1. (a) Write a Matlab code `diagdom.m` to check whether a given matrix $A$ is strictly diagonally dominant or not. Use your code to determine whether the following two matrices are strictly diagonally dominant.

(i) 

$$
A = \begin{bmatrix}
28 & -6 & 4 & 1 & -2 & -5 & 8 & 0 \\
-4 & 26 & -1 & 4 & 0 & 4 & 1 & 4 \\
1 & -6 & 26 & -5 & 1 & -1 & -6 & 0 \\
-5 & -6 & 1 & 21 & 0 & -3 & 2 & 2 \\
4 & 1 & 3 & -3 & 17 & 0 & -3 & 3 \\
4 & -2 & 3 & 0 & 2 & 14 & 0 & 1 \\
1 & 2 & 4 & 3 & -2 & 1 & 17 & 4 \\
1 & 3 & 1 & 3 & 0 & -1 & 3 & 15 \\
\end{bmatrix}
$$

(ii) 

$$
A = \begin{bmatrix}
8 & 1 & 0 & 0 & 1 & 3 & 0 & 2 \\
-2 & 17 & -2 & 3 & 2 & 4 & 1 & 2 \\
-2 & 4 & 18 & -3 & 0 & -2 & 1 & 4 \\
-1 & 1 & 9 & 28 & 4 & 5 & 3 & 2 \\
-5 & -1 & 6 & -3 & 27 & 3 & 6 & 1 \\
4 & 5 & 2 & 2 & 21 & 2 & 3 & 3 \\
3 & 0 & -3 & 2 & -2 & 0 & 14 & 2 \\
1 & -1 & -2 & 2 & 0 & 0 & 3 & 11 \\
\end{bmatrix}
$$

(b) For both matrices, determine the $LU$-decomposition by the Crout factorization algorithm, if such a decomposition exists.

Sol 1 (a):

```matlab
1 n=length(A);
2 AA=A;
3 dd=abs(diag(AA));
4 for i=1:n
5 AA(i,i)=0;
6 end;
7 SS=sum(abs(AA),2);
8 for i=1:n
9 if dd(i)<=SS(i)
10 disp(['A not strictly diagonally dominant in row ', num2str(i)])
11 return
12 end
13 end
14 disp(['A strictly diagonally dominant'])
```

(b)
It is important to keep in mind that strict diagonal dominance is merely a sufficient and not a necessary condition for existence of an $LU$-decomposition. As the second example shows, an $LU$-decomposition can exist even when the matrix $A$ is not strictly diagonally dominant.

2. (a) Numerically determine whether the following two symmetric matrices $A$ are positive-definite and, if so, calculate the Cholesky factor $L$ :

$$A = \begin{bmatrix} 6 & 6 & 6 & 12 & 0 & 12 & 6 & 0 \\ 6 & 7 & 7 & 15 & 2 & 13 & 7 & 1 \\ 6 & 7 & 9 & 15 & 4 & 19 & 9 & 1 \\ 12 & 15 & 15 & 36 & 9 & 27 & 18 & 3 \\ 0 & 2 & 4 & 9 & 13 & 12 & 7 & 6 \\ 12 & 13 & 19 & 27 & 12 & 49 & 21 & 5 \\ 6 & 7 & 9 & 18 & 7 & 21 & 12 & 1 \\ 0 & 1 & 1 & 3 & 6 & 5 & -1 & 4 \end{bmatrix}$$

(i) $A = \begin{bmatrix} 6 & 6 & 6 & 12 & 0 & 12 & 6 & 0 \\ 6 & 7 & 7 & 15 & 2 & 13 & 7 & 1 \\ 6 & 7 & 9 & 15 & 4 & 19 & 9 & 1 \\ 12 & 15 & 15 & 36 & 9 & 27 & 18 & 3 \\ 0 & 2 & 4 & 9 & 13 & 12 & 7 & 6 \\ 12 & 13 & 19 & 27 & 12 & 49 & 21 & 5 \\ 6 & 7 & 9 & 18 & 7 & 21 & 12 & 1 \\ 0 & 1 & 1 & 3 & 6 & 5 & -1 & 4 \end{bmatrix}$

(ii) $A = \begin{bmatrix} 6 & 6 & 6 & 12 & 0 & 12 & 6 & 0 \\ 6 & 7 & 7 & 15 & 2 & 13 & 7 & 1 \\ 6 & 7 & 9 & 15 & 4 & 19 & 9 & 1 \\ 12 & 15 & 15 & 36 & 9 & 27 & 18 & 3 \\ 0 & 2 & 4 & 9 & 13 & 12 & 7 & 6 \\ 12 & 13 & 19 & 27 & 12 & 49 & 21 & 5 \\ 6 & 7 & 9 & 18 & 7 & 21 & 12 & 1 \\ 0 & 1 & 1 & 3 & 6 & 5 & -1 & 4 \end{bmatrix}$

(b) Find an example of a non-symmetric $2 \times 2$ real matrix which is positive-definite.

