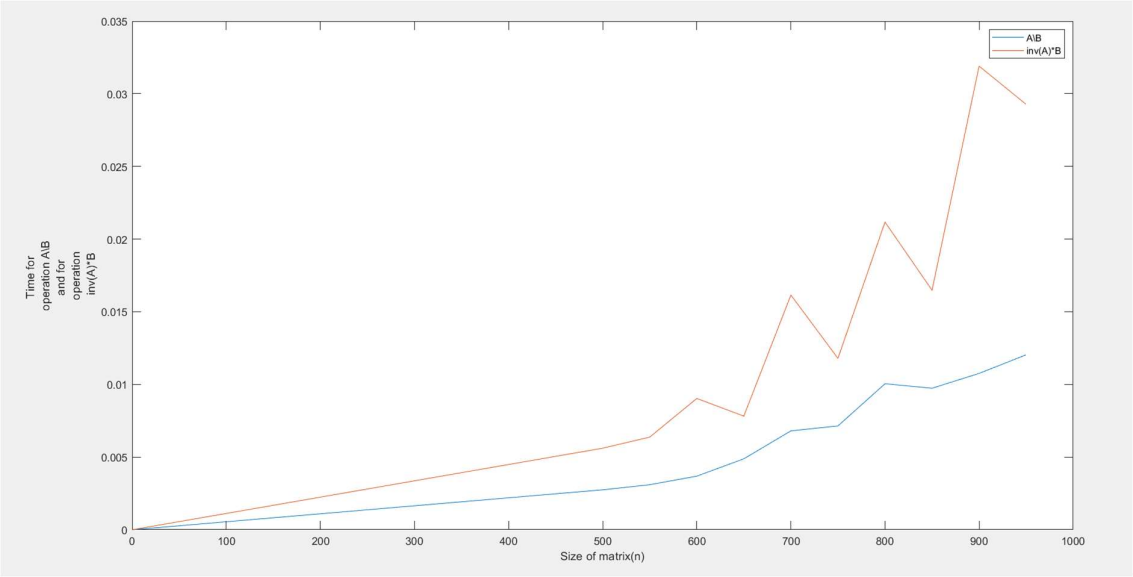
```
%  Task 1
%Setting the value of n as per mentioned in the question
n = 500:50:1000;
j=1;

time1=0;
val1=0;
time2=0;
val2=0;

%Starting a for loop which will run 10 times
for i=1:10
disp('The result is:');
%creating a random sized square matrix A and a column matrix b using rand
% inbuilt function
A=rand(n(1,j),n(1,j));
b=rand(n(1,j),1);

% To measure the time in an operation A\b we are using tic toc function
tic     %stopwatch starts
x= A\b;
y1=toc;  %stopwatch stops and the value of time taken is updated in y1

disp(x);
fprintf('It tooks %f sec for evaluation',y1);

%Creating an array which will store the all values of time taken in 10
%operations
time1=[time1;y1];
val1=[val1;n(1,j)];

%Writing the vales in an excel sheet
a=xlswrite('L1Q1.xlsx',time1);
d=xlswrite('L1Q1.xlsx',val1,'C1:C11');

%Now using tic toc function we are trying to get the time taken in inv(A)*b
tic             %stopwatch starts
x1= inv(A)*b;
y2=toc;         %stopwatch stops
time2=[time2;y2];

%Writing the values of time taken in same excel sheet in column B
c=xlswrite('L1Q1.xlsx',time2,'B1:B11');
j=j+1;
end         %Ending the for loop

%Now as per mentioned in the question we are trying to read the values from
%the created excel sheet
temp=readmatrix('L1Q1.xlsx');
```

```
50 x_1 = temp(:, 1);    %reading first column
51 y_1 = temp(:, 2);    %reading second column
52 z_1 = temp(:, 3);    %reading third column
53 plot(z_1,x_1);       %plotting the time taken in operation A\b wrt values of n
54
55 %plot(val1,time1);
56 %To plot the curve on same plane we have used hold on and hold off function
57 hold on
58 plot(z_1,y_1);       %plotting the time taken in operation inv(A)*b wrt values of n
59 hold off
60 legend('A\B','inv(A)*B')
```



	A	B	C
		L1Q1	
	time1	**time2**	**n**
	Number	Number	Number
1	0	0	0
2	0.0027	0.0056	500
3	0.0031	0.0064	550
4	0.0037	0.0090	600
5	0.0049	0.0078	650
6	0.0068	0.0162	700
7	0.0071	0.0118	750
8	0.0100	0.0212	800
9	0.0097	0.0165	850
10	0.0108	0.0319	900
11	0.0120	0.0293	950

```
% Task2
% To solve this question we are creating a function FdSubs which is taking
% two input L and b and returning an ouput matrix x
% Note----We have to first give the L and b as input
function [x] = FdSubs(L, b)

[m, n] = size(L);       %copying the size of U in m and n variables

%As we are doing this procedure only for Square matrix so program will give
%an error if L is not square matrix
if m ~= n
error('L must be a square matrix');
end

%if L is square matrix then we create a column matrix with all elements zero
x = zeros(n, 1);

for i = 1:n
%now we are using this equation
% In reality this equation is just back substitution method
% We can observe this thing when we run the for loop and put the required
% values in it

x(i) = (b(i) - L(i, 1:n) * x) / L(i, i);

end

end
```

```
>> L=[1,0,0;2,1,0;3,2,1]

L =

     1     0     0
     2     1     0
     3     2     1

>> b=[1;4;7]

b =

     1
     4
     7

>> FdSubs(L,b)

ans =

     1
     2
     0

>>
```

```
% To solve this question we are creating a function BsSubs which is taking
% two input U and b and returning an ouput matrix x
% Note----We have to first give the U and b as input

function [x] = BdSubs(U, b)

[m, n] = size(U);               %copying the size of U in m and n variables

%As we are doing this procedure only for Square matrix so program will give
%an error if U is not square matrix
if m ~= n
error('U must be a square matrix');
end

%if U is square matrix then we create a column matrix with all elements zero
