重庆大学

机械工程中的数值分析技术课程期中项目报告

2020 至 2021 学年第二学期

项目名称：A Linear Equations Solver Based on Matlab

学生姓名：易弘睿

学院专业班级：UC 联合学院 2018 级机械 01 班

学　　号：20186103

任课教师：温罗生

重庆大学数学与统计学院

Catalog

1. Abstract

In this project, a linear equation solver based on MATLAB is designed by combining the knowledge of Gauss elimination, partial pivoting, upper / lower triangular matrix, triangular system, LU decomposition, Jacobi, Gauss-Seidal, successive over relaxation method (SOR) method and so on. This report makes the introduction for users, introduces the calculation logic of the solver in detail, especially the automatic selection of the optimal method. The part introduces how the solver automatically selects the optimal algorithm and solves special problems with high efficiency as the goal. Finally, a series of examples show that the solver could solve different kinds of linear equations, from small to large matrix problems, and could reach the results quickly and accurately.

2. Instruction of operation

2.1 Function and parameters

The function of this solver is

```
ans = Solver(A,b,delta,omega)
```

The meaning of each parameter is as follows:

A: Coefficient Matrix

B: RHS Vector

delta: Relative error used to judge when to stop for iterative methods such as Jacobi, Gauss-Seidal and SOR, which is not necessary.

omega: Weight coefficient for SOR method, which is not necessary.

2.2 Input

In this solver, the user needs to enter the function ahead of time. For example:

```
ans = Solver([1 2 0 3 0 0;4 0 5 0 6 0;0 7 8 9 0 0;10 0 11 12 13 0;0 14 0 15 16 17;0 0 18 0 19 0], ones(6,1));
```

2.3 Output

We could get the output in command window of matlab using the example above:

```
Cool!There is only one unique solution.
----------------------List of different methods for solving---------------------
Method 1:Gauss elimination without pivoting (Only for square matrix)
```

```
Method 2:Partial Pivoting (For square matrix which can be divided by 0 but not
2)
Method 3:Upper triangular matrix(For upper triangular matrix
Method 4:Lower triangular matrix(For lower triangular matrix
Method 5:Tridiagonal System (For tridiagonal banded matrix)
Method 6:LU Decomposition (For very large matrix)
Method 7:Jacobi(For all elements in diagonal are not 0; sparse matrix;
diagonally dominant)
Method 8:Gauss-Seidal Iteration(The same as Jacobi)
Method 9:SOR(The same as Jacobi but need an extra nargin lambda)
Method 10:Just give me the most exact solution!
--------------------------------------------------------------------------------
--
----------------Please choose the solving methods----------------
Input 's'if you want me to select the optimal method automatically.
Otherwise, enter number 1~10 to choose a method you prefer.
-----------------------------------------------------------------
Now please make your choice: s
--------------Gaussian Elimination with pivoting----------------
The running time for the Gaussian Elimination with pivoting is 0.00563380 s.
The maximum relative error for the Gaussian Elimination with pivoting is
0.0000  %.
----------------------------------------------------------------
```

