fire

MDPI

Article

Optimization of Numerical Simulation Algorithm for Spontaneous Combustion in Goaf via a Compression Storage and Solution Method of Coefficient Matrix

Yongbo Cai [1], Yanlu Zhang [1], Qingjie Qi [1,*], Yueping Qin [2], Tianbai Zhou [3] and Zuo Sun [1]

[1] Emergency Science Research Academy, China Coal Research Institute, Beijing 100013, China; caiyongbo@mail.ccri.ccteg.cn (Y.C.); zhangyanlu@mail.ccri.ccteg.cn (Y.Z.); sunzuo@mail.ccri.ccteg.cn (Z.S.)
[2] School of Emergency Management and Safety Engineering, China University of Mining and Technology (Beijing), Beijing 100083, China; yanpp@ccteg-bigdata.com
[3] Research Institute of Mine Big Data, China Coal Research Institute, Beijing 100013, China; zhoutb@ccteg-bigdata.com
* Correspondence: qiqingjie@mail.ccri.ccteg.cn; Tel.: +86-13898568886

Abstract: In coal mine engineering, numerical software is used to analyze the behavior of coal rock damage and fluid migration. The order of the coefficient matrix used in numerical calculations is increasing, and this increases the computation steps in obtaining the coefficient matrix solution. The storage and solution of the coefficient matrix are key factors influencing the efficiency of the numerical software. Therefore, to save storage space and reduce the computation steps, the coefficient matrix must be effectively compressed and stored. In this work, the structural characteristics of different coefficient matrices are analyzed in detail, and we find that for different computational regions, as long as the nodes are numbered according to certain rules, the corresponding coefficient matrices will have similar structural characteristics. The nonzero elements are symmetrically distributed in the diagonal band, and all the elements on both sides outside the band are zero. Based on this, the coefficient matrix is compressed by a pivoting scheme, and the compressed matrix is directly eliminated by dislocation Gaussian elimination. Thus, a compressed storage method that integrates the compression and solution of the coefficient matrix is established. The compressed storage and calculation module is incorporated into our self-developed simulation software COMBUSS-3D to simulate the evolution of the temperature field in the goaf of Luling Coal Mine. Compared with the conventional method, the compressed storage module can significantly improve the computing rate of the simulation, by approximately 80%.

Keywords: coefficient matrix; band matrix; sparse matrix; compressed storage; Gaussian elimination

Citation: Cai, Y.; Zhang, Y.; Qi, Q.; Qin, Y.; Zhou, T.; Sun, Z. Optimization of Numerical Simulation Algorithm for Spontaneous Combustion in Goaf via a Compression Storage and Solution Method of Coefficient Matrix. *Fire* **2022**, *5*, 71. https://doi.org/10.3390/fire5030071

Academic Editor: Minbo Zhang

Received: 6 May 2022
Accepted: 23 May 2022
Published: 29 May 2022

1. Introduction

Common numerical calculation methods include the finite difference method, finite element method, and finite volume method. A numerical calculation process eventually evolves into a solution of large linear equations [1–5]. The matrix used to store such equations is called the coefficient matrix. To some extent, the solution speed of the coefficient matrix determines the computational efficiency of the numerical software [6–10]. A coefficient matrix contains a large number of zero elements, occupying considerable space in the matrix storage and calculation when solving by Gaussian elimination, resulting in an unnecessarily high computational load [11,12]. Therefore, compressed storage is applied to improve the computational efficiency of numerical software [13,14].

Many studies have focused on the compressed storage of sparse matrices, and some feasible methods have been proposed to solve the compression process of various types of matrices. Stabrowski proposed two methods for compressing asymmetric sparse matrices, which were comprehensively compared with various matrices derived from engineering applications [15]. Lin et al. concluded that the compressed row/column storage

schemes are not perfect and proposed two types of multidimensional sparse-matrix compression schemes to obtain better results [16]. Katherine et al. developed two optimization techniques to improve the storage efficiency of sparse-matrix vector multiplication and evaluated their optimization results [17]. Other scholars studied targeted compression schemes to meet the storage requirements of the various forms of sparse matrices [18–20]. The existing compression methods have been proposed to compress sparse matrices with a general form. Although these compression schemes have a wide application scope, they do not focus on the structural particularity of the coefficient matrix. Moreover, the existing compression schemes are used for storage, which do not directly involve the solution of the matrices and are not effectively combined with Gaussian elimination.

Gaussian elimination is a classical algorithm for solving large linear equations and has been widely used in numerical computation. The core of the algorithm is to convert the matrix elements into a triangular form by basic transformation. Optimization schemes for Gaussian elimination have been proposed to improve the computational efficiency of numerical software. Peña developed a pivoting strategy to modify the Gaussian elimination process and applied it to some important matrix classes, and the results obtained using this pivoting strategy were compared with those obtained by general partial rotation [21]. Alanelli and Hadjidimos proposed a block transformation strategy to optimize the Gaussian elimination process and combined it with the conventional iterative method for the solution of linear systems with a high convergence rate [6]. Xiao et al. integrated the elimination processes of the coefficient and column matrices by analyzing the triangle process, and the calculation results obtained using the improved Gaussian method implemented by OpenCL were analyzed in detail [22]. The column pivot elimination is another improved strategy of Gaussian elimination. The modification made to this strategy is the selection of the column pivot element as the principal element in turn among the principal diagonal elements of the coefficient matrix and the elements below it. Subsequently, by moving the principal element to the principal diagonal, the elements below the principal diagonal can be eliminated, and the matrix is finally transformed into a triangular system [23]. Through the above optimization strategies for Gaussian elimination, it can be found that reducing the calculation of the irrelevant elements in the elimination process can help effectively improve the computing rate. In fact, based on the structural characteristics of the coefficient matrix, the compressed storage and solution of the matrix can be integrated to improve the computational efficiency.

