Homework 3

SYSTEMS OF LINEAR EQUATIONS.

Problem 1 c.

Implementation of strass for matrix multiplication.

The code is as below

```
function c = strass(a,b)
nmin = 16;
format long;
[n,n] = size(a);
if n <= nmin
   c = a*b;
else
   m = n/2; u = 1:m; v = m+1:n;
   p1 = strass(a(u,u)+a(v,v),b(u,u)+b(v,v));
   p2 = strass(a(v,u)+a(v,v),b(u,u));
   p3 = strass(a(u,u),b(u,v)-b(v,v));
   p4 = strass(a(v,v),b(v,u)-b(u,u));
   p5 = strass(a(u,u)+a(u,v),b(v,v));
   p6 = strass(a(v,u)-a(u,u),b(u,u)+b(u,v));
   p7 = strass(a(u,v)-a(v,v),b(v,u)+b(v,v));
   c = [p1+p4-p5+p7,p3+p5; p2+p4, p1-p2+p3+p6];
end
end
```

Implementation of the function:

```
for m=4:7
    n=2^m;
    rand('seed',10042022)
    a=rand(n,n);
    b=rand(n,n);
    C_Result_strass = strass(a,b);
    C_Result_normal = a*b;
    difference = norm(C_Result_strass) - norm(C_Result_normal)


end
```

```
difference =
     0
difference =
     0
difference =
     0
difference =
     9.094947017729282e-13
```

```
fprintf(['The error in computing the product using strass as rather than the normal wa
```

```
    'so there is not big difference in the methods.'])
```

```
The error in computing the product using strass as rather than the normal way is sufficiently smallso ther
```

Problem 4.

12/15

```
function [a,x] = elimination_algo(A,b)
a=[A,b];
[n,m] =size(a);
for i = 2:n-1
    if n-i+1 ~= i
        value=a(i,:)*a(n-i+1,i);
        a(n-i+1,:) = a(n-i+1,:)-value/a(i,i);
    end
end
for j = 1:n-1
    a(n,:) = a(n,:) - a(j,:)*a(n,j)/a(j,j);
end
%Updatig the values
value = a(1,:);
a(1,:) =a(n,:);
a(n,:) = value;
for j =2:n
    a(j,:) = a(j,:)-a(1,:)*a(j,1)/a(1,1);
end
for j = 2:n-1
     a(n,:) = a(n,:)-a(j,:)*a(n,j)/a(j,j);
end

x=ones(n,1); %Initializing x
x(n) = a(n,m)/a(n,n); %Solving for the first value of x

%Performing backward substitution
for i = n-1:-1:1
   x(i)=(a(i,m)-(a(i,n)*x(n)))/a(i,i);
end

end
```

Implementation of the algorithm.

```
n=9; m=9;
b=randn(n,1);
A = Mat(m,n);
rand('seed',10042022)
xk=A\b;
[a,x_alg] = elimination_algo(A,b);
%Forming an n by m matrix for verification.
```

```
format short
A
```

```
A = 9×9
    0.1410    0.5119    0.2628    0.0832    0.2044    0.1751    0.4490    0.9810 ···
    0.3615    0.4321         0         0         0         0         0    0.3848
    0.0895         0    0.6353         0         0         0    0.7522         0
    0.7115         0         0    0.1806         0    0.3094         0         0
    0.1613         0         0         0    0.2387         0         0         0
    0.2221         0         0    0.5260         0    0.8330         0         0
    0.4902         0    0.8715         0         0         0    0.8241         0
    0.5075    0.0879         0         0         0         0         0    0.1636
    0.1914    0.3845    0.4959    0.4679    0.2931    0.3800    0.9213    0.2905
```

```
xk'
```

```
ans = 1×9
   22.0875   81.0549  -49.1637 -211.7528    0.8061  133.9952   67.9102  -66.9349 ···
```

```
x_alg'
```

```
ans = 1×9
   22.0875   81.0549  -49.1637 -211.7528    0.8061  133.9952   67.9102  -66.9349 ···
Error1 = -2.0000e-15
```

The true value obtained using MatLab solver and the algorithm designed produce almost the same results. Hence the algorithm works well for any $n - by - n$ where n is an odd number.

19/20

Problem five; Implementation of the algorithm.

```
%LU decomposition with partial pivoting. for B and its transpose.
function [L,U,LU,p1] = decomposition(B')
n = size(B,1);
p1 = (1:n)';
for k=1:n-1
    [~,pos] = max(abs(B(k:n,k)));
    row2swap = k-1+pos;
    B([row2swap, k],:) = B([k, row2swap],:);
    p1([row2swap, k]) = p1([k, row2swap]);
    J = k+1:n;
    B(J,k) = B(J,k)/B(k,k);
    B(J,J) = B(J,J) - B(J,k)*B(k,J);
end
L = tril(B,-1)+eye(n);
K=L';
U = triu(B);
Y=U';
LU=Y*K;
%Forward substition using the function forsub given.

%Backward substition using the function backsub given
end
```

```
%For positive semi-definite matrix, A
function [R] = SPD(A)
[~, n]= size(A);
R = zeros(n,n);
for j = 1:n
    for i = 1:j-1
        sum1 = 0;
        for k =1:i-1
        sum1 =sum1 + (R(k,i)*R(k,j));
        end
        R(i,j)=(A(i,j)-sum1)/R(i,i);
    end
    sum2=0;
    for k =1:j-1
        sum2 =sum2+R(k,j)*R(k,j);
    end
    R(j,j) = sqrt((A(j,j))-sum2);

end
%Forward and backward substitution at this level using forsub and backsub.
end
%For tridiagonal matrix, T
function [e] = Tridiagonal(T,r)
n = length(T);
for i =1:n-1
    mu = T(i+1,i)/T(i,i);
    T(i+1,i+1) = T(i+1,i+1) - mu*T(i,i+1);
    r(i+1) = r(i+1) - mu*r(i);


end


  %Backward substitution
  e(n) = r(n)/T(n,n);
  for i = n-1 : -1:1
      e(i) = (r(i)-T(i,i+1)*e(i+1))/T(i, i);
  end

end
%Multiplying C*e
z=C*e
%LU decomposition on vector B with partial pivoting
function [L,U,LU,p1] = decomposition(B)
L = tril(B,-1)+eye(n);
U = triu(B);
LU=L*U;
%The result is a vector, multiply that with x to get a constant, vector2, then
f_x_algo =  vector2+vector1
```

