15 April 2019

Patryk Ẇalczak
Group 2

Project 14

# Solving a system of equations $Ax = b$, where $A \in \mathbb{R}^{n\times n}$ is a Hessenberg matrix by the Jacobi method.

# 1 Method description

In linear algebra and numerical analysis, a Hessenberg matrix is a special kind of square matrix, which has zero entries below the first subdiagonal.

$$\begin{pmatrix} x_{11} & x_{12} & x_{13} & x_{14} & \dots & x_{1n} \\ x_{21} & x_{22} & x_{23} & x_{24} & \dots & x_{2n} \\ 0 & x_{32} & x_{33} & x_{34} & \dots & x_{3n} \\ 0 & 0 & x_{43} & x_{44} & \dots & x_{4n} \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & x_{nn-1} & x_{nn} \end{pmatrix}$$

The Jacobi method is an iterative process for solving system of linear equations $Ax = b$, where $A \in \mathbb{R}^{n \times n}$,
$b \in \mathbb{R}^n$ and $a_{ii} \neq 0$ for $i = 1, ..., n$.

The method starts from a given initial guess $x(0)$ it constructs a sequence $x(k)$ of consecutive approximations, such that $x(k) \to x$ as $k \to \infty$.

$$x_1 = (b_1 - \sum_{j=2}^{n} a_{1j} \cdot x_j)/a_{11}$$

$$x_2 = (b_2 - \sum_{j=1, j\neq 2}^{n} a_{2j} \cdot x_j)/a_{22}$$

$$\vdots$$

$$x_n = (b_n - \sum_{j=n-1}^{n} a_{nj} \cdot x_j)/a_{nn}$$

But in the project considering is only Hessenberg matrix, so the amount of iteration can be lower.

$$x_1 = (b_1 - \sum_{j=2}^{n} a_{1j} \cdot x_j)/a_{11}$$

$$\vdots$$

$$x_m = (b_m - \sum_{j=m-1, j\neq m}^{n} a_{mj} \cdot x_j)/a_{mm} \text{ for } m \in \langle 2, n)$$

Starting from a given initial guess $x^{(0)} = (x_1^{(0)}, ..., x_n^{(0)})^T \in \mathbb{R}^n$ we use the above formulas to compute the approximations $x^{(k)}$.

$$x_1^{(0)} = (b_1 - \sum_{j=2}^{n} a_{1j} \cdot x_1^{(0)})/a_{11}$$

$$\vdots$$

$$x_m^{(0)} = (b_m - \sum_{j=m-1, j\neq m}^{n} a_{mj} \cdot x_j^{(0)})/a_{mm} \text{ for } m \in \langle 2, n)$$

Possible stopping criteria in my algorithm:

1. $\|x^{(k+1)} - x^{(k)}\| <$ tolerance, which is set up by user.
2. Number of iteration reaches the limit, which is set up by user.

When the Jacobi method convergent, then it gives correct solution, otherwise limit of iteration has to occure.

## 2 Description of Matlab program

After run the program, the Menu appears:

| |
|---|
| **Menu** |
| Change Hessenber matrix A |
| Change vector b |
| Change number of maximal iterations |
| Change tolerance |
| Display variables |
| Change example |
| Compute Ax=b |
| FINISH |

After pushing:

- Change Hessenber matrix A - user will be able to input their own matrix. The program checks if it is Hessenberg matrix and if not the substitue matrix is choosen
- Change vector b - user will be able to input their own vector.
- Change number of maximal iterations - user will be able to input their own limit of iterations.
- Change tolerance - user will be able to input their own tolerance of error.
- Display variables - user will be able to see all variables.
- Change example - change the default variables.
- Compute Ax=b - program will check if size of A and b are correct, if that's true program will compute spectral radius of matrix A, condition number of matrix A, values of x and number of needed iterations.
- FINISH - program will finish.