First, the structural characteristics of the coefficient matrices were analyzed in detail by combining three coal mine engineering examples. All the nonzero elements of the coefficient matrix were confined within a band with diagonal symmetry, and the bandwidth was related to the meshing method used and the number of adjacent nodes. Based on the structural characteristics of the coefficient matrix, the zero elements outside the band were removed, and the remaining elements were pivoted diagonally into a new matrix, thus realizing the compressed storage of the coefficient matrix. Subsequently, the coefficient matrix could be directly solved by dislocation Gaussian elimination of the compressed matrix. The advantage of the compressed storage method was preliminarily verified by comparing the calculation rates of coefficient matrices of different orders using the conventional Gaussian method. Finally, a compressed storage module was incorporated into our self-developed simulation software COMBUSS-3D to simulate the distributions of the temperature and oxygen concentration in the goaf area of Luling Coal Mine, Anhui Province, China. The incorporation of the compression module into the COMBUSS-3D software significantly improved its computational efficiency, and the computational time was reduced by approximately 80% on average. The proposed compression strategy integrates the compressed storage and solution of the coefficient matrix, thus improving the computational efficiency of the simulation software and enabling its application to engineering calculations.

2. Structural Analysis of Coefficient Matrix

For mesh generation, any grid node P is taken as an example. In the synthesis of linear equations, all grid nodes adjacent to node P contribute to the node equation of node P, whereas the grid nodes that are not adjacent to it do not. In the finite difference method, the grid nodes adjacent to node P are the nodes above, below, left, and right of it, while in the finite element method, the grid nodes adjacent to node P are the other nodes in the same grid element with node P. Assuming that there are n nodes in the mesh generation and the number of nodes adjacent to node P is m, the grid linear equation of node P will contain n coefficients. The number of nonzero coefficients is m + 1, and the other coefficients are zero. Since only the coefficients corresponding to the adjacent nodes in the node linear equation are nonzero, the other nodes have no contribution to the equation. Therefore, the coefficient matrix containing a large number of zero elements is a sparse matrix. To thoroughly explore the general structural characteristics of coefficient matrices, the following analysis was performed in combination with three engineering examples in the numerical simulation of coal mines.

The first engineering example is a model of the spontaneous combustion of coal in the goaf area of the coal mine, which involves solving a 2D rectangular computational region using the finite difference method to simulate the temperature distribution in the goaf area [24]. In fact, this example is representative of not only the simulation of heat conduction, but also other mechanical or other physical field problems. As shown in Figure 1, the rectangular computational region is meshed, and the grid nodes are numbered in turn. The computational region is divided into 20 nodes, i.e., the linear system contains 20 linear equations. The internal nodes are adjacent to the four surrounding nodes at most, and the number of nodes adjacent to the boundary nodes is less than four; therefore, each node equation contains at most five nonzero coefficients. By writing the nodal equations into a matrix, the coefficient matrix **M** of the computational region can be obtained, as expressed in Equation (1). From Equation (1), it can be found that the nonzero elements in the coefficient matrix **M** are distributed in a symmetrical band along the matrix diagonal, with a bandwidth of 9.

$$\mathbf{M} = \begin{bmatrix} k_{0,0} & k_{0,1} & & & k_{0,4} & & & \\ k_{1,0} & k_{1,1} & k_{1,2} & & & k_{1,5} & & \\ & k_{2,1} & k_{2,2} & k_{2,3} & & & \ddots & \\ & & k_{3,2} & k_{3,3} & & & & \ddots \\ k_{4,0} & & & & k_{4,4} & k_{4,5} & & \\ & k_{5,1} & & & k_{5,4} & k_{5,5} & \ddots & \\ & & \ddots & & & \ddots & \ddots & k_{18,19} \\ & & & \ddots & & & k_{19,18} & k_{19,19} \end{bmatrix}_{20\times 20} \tag{1}$$

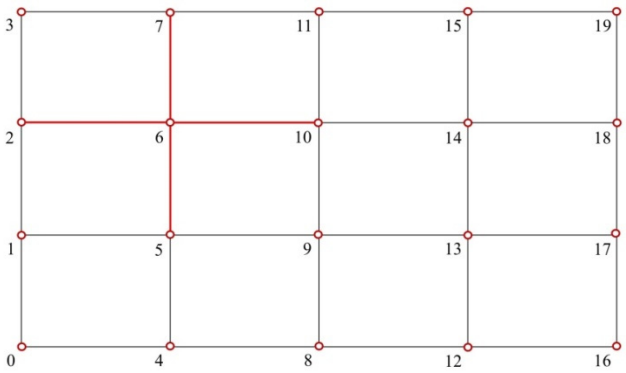

Figure 1. Mesh generation of the model of spontaneous combustion of coal in goaf area.

Here, $k_{x,y}$ represents the nonzero elements of the coefficient matrix, x and y represent the row and column positions, respectively. The zero elements in the coefficient matrix are omitted.

The second example is the calculation of the heat emitting from a surrounding rock at the drifting face using the finite element method [25]. In the numerical simulation, the drifting face can be considered axisymmetric; therefore, half of its section is taken as the computational region. In this model, triangular elements are used as basic elements to discretize the continuum in the computational region and divide the entire area into a series of elements. The division of the elements should satisfy two requirements. (1) It should be based on the temperature distribution in the computational region, elements in the area with large temperature change should be encrypted, and elements in the area with small temperature change should be reduced. (2) This model focuses on the temperature distribution in the surrounding rock at the drifting face. Encryption elements are required in the computational region close to the wall of the surrounding rock to obtain the node temperature with a high accuracy. Therefore, by increasing the element size in equal proportion, the element density near the wall of the surrounding rock can be increased, and the element becomes smaller. The farther away from the wall of the surrounding rock, the lower the element density and the larger the element. The nodes are numbered sequentially from right to left and from near to far. There are 99 nodes in the computational region, and 99 equations are correspondingly obtained. Figure 2 shows the element division and node number; only some representative nodes are identified in the figure. As shown in the red region, each node equation has at most seven nonzero coefficients. The coefficient matrix **L** is expressed in Equation (2), and the matrix order is 99 × 99. The nonzero elements are distributed in the band with a bandwidth of 23.

$$\mathbf{L} = \begin{bmatrix} k_{0,0} & k_{0,1} & & & & k_{0,11} & & & & & & & \\ k_{1,0} & k_{1,1} & k_{1,2} & & & k_{1,11} & k_{1,12} & & & & & & \\ & k_{2,1} & k_{2,2} & k_{2,3} & & & k_{2,12} & k_{2,13} & & & & & \\ & & \ddots & \ddots & \ddots & & & \ddots & \ddots & & & & \\ & & & \ddots & \ddots & \ddots & & & \ddots & \ddots & & & \\ k_{11,0} & k_{11,1} & & & & k_{11,11} & k_{11,12} & & & k_{11,22} & & & \\ & k_{12,1} & k_{12,2} & & & k_{12,11} & k_{12,12} & k_{12,13} & & k_{12,22} & k_{12,23} & & \\ & & \ddots & \ddots & & & \ddots & \ddots & \ddots & & \ddots & \ddots & \end{bmatrix}_{99\times 99} \quad (2)$$

