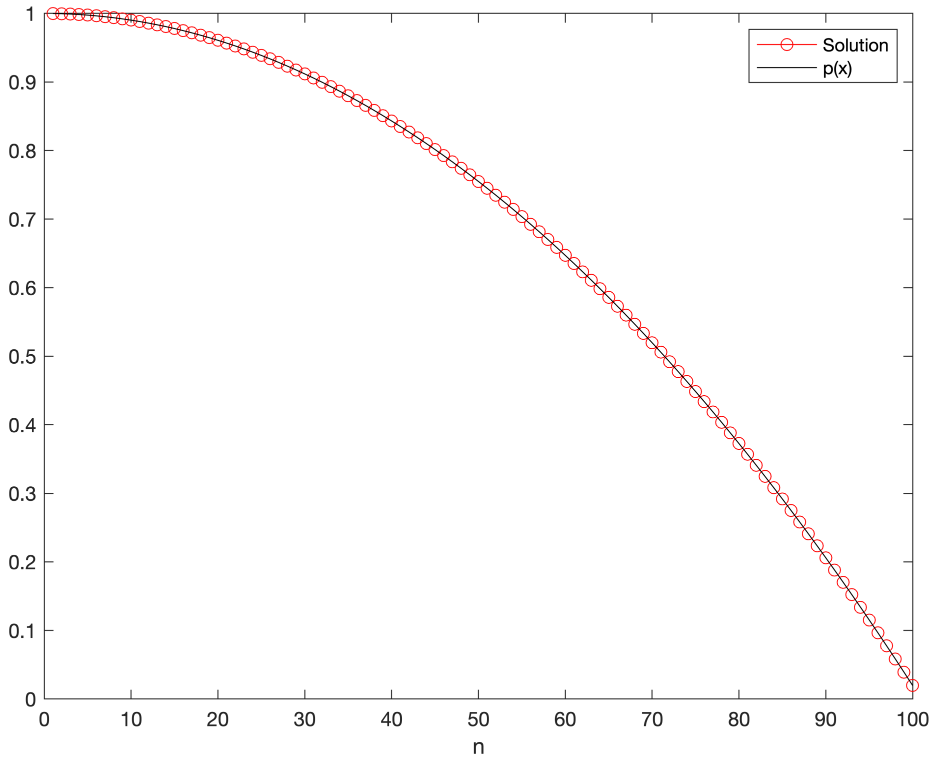
```
x = fast_algorithm(u,v,b,A)
```

```
x = 100×1
    0.9999
    0.9996
    0.9991
    0.9985
    0.9976
    0.9965
    0.9952
    0.9938
    0.9921
    0.9902
      ⋮
```

```
px =1- (h.*(1:1:n)').^2
```

```
px = 100×1
    0.9999
    0.9996
    0.9991
    0.9984
    0.9975
    0.9965
    0.9952
    0.9937
    0.9921
    0.9902
      ⋮
```

```
plot(x,'r-o','DisplayName','Solution'), hold on
plot(px,'k','DisplayName','p(x)')
xlabel('n')
legend();
```