Sol 2 (a):


(b): We wish to show that \( x^T A x > 0 \) for all \( x \neq 0 \). Consider the \( 2 \times 2 \) matrix:

\[
\begin{pmatrix}
1 & 1 \\
0 & 1
\end{pmatrix}
\]

This is obviously not symmetric. However, it is positive-definite since

\[
x^T \begin{pmatrix}
1 & 1 \\
0 & 1
\end{pmatrix} x = x_1^2 + x_1 x_2 + x_2^2 = \frac{1}{2} (x_1 + x_2)^2 + \frac{1}{2} x_1^2 + \frac{1}{2} x_2^2 \geq \frac{1}{2} \| x \|^2 > 0 \quad \text{for } x \neq 0.
\]

Thus, in all cases, \( x^T A x > 0 \) and \( A \) is positive-definite.

3. (a) Modify the Cholesky algorithm in code demo_alg066.m to obtain a code choleskysolve which solves linear systems \( Ax = b \) when \( A \) is self-adjoint, positive-definite. See the suggestions in section 6.6 of Burden & Faires, 10th Ed.

(b) Show that the optimal operation counts for your algorithm are as follows:

- \( \frac{1}{2} n^3 + \frac{3}{2} n^2 + \frac{1}{2} n \) multiplications/divisions
- \( \frac{1}{2} n^3 + n^2 - \frac{1}{2} n \) additions/subtractions
- \( n \) square roots

(c) Use your code to solve \( Ax = b \) for

\[
A = \begin{pmatrix}
4 & 2 & 6 & 2 & 6 & 2 & 0 & 2 & 2 & 6 \\
2 & 10 & 6 & 10 & 5 & 11 & 7 & 0 & 9 & 9 \\
2 & 10 & 10 & 63 & 13 & 9 & 13 & 14 & 7 & 17 \\
6 & 3 & 5 & 13 & 33 & -9 & -8 & 19 & 5 & 25 \\
2 & 1 & 11 & 9 & -9 & 19 & -7 & 11 & 1 & 10 \\
0 & 9 & 7 & 13 & -8 & 9 & 39 & -8 & -11 & -3 \\
2 & 4 & 0 & 14 & 19 & -7 & -8 & 35 & 6 & 19 \\
2 & 1 & 9 & 7 & 5 & 11 & -11 & 6 & 25 & 11 \\
6 & 3 & 9 & 17 & 25 & 1 & -3 & 19 & 11 & 34
\end{pmatrix}, \quad b = \begin{pmatrix}
12 \\
24 \\
26 \\
54 \\
32 \\
78 \\
26 \\
21
\end{pmatrix}
\]