You can use cholesky on A too

Testing the code

```
n=10;
T=spdiags(ones(n,1)*[-1,2,1],-1:1,n,n);
A1=tril(rand(n));
B=rand(n);
C=rand(n);
b=rand(n,1);
x=ones(n,1);
rand('seed', 10082022)
A=A1*A1';
%%%%%%%%%%%%%
[K,Y,YK,p1] = decomposition(Z);
P1 = eye(n);
P1=P1(p1,:);
k= forsub(K,P1*x);
y= backsub(Y,k);
vector1 = b'*y;
%*****************
R=SPD(A);
Q=R';
A_dec = Q*R;
r1 = R\(R'\x);%Forward and backward substitution to obtain the vector needed.
%%%%%%%%%%%%%%%%%%%
e = Tridiagonal(T,r1);
%*******************
z=C*e';
%%%%%%%%%%%%%%%%%%%%%%%%
[L,U,~,p1] = decomposition(B);
P1 = eye(n);
P1=P1(p1,:);
k1= forsub(L,P1*z);
m= backsub(U,k1);
vector2=x'*m;
f_x_algo = vector2+vector1;

f_x_algo
```

Write this a general function

```
f_x_algo = 4.3690e+03
```

```
%*****************************Using Matlab solver
[L,U,p] = lu(B, 'vector');
[L1,U1,p1] = lu(B', 'vector');
yk=U1\(L1\x(p1));
vec=b'*yk;
R=chol(A);
r2= R\(R'\x);
pol=T\r2;
vec1=C*pol;
yv=U\(L\vec1(p));
vec2 = x'*yv;
f_x_exact = vec2+vec;
f_x_exact
```

```
f_x_exact = 4.3715e+03
```

```
%This is only computed for comparison witht the algoritm.

error = abs(f_x_exact-f_x_algo)/f_x;
error
```

```
error = 5.6740e-04
```

The absolute relative error in evaluating $f(x)$ using the algorithm is somewhat small so it is safe to conclude that the algorithm works well.

Problem six

Pentadiagonal systmes of linear equations.

Code for the algorithm.

```
function x = Pentadiagonali(a,b,c,d,e,f)
%setting n using the a entries in the major diagonal.
n=length(a);
% %initializing the vectors for the multipliers
M1 = ones(n-2,1);
M2 = ones(n-2,1);
%for loop to update the multipliers and also calculate the entries for the
%argumented matrix.
%Gaussian Elimination is done in this for loop using the multipliers

for i = 1:n-2
    M1(i) = b(i)/a(i);
    M2(i) = d(i)/a(i);

    a(i+1) = a(i+1) - M1(i)*c(i);
    c(i+1) = c(i+1) - M1(i)*e(i);
    b(i+1) = b(i+1) - M2(i)*c(i);
    a(i+2) = a(i+2) - M2(i)*e(i);

    f(i+1) = f(i+1) - M1(i)*f(i);
    f(i+2) = f(i+2) - M2(i)*f(i);



 end

%For the final entries of the matrix
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);

%Solving for the last value of x so we that we can use backward substition.
% we employ the function backsub to do this for us.
x(n) = f(n)/a(n);
%for the second last value of the solution, we use x(n) to compute it as
```

```
%follows

x(n-1) = (f(n-1)-c(n-1)*x(n))/a(n-1);
%For loop for other solutions

for i = n-2:-1:1
    x(i) = (f(i)-c(i)*x(i+1)-e(i)*x(i+2))/a(i);
end
%The formula is so since we have only three entries per row

end
```

Implementation of the algorithm designed for solving a pentadiagonal system of linear equations.

```
function [A,f,x] = pentadiagonal_test(n)
%Entries of matrix A
i = (1:n);
a=i;
j = (1:n-1);
b = -(j+1)/3;
c=b;
k=(1:n-2);
d=-(k+2)/6;
e=d;
%Forming the vector f

f=ones(length(n))';
f(1)=1/2; f(2)=1/6; f(n-1) = 1/6; f(n) = 1/2; f(3:n-2)=0;

x = Pentadiagonali(a,b,c,d,e,f);
%Forming matrix A using the entries a,b,c,d,and e
A = diag(d,-2)+diag(b,-1)+diag(a)+diag(c,1)+diag(e,2);




end
```

Using the code and comparison.