x = zeros(n, 1);

for i = 1:n
j = n+1-i;
%now we are using this equation
% In reality this equation is just back substitution method
% We can observe this thing when we run the for loop and put the required
% values in it
%Here we are creating a variable j and first we are calculating the value
%nth row of x ......We can easily observe this all just write these all
%matrix in systematic manner on notebook and then we can see that first we
%calculate the nth element of x then using that n-1th element of x and so on...
x(j) = (b(j) - U(j, 1:n) * x) / U(j, j);

end

end
```

```
>> U=[1,0,1;0,2,0;0,0,2]

U =

     1     0     1
     0     2     0
     0     0     2

>> b=[1;2;0]

b =

     1
     2
     0

>> BdSubs(U,b)

ans =

     1
     1
     0

>>
```

```
% Initial three lines are as per mentioned in question
p=8;
x = [2 -1 zeros(1, p-2)];
A = toeplitz(x);            %Creating a toeplitz matrix
[m ,n]=size(A);             %Copying the size of matrix A in m and n
B=ones(1,8);                %Creating another row matrix B in which all elements are 1

%Algorithm to calculate L and U
  L=zeros(m,m);
  U=zeros(m,m);
  for i=1:m

  for k=1:i-1
  L(i,k)=A(i,k);
  for j=1:k-1
  L(i,k)= L(i,k)-L(i,j)*U(j,k);
  end
  L(i,k) = L(i,k)/U(k,k);
  end

  for k=i:m
  U(i,k) = A(i,k);
  for j=1:i-1
  U(i,k)= U(i,k)-L(i,j)*U(j,k);
  end
  end
  end
  for i=1:m
  L(i,i)=1;
  end

  %Now we are using FdSubs and BdSubs functions and calculating the value
  %of X

  Y=FdSubs(L, B);
  X=BdSubs(U, Y);
   disp('x=');
   disp(x);
   disp('A=');
   disp(A);
   disp('L=');
   disp(L);
   disp('U=');
   disp(U);
   disp('X=');
   disp(X);
```

```
>> luSelfnP
x=
     2    -1     0     0     0     0     0     0

A=
     2    -1     0     0     0     0     0     0
    -1     2    -1     0     0     0     0     0
     0    -1     2    -1     0     0     0     0
     0     0    -1     2    -1     0     0     0
     0     0     0    -1     2    -1     0     0
     0     0     0     0    -1     2    -1     0
     0     0     0     0     0    -1     2    -1
     0     0     0     0     0     0    -1     2

L=
    1.0000         0         0         0         0         0         0         0
   -0.5000    1.0000         0         0         0         0         0         0
         0   -0.6667    1.0000         0         0         0         0         0
         0         0   -0.7500    1.0000         0         0         0         0
         0         0         0   -0.8000    1.0000         0         0         0
         0         0         0         0   -0.8333    1.0000         0         0
         0         0         0         0         0   -0.8571    1.0000         0
         0         0         0         0         0         0   -0.8750    1.0000

U=
    2.0000   -1.0000         0         0         0         0         0         0
         0    1.5000   -1.0000         0         0         0         0         0
         0         0    1.3333   -1.0000         0         0         0         0
         0         0         0    1.2500   -1.0000         0         0         0
         0         0         0         0    1.2000   -1.0000         0         0
         0         0         0         0         0    1.1667   -1.0000         0
         0         0         0         0         0         0    1.1429   -1.0000
         0         0         0         0         0         0         0    1.1250

X=
    4.0000
    7.0000
    9.0000
   10.0000
   10.0000
    9.0000
    7.0000
    4.0000

>>
```