MATLAB functions:

1. Menu.m - script for graphic interface of menu, which uses rest of functions.
2. Jacobi_Method.m - function strictly for computing system of equationm Ax=b using Jacobi method, spectral radius of matrix A, condition number of matrix A, return solve of equation and number of iterations.
3. Is_Hessenberg.m - fuction checking if the input Matrix is a Hesseneberg matrix, return input matrix if it is true, and hess(input matrix) if that's false.

   (**Note.** *All source codes can be found in section 5.*)

# 3 Numerical tests

All the numerical tests are included in the Menu.m file, while user pushes button 'Change example', the example data are changed. Program computes spectral radius of matrix A, condition number of matrix A, values of x and number of needed iterations for input data. In every test the variables are different, for checking the correctnes of program there is error also.

$$error = \frac{\|X - Z\|}{\|Z\|},$$

The spectral radius of matrix $A$: rho_A=max(abs(eig(A))).
Find the condition number of $A$: $cond(A) = \|A^{-1}\| \cdot \|A\|$.

**Tests:**
$x$ is the result we obtain using our implemented Jacobi_method function, whereas $Z$ is the predefined solution wich was used to get vector $b$ by multiplication $b = A * Z$.

1.
$$A = \begin{pmatrix} 10.0000 & -0.0000 & 0 & 0 \\ -0.0000 & 10.0000 & -0.0000 & 0 \\ 0 & -0.0000 & 13.0000 & -1.7321 \\ 0 & 0 & -1.7321 & 11.0000 \end{pmatrix}, \; b = \begin{pmatrix} 10.0000 \\ 20.0000 \\ 32.0718 \\ 38.8038 \end{pmatrix},$$

$$\text{max iterator} = 100, \text{ tolerance} = 1.0000e^{-6}$$

Result:

Spectral radius of A is equal 0.144841
Condition number of A is equal 1.400000
After 8 iteriations result is:

$$\begin{pmatrix} 1.0000 \\ 2.0000 \\ 3.0000 \\ 4.0000 \end{pmatrix}$$

$$\text{And error is equal:} 2.518573e^{-8}$$

In the test the error is really small, and number of iteration didn't reach the limit, so result is correct.

2.

$$A = \begin{pmatrix} 4.0000 & 0 & 3.6056 \\ -3.6056 & -3.3077 & 5.5385 \\ 0 & 4.5385 & 6.3077 \end{pmatrix}, \; b = \begin{pmatrix} 129.3499 \\ 167.0017 \\ 106.7692 \end{pmatrix},$$

$$\text{max iterator} = 100, 1.0000e^{-6}$$

Result:

Spectral radius of A is equal 1.201809
Condition number of A is equal 2.281605
After 100 iteriations result is:

$$\begin{pmatrix} 1.0e+09 \\ -0.6466 \\ 3.0451 \\ -2.3594 \end{pmatrix}$$

$$\text{And error is equal: } 1.242046e^{8}$$

In the test the error is really big, and number of iteration reached the limit, so result is incorrect.

3.

$$A = \begin{pmatrix} 8.6364 & 1.1499 & 0 & 0 \\ 1.1499 & 10.8636 & 1.6583 & 0 \\ 0 & 1.6583 & 11.5000 & 3.1623 \\ 0 & 0 & 3.1623 & 8.0000 \end{pmatrix}, \; b = \begin{pmatrix} 10.9362 \\ 27.8521 \\ 50.4657 \\ 41.4868 \end{pmatrix},$$

$$\text{max iterator} = 100, \text{ tolerance} = 1.0000e^{-6}$$

Result:

Spectral radius of A is equal 0.365116
Condition number of A is equal 2.359728
After 16 iteriations result is:

$$\begin{pmatrix} 1.0000 \\ 2.0000 \\ 3.0000 \\ 4.0000 \end{pmatrix}$$

And error is equal: $3.517808e^{-8}$

In the test the error is really low, and number of iteration didn't reach the limit, so result is correct.

# 4 Conclusion

As the test shows, the Jacobi method is not correct when it is divergent. But if it is convergent the correct answer occurs. For Hessenberg matrix many computations are ommited, so the result appears after lower amount of iterations. In the second test after one hunderd iteration there is no result, and the error is huge. On the other hand, in first and second test Jacobi methos gived result. Moreover, when spectral radius is bigger then one (as in second test) method is not convergent, but if it is lower then one (as in the rest of tests) method is convergent.

