International Core Journal of Engineering Volume 7 Issue 11, 2021
ISSN: 2414-1895 DOI: 10.6919/ICJE.202111_7(11).0052

Comparison and exploration of sparse matrix solution based on iterative methods

Yan Rong

Shanghai University of International Business and Economics, Shanghai, 201620, China

Abstract

The solution of sparse linear equations is the core of many scientific computing tasks and engineering problems. With the increase of the complexity of practical problems, the optimization of solving sparse linear equations becomes more and more important. In this paper, based on the iterative method, the solutions of sparse matrix equations are analyzed, and three iterative methods are studied: Jacobi iterative method, Gauss-Seidel iterative method, and SOR iterative method. The sparse matrix solution performance of triplet storage is investigated, and the influence of relaxation factor ω on the number of SOR iterations is discussed. Finally, we compare and analyze the performance difference between the classical direct method and the iterative method, and give the applicable matrix of each method and the shortcomings.

Keywords

Sparse matrix, Classical iterative methods, pretreatment, Direct solution.

1. Introduction

1.1 Sparse matrix background

A sparse matrix refers to a matrix in which there are many elements with a value of 0 and the distribution of non-0 elements is irregular.

The sparse matrix is generally stored by compression, which has the advantage that each element can be accessed randomly, so it is easy to solve various operations of the matrix. For a sparse matrix, it usually has large dimensions, sometimes so large that zero elements take up most of the memory. Therefore, the storage method suitable for dense matrix will waste a large number of memory cells to store zero elements. A lot of time is also wasted on invalid operations of zero elements.

There are two kinds of common storage methods, namely, sequential storage structure and chain storage structure.

There are two common ways of sequential storage structure: triplet representation and pseudo address representation. Triplet notation, which defines a data structure that holds the x-coordinate, column coordinate, and value of the element. The pseudo-address representation is more space-saving than the triplet representation. It only defines a structure in which a member variable is used to store the value of the element and another member variable is used to store its pseudo-address. There are also two common methods of list storage structure, which are adjacency list and orthogonal list respectively.

1.2 Data background

The matrix selected in this paper comes from the optimal control problem of Vehicle Dynamics and Optimization Laboratory.

Each optimal control problem produces a series of matrices of different sizes. The researchers have created a graph coarsening strategy that matches node pairs and given the mapping of this matrix.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202111_7(11).0052

This matrix consists of a set of nodes, where $map(i) = k$represents the mapping of node i in the original graph to node k in the smaller graph. $Map(i) = Map(j) = k$ indicates that nodes i and j are mapped to the same node k, and finally, nodes i and j are merged.

The goal of the station attitude control problem is to determine the state and control to minimize the size of the final momentum. When the station finally reaches a direction, it can remain stable without the use of additional control torque. The system state is defined by the angular velocity of the spacecraft relative to the inertial frame of reference, Euler-Rodriguez parameters are used to define the attitude of the vehicle, and the angular momentum of the control torque gyroscope and control system is defined as torque. After 13 mesh iterations, the specified precision tolerance is satisfied. As the mesh was refined, the size of the matrix was increased from 99 to 1640.

2. Problem analysis

2.1 Jacobi iteration method

2.1.1 Sub-section Headings

Suppose the coefficient matrix A of the linear equations $Ax = b$ is invertible, and the principal diagonal elements $a_{11}, a_{22}, \dots, a_{mm}$ is not 0. The $D = diag(a_{11}, a_{22}, \dots, a_{mm})$, and decompose A into $A = (A - D) + D$, so

$$Dx = (D - A)x + b$$

Let $x = B_1 x + f_1$,where

$$B_1 = I - D^{-1}A \ , \quad f_1 = D^{-1}b$$

Iteration method with B_1 as iteration matrix

$$x^{(k+1)} = B_1 x^{(k)} + f_1$$

It's called Jacobi iteration, and it's expressed in terms of the components of a vector

$$x_i^{(k+1)} = \frac{1}{a_{ii}}[b_i - \sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j^{(k)}] \ \ (i = 1,2 \dots, n; k = 0,1,2, \dots)$$

Among them, the $x^{(0)} = (x_1^{(0)}, x_2^{(0)}, \dots, x_n^{(0)})^T$ as the initial vector.