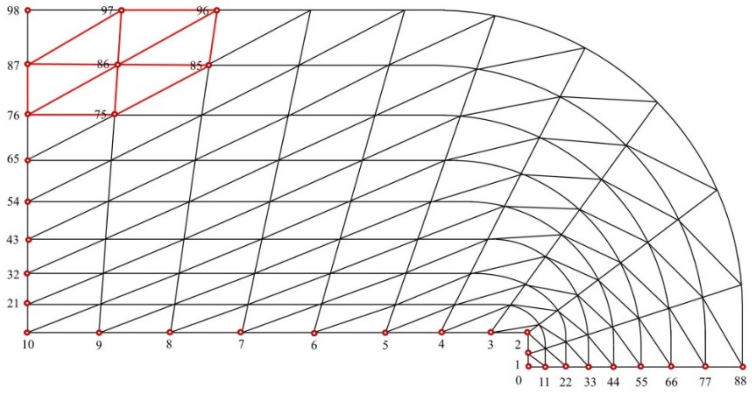

Figure 2. Mesh generation of the model of heat emitting from a surrounding rock at the drifting face.

The third example is the analysis of the heat dissipation in the surrounding rock of trapezoid roadways in a coal mine using the finite volume method [26,27]. To improve the adaptability of the element division to the wall and corner area of the surrounding rock, triangular elements are used as basic elements to divide the computational region. The computational region is divided into 16 elements along the circumference and 8 elements

along the radial direction. Figure 3 shows the mesh generation; only representative nodes are identified in the figure. Based on the characteristics of the temperature distribution in the surrounding rock, the element size is determined using the equal proportion increment method, and the element density is increased appropriately near the wall and corner area in the computational region; thus, the node temperature can be obtained with a relatively high accuracy. The nodes are numbered clockwise from outside to inside, totaling 144 nodes. There are at most seven nonzero coefficients in each node equation. The corresponding coefficient matrix is denoted by **N**. As expressed in Equation (3), the order of the matrix is 144 × 144, and the nonzero elements are distributed in the band with a bandwidth of 35.

$$\mathbf{N} = \begin{bmatrix} k_{0,0} & k_{0,1} & & & & & & k_{0,15} & k_{0,16} & k_{0,17} & & & \\ k_{1,0} & k_{1,1} & k_{1,2} & & & & & & & k_{1,17} & k_{1,18} & & \\ & k_{2,1} & k_{2,2} & k_{2,3} & & & & & & & k_{2,18} & k_{2,19} & \\ & & \ddots & \ddots & \ddots & & & & & & \ddots & \ddots & \ddots \\ & & & \ddots & \ddots & \ddots & & & & & & & \\ & & & & \ddots & \ddots & \ddots & & & & & & \\ k_{15,0} & & & & & k_{15,14} & k_{15,15} & k_{15,16} & & & & & \\ k_{16,0} & & & & & & k_{16,15} & k_{16,16} & k_{16,17} & & & & \\ k_{17,0} & k_{17,1} & & & & & & k_{17,16} & k_{17,17} & k_{17,18} & & & \\ & \ddots & \ddots & \ddots & & & & & \ddots & \ddots & \ddots & & \end{bmatrix}_{144\times144} \tag{3}$$

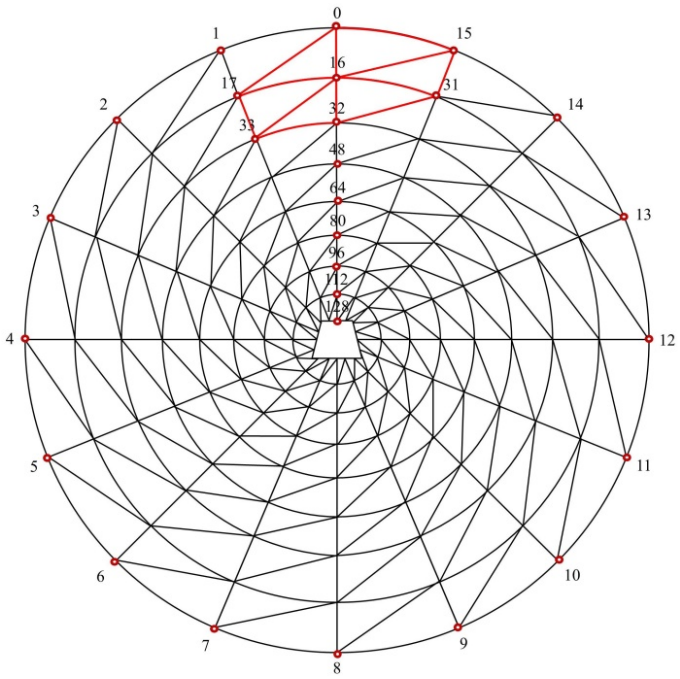

Figure 3. Mesh generation of the model of surrounding rock heat dissipation for trapezoid roadways in a coal mine.

A further analysis, as presented in Figure 1, shows two adjacent nodes at the 0th, 3rd, 16th, and 19th nodes, and the linear equations of these nodes have three nonzero coefficients. Similarly, there are four adjacent nodes in the 5th, 6th, 9th, 10th, 13th, and 14th nodes, and the linear equations of these nodes contain five nonzero coefficients. The linear equations of the remaining nodes contain four nonzero coefficients. Thus, as shown in Figure 1, the coefficient matrix **M** has at least three nonzero elements and at most five nonzero elements per row. For example, the 0th and 4th nodes are adjacent to each other; therefore, $k_{0,4}$ and $k_{4,0}$ are nonzero elements and symmetrical about the diagonal. In other

words, all the diagonal elements of the coefficient matrix **M** are nonzero elements, and the remaining nonzero elements appear in pairs with diagonal symmetry. Similarly, the coefficient matrices **L** and **N** also conform to the above rules. Figure 4 shows the general form of the coefficient matrix. The red and blue circles in the figure represent the diagonal elements and the remaining nonzero elements, respectively. According to the examples of the three numerical calculation methods mentioned above, regardless of the shape of the basic element and regardless of whether the element size changes or not, as long as the nodes are numbered in sequence according to certain rules, the coefficient matrix will exhibit the following common characteristics: (1) It contains a large number of zero elements; (2) its nonzero elements are regularly distributed in the band; (3) its diagonal elements are all nonzero elements; (4) the nonzero elements outside the diagonal have diagonal symmetry; (5) its bandwidth depends on the number of adjacent nodes in the grid and the numbering method of the nodes.