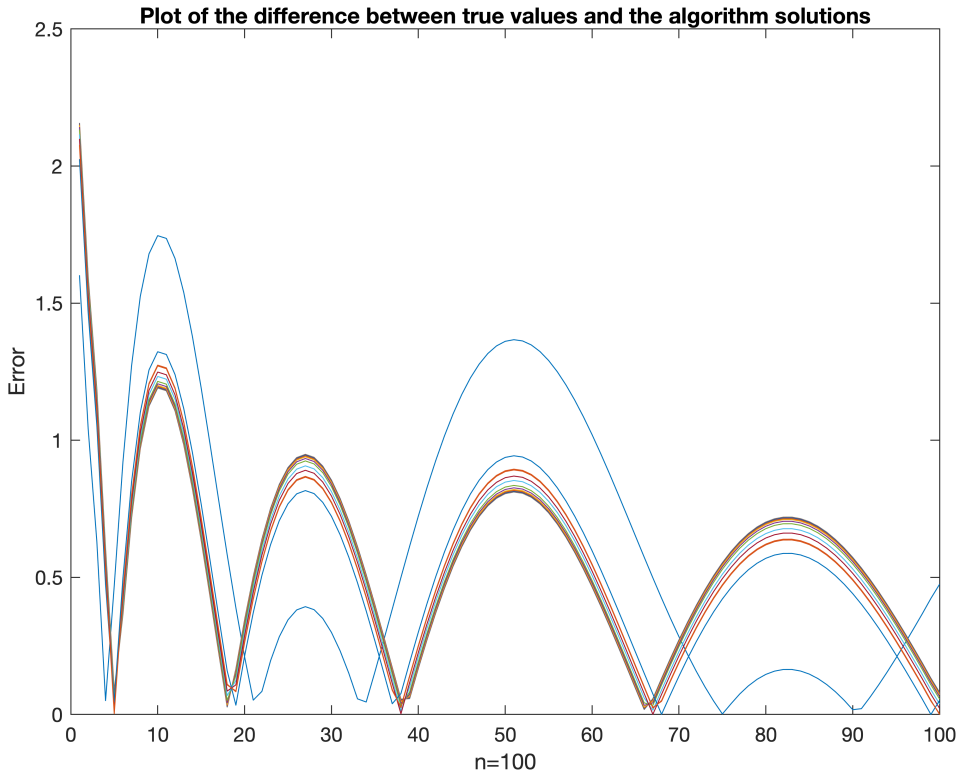
```
%Implementation of pentadiagonal to print the matrix A, vector f and
%solution x for Ax =f using different values of n.
n1=100;
[A,f,x_1] = pentadiagonal_test(n1);
x_cal=A\f';

%
```

```
n2=1000;
[A,f1,x_2] = pentadiagonal_test(n2);
x2=A\f1';
%Computing the difference in the solutions
diff1 = abs(x_1'-x_cal);
diff2=abs(x_2'-x2);
```

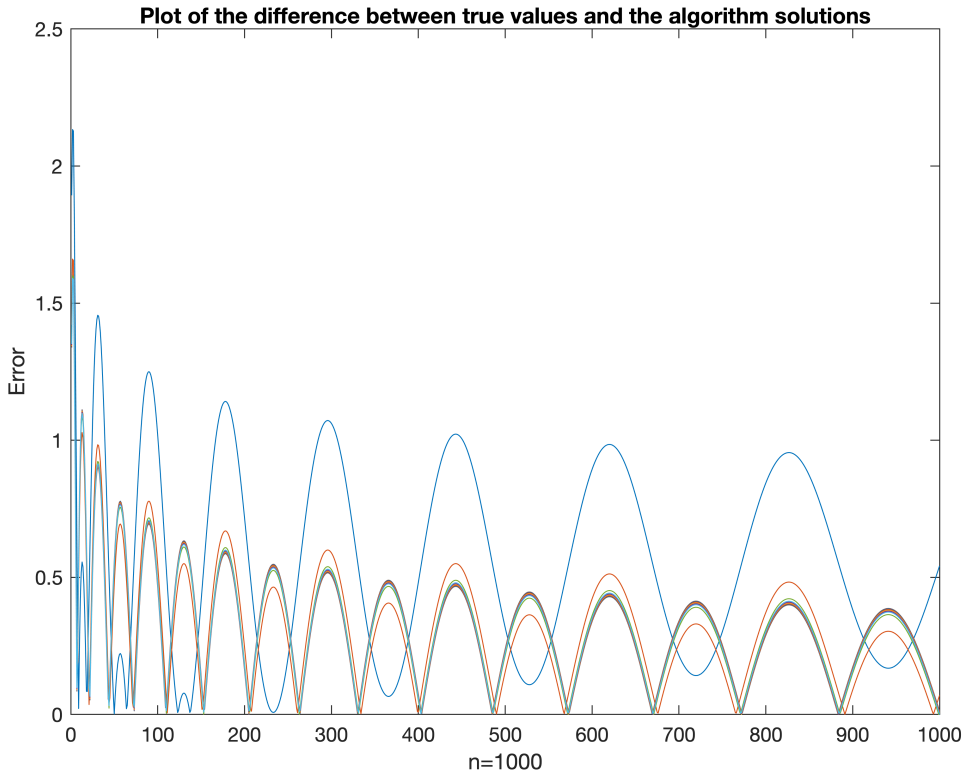
It is better to use 2-norm

Plot of the error distribution in the two computed results.

```
plot(diff1)
xlabel('n=100')
ylabel('Error')
title('Plot of the difference between true values and the algorithm solutions')
```




```
plot(diff2)
xlabel('n=1000')
ylabel('Error')
title('Plot of the difference between true values and the algorithm solutions')
```



For both n, the difference in the results of MatLab solver and those of the algorithm is slightly differ by a very small error and reduce much by a factor of ten with increase in n by a factor of 10.