Sol 3 (a): Below is the code choleskysolve:

```matlab
% disp(['tril(A)='), disp(single(tril(A)))
[disp('tril(A)='), disp(single(tril(A)))
pause
N=length(A);
% STEP 1
if A(1,1) <= 0
disp('A not positive definite')
return
end
A(1,1) = sqrt(A(1,1));
single(tril(A))
pause
% STEP 2
for J = 2 : N
A(J,1) = A(J,1)/A(1,1);
single(tril(A))
pause
end;
% STEP 3
NN = N-1;
for I = 2 : NN
% STEP 4
KK = I-1;
S = 0;
for K = 1 : KK
S = S - A(I,K)*conj(A(I,K));
end;
if A(I,I) + S <= 0
disp('A not positive definite')
return
end
A(I,I) = sqrt(A(I,I) + S);
single(tril(A))
pause
% STEP 5
JJ = I+1;
for J = JJ : N
A(J,I) = (A(J,I) + S)/A(I,I);
single(tril(A))
pause
end;
% disp(['tril(A)='), disp(single(tril(A)))
```
end;
47  % STEP 6
48  S = O;
49  for K = 1 : NN
50    S = S-A(N,K)*conj(A(N,K));
51  end;
52  A(N,N) = sqrt(A(N,N)+S);
53  single(tril(A))
54  pause
55  % SOLVE AX=b FOR X
56  X=b
57  pause
58  % STEP 8
59  X(1)=X(1)/A(1,1);
60  X=X
61  pause
62  % STEP 9
63  for I=2:N
64    J=I-1;
65    for J=1:J
66      X(I)=X(I)-A(I,J)*X(J);
67    end
68    X(I)=X(I)/A(I,I);
69  end
70  X=X
71  pause
72  % STEP 10
73  X(N)=X(N)/A(N,N);
74  X=X
75  pause
76  % STEP 11
77  for I=N-1:-1:1
78    J=I+1;
79    for J=I:N
80      X(I)=X(I)-A(J,I)*X(J);
81    end
82    X(I)=X(I)/A(I,I);
83  end
84  X=X
85  pause
86  L=tril(A);
87  disp('L='), disp(single(L))
88  disp('X='), disp(single(X))
We count the flops in each step of the algorithm:

**Cholesky Factorization:**

**Step # 1:** SR : 1

**Step # 2:** MD : \((n - 1)\)

**Step # 3:** 0

**Step # 4:** SR : \((n - 2)\)

\[
\text{AS} : (n - 1) + \sum_{i=2}^{n-1} \sum_{k=1}^{i-2} 1 = \frac{1}{2}(n^2 - 3n + 2)
\]

**Step # 5:** MD/AS: \[
\sum_{i=2}^{n-1} \sum_{j=i+1}^{n} \sum_{k=1}^{i-2} 1 = \frac{1}{6}(n^3 - 3n^2 + 2n)\]

\[
\text{AS} : \sum_{i=2}^{n-1} \sum_{j=i+1}^{n} \sum_{k=1}^{i-1} 1 = \frac{1}{6}(n^3 - 6n^2 + 11n - 6)
\]

**Step # 6:** SR : 1

\[
\text{AS} : 1 + \sum_{k=1}^{n-2} 1 = (n - 2) + 1 = n - 1
\]

**Step # 7:** 0

**Step # 8:** MD : 1

**Step # 9:** MD/AS : \((n - 1)\)

\[
\text{AS} : \sum_{i=2}^{n} \sum_{j=i+1}^{n} \sum_{k=1}^{i-1} 1 = \frac{1}{2}(n^2 - 3n + 2)
\]

**Step # 10:** MD : 1

**Step # 11:** same # as Step # 9

Therefore, in total for the Cholesky Factorization we require

- Multiplications/divisions: \(\frac{1}{6}(n^3 + 3n^2 - 4n)\)
- Additions/subtractions: \(\frac{1}{6}(n^3 - n)\)
- Squareroots: \(n\)

and for solving the linear system:

- Multiplications/divisions: \(n^2 + n\)
- Additions/subtractions: \(n^2 - n\)

so the grand total of flops is:

- Multiplications/divisions: \(\frac{1}{6}n^3 + \frac{3}{2}n^2 + \frac{1}{3}n\)
- Additions/subtractions: \(\frac{1}{6}n^3 + n^2 - \frac{7}{6}n\)
- Squareroots: \(n\)

(c):
4. The \( LDU \)-decomposition of a square matrix \( A \) writes it, if possible, as \( A = L D U \) where \( L \) is lower-triangular with all 1's on the diagonal, \( D \) is a diagonal matrix with non-zero entries, and \( U \) is upper-triangular with all 1's on the diagonal.

(a) Show that the \( LDU \)-decomposition exists if and only if Gaussian elimination without pivots succeeds for the matrix \( A \) and, furthermore, the factors \( L, D, \) and \( U \) are then unique. In fact, \( D \) consists of the pivot elements in Gaussian elimination.

(b) If the matrix \( A \) is self-adjoint, then show furthermore that \( U = L^* \).

(c) If all of the pivot elements in Gaussian elimination are positive for some self-adjoint matrix \( A \), then use parts (a),(b) to show that \( A = L L^* \) for some lower-triangular matrix \( L \) or some upper-triangular matrix \( U \), with \( U = L^* \).

Hint: Show that \( D = S S^* \) for some diagonal matrix \( S \).