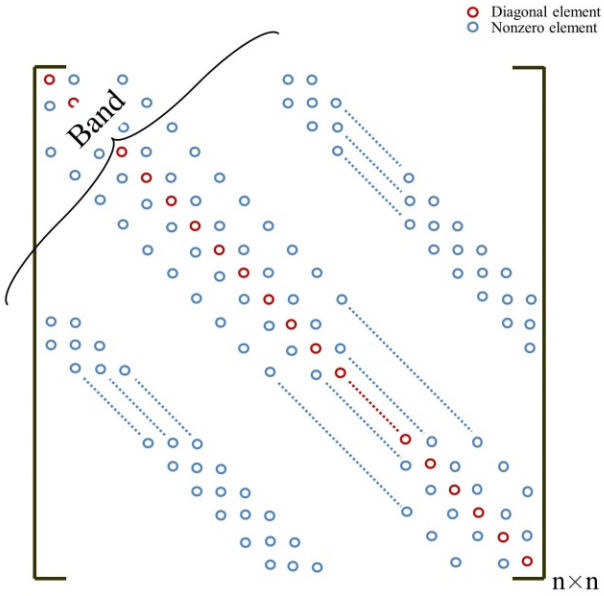

Figure 4. General form diagram of the coefficient matrix.

3. Compressed Storage and Solution Method

On the one hand, the coefficient matrix is a sparse matrix containing a large number of zeros and occupies considerable storage space. On the other hand, in the process of Gaussian elimination, if all the zero elements are involved in the elimination, the calculation steps will be significantly high. Hence, to improve the computational efficiency of the numerical software, the coefficient matrix must be compressed. Many studies have been conducted on compressed storage, mainly focusing on the compression of general sparse matrices. These compression methods have a wide range of applications, and there is no restriction on the matrix structure when compressing a sparse matrix. They are applicable for almost any large sparse matrix. However, through the previous analysis, it can be concluded that the distinct structural characteristics of the coefficient matrix should be utilized to further optimize the compressed storage. Based on the diagonally symmetrical band distribution of the nonzero elements in the coefficient matrix, a concise compressed storage and solution scheme is proposed in this paper. The flowchart is shown in Figure 5.

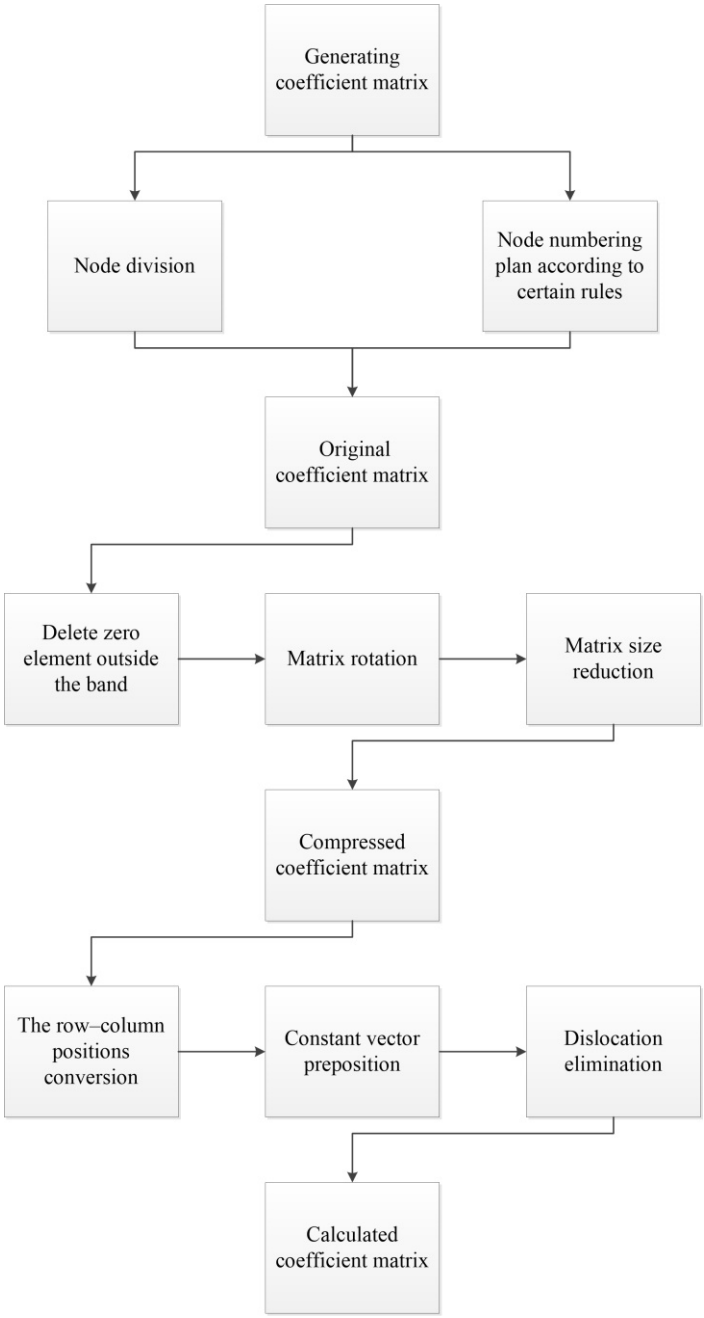

Figure 5. Flowchart of the compressed storage and solution method.