Queston 5

```
A = [3 -1 -1 0 0;
    -1 4 -1 -1 0;
    -1 -1 5 -1 -1;
    0 -1 -1 4 -1;
    0 0 -1 -1 3];
b = [3 -5 5 -5 3]';
```

```
function [iter] = iterative_method(A,b,tol,method)
x_init = zeros(length(b),1);
iter = 0;
switch method

    case 'jacobi'
        D = diag(diag(A));
        while (true)
            zk = D\(b - A*x_init);
            x = x_init+ zk;
```

```
                if norm(b - A*x,2)< tol*norm(b,2)
                    break;
                end
                x_init = x;
                iter = iter + 1;
            end

        case 'gauss_seidel'

            n = length(b);
            x =zeros(length(b),1);

            while (true)
                  for i = 1:n
                    x(i) = (b(i) - A(i,1:i-1) * x(1:i-1) - A(i,i+1:n) * x_init(i+1:n)) / A(i,i);
                  end
                if norm(b - A*x,2)< tol*norm(b,2)
                    break;
                end
                x_init = x;
                iter = iter + 1;

            end
    end
    end
```

```
% 5a
[iter] = iterative_method(A,b,1e-6,'jacobi')
```

```
iter = 34
```

```
%5b
[iter] = iterative_method(A,b,1e-6,'gauss_seidel')
```

```
iter = 18
```

```
function x = fast_pentadiag(a,b,c,d,e,f)
% STEP 1
n = length(f);
x = zeros(n,1);
B = zeros(n-2,1);
K = zeros(n-2,1);
for i = 1:n-2
    B(i) = b(i)/a(i);
    K(i) = d(i)/a(i);
    a(i+1) = a(i+1) - B(i)*c(i);
    c(i+1) = c(i+1) - B(i)*e(i);
    b(i+1) = b(i+1) - K(i)*c(i);
    a(i+2) = a(i+2) - K(i)*e(i);
    f(i+1) = f(i+1) - B(i)*f(i);
```

```
    f(i+2) = f(i+2) - K(i)*f(i);
end
a(n) = a(n) - (b(n - 1)/a(n - 1))*c(n-1);
f(n) = f(n) - (b(n - 1)/a(n - 1))*f(n - 1);
x(n) = f(n)/a(n);
x(n - 1) = (f(n - 1) - x(n)*c(n - 1))/a(n - 1);
%STEP 2
for i = n-2:-1:1
    x(i) = (f(i) - x(i+1)*c(i) - x(i+2)*e(i))/a(i);
end
end

function [iter] = iterative_method(A,b,tol,method)
x_init = zeros(length(b),1);
iter = 0;
switch method

    case 'jacobi'
        D = diag(diag(A));
        while (true)
            zk = D\(b - A*x_init);
            x = x_init+ zk;


            if norm(b - A*x,2)< tol*norm(b,2)
                break;
            end
            x_init = x;
            iter = iter + 1;
        end

    case 'gauss_seidel'

        n = length(b);
        x =zeros(length(b),1);

        while (true)
              for i = 1:n
                x(i) = (b(i) - A(i,1:i-1) * x(1:i-1) - A(i,i+1:n) * x_init(i+1:n)) / A
              end
            if norm(b - A*x,2)< tol*norm(b,2)
                break;
            end
            x_init = x;
            iter = iter + 1;

        end
end
end
```

```
function x = fast_algorithm(u,v,b,A)

        p1 = A\b;

        p2 = A\u;

        alpha = 1 - v*p2;

        x = p1 + (1/alpha).*p2*v*p1;
end
```

Question 1

ⓐ STEP 1: Upper triangular form

for $i = 1$ to $n-2$ do

$B_i = b_i / a_i$

$K_i = d_i / a_i$

$a_{i+1} = a_{i+1} - B_i c_i$

$c_{i+1} = c_{i+1} - B_i e_i$

$b_{i+1} = b_{i+1} - k_i c_i$

$a_{i+2} = a_{i+2} - k_i e_i$

$f_{i+1} = f_{i+1} - B_i f_i$

$f_{i+2} = f_{i+2} - k_i f_i$

end for

$$a_n = a_n - \frac{b_{n-1}}{a_{n-1}} c_{n-1}$$

$$f_n = f_n - \frac{b_{n-1}}{a_{n-1}} f_{n-1}$$

$x_n = f_n / a_n$

$x_{n-1} = (f_{n-1} - x_n c_{n-1}) / a_{n-1}$

(a) STEP 2: Backward substitution.

for $i = n-2$ to 1 do

$x_i = (f_i - x_{i+1} c_i - x_{i+2} e_i) / a_i$

end for

(b) Computing the total FLOPS count

STEP 1:

$6(n-2)+3$ ~~Addition~~ subtractions

$6(n-2)+3$ Multiplications

$2(n-2)+4$ Divisions

Total for step 1 $= 14(n-2)+10 = 14n-18$

STEP 2:

$2(n-2)$ subtractions

$2(n-2)$ Multiplications

$(n-2)$ Divisions

Total for step 2 $= 5(n-2) = 5n-10$

Therefore, total number of FLOPS $= 14n-18+5n-10$

$= 19n-28$

$\in O(n)$

Question 3

(a) $Ay = c$ have a fast algorithm. We want to obtain a fast algorithm for solving $(A - uv^T)x = b$.