(d) Use part (c) to show that if all of the pivot elements in Gaussian elimination are positive for some self-adjoint matrix \( A \), then \( A \) is a positive-definite matrix.

(a) Proof: It was proved in class and in the textbook that Gaussian elimination without pivots succeeds for the matrix \( A \) if and only if it has an \( LU \)-decomposition of the form \( A = L U \), where \( L \) is lower-triangular with all 1's on the diagonal and \( U \) is upper-triangular with the pivot elements \( u_{kk} = u_{kk}, k = 1, \ldots, n \) on the diagonal. If such an \( LU \)-decomposition exists then define \( D = \text{diag}(u_{11}, u_{22}, \ldots, u_{nn}) \), so that

\[
U = \begin{pmatrix}
    u_{11} & u_{12} & u_{13} & \cdots & \cdots & \cdots & \cdots & \cdots & u_{1n} \\
    0 & u_{22} & u_{23} & \cdots & \cdots & \cdots & \cdots & \cdots & u_{2n} \\
    \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
    \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
    0 & 0 & 0 & \cdots & u_{n-1,n-1} & u_{n-1,n} & u_{n-1,n} & u_{nn} \\
\end{pmatrix}
\]

\( = \begin{pmatrix}
    u_{11} & 0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\
    0 & u_{22} & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\
    0 & 0 & u_{33} & \cdots & \cdots & \cdots & \cdots & \cdots & 0 \\
    \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
    0 & 0 & 0 & \cdots & u_{n-1,n-1} & 0 & u_{n-1,n} & u_{nn} \\
\end{pmatrix}
\]

\[ \times \begin{pmatrix}
    1 & u_{12}/u_{11} & u_{13}/u_{11} & \cdots & \cdots & \cdots & \cdots & \cdots & u_{1n}/u_{11} \\
    0 & 1 & u_{23}/u_{22} & \cdots & \cdots & \cdots & \cdots & \cdots & u_{2n}/u_{22} \\
    0 & 0 & 1 & \cdots & \cdots & \cdots & \cdots & \cdots & u_{33}/u_{33} \\
    \vdots & \vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\
    0 & 0 & 0 & \cdots & 1 & u_{n-1,n}/u_{n-1,n} & u_{n-1,n} & u_{nn} \\
\end{pmatrix} \]

\[ = D \hat{U} \]  

where \( \hat{U} \) is now upper-triangular with all 1's along the diagonal and we used the fact that left-multiplication of a matrix by a diagonal matrix just rescales the rows of the matrix. This gives the \( LDU \)-decomposition \( A = L D U \). Conversely, if the \( LDU \)-decomposition is given, then define \( U = D \hat{U} \) to obtain the \( LU \)-decomposition \( A = L U \).

To see that the \( A = L D U \) decomposition is unique, note that if there exists another decomposition \( A = L' D' U' \) then it must be the case that:

\[ (L')^{-1} L D = (L')^{-1} L U^{-1} = D' U^{-1} \]

The left-most matrix is lower triangular but the right-most is upper, so that both must be diagonal! Moreover, since \( L_{ii} = (L')_{ii}^{-1} = 1 \) and likewise \( U_{ii} = U'_{ii} = U''_{ii} = 1 \), it follows that:

\[ (L')^{-1} L = I, \quad U^{-1} U' = I, \quad D = D'. \]

This yields the desired uniqueness result, i.e. \( D = D' \), \( U = U' \) and \( L = L' \).

(b) If \( A = A^* \), then \( U' D' L' = L D U \) and since we know the representation is unique by part (a), we know that the upper triangular matrices \( U = U' \). We also deduce that \( D \) is self-adjoint.

(c) Note that any diagonal matrix \( D \) with all positive entries can be written as \( S S^* \) for another diagonal matrix \( S \) with entries \( \sqrt{a_{ii}}, i = 1, \ldots, n \). Then

\[ A = L D U = L S S^* L' = (L S)(L S)^* = L L^* \]
with \( \mathbf{L S} \equiv \mathbf{L} \) where \( \mathbf{L} \) is lower triangular since it is the product of two lower triangular matrices. If we wish, we can recast this as:
\[
\mathbf{A} = \mathbf{U}^* \mathbf{U}
\]
where \( \mathbf{U} \) is invertible.