3.1. Compressed Storage of Coefficient Matrix

The compression scheme involves deleting a large number of zero elements on both sides of the band of the coefficient matrix, and the remaining elements are pivoted clockwise by 45°. Taking the coefficient matrix **M** as an example, the bandwidth of the matrix is 9. All the zero elements in the lower left and upper right of the nine column elements along the diagonal direction are deleted, and the matrix is rotated clockwise by 45°. Figure 6 shows the compression process of the coefficient matrix. The different colored circles in the figure represent the diagonal, nonzero, and zero elements. Thus, the original coefficient

matrix **M** is compressed into a new matrix **M′** of 20 rows and 9 columns, as expressed in Equation (4).

$$\mathbf{M}' = \begin{bmatrix} & & & & k_{0,0} & k_{0,1} & 0 & 0 & k_{0,4} \\ & & & k_{1,0} & k_{1,1} & k_{1,2} & 0 & 0 & k_{1,5} \\ & & 0 & k_{2,1} & k_{2,2} & k_{2,3} & 0 & 0 & k_{2,6} \\ & 0 & 0 & k_{3,2} & k_{3,3} & 0 & 0 & 0 & k_{3,7} \\ k_{4,0} & 0 & 0 & 0 & k_{4,4} & k_{4,5} & 0 & 0 & k_{4,8} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ k_{15,11} & 0 & 0 & k_{15,14} & k_{15,15} & 0 & 0 & 0 & k_{15,19} \\ k_{16,12} & 0 & 0 & 0 & k_{16,16} & k_{16,17} & 0 & 0 & \\ k_{17,13} & 0 & 0 & k_{17,16} & k_{17,17} & k_{17,18} & 0 & & \\ k_{18,14} & 0 & 0 & k_{18,17} & k_{18,18} & k_{18,19} & & & \\ k_{19,15} & 0 & 0 & k_{19,18} & k_{19,19} & & & & \end{bmatrix}_{20\times 9} \tag{4}$$

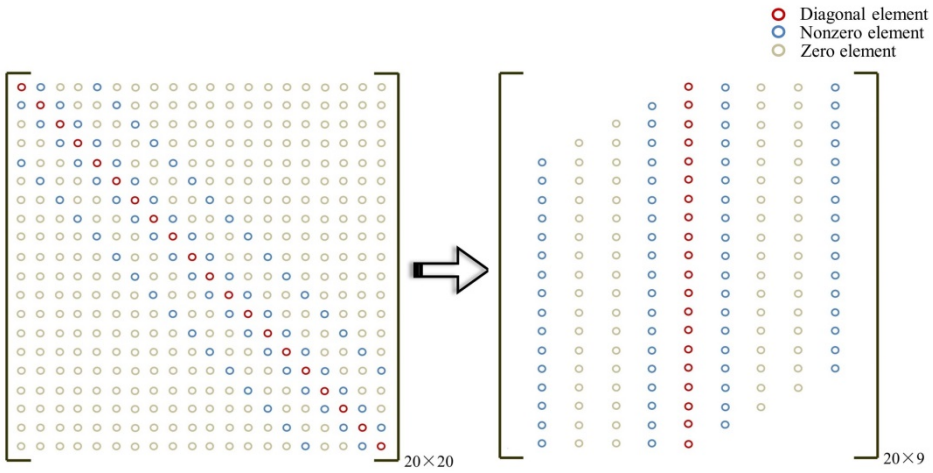

Figure 6. Compression process of the coefficient matrix **M**.

For an arbitrary matrix $M_{m\times n}$, it is assumed that the matrix storage is R in bytes. The size (double) represents the size of a double-precision floating-point variable. The storage of the coefficient matrix can be calculated using Equation (5):

$$\mathrm{R} = \mathrm{m} \times \mathrm{n} \times \mathrm{Size(double)} \tag{5}$$

The coefficient matrix **M** contains 400 elements, of which 81 are nonzero. According to Equation (5), the storage values of matrix **M** and compressed matrix **M′** are calculated, as shown in Table 1.

Table 1. Comparison of the coefficient matrix storage.

Coefficient Matrix	Uncompressed Matrix M	Compressed Matrix M′
Matrix storage (Bytes)	3200	1440

As shown in Table 1, the storage capacity of the coefficient matrix can be significantly reduced using the proposed compressed storage scheme. Theoretically, with the increase in the dimension and sparsity of the coefficient matrix, the compression efficiency should increase. The most important advantage of this storage method is that the compressed matrix can be directly eliminated by Gaussian elimination, which integrates the storage and solution of the matrix.

3.2. Solution of Coefficient Matrix

By compressing the coefficient matrix **M**, the solution of matrix **M** is transformed into a solution of the compressed matrix **M′**. Unlike the conventional Gaussian elimination scheme, where the original coefficient matrix is eliminated, the proposed compression scheme eliminates the compressed coefficient matrix, thus avoiding the substitution of a large number of zero elements for calculation and improving the calculation efficiency. Notably, the address of the compressed matrix **M′** is used directly in the programming. The relationship between the matrices **M** and **M′** in terms of their row and column positions is expressed in Equation (6):

$$\begin{cases} I = i \\ J - I + 5 = j \end{cases} \tag{6}$$

Here, I and J are the row and column positions of matrix **M**, respectively. i and j are the row and column positions of matrix **M′**, respectively.

Similarly, the row–column positions of an arbitrary coefficient matrix before and after the compression can be expressed as follows:

$$\begin{cases} I = i \\ J - I + \frac{B}{2} + 1 = j \end{cases} \tag{7}$$

Here, B represents the bandwidth of the coefficient matrix.

Taking the calculation of matrix **M** as an example, it is assumed that:

$$\mathrm{MX} = b \tag{8}$$

Here, b represents the constant vector.

The original coefficient matrix **M** is compressed into a new coefficient matrix **M′**. The constant vector b is stored in the 0th column of the matrix such that an augmented matrix **A** can be obtained, as expressed in Equation (9). The matrix **A** is the matrix actually stored and solved in the computer.