3. Introduction of calculation logic

3.1 Overall progress

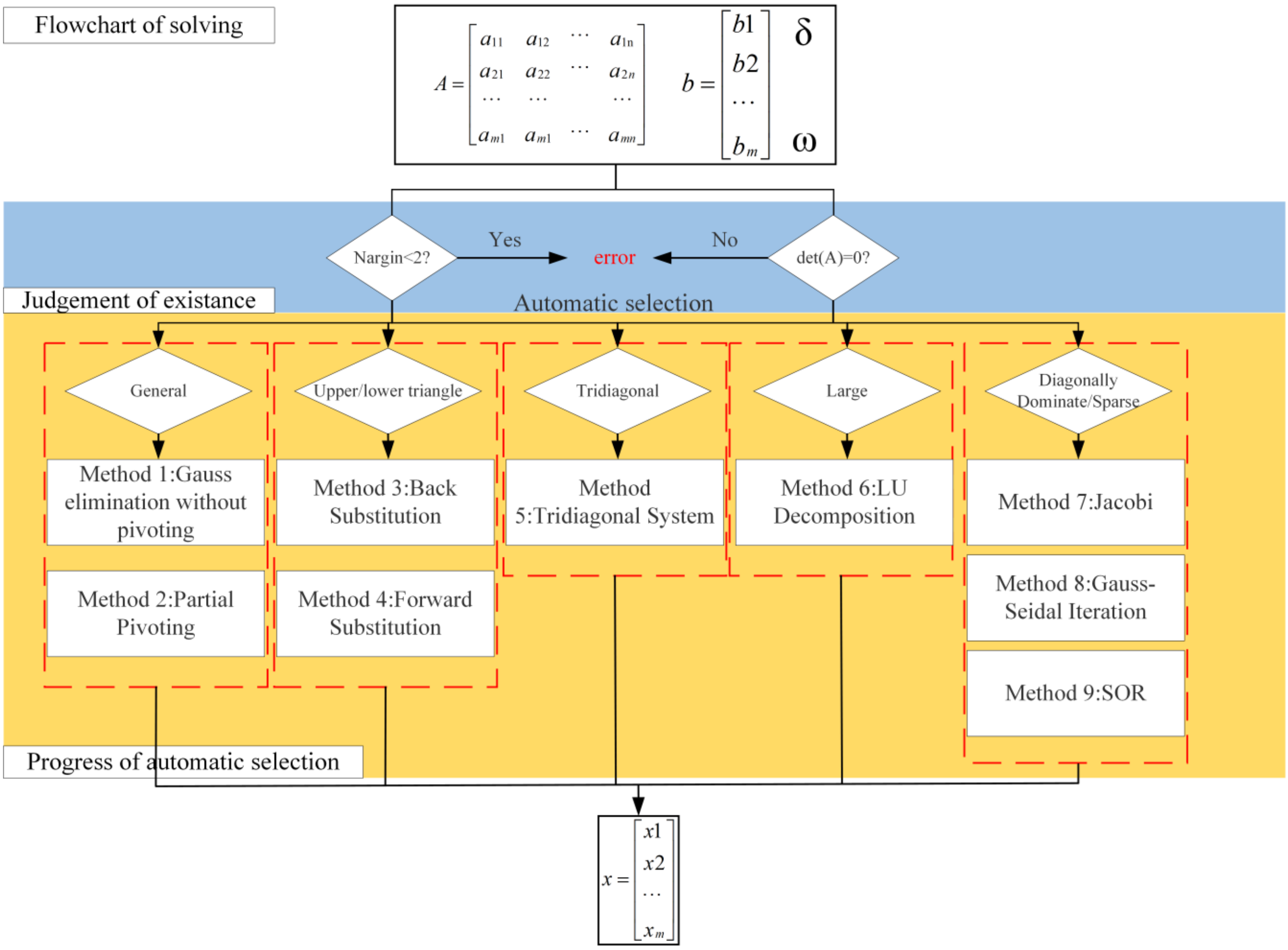

Fig. 1. Flowchart of solving procedure

In practice, the solving procedure based on matlab mainly contains three processes. The user could choose to solve equations manually or automatically. Specifically, the procedure of the automatic solving method is described as follows:

1) The coefficient matrix $A=\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & & \cdots \\ a_{m1} & a_{m1} & \cdots & a_{mn} \end{bmatrix}$ and RHS vector $b=\begin{bmatrix} b1 \\ b2 \\ \cdots \\ b_m \end{bmatrix}$ are entered by the user.

2) Judgment if det(A)=0 and Nargin<2 are determined by the algorithm.

3) Decision of manual solving and automatic solving is chosen by the user.

4) If manual solving method is chosen, $x = \begin{bmatrix} x1 \\ x2 \\ \cdots \\ x_m \end{bmatrix}$ would be achieved by corresponding solving method with relative error and running time; if automatic solving method is chosen, the procedure will further proceed.
5) Judgment of different conditions for corresponding method is determined by the algorithm.
6) $x = \begin{bmatrix} x1 \\ x2 \\ \cdots \\ x_m \end{bmatrix}$ is achieved by the optimal method with corresponding relative error and running time.

There are totally ten methods for users to choose in the algorithm as follows:

Method 1: Gauss elimination without pivoting (Only for square matrix);

Method 2: Partial Pivoting (For square matrix which can be divided by 0 but not 2);

Method 3: Back substitution (For upper triangular matrix);

Method 4: Forward substitution (For lower triangular matrix);

Method 5: Tridiagonal System (For tridiagonal banded matrix);

Method 6: LU Decomposition (For very large matrix);

Method 7: Jacobi (For all elements in diagonal are not 0; sparse matrix; diagonally dominant);

Method 8: Gauss-Seidal Iteration (The same as Jacobi);

Method 9: SOR(The same as Jacobi but need an extra parameter omega);

Method 10: Exact solution.

3.2 Judgement of the input

The MATLAB code is below:

```
% 判断输入参数个数并由此确定函数变量
if nargin < 2
    error('At least 2 arguments are required. Please check the input.')
elseif nargin < 3
    delta = 0.00001;
    omega = 0.1:0.1:2;
elseif nargin < 4
    omega = 0.1:0.1:2;
end

% 判断行列式并由此确定方程组解的情况
if det(A) > 10^(-6)
    fprintf('Cool!There is only one unique solution.\n')
elseif det(A) <= 10^(-6) && rank(b) == 0
    error('Oh! There are infinite solutions.\n')
elseif det(A) <= 10^(-6) && rank(b) ~= 0
    error('Oh! There generally no solution at all.\n')
end

[A_row,A_col] = size(A);
[B_row,B_col] = size(b);

% 判断是否为square matrix
if A_row ~= A_col
    error( 'A is not square. Please check the input.')
elseif B_col ~= 1
    error( 'B is not a column vector. Please check the input.')
elseif A_row ~= B_row
    error( 'The size does not match. Please check the input.')
end
```

3.3 Choosing progress

```
% 用户选择
choice = zeros(1,10);
disp('-----------------------List of different methods for solving-------------------------')
disp('Method 1:Gauss elimination without pivoting (Only for square matrix)')
```

```
disp('Method 2:Partial Pivoting (For square matrix which can be divided by 0 but not 2)')
disp('Method 3:Back substitution(For upper triangular matrix')
disp('Method 4:Forward substitution(For lower triangular matrix')
disp('Method 5:Tridiagonal System (For tridiagonal banded matrix)')
disp('Method 6:LU Decomposition (For very large matrix)')
disp('Method 7:Jacobi(For all elements in diagonal are not 0; sparse matrix; diagonally dominant)')
disp('Method 8:Gauss-Seidal Iteration(The same as Jacobi)')
disp('Method 9:SOR(The same as Jacobi but need an extra nargin omega)')
disp('Method 10:Just give me the most exact solution!')
disp('--------------------------------------------------------------------------------------')

while 1
    disp('----------------Please choose the solving methods----------------')
    disp('Input ''s''if you want me to select the optimal method automatically.')
    disp('Otherwise, enter number 1~10 to choose a method you prefer.')
    disp('-----------------------------------------------------------------')
    str = input('Now please meke your choice: ','s');

    if str == 'q'
        break
    elseif str == 's'
        break
    else
        choice(eval(str)) = 1;

        break

    end

end
```