(d) By part (c)
\[
\mathbf{x}^T \mathbf{Ax} = \mathbf{x}^T \mathbf{U}^* \mathbf{U} \mathbf{x} = \| \mathbf{U} \mathbf{x} \|^2 \geq 0
\]
and = 0 iff \( \mathbf{U} \mathbf{x} = 0 \). Since \( \mathbf{U} \) has the non-zero elements of \( \mathbf{S} \) along its diagonal and these are all positive, \( \mathbf{U} \) is invertible and thus \( \mathbf{U} \mathbf{x} = 0 \) iff \( x = 0 \). Therefore \( \mathbf{A} \) is positive definite.

5. (a) Write a code \( \text{LDUFact} \) which obtains, if possible, the \( \text{LDU} \)-decomposition of a given square matrix \( \mathbf{A} \). To do this, modify the direct \( \text{LU} \)-factorization code \( \text{demo_alg064.m} \) in a manner similar to the way the Cholesky factorization code \( \text{demo_alg066.m} \) was modified to obtain the \( \text{LDL}^* \)-factorization code \( \text{demo_alg065.m} \).

(b) Use your code to determine, if possible, the \( \text{LDU} \)-decompositions of the two symmetric matrices in Problem 5-2. (Note that your code from (a) could be rewritten to be more efficient for symmetric matrices, but you need not do this.) Determine from your \( \text{LDU} \)-decomposition results whether the two matrices are positive-definite and explain your reasoning.

Sol 5 (a):

```matlab
1 disp('A='), disp(single(A))
2 pause
3 N=length(A);
4 TRUE = 1;
5 FALSE = 0;
6 OK=TRUE;
7 % STEP 1
8 if abs(A(1,1)) <= 1.0e-20
9   OK = FALSE;
10 else
11   % the entries of L below the main diagonal will be placed
12   % in the corresponding entries of A; the entries of U
13   % above the main diagonal will be placed in the
14   % corresponding entries of A; the diagonal matrix D will
15   % become the main diagonal of A;
16   % STEP 2
17   for J = 2 : N
18     % first row of U
19     A(1,J) = A(1,J)/A(1,1);
20     % first column of L
21     A(J,1) = A(J,1)/A(1,1);
22     single(A)
23     pause
24   end;
25   % STEP 3
26   M = N-1;
27   I = 2;
28   while I <= M & OK == TRUE
29     % STEP 4
30     KK = I-1;
31     for K = 1 : KK
32       VV(K)=A(I,K)*A(K,K);
33       V(K)=A(K,K)*A(K,I);
34       A(I,I) = A(I,I)-VV(K)*A(K,I);
35     end;
36     single(A)
37     pause
38     if abs(A(I,I)) <= 1.0e-20
39       OK = FALSE;
40     else
41       % STEP 5
42       JJ = I+1;
43       for J = JJ : N
44         SS = 0;
45         S = 0;
46         for K = 1 : KK
47           SS = SS-VV(K)*A(K,J);
48           S = S-A(J,K)*V(K);
49         end;
50         % Ith row of U
51         A(I,J) = (A(I,J)+SS)/A(I,I);
52         % Ith column of L
53         A(J,I) = (A(J,I)+S)/A(I,I);
54         single(A)
55         pause
56       end;
57     end;
58   end;
59   if OK == TRUE
60     % STEP 6
61     for K = 1 : M
62       A(N,N) = A(N,N)-A(N,K)*A(K,N);
63     end;
64     single(A)
65     pause
66     if abs(A(N,N)) <= 1.0e-20
67       OK = FALSE;
68   end;
```

% If A(N,N) = 0 then A = LU but the matrix is singular.
% Process is complete, all entries of A have been determined.
% STEP 7
L=tril(A,-1)+eye(N,N); U=triu(A,1)+eye(N,N); D=diag(diag(A));
disp(‘L=‘), disp(single(L))
disp(‘D=‘), disp(single(D))
disp(‘U=‘), disp(single(U))

end;

if OK == FALSE
fprintf(1,'The matrix does not have an LDU factorization.\n');
end;

(b):

A=[ 6 6 6 12 0 12 6 0; 6 7 7 15 2 13 7 1; 6 7 9 15 4 19 9 1; 12 15 15 36 9 27 18 3; 0 2 4 9 13 12 7 6; 12 13 19 27 12 49 21 5; 6 7 9 18 7 21 12 -1; 0 1 1 3 6 5 -1 4]
disp(‘running LDUfact.m gives:’)