$$\mathbf{A} = \begin{bmatrix} b_0 & & & & & a_{0,5} & a_{0,6} & 0 & 0 & a_{0,9} \\ b_1 & & & & a_{1,4} & a_{1,5} & a_{1,6} & 0 & 0 & a_{1,9} \\ b_2 & & & 0 & a_{2,4} & a_{2,5} & a_{2,6} & 0 & 0 & a_{2,9} \\ b_3 & & 0 & 0 & a_{3,4} & a_{3,5} & 0 & 0 & 0 & a_{3,9} \\ b_4 & a_{4,1} & 0 & 0 & 0 & a_{4,5} & a_{4,6} & 0 & 0 & a_{4,9} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ b_{13} & a_{13,1} & 0 & 0 & a_{13,4} & a_{13,5} & a_{13,6} & 0 & 0 & a_{13,9} \\ b_{14} & a_{14,1} & 0 & 0 & a_{14,4} & a_{14,5} & a_{14,6} & 0 & 0 & a_{14,9} \\ b_{15} & a_{15,1} & 0 & 0 & a_{15,4} & a_{15,5} & 0 & 0 & 0 & a_{15,9} \\ b_{16} & a_{16,1} & 0 & 0 & 0 & a_{16,5} & a_{16,6} & 0 & 0 & \\ b_{17} & a_{17,1} & 0 & 0 & a_{17,4} & a_{17,5} & a_{17,6} & 0 & & \\ b_{18} & a_{18,1} & 0 & 0 & a_{18,4} & a_{18,5} & a_{18,6} & & & \\ b_{19} & a_{19,1} & 0 & 0 & a_{19,4} & a_{19,5} & & & & \end{bmatrix}_{20\times 10} \tag{9}$$

Here, $a_{x,y}$ represents the nonzero elements of the augmented matrix, and $b_{x,y}$ represents the elements of the constant vector.

The specific elimination step is first to eliminate the 5th column elements of the augmented matrix to 1, i.e., the diagonal elements of the original coefficient matrix **M**. Subsequently, the matrix elements are dislocation eliminated from top to bottom, while the constant-vector elements are eliminated directly from the 0th row to the 19th row. Finally, all the elements before the 5th column of the augmented matrix are eliminated to zero. Figure 7 shows the elimination process of the augmented matrix, in which the black circles represent the constant-vector elements, and the arrows represent the elimination direction. Notably, the blank areas in Equation (9), which are generated by the rotation of the original

coefficient matrix, do not actually exist; therefore, they are not involved in the calculation. The compressed storage program module can be written using Visual Basic. The program module can be found in the Appendix A.

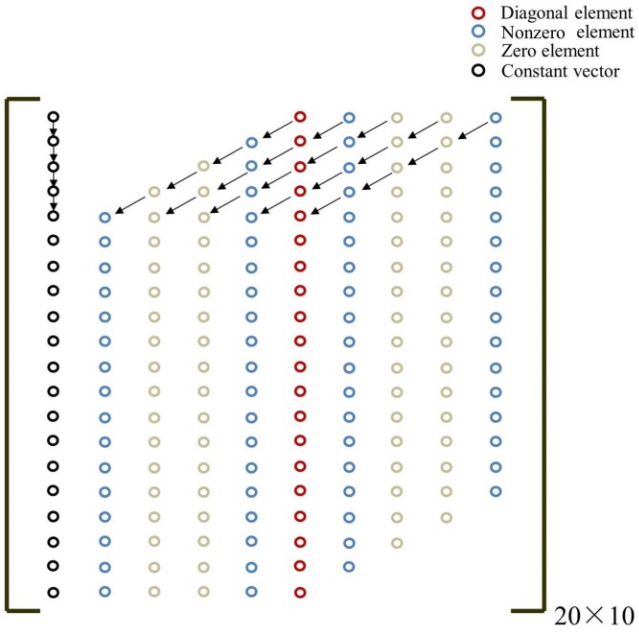

Figure 7. Elimination process of the augmented matrix **A**.

The compressed storage module has been applied in a simulation software developed by the authors of this study; this module not only helps perform accurate calculations, but also significantly improves the calculation rate of the software [28–31]. The advantages of the compressed storage method can be initially verified by solving sparse matrices of different sizes. Taking the solution of AX = b as an example and assuming that the matrix is $A_{n\times n}$ with a bandwidth of 9, the operation times of the two methods for n = 20, 50, and 200 were recorded. The average of 10 simulations of solving the same matrix was taken as the calculation time. Table 2 presents the results.

Table 2. Comparison of the computation time.

Matrix Order	20 × 20	50 × 50	200 × 200
Computation time without compression (s)	0.000148	0.002077	0.092566
Computation time of compressed storage (s)	0.000064	0.000118	0.000388
$\frac{\text{Computation time of compression storage}}{\text{Computation time without compression}}$	0.432	0.0568	0.00419

The results show that the compressed storage method requires significantly less computation time than the conventional Gaussian elimination. With the increase in the order of the matrix, the advantages of the compressed storage method become more prominent. In a numerical calculation, the solution to large linear equations is a key factor affecting the calculation rate. The proposed method that combines the matrix compressed storage and solution can play a certain role in improving the efficiency of numerical calculations.

4. Engineering Application

The compressed storage module is proposed to optimize the numerical software and apply it to engineering calculations. To further verify the practicality of the compression strategy, the compressed storage module is applied to a field engineering calculation, and the effect of the compression module on improving the efficiency of the engineering

calculation is explored. Spontaneous combustion in the goaf is a common disaster in coal mines, causing economic losses and casualties. Model research and numerical simulations of the spontaneous combustion in goaf areas have been widely conducted. COMBUSS-3D is a numerical simulation software for simulating the spontaneous combustion in the goaf area, developed independently to solve the distributions of the temperature and oxygen concentration in the goaf area and provide theoretical support for controlling spontaneous combustion. The COMBUSS-3D software used in this work is self-developed, which adopts serial mode and single thread calculation. For wider engineering applications, more works of processor vectorization and processor hyper threading will be presented in our further work. By matching field measurement results obtained from engineering projects, this software has been verified as an effective method for predicting the spontaneous combustion in goaf areas [29,32].

The Luling Coal Mine, located in Huaibei City, Anhui Province, China, was taken as the study object. Figure 8 shows the computational region of the goaf area. As shown, the two wings of the working face are the directions of the air intake roadway and air return roadway, and Γ_1, Γ_2, Γ_3, and Γ_4 are the boundaries of the computational region.