13/15

Problem number seven a)

Complete pivoting. The code below implements the solution to a linear sysytem using the $lupp.m$ code tempelate provided.

```
function [LU,p,q,L,U] = lucp(A)
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
    % Find the column in row K that contains the largest entry in magnitude
    [temp,pos1] = max(abs(A(k:n,k:n)));
    [~,pos2] = max(temp);
    row2swap = k-1+pos1;
    column2swap=k-1+pos2;

    % Swap the rows of A and p and q (perform complete pivoting)
    A([row2swap, k],:) = A([k, row2swap],:);
    p([row2swap, k]) = p([k, row2swap]);
    A(:,[column2swap, k]) = A(:,[k,column2swap]);
    q([column2swap, k]) = p([k, column2swap]);
```

10/10

```
        % 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
    % Extracting the Lower triangular matrix, L
    L = eye(length(n))+tril(A,-1);
    % Extracting the Upper triangular matrix, U
    U = triu(A);
    LU=L*U;
    end
```

Problem 7b

Implementation of complete pivoting using the tempelate codes provided

```
n=12;
A=((1:n)').^(0:n-1);
b=A(:,n);
%For complete pivoting, we compute the permutaion matrices, P and Q
[LU, p,q,L,U] = lucp(A);
P = eye(n);
P=P(p,:);
Q = eye(n);
Q = Q(:,q);
Ly= forsub(L,P*b);
Ux= backsub(U,Ly);
X_1=Q*Ux;
% y = forsub(LU,b(p));
% x = backsub(LU,y);%x = x(q);
x1 = A\b;
```

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

```
x1'
```

```
ans = 1×12
     0     0     0     0     0     0     0     0     0     0     0     1
```

```
X_1'
```

```
ans = 1×12
     0     0     0     0     0     0     0     0     0     0     0     1
```

Codes used in compling the homework.

You have not completed for partial pivoting

Codes for number 4.

```
function [A] = Mat(m,n)
A =diag(randn(n,1));
k=1;
for j =1:m
    for i = 1:n
        if i==j
            A(i,j) =rand(k); %For the major diagonal
        elseif n-j==i-1
            A(i,j) =rand(k); %For the minor diagonal
        elseif i==1
            A(j,i)=rand(k);  %For the fist row

        elseif j==1
            A(j,i)=rand(k); %For the first column
        elseif i==n
            A(j,i) = rand(k); %For the last row
        elseif j==n
            A(j,i) = rand(k); %For the first column
        end
    end
end
end
function [a,x] = elimination_algo(A,b)
a=[A,b];
[n,m] =size(a);
for i = 2:n-1
    if n-i+1 ~= i
        value=a(i,:)*a(n-i+1,i);
        a(n-i+1,:) = a(n-i+1,:)-value/a(i,i);
    end
end
for j = 1:n-1
    a(n,:) = a(n,:) - a(j,:)*a(n,j)/a(j,j);
end
%Updatig the values
value = a(1,:);
a(1,:) =a(n,:);
a(n,:) = value;
for j =2:n
    a(j,:) = a(j,:)-a(1,:)*a(j,1)/a(1,1);
end
for j = 2:n-1
     a(n,:) = a(n,:)-a(j,:)*a(n,j)/a(j,j);
end

x=ones(n,1); %Initializing x
x(n) = a(n,m)/a(n,n); %Solving for the first value of x

%Performing backward substitution
for i = n-1:-1:1
```

```
   x(i)=(a(i,m)-(a(i,n)*x(n)))/a(i,i);
end

end
```

Code function for pentadiagonal matrices.

```
function x = Pentadiagonali(a,b,c,d,e,f)
%setting n using the a entries in the major diagonal.
n=length(a);
% %initializing the vectors for the multipliers and that of the solution, x
x=ones(n,1);
M1 = ones(n-2,1);
M2 = ones(n-2,1);
%for loop to update the multipliers and also calculate the entries for the
%argumented matrix.
%Gaussian Elimination is done in this for loop using the multipliers

for i = 1:n-2
    M1(i) = b(i)/a(i);
    M2(i) = d(i)/a(i);

    a(i+1) = a(i+1) - M1(i)*c(i);
    c(i+1) = c(i+1) - M1(i)*e(i);
    b(i+1) = b(i+1) - M2(i)*c(i);
    a(i+2) = a(i+2) - M2(i)*e(i);

    f(i+1) = f(i+1) - M1(i)*f(i);
    f(i+2) = f(i+2) - M2(i)*f(i);



 end

%For the final entries of the matrix
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);

%Solving for the last value of x so we that we can use backward substition.
% we employ the function backsub to do this for us.
x(n) = f(n)/a(n);
%for the second last value of the solution, we use x(n) to compute it as
%follows

x(n-1) = (f(n-1)-c(n-1)*x(n))/a(n-1);

%For loop for other solutions

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

```
%The formula is so since we have only three entries per row

end
```

Implemenation code

```
function [A,f,x] = pentadiagonal_test(n)
%Entries of matrix A
i = (1:n);
a=i;
j = (1:n-1);
b = -(j+1)/3;
c=b;
k=(1:n-2);
d=-(k+2)/6;
e=d;
%Forming the vector f

f=ones(length(n))';
f(1)=1/2; f(2)=1/6; f(n-1) = 1/6; f(n) = 1/2; f(3:n-2)=0;

x = Pentadiagonali(a,b,c,d,e,f);
%Forming matrix A using the entries a,b,c,d,and e
A = diag(d,-2)+diag(b,-1)+diag(a)+diag(c,1)+diag(e,2);

%For loop for other solutions,

for i = n-2:-1:1
    x(i) = (f(i)-c(i)*x(i+1)-e(i)*x(i+2))/a(i);
end
%The formula is so since we have only three entries per row

end
```

Code function for forward substitution.