L=[ 1 0 0 0 0 0 0 0; 1 1 0 0 0 0 0 0; 1 1 1 0 0 0 0 0; 2 3 0 1 0 0 0 0; 0 2 1 1 1 0 0 0; 2 1 3 0 1 1 0 0; 1 1 1 0 1 1 0 0; 0 1 0 0 1 0 1 1] D=[ 6 0 0 0 0 0 0 0; 0 1 0 0 0 0 0 0; 0 0 2 0 0 0 0 0; 0 0 0 3 0 0 0 0; 0 0 0 0 4 0 0 0; 0 0 0 0 0 2 0 0; 0 0 0 0 0 0 -2 0; 0 0 0 0 0 0 0 1] U=[ 1 1 1 2 0 2 1 0; 0 1 1 3 2 1 1 1; 0 0 1 0 1 3 1 0; 0 0 0 1 1 0 1 0; 0 0 0 0 0 1 1 0; 0 0 0 0 0 0 1 1; 0 0 0 0 0 0 0 1] disp(‘One element of D is negative, so A is not positive-definite’) pause

A=[ 4 2 8 -2 6 4 2 -2; 2 2 2 0 2 3 3 -2; 8 2 45 9 44 21 0 8; -2 0 9 20 11 14 1 6; 6 2 44 11 63 29 5 18; 4 3 21 14 29 21 9 7; 2 3 0 1 5 9 16 2; -2 -2 8 6 18 7 2 20]
disp(‘running LDUfact.m gives:’)

L=[ 1 0 0 0 0 0 0 0; 1/2 1 0 0 0 0 0 0; 2 -2 1 0 0 0 0 0; -1/2 1 3/5 1 0 0 0 0; 3/2 -1 6/5 -1/3 1 0 0 0; 1 1 3/5 2/3 1/2 1 0 0; 1/2 1 0 0 0 0 0 0; -1/2 -1 2/5 0 1/2 0 3 1] D=[ 4 0 0 0 0 0 0 0; 0 1 0 0 0 0 0 0; 0 0 25 0 0 0 0 0; 0 0 0 9 0 0 0 0; 0 0 0 0 16 0 0 0; 0 0 0 0 0 1 0 0; 0 0 0 0 0 0 1 0; 0 0 0 0 0 0 0 1] U=[ 1 1/2 2 -1/2 3/2 1 1/2 -1/2; 0 1 -2 1 -1 1 2 -1; 0 0 1 3/5 6/5 3/5 0 2/5; 0 0 0 1/2 1/2 1/2 0 0]
6. Consider functions \( f(u) \) on the unit interval \( u \in [0,1] \) such that \( f(0) = f(1) = 0 \). If the interval is discretized with \( u_i = i/n \) for \( i = 0, 1, \ldots, n \) and the function \( f \) is represented by its interior values \( f_i = f(u_i) \), \( i = 1, \ldots, n-1 \), then the second-derivative operator \(-d^2/du^2\) can be approximated by

\[
-f''(u_i) \approx n^2[-f_{i-1} + 2f_i - f_{i+1}], \quad i = 1, \ldots, n-1.
\]

The second-order differential equation \(-f''(u) = b(u)\) for a given function \( b \) with \( b(0) = b(1) = 0 \) is thus approximated by a linear system of equations

\[
n^2[-f_{i-1} + 2f_i - f_{i+1}] = b_i, \quad i = 1, \ldots, n-1,
\]

with \( b_i = b(u_i), i = 1, \ldots, n-1 \).

(a) Show that this can be written as a linear equation \( Af = b \) for an \( n \times n \) symmetric, tridiagonal matrix \( A \) and give the diagonal elements \( a_{ii} \), \( i = 1, \ldots, n \) and off-diagonal elements \( c_i, i = 1, \ldots, n-1 \) of this matrix.