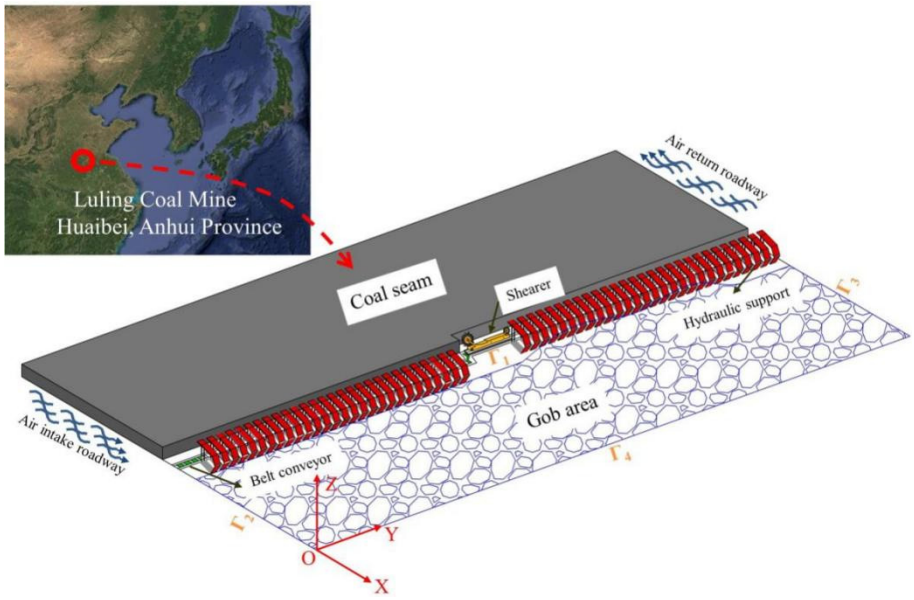

Figure 8. Computational region of the goaf area.

The physical model of spontaneous combustion in goaf includes the interaction between flow field, oxygen concentration field, gas temperature field, and solid temperature field. The partial differential equations of the four fields are shown in Equations (10)–(13).

Flow field:

$$\begin{aligned}&\sum \frac{K}{g} \cdot \frac{\partial P}{\partial x} \cos\alpha \Delta S + \sum \frac{K}{g} \cdot \left(\frac{\partial P}{\partial y} + \rho_g g \sin\theta\right) \cos\beta \Delta S + \\ &\sum \frac{K}{g} \cdot \left(\frac{\partial P}{\partial z} + \rho_g g \cos\theta\right) \cos\gamma \Delta S - \sum n \frac{\partial \rho_g}{\partial t} \Delta V = 0\end{aligned} \tag{10}$$

Here, K represents the permeability coefficient of porous media, m/s; g represents the gravitational acceleration, m/s^2; ρ_g represents the air density, kg/m^3; P represents the sum of static pressure and dynamic pressure, Pa; and n represents the porosity of float coal, %;

Oxygen concentration field:

$$\begin{aligned}&\sum n v k_{o_2} \frac{\partial C_{o_2}}{\partial \vec{n}} \Delta S - \sum C_{o_2} \left(v_x \cos\alpha + v_y \cos\beta + v_z \cos\gamma\right) \Delta S - \\ &\sum u(t) \Delta V - \sum n \frac{\partial C_{o_2}}{\partial t} \Delta V = 0\end{aligned} \tag{11}$$

Here, C_{O_2} represents the oxygen concentration, mol/m^3; k_{O_2} represents the constant term of oxygen diffusion coefficient; $\underset{n}{\rightarrow}$ represents the normal vector outside the unit of the area element on the boundary of the control volume; $u(t)$ represents the oxygen consumption per unit volume of coal in unit time, mol·m^{-3}·s^{-1}.

Gas temperature field:

$$\sum n\lambda_g \frac{\partial T_g}{\partial \vec{n}} \Delta S + \sum K_e S_e (T_s - T_g) \Delta V - \sum n\rho_g C_g t_g \vec{v} \cdot \vec{n} \Delta S - \sum n\rho_g C_g \frac{\partial T_g}{\partial t} \Delta V = 0 \quad (12)$$

Here, T_g represents the gas temperature, K; λ_g represents the thermal conductivity of gas, W/(m·K); C_g represents the specific heat of gas, KJ/(kg·K).

Solid temperature field:

$$\sum (1-n)\lambda_s \frac{\partial T_s}{\partial \vec{n}} \Delta S - \sum K_e S_e (T_s - T_g) \Delta V + \sum q(t) \Delta V - \sum (1-n)\rho_s C_s \frac{\partial T_s}{\partial t} \Delta V = 0 \quad (13)$$

Here, λ_s represents the thermal conductivity of coal and rock, W/(m·K); T_s represents the solid temperature, K; S_e represents the surface area of solid per unit volume in goaf, m^2; K_e represents the convective heat transfer coefficient, J/(m^2·s·K); ρ_s represents the density of coal and rock, kg/m^3; C_s represents the specific heat capacity of coal and rock, KJ/(kg·K).

The calculation area was divided into hexahedral meshes. The number of grid points was 5320, and the boundary conditions and mesh or grid points in different simulations were not changed. The mesh size and types are shown in Figure 9.

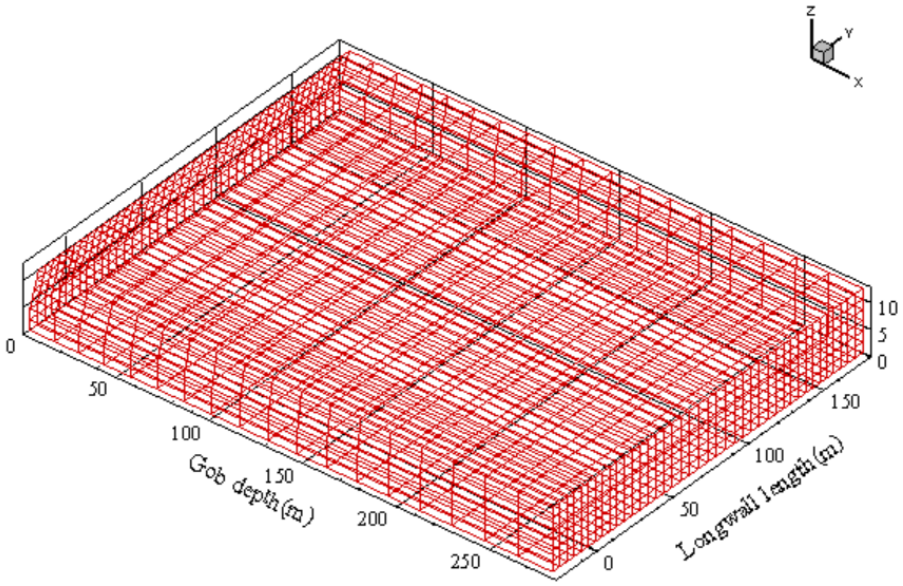

Figure 9. Mesh generation of the model of the goaf area.

