UNIVERSITY OF LEICESTER

Computer Assignment 2

For Scientific Computing MA3012/MA7012

Gleb Vorobchuk

Leicester

October 2017

UNIVERSITY OF LEICESTER

Task №1.

Consider the system of linear equations

$$\begin{bmatrix} 5 & 3 & 36 & 7 \\ 15 & 9 & 28 & -1 \\ -23 & -4 & -13 & 7 \\ 9 & 3 & 40 & -2 \end{bmatrix} x = \begin{bmatrix} 67 \\ 63 \\ -52 \\ 88 \end{bmatrix}.$$

A)Try to solve this system using Matlab function gaussel.

Code of the function:

```
function [x] = gaussel(A,b)
% [x] = gaussel(A,b)
%
%   This subroutine will perform
Gaussian elimination
%   and back substitution to solve the
system Ax = b.
%   INPUT :  A - matrix for the left
hand side.
%            b - vector for the right
hand side
%
%   OUTPUT : x - the solution vector.
N = max(size(A));
% Perform Gaussian Elimination
 for j=2:N,
     for i=j:N,
        m = A(i,j-1)/A(j-1,j-1);
        A(i,:) = A(i,:) - A(j-1,:)*m;
        b(i) = b(i) - m*b(j-1);
     end
     A
% Perform back substitution
 x = zeros(N,1);
 x(N) = b(N)/A(N,N);
 for j=N-1:-1:1,
   x(j) = (b(j)-A(j,j+1:N)*x(j+1:N))/
A(j,j);
 end
end
% End of function
```

Results of computation:

```
ans =

   NaN
   NaN
   NaN
   NaN
```

We can see that this function is not working properly.

This is due to the fact that we need to eliminate the first element in the second row of the matrix. To do this, we need to multiply the first line by 3 and take it from the second line.

Matrix A after first iteration:

```
A =

    5.0000    3.0000   36.0000    7.0000
         0         0  -80.0000  -22.0000
         0    9.8000  152.6000   39.2000
         0   -2.4000  -24.8000  -14.6000
```

Now we have the second element in second row that equal to zero.

In order to continue elimination we need to multiply second row by $\frac{9.8}{0}$ which leads us to divide by zero.

Matrix A after second iteration:

```
A =

     5      3     36      7
     0      0    -80    -22
   NaN    NaN    Inf    Inf
   NaN    NaN   -Inf   -Inf
```

Matrix A after third iteration:

```
A =

     5     3    36     7
     0     0   -80   -22
   NaN   NaN   Inf   Inf
   NaN   NaN   NaN   NaN
```

And vector *x* is equal to:

```
ans =

   NaN
   NaN
   NaN
   NaN
```

B)Modify the function gaussel to implement gaussian elimination with scaled partial pivoting.

Matlab code:

```
function [x,l] = gaussel_spp(A,b)
format long
x_hat=A\b
% [x] = gaussel(A,b)
%
%   This subroutine will perform Gaussian elimination
%   and back substitution to solve the system Ax = b.
%   INPUT :  A - matrix for the left hand side.
%            b - vector for the right hand side
%
%   OUTPUT : x - the solution vector
N = max(size(A));
%Define vector l with pivot rows on each step
 for i=1:N,
    l(i)=i;
 end
%Define vector s
for i=1:N,
     s(i)=max(abs(A(i,:)));
end
 % Perform Gaussian Elimination
 for j=1:N,
     for i=1:N,
        if i<j
            p(i)=0; %Eliminate to chose new pivot every time
        else
        p(i)=abs(A(i,j)/s(i));% Find pivot element
        end
        [M,I] = max(p(:));%Get index of pivot element
     end
     A([I,j],:)=A([j,I],:) %Swap rows
     b([I,j],:) = b([j,I],:)%Swap values of b
     s([I j])=s([j I]) %Swap s values to prevent elimination of unused values.
     l([I j])=l([j I]) %Swap values of l
      for i=j:N-1,
         m(i)=A(i+1,j)/A(j,j)
         A(i+1,:) = A(i+1,:) - A(j,:)*m(i);
         b(i+1) = b(i+1) - m(i)*b(j);
      end
      A
 end
% Perform back substitution
 x = zeros(N,1);
 x(N) = b(N)/A(N,N);
 for j=N-1:-1:1,
   x(j) = (b(j)-A(j,j+1:N)*x(j+1:N))/A(j,j);
 end
 l
 %Calculating the error
 x_r=[1;-1;2;-1]
 e_1=abs(x-x_r)%error with calculated values.
 e_2=abs(x_hat-x_r)%error with A\b values
 n_e_1 = sqrt(sum(e_1.^2)) % norms of the errors
 n_e_2 = sqrt(sum(e_2.^2))
end
% End of function
```

Results:

```
ans =

   1.000000000000000
  -0.999999999999999
   2.000000000000000
  -0.999999999999999
```

```
A =

 -23.000000000000000  -4.000000000000000 -13.000000000000000   7.000000000000000
                   0   6.391304347826087  19.521739130434781   3.565217391304348
                   0                   0  30.530612244897959  -0.061224489795918
                   0                   0                   0   7.386809269162210


l =

     3     2     4     1
```

The error of x.

```
e_1 =

   1.0e-14 *

   0.044408920985006
   0.122124532708767
   0.022204460492503
   0.144328993201270
```

The error of A\b:

```
e_2 =

   1.0e-14 *

   0.088817841970013
   0.022204460492503
   0.044408920985006
   0.155431223447522
```

Norms of errors X and A\b respectively:

```
n_e_1 =

     1.954749347017227e-15


n_e_2 =

     1.857758450483250e-15
```

As we can see the error is less in case of using A\b. This method is more accurate.

Task №2.

Implement the Newton's formula.

Matlab code:

```
function [x]=run_newton(f,df,x0,N,sens)
x=x0;
i=0;
x_hat = 1;%real root of polynom
while i<N
    if abs(f(x)) < sens
        x
        i = N+1;
        else if abs(df(x)) < sens
            disp("error")
            disp(x)
            i = N+1;
        else
            x = x - (f(x)/df(x)); %implementation of
                                    Newton's formula
            i = i + 1;
            l(i) = f(x);
            e_k(i) = x - x_hat;
        end
    end
end
    %Calculate e_k and e_k+1
    e_k_1 = [];
    e_k_1 = e_k;
    e_k(length(e_k))=[];

    %linear_model=fitlm(log(e_k),log(e_k_1));
    %Coefficients of the linear model
    %lin_coef=linear_model.Coefficients.Estimate;
    %disp(lin_coef)
    %r = lin_coef(2)%r is the second linear_coeficient
    n_e_1  = sqrt(sum(e_k.^2)) %norms of the errors
    n_e_2 = sqrt(sum(e_k_1.^2))
end
```

Function was started with following parameters:

```
run_newton(@(x)(x.^5-x.^4-4*x.^3+4*x.^2+5*x-5),@(x)(5*x.^4-4*x.^3-12*x.^2+8*x+5),0.001,100,0.001)
```

Results:

```
ans =

   0.999994940944266


x =

   0.999994940944266


n_e_1 =

   0.001594253822966


n_e_2 =

   0.001594261849912
```