```
function x = forsub(L,b)
%forsub  Solves the unit lower triangular system Lx=b using forward
%        substitution.
%
%   x = forsub(L,b) Solves the unit lower triangular system Lx=b using
%   forward substition.
%
%   Use in conjuction with lupp and backsub to solve a general system Ax=b
n = size(L,1);   % How big is the system
x = zeros(n,1);  % Initialize the solution vector
x(1) = b(1);
```

```
for j = 2:n
    x(j) = b(j) - L(j,1:j-1)*x(1:j-1);
end
end
```

backsub tempelate given.

```
function x = backsub(U,b)
%forsub  Solves the upper triangular system Ux=b using backward
%        substitution.
%
%   x = backsub(U,b) Solves the upper triangular system Ux=b using
%   backward substition.
%
%   Use in conjuction with lupp and forsub to solve a general system Ax=b
n = size(U,1);   % How big is the system
x = zeros(n,1);  % Initial the solution vector
x(n) = b(n)/U(n,n);
for j = n-1:-1:1
    x(j) = ( b(j) - U(j,j+1:n)*x(j+1:n) )/U(j,j);
end
end
```

lupp.m function

```
function [LU,p,q,L,U] = lucp(A)
n = size(A);
p = (1:n)';
q=(1:n)';
for k=1:n-1
    % Find the row in column k that contains the largest entry in magnitude
    % Find the column in row K that contains the largest entry in magnitude
    [temp,pos1] = max(abs(A(k:n,k:n)));
    [~,pos2] = max(temp);
    row2swap = pos1(pos2)+k-1;
    column2swap=k-1+pos2;

    % Swap the rows of A and p and q (perform complete pivoting)
    A([k, row2swap],:)=A([row2swap, k],:);
    p([k, row2swap])=  p([row2swap, k]) ;
    A(:,[ k,column2swap]) = A(:,[column2swap,k]);
    q([k,column2swap]) = q([ column2swap,k]);
    % Perform the kth step of Gaussian elimination
    if A(k,k) == 0
      break
    end
    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
% Extracting the Lower triangular matrix, L
L = tril(A,-1)+eye(n);
```

```
% Extracting the Upper triangular matrix, U
U = triu(A);
LU=L*U;
end
```

MatLab function for strass