(b) When a tridiagonal matrix is also symmetric, then the Crout factorization employed in Algorithm 6.7 is not the most efficient one. Instead, one can use a compact version of an \( LDL^T \)-factorization taking advantage of the fact that \( U = L^T \), which makes it unnecessary to calculate and store the elements of \( U \). For given \( n \)-dimensional diagonal \( a \) and \((n-1)\)-dimensional off-diagonal \( c \) of \( A \), write a Matlab code \texttt{symtridiLDLt.m} to obtain its \( LDL^T \)-factorization. Your code should output the \( n \)-dimensional diagonal \( d \) of \( D \) and the \((n-1)\) off-diagonal \( l \) of \( L \). Try to minimize the number of arithmetic operations employed. Also write your code to then solve linear equations of the form \( Ax = b \) by forward substitution with \( L \), directly inverting \( D \), and backward substitution with \( L^T \).

(c) Taking \( n = 31 \) and \( b(u) = -u \ln u \), use your code \texttt{symtridiLDLt.m} to solve the system in (a) for the vector \( f \). Compare your solution \( f \) of the linear system with the discrete values \( f_i = f(u_i) \), \( i = 1, \ldots, n \) for the exact solution \( f(u) = \frac{u}{n} \ln n - \frac{1}{n} (u^2 - u) \) of the differential equation \(-f''(u) = b(u)\).

\[
\text{Sol 6 (a): The matrix } A \text{ is given (element-wise) by}
\]

\[
A_{ii} = 2n^2 \quad \text{for } i = 1, 2, \ldots, n-1
\]
\[
A_{i,i+1} = -n^2 \quad \text{for } i = 1, 2, \ldots, n-2
\]
\[
A_{i,i-1} = -n^2 \quad \text{for } i = 2, \ldots, n-1
\]
\[
A_{ij} = 0 \quad \text{for } j \neq (i \pm 1)
\]

The direct matrix multiplication with the \((n-1) \times (n-1)\) matrix:

\[
A = n^2
\]

\[
\begin{pmatrix}
2 & -1 & 0 & 0 & 0 & 0 & 0 \\
-1 & 2 & -1 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & \ddots & \ddots & 0 & 0 \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & 0 \\
\vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
0 & \ddots & \ddots & \ddots & \ddots & \ddots & -1 \\
0 & 0 & \ddots & \ddots & \ddots & \ddots & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 2
\end{pmatrix}
\]

shows that we can write the second-order difference equation as the linear equation \( Af = b \).
% solve Ax=b by forward & backward substitution
x=bb;
for i=2:n
x(i)=x(i)-l(i-1)*x(i-1);
x(i-1)=x(i-1)/d(i-1);
end
x(n)=x(n)/d(n);

for i=n-1:-1:1
x(i)=x(i)-l(i)*x(i+1);
end

(c):

n=31;
c=-ones(1,n-1)+(n+1)*(n+1); a=2*ones(1,n)*(n+1)*(n+1);
u=(1:n)/(n+1);
bb=-u.*log(u);
f=@(u) u.^3.*log(u)/6-5*(u.^3-u)/36;
A=full(gallery('tridiag',c,a,c))
disp('Running symtridiLDLt.m gives:'),
d=[2.0480 1.5360 1.3653 1.2800 1.2288 1.1947 1.1703 ... 1.1520 1.1378 1.1264 1.1171 1.1093 1.1028 1.0971 ... 1.0923 1.0880 1.0842 1.0809 1.0779 1.0752 1.0728 ... 1.0705 1.0685 1.0667 1.0650 1.0634 1.0619 1.0606 ... 1.0593 1.0581 1.05701e3
l=[-0.5000 -0.6667 -0.7500 -0.8000 -0.8333 -0.8571 -0.8750 ... -0.8889 -0.9000 -0.9091 -0.9167 -0.9231 -0.9286 -0.9333 ... -0.9375 -0.9412 -0.9444 -0.9474 -0.9500 -0.9524 -0.9545 ... -0.9565 -0.9583 -0.9600 -0.9615 -0.9630 -0.9643 -0.9655 ... -0.9667 -0.9687];
x=[ 0.0043 0.0085 0.0126 0.0164 0.0200 0.0233 0.0263 ... 0.0290 0.0313 0.0333 0.0349 0.0362 0.0371 0.0376 ... 0.0378 0.0377 0.0372 0.0364 0.0352 0.0338 0.0321 ... 0.0301 0.0278 0.0254 0.0227 0.0198 0.0168 0.0136 ... 0.0103 0.0069 0.0035];
disp('Comparison with solution of the differential equation:'),
ans =

The exact solution and the discretized numerical solution are plotted together on the following page:
Comparison of Solution for Discretized and Exact Problem

![Graph showing comparison between discrete and exact solutions]