It can be seen that the Jacobi iteration method is simple, and only one iteration and vector multiplication need to be calculated for each iteration. When the computer computes the formula, it needs two sets of memory cells to hold $x^{(k)}$ and $x^{(k+1)}$.

2.2 Gauss-Seidel iteration method

On the basis of Jacobi iterative method, Gauss-Seidel iterative method uses the$i - 1$ components to calculate the i-th approximate solution in the $k + 1$ step. The vector expression is:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}[b_i - \sum_{j=1}^{i-1} a_{ij} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij} x_j^{(k)}](i = 1,2 \dots, n; k = 0,1,2, \dots)$$

It can be seen that Gauss-Seidel method needs to calculate the components of the solutions in order from 1 to n. One obvious advantage, however, is that when the computation is run, only one set of storage units is required, and no additional variables are required to hold the iterative solution of the previous step. In terms of the amount of calculation, it is almost the same as jacobian iterative method, and each step of calculation is equivalent to a matrix and vector multiplication.

2.3 Gauss-Seidel iteration method

Working on the basis of Gauss-Seider method, SOR calculates the weighted average of$\tilde{x}_1^{(k+1)}$ and the preceding iteration solution $x_1^{(k)}$ to produce the next iteration solution. The vector representation is of the form

ISSN: 2414-1895 DOI: 10.6919/ICJE.202111_7(11).0052

$$x_i^{(k+1)} = (1-\omega)x_i + \frac{\omega}{a_{ii}}[b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}]$$

Where, ω is the relaxation factor. When $\omega < 1$, it corresponds to the low-slack iteration method; when $\omega > 1$, it corresponds to the over-slack iteration method; when $\omega = 1$, SOR iteration method is equivalent to Gauss-Seidel iteration method.

SOR method only adds the calculation of relaxation factor ω, and its premise is the same as the previous two methods, that the diagonal element of matrix A cannot be 0, and SOR method also needs to calculate the components of the solution in the order from 1 to n.

3. Problem-solving

3.1 Section Headings

Because the iterative method requires that the diagonal element of matrix *A* cannot be 0, but in reality, many matrices have the diagonal element of 0, so the matrix needs to be preprocessed.

First, the method of selecting all pivot elements is adopted. Think of a matrix as an iterated partitioned matrix. When the value on the main diagonal is 0, look for the largest element a_{mn} in the partitioned matrix. Take the transformation of the column and row of A, and put a_{mn} in the position of the diagonal element. At the same time, the corresponding row and column transformation of the right vector b and the solution vector x is carried out to achieve the preliminary processing of matrix A.

After selecting the principal element method, the matrix A is filled in for the case that the diagonal element still has 0. The specific method is as follows: if the diagonal element a_{ii} of the matrix is 0, the largest element in this column is found on the ith column, denoted as a_{ki}. Add the coefficient corresponding in row k to row i, and change the right-hand value ofb_i to $b_i + b_k$. A matrix with non-zero diagonal elements can be obtained by processing all rows with pivot elements of 0.

A 99×99 sparse matrix is taken as an example to preprocess the matrix. The matrix before and after processing is visualized as shown in the figure below.