# 5 Source Codes

**Is_Hessenberg.m :**

```
function M = Is_Hessenberg(A)
% Is A a Hessenber matrix
M=hess(A);
[m,n]=size(A);
if m~=n
    disp('Your matrix is not a Hassenberg')
    return;
end
Z=zeros(n);
C=tril(A,-2);
if ~isequal(Z,C)
    disp('Your matrix is not a Hassenberg')
    return;
end
M=A;
end
```

**Jacobi_Method.m :**

```
function [x,k] = Jacobi_Method(A,b,max_iterations,tolerance)
% Jacobi's method for Ax=b

n=max(size(A));
x0=zeros(n,1);
x=x0;
dx=tolerance+1;
k=0;

I=eye(n);
D=diag(diag(A));
B=I-D\A;
rho_B=max(abs(eig(B)));
fprintf('Spectral radius of A is equal %f \n',rho_B);
cond_A=cond(A);
fprintf('Condition number of A is equal %f \n',cond_A);

d=diag(A);
if ~all(d)
    disp('Diagonal element of A is equal to 0!');
    return;
end

while norm(dx)> tolerance && k<= max_iterations
       for i=1:n
           s=b(i);    %i-th element of vector b
           m=1;
           if i>=2
               m=i-1;
           end
           for j=[m:i-1,i+1:n]
               %bi- Sum(from 1 to n without j) * aij * x0j
                               s=s-A(i,j)*x0(j);
           end
           x(i)=s/A(i,i);      %xi= s/aii
       end

      dx=x-x0;  %compute diference between x and x0
      k=k+1;     %next iteration
      x0=x;      %save x as x0
end
k=k-1;
end
```

**Menu.m:**

```
% MENU
clear
clc
finish=8;
control=1;
ex=0;

%default data %
A=ones(4)+10*eye(4);
A=hess(A);
Z=(1:4)';b=A*Z;
max_iter=100;
tol = 1e-6;

while control~=finish

    control=menu('Menu', 'Change Hessenber matrix A', 'Change vector b',
        'Change number of maximal iterations','Change tolerance',
        'Display variables','Change example','Compute Ax=b','FINISH');

    switch control
        case 1
            A=input('A= ');
            A=Is_Hessenberg(A);

        case 2
            b=input('b= ');

        case 3
            max_iter=input('max iterations= ');

        case 4
            tol=input('tolerance= ');
        case 5
            disp('A= ');disp(A)
            disp('b= ');disp(b)
            disp('max iterator= ');disp(max_iter)
            disp('tolerance= ');disp(tol)

        case 6
            if ex==0
                A=[4,2,3;3,-5,2;-2,3,8];
                A=hess(A);
                Z=[8;-14;27];b=A*Z;
```

```
            end
            if ex==1
                A=ones(4)+10*eye(4);
                A=hess(A);
                Z=(1:4)';b=A*Z;
            end
            if ex==2
                A=[10,-1,2,0;-1,11,-1,3;2,-1,10,-1;0,3,-1,8];
                A=hess(A);
                Z=[1;2;3;4];b=A*Z;

            end
            if ex==3
                disp('Teraz')
                A=triu(ones(6),-1);
                Z=[1;2;3;4;5;6];b=A*Z;

            end
            ex=ex+1;
            ex=mod(ex,4);

        case 7
            n=max(size(A));
            m=max(size(b));
            disp(n)
            disp(m)
            if m==n
                [result,iteration] = Jacobi_Method(A,b,max_iter,tol);
                err_x=norm(result-Z)/norm(Z);
                fprintf('After %d iteriations result is:\n',iteration);
                disp(result)
                fprintf('And error is equal: %d\n',err_x);
            else
                fprintf('Iccorect size of A and b\n');
            end

        case 8
            disp('FINISH')
    end
end
```