3.4 Automatic solving progress

3.4.1 Upper/lower triangular matrix

```
% 判断是否为上三角或下三角矩阵
        if A == triu(A)
            choice = zeros(1,10);
```

```
            choice(3) = 1;
            break
        elseif A == tril(A)
            choice = zeros(1,10);
            choice(4) = 1;
            break
        else
            choice(3:4) = 0;
        end
```

3.4.2 Tridiagonal system

```
% 判断是否为三对角矩阵
        f = diag(A);
        n = length(diag(A));
        new_A = zeros(n,n);
        g = zeros(1,n);
        e = zeros(1,n);
        for i = 1:length(g)-1
            g(i) = A(i,i+1);
            e(i+1) = A(i+1,i);
        end
        for i = 1:n
            new_A(i,i) = f(i);
        end
        for i = 1:n-1
            new_A(i,i+1) = g(i);
            new_A(i+1,i) = e(i+1);
        end
        if A == new_A
            istri_banded = true;
            choice = zeros(1,10);
            choice(5) = 1;
            break
        else
            istri_banded = false;
            choice(5) = 0;
        end
```

3.4.3 *LU decomposition*

```
% 判断是否为大型矩阵
        for j = 1:size(A,2)
            D(1,j) = det(A(1:j,1:j));
        end
        if A_row*A_col>= 10000  && all(D)
            choice = zeros(1,10);
            choice(6) = 1;
            break
        else
            choice(6) = 0;

        end
```

3.4.4 *Jacobi, Gauss-seidal and SOR*

```
% 判断对角线是否有0
        if any(diag(A))
            choice(7:9) = 0;
        else
            count = 0;
            [rows,cols] = size(A);
            matrix_size = rows*cols;
            for i = 1:rows
                for k = 1:cols
                    if A(i,k) == 0
                        count = count + 1;
                    end
                end
            end

            % 判断是否为稀疏矩阵
            if count > matrix_size/2
                issparse = true;
            else
                issparse = false;
            end
            if issparse
                is_dominant = Diagonally_dominant(A);

                if is_dominant
```

```
                choice = zeros(1,10);
                choice(8) = 1;
                break
            else
                choice(7:9) = 0;
            end
        else
            choice(7:9) = 0;
        end
    end
```

3.5 Solving progress for manual Choosing

```
if choice(1)
    disp('--------------Gaussian Elimination without pivoting-------------')
    ans = Gaussian_Elimination_LinearEQ(A,b);
    relative_error = max(Err(ans,exact_solution));
    fprintf('The maximum relative error for the Gaussian Elimination
is %.4f  %%.\n',relative_error)
    disp('-----------------------------------------------------------------')
end
if choice(2)
    disp('--------------Gaussian Elimination with pivoting----------------')
    ans = Gaussian_Elimination_Pivoting_LinearEQ(A,b);
    relative_error = max(Err(ans,exact_solution));
    fprintf('The maximum relative error for the Gaussian Elimination with
pivoting is %.4f  %%.\n',relative_error)
    disp('-----------------------------------------------------------------')
end
if choice(3)
    disp('---------Back Substitution for Upper Triangular Matrix-----------')
    if A == triu(A)
        ans = BackSub_TriU(A,b);
        relative_error = max(Err(ans,exact_solution));
        fprintf('The maximum relative error for the Back Substitution for Upper
Triangular Matrix is %.4f  %%.\n',relative_error)
    else
        disp('Not a Upper Triangular Matrix.')
    end
    disp('-----------------------------------------------------------------')
end
if choice(4)
    disp('------Forward Substitution for the Lower Triangular Matrix------')
```

```
    if A == tril(A)
        ans = ForwardSub_TriL(A,b);
        relative_error = max(Err(ans,exact_solution));
        fprintf('The maximum relative error for the Forward Substitution for
Lower Triangular Matrix is %.4f  %%.\n',relative_error)
    else
        disp('Not a Lower Triangular Matrix.')
    end
    disp('----------------------------------------------------------------')
end

if choice(5)
    disp('--------------------------Banded Matrix-------------------------')
    if istri_banded
        ans = Tridiagonal_Matrix_LinearEQ(b,f,g,e,n);
        relative_error = max(Err(ans,exact_solution));
        fprintf('The maximum relative error for the TridiagonalBanded Matrix
Elimination is %.4f  %%.\n',relative_error)
    else
        disp('It not a tridiagonal banded matrix.')
    end
    disp('----------------------------------------------------------------')
end
if choice(6)
    disp('------------------------LU Decomposition------------------------')
    ans = LU_Decomposition_LinearEQ(A,b);
    relative_error = max(Err(ans,exact_solution));
    fprintf('The maximum relative error for the LU Decomposition method
is %.4f  %%.\n',relative_error)
    disp('----------------------------------------------------------------')
end
if choice(7)
    disp('-----------------------Jacobi Iterative-------------------------')
    ans = Jacobi_Iterative_LinearEQ(A,B,delta);
    relative_error = max(Err(ans,exact_solution));
    fprintf('The maximum relative error for the Jacobi method
is %.4f  %%.\n',relative_error)
    disp('----------------------------------------------------------------')
end
if choice(8) == 1
    disp('---------------------Gauss-Seidal Iterative---------------------')
    ans = Gauss_Seidal_Iterative_LinearEQ(A,b,delta);
    relative_error = max(Err(ans,exact_solution));
    fprintf('The maximum relative error for the Gauss-Seidal
is %.4f  %%.\n',relative_error)
```

```
    disp('----------------------------------------------------------------')
end
if choice(9)
    disp('---------------------------SOR Method----------------------------')
    [time,relative_error,best_lambda,ans] = SOR_LinearEQ(A,b,delta,omega);
    fprintf('The best running time for the SOR method is %.8f s.\n',time)
    fprintf('The maximum relative error for the SOR method with best lambda
is %.4f  %%.\n',relative_error)
    disp('----------------------------------------------------------------')
end
if choice(10)
    disp('------------------------Exact solution--------------------------')
    disp('The exact solution is')
    disp(exact_solution)
    disp('----------------------------------------------------------------')
end
```