```
function c = strass(a,b)
nmin = 16;
format long;
[~,n] = size(a);
if n <= nmin
   c = a*b;
else
   m = n/2; u = 1:m; v = m+1:n;
   p1 = strass(a(u,u)+a(v,v),b(u,u)+b(v,v));
   p2 = strass(a(v,u)+a(v,v),b(u,u));
   p3 = strass(a(u,u),b(u,v)-b(v,v));
   p4 = strass(a(v,v),b(v,u)-b(u,u));
   p5 = strass(a(u,u)+a(u,v),b(v,v));
   p6 = strass(a(v,u)-a(u,u),b(u,u)+b(u,v));
   p7 = strass(a(u,v)-a(v,v),b(v,u)+b(v,v));
   c = [p1+p4-p5+p7,p3+p5; p2+p4, p1-p2+p3+p6];
end
end
```

7/7

No 1:

a) Given that $\begin{bmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{bmatrix} = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{bmatrix}$

with $C_{11} = A_{11}B_{11} + A_{12}B_{21}$ ——— ✗

$C_{12} = A_{11}B_{12} + A_{12}B_{22}$ ——— A_2

$C_{21} = A_{21}B_{11} + A_{22}B_{21}$ ——— A_3

$C_{22} = A_{21}B_{12} + A_{22}B_{22}$ ——— A_4

The Strassen Scheme is defined in such a way that;

$P_1 = (A_{11} + A_{22})(B_{11} + B_{22}) = A_{11}B_{11} + A_{11}B_{22} + A_{22}B_{11} + A_{22}B_{22}$

$P_2 = (A_{21} + A_{22})(B_{11}) = A_{21}B_{11} + A_{22}B_{11}$

$P_3 = A_{11}(B_{12} - B_{22}) = A_{11}B_{12} - A_{11}B_{22}$

$P_4 = A_{22}(B_{21} - B_{11}) = A_{22}B_{21} - A_{22}B_{11}$

$P_5 = (A_{11} + A_{12})B_{22} = A_{11}B_{22} + A_{12}B_{22}$

$P_6 = (A_{21} - A_{11})(B_{11} + B_{12}) = A_{21}B_{11} + A_{21}B_{12} - A_{11}B_{11} - A_{11}B_{12}$

$P_7 = (A_{12} - A_{22})(B_{21} + B_{22}) = A_{12}B_{21} + A_{12}B_{22} - A_{22}B_{21} - A_{22}B_{22}$

The entries of Matrix C are defined in such a way that.

$C_{11} = P_1 + P_4 - P_5 + P_7$

or $C_{12} = P_3 + P_5$

$C_{21} = P_2 + P_4$

$C_{22} = P_1 - P_2 + P_3 + P_6$.

We need to show that this definition, as per Strassen's scheme aggrees with the normal multiplication and entry definition given in *1 *2 *3 and *4.

$$C_{11} = P_1 + P_4 - P_5 + P_7$$
$$= A_{11}B_{11} + A_{11}B_{22} + A_{22}B_{11} + A_{22}B_{22} + A_{22}B_{21} - A_{22}B_{11} - A_{11}B_{22}$$
$$- A_{12}B_{22} + A_{12}B_{21} + A_{12}B_{22} - A_{22}B_{21} - A_{22}B_{22}$$

Simplifying this implies that

$$C_{11} = A_{11}B_{11} + A_{12}B_{21}$$ as defined by A_1

Also; $C_{12} = P_3 + P_5$

$$= A_{11}B_{12} - A_{11}B_{22} + A_{11}B_{22} + A_{12}B_{22}$$
$$= A_{11}B_{12} + A_{12}B_{22}$$

as defined in A_2.

Similarly; $C_{21} = P_2 + P_4$

$$= A_{21}B_{11} + A_{22}B_{11} + A_{22}B_{21} - A_{22}B_{11}$$
$$= A_{21}B_{11} + A_{22}B_{21}$$

as it is in A_3.

Lastly: $C_{22} = P_1 - P_2 + P_3 + P_6$

$$= A_{11}B_{11} + A_{11}B_{12} + A_{22}B_{11} + A_{22}B_{22} - A_{21}B_{11} + A_{22}B_{11}$$
$$+ A_{11}B_{12} - A_{11}B_{22} + A_{21}B_{11} + A_{21}B_{12} - A_{11}B_{11} - A_{11}B_{12}$$

Which simplifies to;

$$C_{22} = A_{22}B_{22} + A_{21}B_{12}$$ as defined in *4

Therefore; the scheme aggrees with the normal operations

Note: For any $n < 16$, using Strass's algorithm is as good as doing the multiplication of matrices normally, since it will cost more operations to do it using Strass's algorithm.

For $n \geq 16$, Strass's algorithm is preffered since the cost of using it is minimal compared to it.

The if statement in the code checks for whether the condition of $n < 16$ is valid, and if so, then the multiplication of matrices is done using the normal way, otherwise, Strass is set into use.

10/ No2: Given the system of linear Equations:

$$\begin{cases} 2x_1 + 0x_2 + 4x_3 + 3x_4 = 4 & \text{(I)} \\ -2x_1 + 0x_2 + 2x_3 - 13x_4 = 40 & \text{(II)} \\ x_1 + 15x_2 + 2x_3 - 9/2x_4 = 29 & \text{(III)} \\ -4x_1 + 5x_2 - 7x_3 - 10x_4 = 9 & \text{(IV)} \end{cases}$$

These equations can be written in matrix form as;

$$\begin{pmatrix} 2 & 0 & 4 & 3 \\ -2 & 0 & 2 & -13 \\ 1 & 15 & 2 & -9/2 \\ -5 & 5 & -7 & -10 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 4 \\ 40 \\ 29 \\ 9 \end{pmatrix}$$ and the resulting

Argumented matrix is as follows:

$$\left(\begin{array}{cccc|c} 2 & 0 & 4 & 3 & 4 \\ -2 & 0 & 2 & -13 & 40 \\ 1 & 15 & 2 & -9/2 & 29 \\ -4 & 5 & -7 & -10 & 9 \end{array}\right)$$

Using gaussian Elimination with partial pivoting,

Let us note that $|-4| > |2|$, so we can swap (I) with (IV). The resulting matrix is

$$\left(\begin{array}{cccc|c} -4 & 5 & -7 & -10 & 9 \\ -2 & 0 & 2 & -13 & 40 \\ 1 & 15 & 2 & -9/2 & 29 \\ 2 & 0 & 4 & 3 & 4 \end{array}\right)$$

We then eliminate non zero entries in (ii), (iii) and (i). using the following operations coupled with multipliers:

$$(ii) - \left(+\tfrac{1}{2}\right)(iv)$$
$$(iii) - \left(-\tfrac{1}{4}\right)(iv)$$
$$(i) - \left(-\tfrac{1}{2}\right)(iv)$$

The resulting matrix is:

$$\left[\begin{array}{cccc|c} -4 & 5 & -7 & -10 & 9 \\ 0 & -5/2 & 11/2 & -8 & 71/2 \\ 0 & 65/4 & 1/4 & -7 & 125/4 \\ 0 & 5/2 & 1/2 & -2 & 17/2 \end{array}\right]$$

Performing partial pivoting on the second column, we note that $\left|65/4\right| > \left|-5/2\right|$, so we swap equations (ii) and (iii). The resulting matrix is:

$$\left[\begin{array}{cccc|c} -4 & 5 & -7 & -10 & 9 \\ 0 & 65/4 & 1/4 & -7 & 125/4 \\ 0 & \boxed{-5/2} & 11/2 & -8 & 71/2 \\ 0 & \boxed{5/2} & 1/2 & -2 & 17/2 \end{array}\right]$$

The following operations, with multipliers eliminate non zero entries of the last two entries in column (2).

$$(ii) - \left(\frac{-10}{65}\right)(iii)$$

$$(i) - \frac{10}{65}(iii)$$

This gives;

$$\left[\begin{array}{cccc|c} -4 & 5 & -7 & -10 & 9 \\ 0 & 65/4 & 1/4 & -7 & 121/4 \\ 0 & 0 & 72/13 & -118/13 & 524/13 \\ 0 & 0 & \boxed{6/13} & -12/13 & 48/13 \end{array}\right]$$

Since $|72/13| > |6/13|$, we shall not employ pivoting at this stage. Let us now eliminate $6/13$ from equation (i) using the following operation

$$(i) - \frac{6}{72}(ii)$$

The resulting matrix is;

$$\left[\begin{array}{cccc|c} -4 & 5 & -7 & -10 & 9 \\ 0 & 65/4 & 1/4 & -7 & 121/4 \\ 0 & 0 & 72/13 & -118/13 & 524/13 \\ 0 & 0 & 0 & -1/6 & 1/3 \end{array}\right]$$

If we then use Backward substitution we have that $-x_4/6 = 1/3 \iff x_4 = -2$

Then, $\frac{72}{13}x_3 - \frac{118}{13}x_4 = \frac{524}{13}$

$\Rightarrow x_3 = 4$

Also: $65/4\, x_2 + 16 x_3 - 7x_4 = 125/4$

and $x_2 = 1$

Using equation (10) gives $x_1 = -3$.

$\therefore x_1 = -3,\ x_2 = 1\ \ x_3 = 4$ and $x_4 = -2$

10/10 **No 3]** we need to show that $M^{-1} = L$,

To prove this, it is sufficient to show that

$$M^{-1}L = LM^{-1} = I.$$

That is, if $P = ML$ and $P_{ik} = \sum_{j=1}^{n} a_{ij} b_{jk}$

for $i = 1, \ldots, n$, and $k = 1, \ldots, n$, then

$P_{ii} = 1$ and $P_{ik} = 0$ if $i \neq k$.

Let use Extract rows of matrix M^{-1} and the ~~Re~~ Columns of L and do the Inner product on the respective rows and columns.

$$i \begin{bmatrix} 1, & 0 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 0 \\ \vdots \\ 0 \end{bmatrix} = 1$$

clearly; $\begin{bmatrix} a_{ii}, & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} b_{jj} \\ 0 \\ \vdots \\ 0 \end{bmatrix} = a_{ii} b_{jj} = P_{ii} = 1$

For the first row of M^{-1} and first column of L, the Inner product is 1, with $P_{ii} = 1$ and $P_{ik} = 0$; $i \neq k$ $\forall$ i, k.

The second row and second column of the two matrix.

$$\begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = a_{22} \cdot b_{22} = P_{22} = 1$$

and $a_{ij} b_{ik} = 0 \; ; \; i \neq k$.

Since Both M^{-1} and L are lower triangular, It is sufficient to ~~prove~~ show that the product of ~~on~~ the two is an identity matrix.

Considering ~~Entry~~ rows and columns with more than on non zero entry;

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

$$= \begin{bmatrix} l_{i+1,i} - l_{i+1,i} + 0 & \cdots & 0 \end{bmatrix}$$

$$= 0.$$

and $\begin{bmatrix} 0 & 0 & 0 & 0 & l_{i+2,i} & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$

$$= 0$$

So $\forall \; a_{ij} b_{ik} = 0, \; i \neq k$ and $P_{ii} = 1$ when $i = k$.

This is not always true

Also; $\begin{bmatrix} 0, & 0 & 0 & b_{i+2,i} & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = 1$

Since $l_{i+2,i} \cdot x_0 = 0$.

So $P_{i,i} = 1$ and $\sum a_{ij} b_{jk} = 0$ for $i \neq k$.

Considering the n^{th} row and taking its Inner product with the n^{th} column, then we have,

$$\begin{bmatrix} 0 & 0 & 0 & b_{n,i} & 0 & \cdots & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} = 1$$

Therefore $\forall$ a_{ij} and b_{jk}, $\sum a_{ij} b_{jk} = P_{ik} = 0$

when $i \neq k$ and $P_{ii} = 1$

So $M^{-1} L = I$

Also, since $M^{-1} L = I$, $L M^{-1} = I$ because M^{-1} and L are lower triangular matrices.

$\therefore M^{-1} = L$

3/3

No 4 Gaussian elimination for an n by n matrix with non zero entries only on the first and last rows and columns and on the two main diagonals. For n odd.

Number of operations:

Step 1: Addition — None
subtraction — n-2
multiplication — n-2
division — n-2

Total : $3(n-2)$
$= 3n-6$.

Step 2: No addition
n-1 divisions
n-1 subtractions
and n-1 multiplications.

Total $3(n-1)$
$= 3n-3$

Step 3: No addition, (n-1) multiplications, divisions and subtractions.

Total $3n-3$

Step 4: No additions and $3(n-2)$ as the total with (n-2) additions, divisions and subtraction.

Initializing: 1 division

This division comes when finding the last value of x_n.

Step 5:- No addition
(n-1) subtractions.
(n-1) multiplications
(n-1) divisions.

3(n-1) operations.

Total : $3n-6+3n-3+3n-3+3n-6+1+3n-3$

$= 15n-20$ operations

The method is of order $O(n)$ since it takes in $15n-20$ operations.

No 5: $T \in \mathbb{R}^{n \times n}$, diagonally dominant tridiagonal matrix;

$A \in \mathbb{R}^{n \times n}$, symmetric positive definite.

$B \in \mathbb{R}^{n \times n}$ and $C \in \mathbb{R}^{n \times n}$ non singular.

Define $f(x) := x^T B^{-1} C T^{-1} A^{-1} x + b^T B^{-T} x$

$x, b \in \mathbb{R}^n$ are column vectors;