$Bx = b$

where $B = (A - uv^T)$

$x = B^{-1} b$

$x = (A^{-1} + \frac{1}{\alpha} A^{-1} uv^T A^{-1}) b$

$x = A^{-1} b + \frac{1}{\alpha} A^{-1} \cancel{b} u v^T A^{-1} b$ — (1)

Since we can solve $A^{-1} b$ using our fast algorithm, let $A^{-1} b = P_1$. Then (1) becomes

$$x = P_1 + \frac{1}{\alpha} A^{-1} u v^T P_1$$

Similarly, we can solve $A^{-1} u$ with our fast algorithm

$$x = P_1 + \frac{1}{\alpha} P_2 v^T P_1 \quad -(2)$$

$$\alpha = 1 - v^T \underbrace{A^{-1} u}_{= P_2} = 1 - v^T P_2 \quad -(3)$$

From (3), since P is a vector, the scalar product $V^T p \in O(n)$ and the result from evaluating (3) is a scalar

From (2)

$$x = P_1 + \frac{1}{\alpha} P_2 \underbrace{v^T P_1}_{=} \quad -(4)$$

scalar product ($\in O(n)$)

(3b) $(A - uv^T)x = b$

where $u \in \mathbb{R}^{n \times 1}$ and $v \in \mathbb{R}^{1 \times n}$.

Therefore, $uv^T = \begin{bmatrix} u_1v_1 & u_1v_2 & \cdots & u_1v_n \\ u_2v_1 & u_2v_2 & \cdots & u_2v_n \\ \vdots & & \ddots & \vdots \\ u_nv_1 & u_nv_2 & \cdots & u_nv_n \end{bmatrix}$

Comparing this matrix with what is given in the question, we can deduce that

$$u = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad v^T = [2, 2, 2, \cdots, 2]$$

Question 4

(b) $Ax = b$

$A = D - L - U \implies U = D - L - A$

$(D-L)x^{k+1} = Ux^k + b$

$$x^{k+1} = \underbrace{(D-L)^{-1}U}_{=R_G}x^k + (D-L)^{-1}b$$

$$R_G = (D-L)^{-1}U = (D-L)^{-1}(D-L-A)$$
$$= (D-L)^{-1}(D-L) - (D-L)^{-1}A$$
$$= I - (D-L)^{-1}A$$

4a $A = D - L - U$

$Ax = b$

$Dx^{i+1} = (L+U)x^{i} + b$

$x^{i+1} = D^{-1}(L+U)x^{i} + D^{-1}b$

Let $R_{g} = D^{-1}(L+U)$

where D is of the form $\begin{bmatrix} a_{11} & & & 0 \\ & a_{22} & & \\ & 0 & \ddots & \\ & & & a_{nn} \end{bmatrix}$

L is of the form $\begin{bmatrix} 0 & & & & 0 \\ a_{21} & 0 & & & \\ \vdots & \ddots & 0 & & \\ \vdots & & \ddots & \ddots & \\ a_{n1} & \cdots & \cdots & a_{n,n-1} & 0 \end{bmatrix}$

and $U = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ & 0 & \ddots & \vdots \\ & & 0 & \vdots \\ & 0 & \ddots & a_{n-1,n} \\ & & & 0 \end{bmatrix}$

$D^{-1} = \begin{bmatrix} 1/a_{11} & & & 0 \\ & 1/a_{22} & & \\ & 0 & \ddots & \\ & & & 1/a_{nn} \end{bmatrix}$

$$R_J = D^{-1}(L+U)$$

$$R_J = \begin{bmatrix} 1/a_{11} & & & \\ & 1/a_{22} & & \\ & & \ddots & \\ & & & 1/a_{nn} \end{bmatrix} \left(\begin{bmatrix} 0 & & & 0 \\ a_{21} & 0 & & \\ \vdots & & \ddots & \\ a_{n1} & \cdots & a_{nn-1} & 0 \end{bmatrix} + \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ & 0 & \ddots & \vdots \\ & & \ddots & a_{n-1n} \\ 0 & & & 0 \end{bmatrix} \right)$$

$$R_J = \begin{bmatrix} 0 & a_{12}/a_{11} & \frac{a_{13}}{a_{11}} & \cdots & a_{1n}/a_{11} \\ \frac{a_{21}}{a_{22}} & 0 & \frac{a_{23}}{a_{22}} & & a_{2n}/a_{22} \\ \vdots & & \ddots & & \vdots \\ & & & & \frac{a_{n-1n}}{a_{n-1n-1}} \\ \frac{a_{n1}}{a_{nn}} & \cdots & & \frac{a_{nn-1}}{a_{nn}} & 0 \end{bmatrix}$$

We know that the Jacobi method requires the matrix to be diagonally dominant, i.e the magnitude of the diagonal element in a row be greater than or equal to the sum of the magnitudes of all other non-diagonal elements in that row for each row of the matrix. Also, we note that each row of R_J is scaled or divided by the diagonal element. So $\frac{a_{ij}}{a_{ii}} < 1$; $i \neq j$; $i, j = 1, 2, \ldots, n$

Therefore, $\|R_J\|_\infty < 1$