4. Examples and Results

4.1 General situation

For general situation, which is non-diagonal, not diagonally dominant, small matrix:

```
ans = Solver([1 2 0 3 0 0;4 0 5 0 6 0;0 7 8 9 0 0;10 0 11 12 13 0;0 14 0 15 16 17;0 0 18 0 19 0], ones(6,1))
```

```
-----------------------Partial Pivoting-------------------------
The running time for the partial pivoting is 0.00406440 s.
The maximum relative error for the Partial Pivoting is 0.0000  %.
----------------------------------------------------------------
```

Comparison with other methods:

```
-------------Gaussian Elimination without pivoting-------------
The running time for the Gaussion Elimination without pivoting is 0.00488420 s.
The maximum relative error for the Gaussian Elimination without pivoting is 0.0000  %.
----------------------------------------------------------------
```

```
------------------------LU Decomposition------------------------
The running time for the LU Decomposition is 0.00424310 s.
The maximum relative error for the LU Decomposition method is 0.0000  %.
----------------------------------------------------------------
```

Table 1. Comparison of general situation

Method	Running time (s)	Error (%)
Partial Pivoting	0.0048644	0
Gaussian Elimination without pivoting	0.0040842	0
LU Decomposition	0.0042431	0

4.2 Upper/lower triangular matrix

For upper or lower triangular matrix:

```
ans = Solver([1 2 3 4 5 6;0 7 8 9 10 11;0 0 12 13 14 15;0 0 0 16 17 18;0 0 0 0 19 20;0 0 0 0 0 21],ones(6,1));
```

```
---------Back Substitution for Upper Triangular Matrix-----------
The running time for the Back Substitution for Upper Triangular Matrix is 0.00163660 s.
The maximum relative error for the Back Substitution for Upper Triangular Matrix is 0.0000  %.
```

Comparison with other methods:

```
--------------Gaussian Elimination without pivoting-------------
The running time for the Gaussion Elimination without pivoting is 0.00399830 s.
The maximum relative error for the Gaussian Elimination without pivoting is 0.0000  %.
----------------------------------------------------------------
```

```
-----------------------Partial Pivoting-------------------------
The running time for the partial pivoting is 0.00469570 s.
The maximum relative error for the Partial Pivoting is 0.0000  %.
----------------------------------------------------------------
```

```
------------------------LU Decomposition------------------------
The running time for the LU Decomposition is 0.00410320 s.
The maximum relative error for the LU Decomposition method is 0.0000  %.
----------------------------------------------------------------
```

Table 2. Comparison of upper/lower triangular matrix

Method	Running time (s)	Error (%)

Back Substitution for Upper Triangular Matrix	0.0016366	0
Gaussian Elimination without pivoting	0.0039983	0
Partial Pivoting	0.0046957	0
LU Decomposition	0.0041032	0

4.3 Tridiagonal banded matrix

For tridiagonal banded matrix:

```
ans = Solver([1 2 0 0 0 0;3 4 5 0 0 0;0 6 7 8 0 0;0 0 9 10 11 0;0 0 0 12 13 14;0 0 0 0 15 16],ones(6,1));
```

```
--------------------------Banded Matrix-------------------------
The running time for the Tridiagonal banded matrix Elimination is 0.00183680 s.
The maximum relative error for the TridiagonalBanded Matrix Elimination is
0.0000  %.
----------------------------------------------------------------
```

Comparison with other methods:

```
--------------Gaussian Elimination without pivoting-------------
The running time for the Gaussion Elimination without pivoting is 0.00304440 s.
The maximum relative error for the Gaussian Elimination without pivoting is
0.0000  %.
----------------------------------------------------------------
```

```
-----------------------Partial Pivoting-------------------------
The running time for the partial pivoting is 0.00564110 s.
The maximum relative error for the Partial Pivoting is 0.0000  %.
----------------------------------------------------------------
```

```
------------------------LU Decomposition------------------------
The running time for the LU Decomposition is 0.00398300 s.
The maximum relative error for the LU Decomposition method is 0.0000  %.
```

```
------------------------------------------------------------------
```

Table 3. Comparison of tridiagonal banded matrix

Method	Running time (s)	Error (%)
Banded Matrix	0.0018368	0
Gaussian Elimination without pivoting	0.0030444	0
Partial Pivoting	0.0056411	0
LU Decomposition	0.003983	0

4.4 Large matrix

For large matrix, which is non-diagonal and not diagonally dominant:

```
ans = Solver(randi([1,100],1000,1000),randi([1,100],1000,1));
```

```
------------------------LU Decomposition------------------------
The running time for the LU Decomposition is 2.85665040 s.
The maximum relative error for the LU Decomposition method is 0.0000  %.
------------------------------------------------------------------
```

Comparison with other methods:

```
--------------Gaussian Elimination without pivoting-------------
The running time for the Gaussion Elimination without pivoting is 3.36288170 s.
The maximum relative error for the Gaussian Elimination without pivoting is
0.0000  %.
------------------------------------------------------------------
```

```
-----------------------Partial Pivoting-------------------------
The running time for the partial pivoting is 3.44457470 s.
The maximum relative error for the Partial Pivoting is 0.0000  %.
------------------------------------------------------------------
```