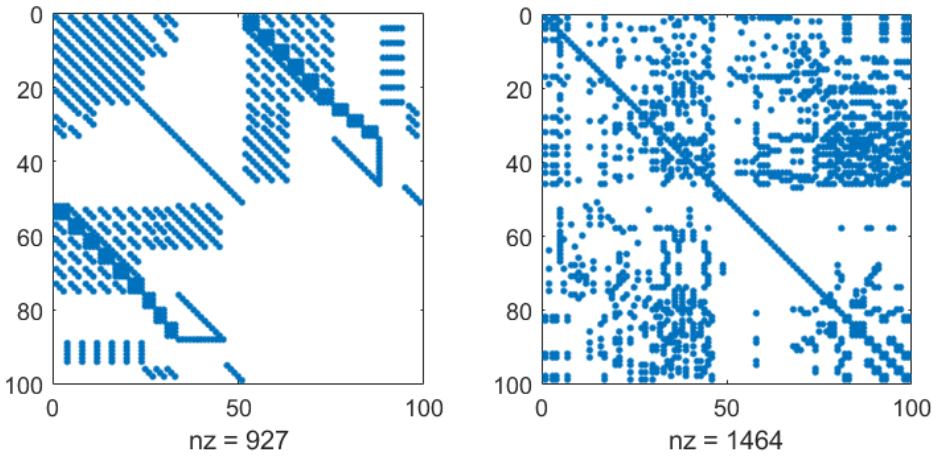

Figure 1. Comparison of matrix non-zero element distribution before and after pretreatment

As can be seen from Figure 1, the sparsity of the pre-processed matrix itself has been destroyed due to the filling of elements. Further analysis, two limitations of the pretreatment method are proposed as follows.

On the one hand, the matrix filling operation makes a large number of 0 elements except the main diagonal be filled with numbers, which destroys the sparsity of the matrix. On the other hand, the addition of the row elements of the matrix makes some large values exist in the matrix, which further leads to the operation of the iterative method, the coefficient of the iterative matrix may be large.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202111_7(11).0052

Therefore, the real solution $x^*$is between x_k andx_{k+1} in the process of iteration from x_k to x_{k+1}. In this case, although the spectral radius $\rho(B)$ of the sparse matrix is less than 0, the iterative method still fails to converge.

The visualization of the coefficient matrix corresponding to $steam3$ data after pre-processing is shown in the figure below.

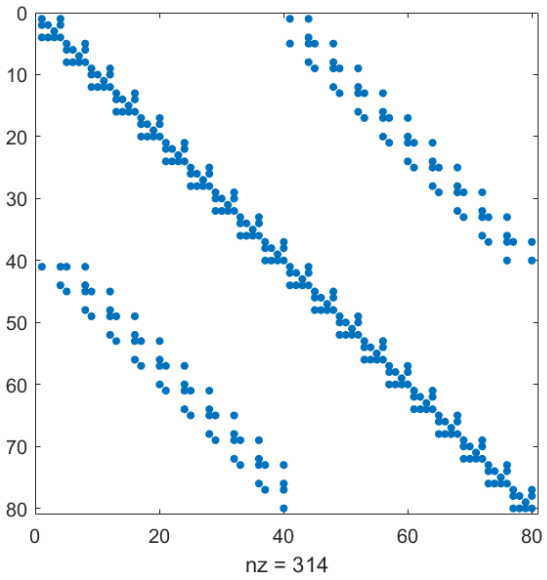

Figure 2. Matrix distribution diagram of steam3 after preprocessing

Figure 2 shows that the sparsity of $steam3$ data still remains good after pretreatment. In the following iterative method, $steam3$data will be taken as an example for further analysis.

3.2 Solving process

In SOR iteration method, the selection of relaxation factor ω has great influence on the convergence property of matrix. Firstly, the value of ω is changed to explore the influence of the selection of relaxation factor ω on the number of iterations.

For ω, values are taken at 0.1 intervals from 0.3 to 1.1. The influence of relaxation factor selection on the number of iterations is shown in the figure below

