Huazhong University of Science and Technology AUT5951

Homework 1 - Foundations of Data Science

闫吕志
人工智能启明实验班2401

Problem - Jacobi and Gauss Iteration

the Idea of Solution

In the first problem,we can simply use 'A \ v' to solve the equation,and that is the comparison that can be used to check the answer afterwards.

Next,we use Jacobi and Gauss Iteration to solve the equation,and we can just use the coding in the class.

But we should pay attention to many details in our coding,such as the reset of the loop count and so on.

Then,we use the 'mean' function to calculate the average number of the iterations of the two paths.

the Explanation of My Coding

To solve the problem,I use 2 files of matlab to complete.

The first one is to solve the system of linear equations using the direct method,and the second one is to solve the system of linear equations using the Jacobi iterative method and the Gauss-Seidel iterative method, and save the average number of iterations in a row vector.

In the first coding,the outcome is shown as the form of row vector($[I1, I2, I3]$),and there is totally 51 vectors with the development of $v1$,so results are in a matrix of 3 columns and 51 rows,which is named as 'X'.

In the second coding,the two different results are 'X1' and 'X2',and the form is identical to the first coding.Then,I use the vector 'avg' to demostrate the average number of the two functions.The former one is Jacobi iteration,and the latter one is Gauss iteration.

The first coding is below:

```
% Problem 1: Solve the system of linear equations using the direct method

% A is the coefficient matrix, and V1, V2, V3 form the vector v
A = [20 + 15 + 40,-20,-15;
-20,20+25+20,-20;
-15,-20,15+20+30];

v = [0,0,200];  % v is a column vector

X = (A \ v')';  % Transpose X to a row vector for subsequent appending

% Loop to increment the first component of v
for i=1:2:100
    v = v + [2, 0, 0];
    x = A \ v';
    X = [X; x'];
end

% Display the results
disp(X)
```

the second coding is below:

```
% Problem 2 3: Solve the system of linear equations using the Jacobi iterative method and the Ga

% A is the coefficient matrix, and V1, V2, V3 form the vector v
A = [20 + 15 + 40,-20,-15;
-20,20+25+20,-20;
-15,-20,15+20+30];

v = [0,0,200];

X1 = [];
X2 = [];
Iter1 = [];
Iter2 = [];

% Use the Jacobi iterative method to solve the system of linear equations
D = diag(diag(A));
T = A - D;

M = -inv(D) * T;

for i=1:2:100
    v = v + [2, 0, 0];
    b_2 = inv(D) * v';
    x = [0;0;0];
    iter1 = 1;
    tol = 1e-6;
    err = tol * 2;
    while (err > tol) && (iter1 < 100)
        x(:, iter1+1) = M * x(:, iter1) + b_2;
        err = norm(x(:, iter1+1) - x(:, iter1), Inf);
        iter1 = iter1 + 1;
    end
    x_add = x(:, end)';
    X1 = [X1; x_add];
    Iter1 = [Iter1 , iter1];
end

% Use the Gauss-Seidel iterative method to solve the system of linear equations
v = [0,0,200];  % Reset v

S = tril(A);
T=A - S;
```

```
for j = 1:2:100         % Use another loop to avoid the previous loop reaching the upper limit
    v = v + [2, 0, 0];
    x = [0;0;0];
    tol = 1e-6;
    err=2*tol;
    iter2 = 1;

    while (err > tol) && (iter2 < 100)
    x(:,iter2+1)=-S\T*x(:,iter2)+S\v';
    err=norm(x(:,iter2+1)-x(:,iter2),Inf);
    iter2=iter2+1;
    end
    x2_add = x(:, end)';
    X2 = [X2; x2_add];
    Iter2 = [Iter2 , iter2];
end

% Calculate the average number of iterations for both methods
avg_iter1 = mean(Iter1);
avg_iter2 = mean(Iter2);
avg = [avg_iter1, avg_iter2];

% Display the results
X1
X2
disp("Average number of iterations (first component is Jacobi, second component is Gauss-Seidel)
disp(avg)
```

Results

```
>> solution_1
    1.1740    1.5368    3.8207
    1.2066    1.5505    3.8324
    1.2393    1.5641    3.8442
    1.2719    1.5778    3.8559
    1.3046    1.5915    3.8677
    1.3372    1.6051    3.8794
    1.3699    1.6188    3.8911
    1.4026    1.6324    3.9029
    1.4352    1.6461    3.9146
    1.4679    1.6598    3.9264
    1.5005    1.6734    3.9381
    1.5332    1.6871    3.9498
    1.5658    1.7007    3.9616
    1.5985    1.7144    3.9733
    1.6312    1.7281    3.9851
    1.6638    1.7417    3.9968
    1.6965    1.7554    4.0085
    1.7291    1.7691    4.0203
    1.7618    1.7827    4.0320
    1.7945    1.7964    4.0438
    1.8271    1.8100    4.0555
    1.8598    1.8237    4.0672
    1.8924    1.8374    4.0790
    1.9251    1.8510    4.0907
    1.9577    1.8647    4.1025
    1.9904    1.8783    4.1142
    2.0231    1.8920    4.1259
    2.0557    1.9057    4.1377
    2.0884    1.9193    4.1494
    2.1210    1.9330    4.1612
    2.1537    1.9466    4.1729
    2.1863    1.9603    4.1846
    2.2190    1.9740    4.1964
    2.2517    1.9876    4.2081
    2.2843    2.0013    4.2199
    2.3170    2.0149    4.2316
    2.3496    2.0286    4.2433
    2.3823    2.0423    4.2551
    2.4149    2.0559    4.2668
    2.4476    2.0696    4.2785
```

```
    2.4803    2.0832    4.2903
    2.5129    2.0969    4.3020
    2.5456    2.1106    4.3138
    2.5782    2.1242    4.3255
    2.6109    2.1379    4.3372
    2.6435    2.1515    4.3490
    2.6762    2.1652    4.3607
    2.7089    2.1789    4.3725
    2.7415    2.1925    4.3842
    2.7742    2.2062    4.3959
    2.8068    2.2199    4.4077
```text
>> solution_2