Table 4. Comparison of large matrix

Method	Running time (s)	Error (%)
LU Decomposition	2.8566504	0
Gaussian Elimination without pivoting	3.3628817	0
Partial Pivoting	3.4445747	0

4.5 Diagonally dominant, sparse, large matrix

For diagonal, sparse and large matrix:

```
ans = Solver([1 2 0 0 0 0;3 4 5 0 0 0;0 6 7 8 0 0;0 0 9 10 11 0;0 0 0 12 13 14;0 0 0 0 15 16],ones(6,1));
```

```
-------------------------Banded Matrix-------------------------
The running time for the Tridiagonal banded matrix Elimination is 0.00231540 s.
The maximum relative error for the TridiagonalBanded Matrix Elimination is
0.0000  %.
----------------------------------------------------------------
```

Comparison with other methods:

```
-------------Gaussian Elimination without pivoting-------------
The running time for the Gaussion Elimination without pivoting is 0.00387730 s.
The maximum relative error for the Gaussian Elimination without pivoting is
0.0000  %.
```

```
----------------------Partial Pivoting-------------------------
The running time for the partial pivoting is 0.00621260 s.
The maximum relative error for the Partial Pivoting is 0.0000  %.
----------------------------------------------------------------
```

Table 5. Comparison of diagonally dominant, sparse, large matrix

Method	Running time (s)	Error (%)
LU Decomposition	0.0023154	0

Gaussian Elimination without pivoting	0.0038773	0
Partial Pivoting	0.0062126	0

4.6 Iteration methods

To figure out the difference of three iteration methods, the comparison of errors varied with time is processed specifically:

```
A_rand = rand(1000, 1000);

b = ones(1000, 1);

A = triu(A_rand, -3);

A = tril(A, 5);

A = A + 10 * eye(1000);
```

```
The iteration of GaussSeidel is 11

The CPU time is 0.068154 s.

The iteration of SOR is 25

The CPU time is 0.102633 s.

The iteration of Jacobi is 9

The CPU time is 0.068347 s.
```