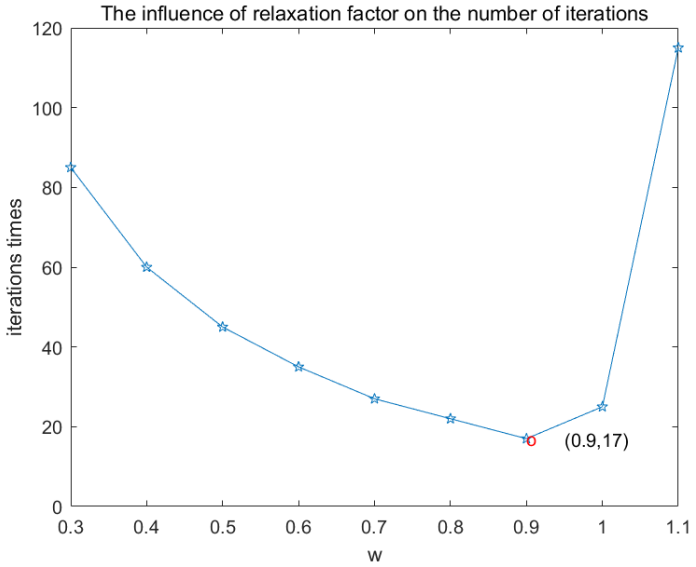

Figure 3. The influence of relaxation factor on the number of iterations

ISSN: 2414-1895 DOI: 10.6919/ICJE.202111_7(11).0052

It can be seen from Figure3 that the number of iterations decreases first and then increases with the increase of relaxation factor ω. When ω approaches 1, the number of iterations is less and the convergence rate is faster. When the relaxation factor ω is 0.9, the number of iterations reaches a minimum of 17 times. The relationship between the absolute value of the eigenvalue and the relaxation factor is further explored.

In general, the necessary condition of SOR iteration method convergence is $0 < \omega < 2$. In order to consider the spectral radius near the boundary, we extend the value of ω from (0,2) to (-1,3) in the experiment. Take the value of ω every 0.001, and the relationship between spectral radius and relaxation factor can be obtained as shown in the figure below.

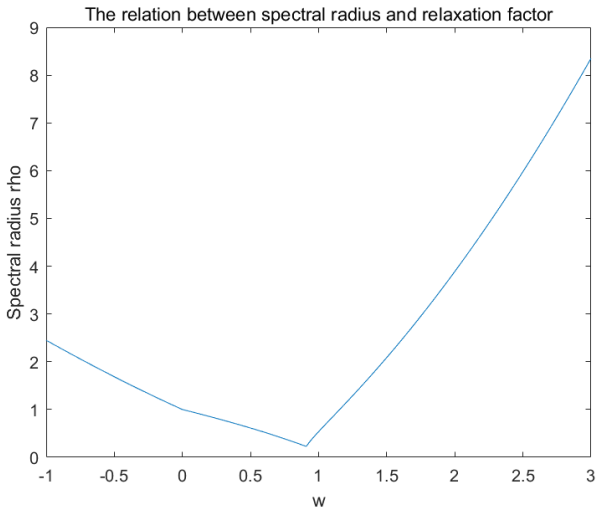

Figure 4. the relation between spectral radius and relaxation factor

It can be seen from Figure4 that when ω is on the interval of (0,1.155), the spectral radius $\rho(B) < 1$ and the iteration converges. When the spectral radius is minimized, the number of iterations reaches the minimum value. That is, when the spectral radius reaches the minimum, the corresponding relaxation factor is the optimal relaxation factor of the matrix under SOR iteration method.

$Steam3$data were substituted into Jacobi iteration method, Gauss-Seidel iteration method and SOR iteration method respectively. The running time, iteration times and residuals of the three running methods are compared as shown in the following table.

Table 1. Comparison of iteration times and running time of the three iterative methods

Methods	Iteration times	Running time
Jacobi iteration method	61	0.0017339
Gauss-Seidel iteration method	25	0.0018858
SOR($\omega = 0.9$)	17	0.0014259
SOR($\omega = 1.1$)	115	0.0046871