X1 =

    1.2066    1.5505    3.8324
    1.2393    1.5641    3.8442
    1.2719    1.5778    3.8559
    1.3046    1.5915    3.8677
    1.3372    1.6051    3.8794
    1.3699    1.6188    3.8911
    1.4026    1.6324    3.9029
    1.4352    1.6461    3.9146
    1.4679    1.6598    3.9264
    1.5005    1.6734    3.9381
    1.5332    1.6871    3.9498
    1.5658    1.7007    3.9616
    1.5985    1.7144    3.9733
    1.6312    1.7281    3.9851
    1.6638    1.7417    3.9968
    1.6965    1.7554    4.0085
    1.7291    1.7690    4.0203
    1.7618    1.7827    4.0320
    1.7944    1.7964    4.0438
    1.8271    1.8100    4.0555
    1.8598    1.8237    4.0672
    1.8924    1.8374    4.0790
    1.9251    1.8510    4.0907
    1.9577    1.8647    4.1025
    1.9904    1.8783    4.1142
    2.0231    1.8920    4.1259
    2.0557    1.9057    4.1377
```
```

```
    2.0884    1.9193    4.1494
    2.1210    1.9330    4.1612
    2.1537    1.9466    4.1729
    2.1863    1.9603    4.1846
    2.2190    1.9740    4.1964
    2.2517    1.9876    4.2081
    2.2843    2.0013    4.2198
    2.3170    2.0149    4.2316
    2.3496    2.0286    4.2433
    2.3823    2.0423    4.2551
    2.4149    2.0559    4.2668
    2.4476    2.0696    4.2785
    2.4803    2.0832    4.2903
    2.5129    2.0969    4.3020
    2.5456    2.1106    4.3138
    2.5782    2.1242    4.3255
    2.6109    2.1379    4.3372
    2.6435    2.1515    4.3490
    2.6762    2.1652    4.3607
    2.7089    2.1789    4.3725
    2.7415    2.1925    4.3842
    2.7742    2.2062    4.3959
    2.8068    2.2198    4.4077

X2 =

    1.2066    1.5505    3.8324
    1.2393    1.5641    3.8442
    1.2719    1.5778    3.8559
    1.3046    1.5915    3.8677
    1.3372    1.6051    3.8794
    1.3699    1.6188    3.8911
    1.4026    1.6324    3.9029
    1.4352    1.6461    3.9146
    1.4679    1.6598    3.9264
    1.5005    1.6734    3.9381
    1.5332    1.6871    3.9498
    1.5658    1.7007    3.9616
    1.5985    1.7144    3.9733
    1.6312    1.7281    3.9851
    1.6638    1.7417    3.9968
    1.6965    1.7554    4.0085
```

```
    1.7291    1.7690    4.0203
    1.7618    1.7827    4.0320
    1.7945    1.7964    4.0438
    1.8271    1.8100    4.0555
    1.8598    1.8237    4.0672
    1.8924    1.8374    4.0790
    1.9251    1.8510    4.0907
    1.9577    1.8647    4.1025
    1.9904    1.8783    4.1142
    2.0231    1.8920    4.1259
    2.0557    1.9057    4.1377
    2.0884    1.9193    4.1494
    2.1210    1.9330    4.1612
    2.1537    1.9466    4.1729
    2.1863    1.9603    4.1846
    2.2190    1.9740    4.1964
    2.2517    1.9876    4.2081
    2.2843    2.0013    4.2199
    2.3170    2.0149    4.2316
    2.3496    2.0286    4.2433
    2.3823    2.0423    4.2551
    2.4149    2.0559    4.2668
    2.4476    2.0696    4.2785
    2.4803    2.0832    4.2903
    2.5129    2.0969    4.3020
    2.5456    2.1106    4.3138
    2.5782    2.1242    4.3255
    2.6109    2.1379    4.3372
    2.6435    2.1515    4.3490
    2.6762    2.1652    4.3607
    2.7089    2.1789    4.3725
    2.7415    2.1925    4.3842
    2.7742    2.2062    4.3959
    2.8068    2.2199    4.4077

 Average number of iterations (first component is Jacobi, second component is Gauss-Seidel):
    25.4200   15.0000
```

Reflection

The average number of Gauss is fewer than the Jacobi one,which shows that the Jacobi function converges faster.