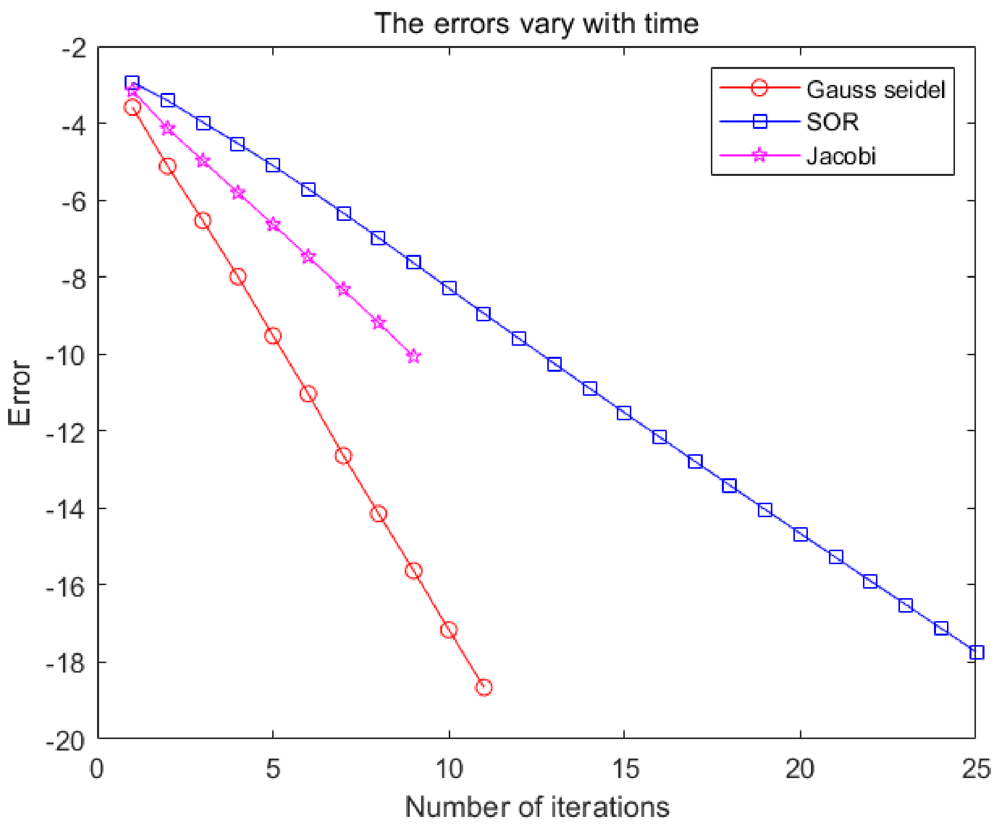

Fig. 2. The errors vary with time

5. Appendix

5.1 Code

5.1.1 main.m

```
%% A Linear Equations Solver
%  Name: Horace
%  Date: June 23 2021
%  Discription: This script makes a solver for linear equations.

clc;clear all;
% disp('This solver is for Ax=b.');
% A = input('A = ');
% b = input('b = ');
% ans = Solver(A, b);

ans = Solver([1 2 0 3 0 0;4 0 5 0 6 0;0 7 8 9 0 0;10 0 11 12 13 0;0 14 0 15 16
17;0 0 18 0 19 0], ones(6,1));

function ans = Solver(A,b,delta,omega)
% 判断输入参数个数并由此确定函数变量
if nargin < 2
    error('At least 2 parameters are required.')
elseif nargin < 3
    delta = 10^-5;
    omega = 0.1:0.1:2;
elseif nargin < 4
    omega = 0.1:0.1:2;
end

% 判断行列式并由此确定方程组解的情况
if det(A) > 10^(-6)
    fprintf('Cool!There is only one unique solution.\n')
elseif det(A) <= 10^(-6) && rank(b) ~= 0
    error('Oh! There is generally no solution at all.\n')
elseif det(A) <= 10^(-6) && rank(b) == 0
    error('Oh! There are infinite solutions.\n')
end

[A_row,A_col] = size(A);
[B_row,B_col] = size(b);
```

```
% 判断是否为square matrix
if A_row ~= A_col
    error( 'A is not square. Please check the input.')
elseif B_col ~= 1
    error( 'B is not a column vector. Please check the input.')
elseif A_row ~= B_row
    error( 'The size does not match. Please check the input.')
end

% 用户选择
choice = zeros(1,10);
disp('-----------------------List of different methods for solving-------------
----------')
disp('Method 1:Gauss elimination without pivoting (Only for square matrix)')
disp('Method 2:Partial Pivoting (For square matrix which can be divided by 0 but
not 2)')
disp('Method 3:Back substitution(For upper triangular matrix')
disp('Method 4:Forward substitution(For lower triangular matrix')
disp('Method 5:Tridiagonal System (For tridiagonal banded matrix)')
disp('Method 6:LU Decomposition (For very large matrix)')
disp('Method 7:Jacobi(For all elements in diagonal are not 0; sparse matrix;
diagonally dominant)')
disp('Method 8:Gauss-Seidal Iteration(The same as Jacobi)')
disp('Method 9:SOR(The same as Jacobi but need an extra nargin omega)')
disp('Method 10:Just give me the most exact solution!')
disp('--------------------------------------------------------------------------
---------')


while 1
    disp('----------------Please choose the solving methods----------------')
    disp('Input ''s''if you want me to select the optimal method
automatically.')
    disp('Otherwise, enter number 1~10 to choose a method you prefer.')
    disp('-----------------------------------------------------------------')
    str = input('Now please make your choice: ','s');

    if str == 'q'
        break
    elseif str == 's'
        break
    else
        choice(eval(str)) = 1;
        break
    end
end
```

```
% 自动选择最优方法的判别过程
if str == 's'
    choice = ones(1,9);
    choice(1) = 0;
    choice(7) = 0;
    choice(9) = 0;
    choice(10) = 0;
    while 1

         % 判断是否为上三角或下三角矩阵
        if A == triu(A)
            choice = zeros(1,10);
            choice(3) = 1;
            break
        elseif A == tril(A)
            choice = zeros(1,10);
            choice(4) = 1;
            break
        else
            choice(3:4) = 0;
        end

        % 判断是否为三对角矩阵
        f = diag(A);
        n = length(diag(A));
        new_A = zeros(n,n);
        g = zeros(1,n);
        e = zeros(1,n);
        for i = 1:length(g)-1
            g(i) = A(i,i+1);
            e(i+1) = A(i+1,i);
        end
        for i = 1:n
            new_A(i,i) = f(i);
        end
        for i = 1:n-1
            new_A(i,i+1) = g(i);
            new_A(i+1,i) = e(i+1);
        end
        if A == new_A
            istri_banded = true;
            choice = zeros(1,10);
            choice(5) = 1;
            break
```

```
    else
        istri_banded = false;
        choice(5) = 0;
    end

    % 判断是否为大型矩阵
    for j = 1:size(A,2)
        D(1,j) = det(A(1:j,1:j));
    end
    if A_row*A_col>= 10000  && all(D)
        choice = zeros(1,10);
        choice(6) = 1;
        break
    else
        choice(6) = 0;
    end

    % 判断对角线是否有0
    if any(diag(A))
        choice(7:9) = 0;
    else
        count = 0;
        [rows,cols] = size(A);
        matrix_size = rows*cols;
        for i = 1:rows
            for k = 1:cols
                if A(i,k) == 0
                    count = count + 1;
                end
            end
        end

        % 判断是否为稀疏矩阵
        if count > matrix_size*0.8
            issparse = true;
        else
            issparse = false;
        end
        if issparse
            is_dominant = Diagonally_dominant(A);

            if is_dominant
                choice = zeros(1,10);
                choice(8) = 1;
                break
```

```
                else
                    choice(7:9) = 0;
                end
            else
                choice(7:9) = 0;
            end
        end
        break
    end
end
ans = nan;
if choice(5)
    f = diag(A);
    n = length(diag(A));
    new_A = zeros(n,n);
    g = zeros(1,n);
    e = zeros(1,n);
    for i = 1:length(g)-1
        g(i) = A(i,i+1);
        e(i+1) = A(i+1,i);
    end
    for i = 1:n
        new_A(i,i) = f(i);
    end
    for i = 1:n-1
        new_A(i,i+1) = g(i);
        new_A(i+1,i) = e(i+1);
    end
    if A == new_A
        istri_banded = true;
    else
        istri_banded = false;
    end
end

exact_solution = A\b;

if choice(1)
    disp('--------------Gaussian Elimination without pivoting-------------')
    ans = Gaussian_Elimination(A,b);
    relative_error = max(Err(ans,exact_solution));
    fprintf('The maximum relative error for the Gaussian Elimination without 
pivoting is %.4f  %%.\n',relative_error)
    disp('-----------------------------------------------------------------')
end
```

```
if choice(2)
    disp('-----------------------Partial Pivoting-------------------------')
    ans = Partial_Pivoting(A,b);
    relative_error = max(Err(ans,exact_solution));
    fprintf('The maximum relative error for the Partial Pivoting
is %.4f  %%.\n',relative_error)
    disp('----------------------------------------------------------------')
end
if choice(3)
    disp('---------Back Substitution for Upper Triangular Matrix-----------')
    if A == triu(A)
        ans = Back_Sub(A,b);
        relative_error = max(Err(ans,exact_solution));
        fprintf('The maximum relative error for the Back Substitution for Upper
Triangular Matrix is %.4f  %%.\n',relative_error)
    else
        disp('Not a Upper Triangular Matrix.')
    end
    disp('----------------------------------------------------------------')
end
if choice(4)
    disp('------Forward Substitution for the Lower Triangular Matrix------')
    if A == tril(A)
        ans = Forward_Sub(A,b);
        relative_error = max(Err(ans,exact_solution));
        fprintf('The maximum relative error for the Forward Substitution for
Lower Triangular Matrix is %.4f  %%.\n',relative_error)
    else
        disp('Not a Lower Triangular Matrix.')
    end
    disp('----------------------------------------------------------------')
end

if choice(5)
    disp('--------------------------Banded Matrix-------------------------')
    if istri_banded
        ans = Tridiagonal(b,f,g,e,n);
        relative_error = max(Err(ans,exact_solution));
        fprintf('The maximum relative error for the TridiagonalBanded Matrix
Elimination is %.4f  %%.\n',relative_error)
    else
        disp('It not a tridiagonal banded matrix.')
    end
    disp('----------------------------------------------------------------')
end
```

```
if choice(6)
    disp('------------------------LU Decomposition------------------------')
    ans = LU_Decomposition(A,b);
    relative_error = max(Err(ans,exact_solution));
    fprintf('The maximum relative error for the LU Decomposition method
is %.4f  %%.\n',relative_error)
    disp('----------------------------------------------------------------')
end
if choice(7)
    disp('-----------------------Jacobi Iterative-------------------------')
    ans = Jacobi(A,B,delta);
    relative_error = max(Err(ans,exact_solution));
    fprintf('The maximum relative error for the Jacobi method
is %.4f  %%.\n',relative_error)
    disp('----------------------------------------------------------------')
end
if choice(8) == 1
    disp('---------------------Gauss-Seidal Iterative---------------------')
    ans = Gauss_Seidal(A,b,delta);
    relative_error = max(Err(ans,exact_solution));
    fprintf('The maximum relative error for the Gauss-Seidal
is %.4f  %%.\n',relative_error)
    disp('----------------------------------------------------------------')
end
if choice(9)
    disp('---------------------------SOR Method----------------------------')
    [time,relative_error,best_lambda,ans] = SOR(A,b,delta,omega);
    fprintf('The best running time for the SOR method is %.8f s.\n',time)
    fprintf('The maximum relative error for the SOR method with best lambda
is %.4f  %%.\n',relative_error)
    disp('----------------------------------------------------------------')
end
if choice(10)
    disp('------------------------Exact solution--------------------------')
    disp('The exact solution is')
    disp(exact_solution)
    disp('----------------------------------------------------------------')
end
end
```