ISSN: 2414-1895 DOI: 10.6919/ICJE.202111_7(11).0052

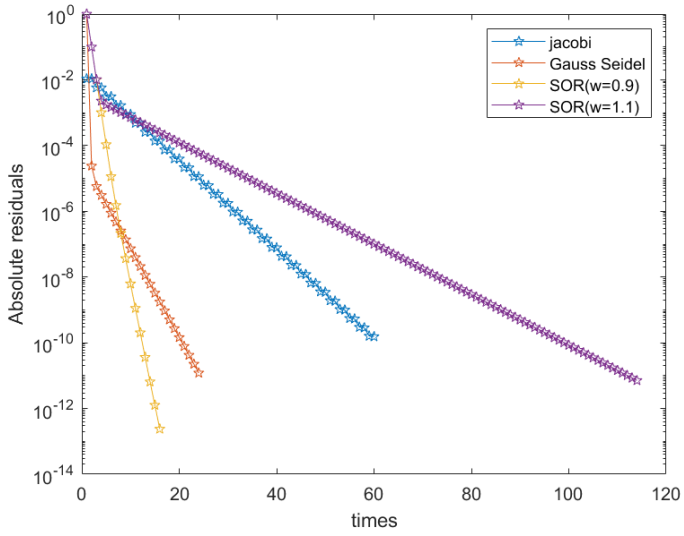

Figure 5. Comparison of iterative residuals and convergence rates

By comparing the running time, the number of iterations and the rate of residual convergence, we can find that the selection of ω has a great influence on the convergence property of the matrix. If ω is the best relaxation factor, SOR iteration method is better than Gauss-Seidel iteration method and Jacobi iteration method. However, if the selection of ω is not appropriate, it will have the opposite effect on the convergence property of equation solution.

4. Conclusion

4.1 Comprehensive comparison

Now we compare the solving effects of all direct methods and iterative methods. The measurement standards we adopted were respectively the solving time of the calling function of MATLAB and the relative residuals of the final solution. The data used in the test is steam3. mat. The performance data of the above three direct methods and three iterative methods are listed as follows:

Table 2. Comprehensive comparative analysis of all solutions

Methods		Running time（s）	Relative residual
Direct method	No selected	1.5725922	8.49890463288078e-13
	columns selected	2.4834406	3.03150101878268e-14
	all selected	0.1420398	4.39585483637832e-14
Classical iterative method	Jacobi	0.0072	1.53840832259825e-10
	Gauss-Seidel	0.0138	1.20520618002085e-11
	SOR	0.0111	2.38221171263545e-13

The residuals of the solution results of all the algorithms tested above are under the control of magnitude 10^{-10}. It can be seen from the above table that the direct LU decomposition method of column selected and all selected is the method with the smallest relative residual, that is, the most accurate solution result. But the direct method running time is relatively long, the shortest running time method is Jacobi iterative method.

ISSN: 2414-1895 DOI: 10.6919/ICJE.202111_7(11).0052

4.2 Application matrix

Since the direct method is, in essence, based on elementary transformations of matrices, it is theoretically equivalent to matrix decomposition. Therefore, all kinds of direct methods in computer implementation will inevitably involve matrix-by-matrix operation, which directly destroys the sparse structure of the matrix, resulting in problems such as memory overflow and long calculation time. As a result, it is not suitable for large-scale sparse matrix.

The iterative method for solving linear equations is specially designed for large sparse matrices. Its characteristic is that each iteration only involves the operation of matrix multiplies vector, which does not destroy the sparse structure of matrix, and is suitable for large-scale sparse matrix.

However, for the classical iterative method, the convergence property depends on some properties of the coefficient matrix, and the principal element of the matrix is required to be non-zero. Therefore, in the process of solving, the matrix needs to be preprocessed, which further leads to the loss of sparsity, and also causes the problem of over-long iteration stride to a certain extent. This will result in non-convergence even though the spectral radius of the matrix is less than 1, which is also a major limitation of the classical iterative method.

References

[1] Zhang Yongjie, SUN Qin. Journal of Changchun University of Science and Technology (Natural Science Edition), 2006(3 issues):38-41.

[2] Huang Dongquan. Journal of Xi 'an Jiaotong University, 1982(06):68-77. (in Chinese)

[3] Yu Wenjian. (2019). Numerical Analysis and Algorithm. 3rd Ed. Tsinghua University Press.

[4] Hu Zhangjie. Application and research of parallel algorithm in solving large-scale linear equations [D]. Chongqing University, 2010.

[5] Zheng Hua, & Luo Liang. Teaching strategy of sparse matrix theory in the course of numerical Analysis.

[6] Arnoldi W E. The principle of minimized iteration in the solution of the matrix eigenproblem[C]// Quart. Appl. Math. 1951:17-29

[7] Saad Y. Iterative methods for sparse linear systems[J]. IEEE Computational Science& Engineering, 1996, 3(4):88-88