a) To efficiently evaluate $f(x)$, we do the following.

Step 1

Let $B^T y = x$.

Then since B is a full non singular matrix, then we can write $B = LU$, (ie we can perform an LU decomposition on B.

Also, since $B = LU$, then $B^T = U^T L^T$

so $U^T L^T y = x$.

Solve this using forward substitution, ie, $U^T k = x$ and then solve $L^T y = k$ using Backward substitution. Then $y = B^{-T} x$.

so, $f(x) = x^T B^{-1} C^T A^{-1} x + b^T y$.

Define $P := b^T y \in \mathbb{R}^n$.

so $f(x) = x^T B^{-1} C^T A^{-1} x + P$. ——— (1)

Step a

Solving $A^{-1}x$.

Let $r \in \mathbb{R}^n$ such that $Ar = x$.

Then, since A is symmetric positive definite then $A = R^T R$. This means we can write A is the form $A = R^T R$ using Cholesky's Algorithm.

so $R^T R r = x$.

Let $Rr = q$. This can be solved using forward substitution. ie $R^T q = x$.

and solve $Rr = q$ solves r using Backward substitution.

so, (1) becomes; $f(x) = x^T B^{-1} C^T r^{-1} + P$

Next: **Step 3**

Solve $T^{-1}r$

Let $Te = r$ for $e \in \mathbb{R}^n$ a Column vector

Then, since T is a tridiagonal matrix (diagonally dominant) we can solve this using Clout's algorithm. A backward substitution is used to find e.

Then, $f(X) = X^T B^{-1} C e + P$.

Step 4

Since C is a non singular matrix, $C \in \mathbb{R}^{n \times n}$ and $e \in \mathbb{R}^n$ a column matrix, then $Ce = Z$ where $Z \in \mathbb{R}^n$, a column matrix/vector.

So $f(X) = X^T B^{-1} Z + P$.

Step 5 solve $B^{-1} Z$ let $m \in \mathbb{R}^n$ column vector, such that $Bm = Z$, then

since B is non singular, then we can do LU decomposition on this, as in **Step 1**

solve $LUm = Z$.

Let $Lh = Z$, solve this using forward substitution, then solve $Um = h$. using Backward substitution.

The resulting expression is then

$$f(x) = x^T m + p$$

Define $x^T m = I$

then $f(x) = I + p$

$$f(x) = I + p = w \quad ; \; w \in \mathbb{R}^n$$

Nos: Algorithm:

① i) LU decomposition with partial pivoting:

for k = 1 to n-1 do;

position = max(abs(B(k:n,k)));

swap row = k-1+pos;

Swap positions of rows;

j = k+1 to n;

$B_{jk} = B_{jk} / B_{kk}$;

$B_{jj} = B_{jj} - B_{jk} * B_{kj}$;

end for

ii) Backward and forward substitution;

a) forward substitution:

x(1) = b(1)

for j = 2 to n do

$x_j = b_j - L_{j,1:j-1} * x(1:j-1)$;

end for

b) Backward substitution;

$x_n = b_n / U_{nn}$

for j = n-1 to 1 do

$x_j = (b_j - U_{j,j+1:n} * x(j+1:n)) / U_{jj}$

end for

Step 2°. for $i = 1$ to n do

$$r_{ii} = \left(q_{ii} - \sum_{k=1}^{i-1} r_{ki}^2 \right)^{1/2}$$

for $j = i+1$ to n do

$$r_{ij} = \frac{1}{r_{ii}} \left(q_{ij} - \sum_{k=1}^{i-1} r_{ki} r_{kj} \right)$$

end for

end for.

forward substitution;

$$q(1) = b(1)$$

for $j = 2:n$

$$q(j) = b(j) - L(j, 1:j-1)\, q(1:j-1)$$

Back ward Substitution:

$r(n) = b(n) / U(n,n)$

for $j = n-1 : -1 : 1$

$r(n) = \left(b(j) - U(j, j+1:n) * r(j+1:n) \right) / U(j,j)$

Step 3°: Implement Clout's algorithm:

a) for $i = 1$ to $n-1$ do

$\mu = c_i / d_i$

$d_{i+1} = d_{i+1} - \mu e_i$

$b_{i+1} = b_{i+1} - \mu b_i$

end for

b) Back ward substitution:

$E_n = b_n / d_n$

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

$E_i = (b_i - e_i E_{i+1}) / d_i$

end for.

Step 4: Multiply C by E to obtain Z.

Step 5: Recall the stored operations ~~values~~ A B^T and reuse them for B.

The resulting expression is then

$$f(x) = x^T m + p$$

define $x^T m = I$

then $f(x) = I + p$

$$f(x) = I + p = w \quad ; w \in \mathbb{R}^n.$$

6a)

Designing a fast algorithm for solving pentadiagonal systems of linear equations using an idea of solving Tridiagonal linear systems of equations discussed in class.

Step 1: Forming the matrix into an upper triangular matrix using G.E.

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

$m_i = b_i / a_i$;

$m'_i = d_i / a_i$;

$a_{i+1} = a_{i+1} - m_i c_i$;

$c_{i+1} = a_{i+1} - m'_i e_i$;

$b_{i+1} = b_{i+1} - m'_i c_i$;

$a_{i+2} = a_{i+2} - m'_i e_i$;

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

$f_{i+2} = f_{i+2} - m'_i f_i$.

end for

Step 2: Find the last value of X using Back ward substitution.

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

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

$x_n = f_n / a_n$;

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

Step 3: Set a for loop to Complete Beckward Substitution.

for $i = n-2$ to 1 do

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

end for;

7/7

b) Number of operations:

No Additions needed anywhere.

Step 1: 6(n-2) subtractions
6(n-2) multiplications.
2(n-2) divisions.

Total 14n - 28 operations

step 2: 10 operations (constants or done one) for divisions, additions and subtractions

Step 3: 2(n-2) multiplications
2(n-2) subtractions.
(n-2) divisions

Total 5n - 10 operations:

Total: 14n - 28 + 10 + 5n - 10
= 19n - 28 operations which is O(n)