5.1.2 *Gaussian_Elimination.m*

```
function res = Gaussian_Elimination(A,B)
tic;
Aug = [A B];
dims_new = size(Aug);
col_new = dims_new(2);

for k = 1:col_new-2
    for i = k+1:col_new-1
        factor = Aug(i,k)/Aug(k,k);
        Aug(i, k:col_new) = Aug(i, k:col_new) - factor*Aug(k, k:col_new);
    end
end

n = dims_new(1);
res = zeros(n,1);
res(n,1) = Aug(n,col_new)/Aug(n,n);
for i = n-1:-1:1
    res(i,1) = (Aug(i,col_new)-Aug(i,i+1:n)*res(i+1:n))/Aug(i,i);
end
fprintf('The running time for the Gaussion Elimination without pivoting is %.8f
s.\n',toc)
end
```

5.1.3 *Partial_Pivoting.m*

```
function res = Partial_Pivoting(A,B)
tic;
Aug = [A B];
dims_new = size(Aug);
col_new = dims_new(2);
n = dims_new(1);
for k = 1:col_new-2
    [maximum,maximum_row] = max(abs(Aug(k:n,k)));
    current_row = maximum_row+k-1;
    if current_row ~= k
        Aug([k,current_row],:) = Aug([current_row,k],:);
    end

    for i = k+1:col_new-1
        factor = Aug(i,k)/Aug(k,k);
```

```
        Aug(i, k:col_new) = Aug(i, k:col_new) - factor*Aug(k, k:col_new);
    end
end

res = zeros(n,1);
res(n,1) = Aug(n,col_new)/Aug(n,n);
for i = n-1:-1:1
    res(i,1) = (Aug(i,col_new)-Aug(i,i+1:n)*res(i+1:n))/Aug(i,i);
end
fprintf('The running time for the partial pivoting is %.8f s.\n',toc)
end
```

5.1.4 Back_*Sub.m*

```
function res = Back_Sub(A,B)
tic
Aug = [A B];
dims_new = size(Aug);
col_new = dims_new(2);
n = dims_new(1);
res = zeros(n,1);
res(n,1) = Aug(n,col_new)/Aug(n,n);
for i = n-1:-1:1
    res(i,1) = (Aug(i,col_new)-Aug(i,i+1:n)*res(i+1:n))/Aug(i,i);
end
fprintf('The running time for the Back Substitution for Upper Triangular Matrix
is %.8f s.\n',toc)
end
```

5.1.5 *Forward_Sub.m*

```
function res = Forward_Sub(A,B)
tic
Aug = [A B];
dims_new = size(Aug);
col_new = dims_new(2);
n = dims_new(1);
res = zeros(n,1);
res(1,1) = Aug(1,col_new)/Aug(1,1);
for i = 2:1:n
    res(i,1) = (Aug(i,col_new)-Aug(i,1:i)*res(1:i))/Aug(i,i);
```

```
end
fprintf('The running time for the Forward Substitution for the Lower Triangular Matrix is %.8f s.\n',toc)
end
```

5.1.6 *Tridiagonal.m*

```
function res = Tridiagonal(r,f,g,e,n)
tic;
n = length(f);
for k = 2:n
    factor = e(k)/f(k-1);
    f(k) = f(k) - factor*g(k-1);
    r(k) = r(k) - factor*r(k-1);
end
res(n) = r(n)/f(n);
for k = n-1:-1:1
    res(k) = (r(k)-g(k)*res(k+1))/f(k);
end
res = res';
fprintf('The running time for the Tridiagonal banded matrix Elimination is %.8f s.\n',toc)
end
```

5.1.7 *LU_Decomposition.m*

```
function res = LU_Decomposition(A,b)
tic
[m,n] = size(A);
L=zeros(n,n);
U=eye(n,n);
L(1,1)=A(1,1);
for k=2:n
    L(k,1)=A(k,1);
    U(1,k)=A(1,k)/L(1,1);
end
for k=2:n-1
```

```
    L(k,k)=A(k,k)-L(k,1:k-1)*U(1:k-1,k);
    for m=k+1:n
        L(m,k)=A(m,k)-L(m,1:k-1)*U(1:k-1,k);
        U(k,m)=(A(k,m)-L(k,1:k-1)*U(1:k-1,m))/L(k,k);
     end
end
L(n,n)=A(n,n)-L(n,1:n-1)*U(1:n-1,n);
d(1,1) = b(1)/L(1,1);
for k=2:n
    d(k,1) = (b(k) - L(k,1:k-1)*d(1:k-1,1))/L(k,k);
end
x(n,1) = d(n);
for k=n-1:-1:1
    x(k,1) = d(k) - U(k,k+1:n)*x(k+1:n,1);
end
res = x;
fprintf('The running time for the LU Decomposition is %.8f s.\n',toc)
end
```

5.1.8 Jacobi.m

```
function res = Jacobi(A,b,delta)
tic;
[m,n] = size(A);
mc = 0;
x0 = ones(n,1);
fprintf('Each iteration result is as follows:\n')
while 1
    xold = x0;
    for k=1:n
        x(k,1)=x0(k,1)+(b(k,1)-A(k,1:n)*xold(1:n,1))/A(k,k);
        err(k)=abs(x(k,1)/x0(k,1)-1);
        x0(k,1)=x(k,1);
    end
    errmax = max(err);
    mc=mc+1;
    fprintf('%d, %10.2e\n', mc, errmax);
    if (errmax <= delta || mc >= 1000)
        break
    end
end
res = x0;
fprintf('Jacobi Iteration for convergence: %d.\n',mc)
```

```
fprintf('The running time for the Jacobi Iteration is %.8f s.\n',toc)
end
```

5.1.9 *Gauss_Seidal.m*

```
function res = Gauss_Seidal(A,b,es)
tic;
[m,n] = size(A);
mc = 0;
x0 = ones(n,1);
fprintf('Each iteration result is as follows:\n')
while 1
    for k=1:n
        x(k,1)=x0(k,1)+(b(k,1)-A(k,1:n)*x0(1:n,1))/A(k,k);
        err(k)=abs(x(k,1)/x0(k,1)-1);
        x0(k,1)=x(k,1);
    end
    errmax = max(err);
    mc=mc+1;
    fprintf('%d, %10.2e\n', mc, errmax);
    if (errmax <= es || mc >= 1000)
        break
    end
end
res = x0;
fprintf('Gauss Seidal Iteration for convergence: %d.\n',mc)
fprintf('The running time for the Gauss_Seidal is %.8f s.\n',toc)
end
```

5.1.10 *SOR.m*

```
function [time,relative_error,best_omega,res] = SOR(A,b,es,lambda)
exact_solution = A\b;
tic;
[m,n] = size(A);
time_all = zeros(1,length(omega));
for h = 1:length(omega)
    tic;
    mc = 0;
    x0 = ones(n,1);
```

```
    while 1
        for k=1:n
            x_sor(k,1)=x0(k,1)+(b(k,1)-A(k,1:n)*x0(1:n,1))/A(k,k);
            x_sor(k,1) = omega(h)*x_sor(k,1)+(1-omega(h))*x0(k,1);
            err(k)=abs(x_sor(k,1)/x0(k,1)-1);
            x0(k,1)=x_sor(k,1);
        end
        errmax = max(err);
        mc=mc+1;
        if (errmax <= es || mc >= 1000)
            break
        end
    end
    mc_all(1,h) = mc;
    time_all(1,h) = toc;
    fprintf('Lambda is: %.1f.\n',omega(h))
    fprintf('The solutions obtained by SOR method are:\n')
    fprintf('SOR for convergence: %d.\n',mc)
    time_there = toc;
    fprintf('The running time for the SOR is %.8f s.\n',time_there)
    cal_error = max(Err(x0,exact_solution));
    fprintf('The maximum relative error for the SOR method
is %.4f  %%.\n',cal_error)
    if mc == min(mc_all)
        res = x0;
        best_omega = omega(h);
        relative_error = cal_error;
        time = time_there;
    end
end
disp('------------------Final Result-----------------')
fprintf('The best lambda is %.1f.\n',best_omega)
end
```

5.2 Reference

Applied Numerical Methods with MATLAB® for Engineers and Scientists Third Edition, Steven C. Chapra Berger Chair in Computing and Engineering Tufts University.