Figures 10 and 11 show the simulation results of the temperature and oxygen concentration distributions in the goaf, respectively.

The compressed storage module can be incorporated into the spontaneous combustion simulation software COMBUSS-3D. Under the same initial simulation conditions, the distributions of the temperature and oxygen concentration in the goaf area of the Luling Coal Mine were simulated. The computation times of the original COMBUSS-3D software and the COMBUSS-3D software with the compressed storage module were recorded. The calculation efficiency of Gaussian elimination method was compared with that of compression storage method, and the calculation results were exactly the same. The compression method does not affect the quality of solved fields. The time required to

perform ten simulations was recorded, as shown in Table 3, in hours, with two decimal digits reserved.

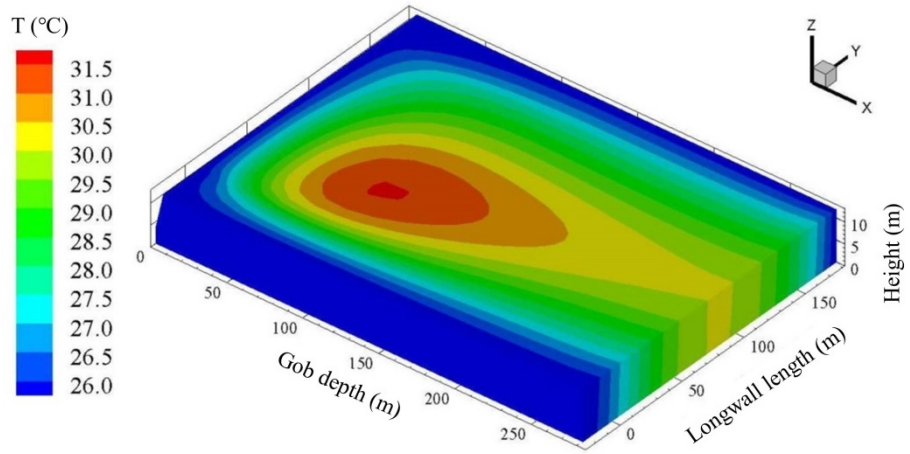

Figure 10. Simulation result of the temperature distribution in the goaf area.

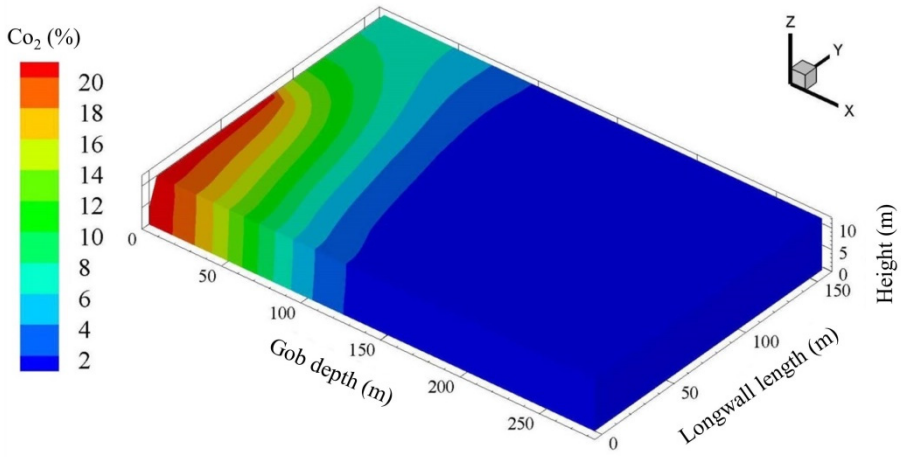

Figure 11. Simulation result of the oxygen concentration distribution in the goaf area.

Table 3. Comparisons of the computation time.

Computation Time after Incorporating the Compression Module T' (h)	Computation Time without the Compression Module T (h)	Increase in the Computational Efficiency $1-\frac{T'}{T}$ (%)
2.66	13.23	79.89
2.64	13.49	80.43
2.61	13.16	80.17
2.69	13.34	79.84
2.66	13.03	79.59
2.60	12.95	79.92
2.65	13.12	79.80
2.67	13.25	79.85
2.66	13.36	80.09
2.67	13.33	79.97

In fact, when the initial simulation conditions did not change considerably, the computation time required to simulate the same operating point was less different. Therefore, the difference in the computation times between the software for ten simulations was only a few minutes. As shown in Table 3, after incorporating the compressed storage module, the

average computation time of the software was approximately 2.65 h; the average computation time of the original software was approximately 13.23 h. The average computation efficiency was improved by 79.96%. Therefore, the compressed storage module was verified to be effective in engineering calculations and can significantly improve the calculation rate of numerical software.

5. Discussions

Method applicability to numerical simulation of spontaneous combustion in goaf is confirmed, and its compressed storage and solution in numerical simulation study should be applied more widely. As long as the grid division and node numbering are carried out according to the fixed law, the compressed storage and solution method can be used to optimize the calculation. In fact, each coefficient matrix has an optimal node numbering scheme to further reduce the bandwidth and improve the computational efficiency. Through the above analysis, compared with the computational efficiency of Gaussian elimination method, the proposed method has significant advantages. For very large coefficient matrices, the significant advantages of the method cannot be proven compared with the iterative method. However, the matrix size required in the current engineering examples simulation could be compressed by the proposed method. In addition, in the numerical calculation software using the iterative method, the method proposed in this paper can also be used to improve the efficiency of partial matrix calculation.

6. Conclusions

In this study, a method that integrates compressed storage and solution for coefficient matrices was developed to improve the computational rate of the Gaussian elimination method. The following conclusions can be drawn from the study:

(1) In numerical calculations, as long as the nodes are numbered according to certain rules, the coefficient matrix will exhibit evident structural characteristics. Typically, the nonzero elements are symmetrically distributed in the diagonal band, and all the elements on both sides outside the band are zero.

(2) Based on the structural characteristics of the coefficient matrix, a new scheme that integrates compressed storage and Gaussian elimination was developed. In this compression method, a large number of zero elements is deleted through a pivoting scheme, and the matrix order is reduced, thus significantly saving the storage space required for the coefficient matrix.

(3) When solving the coefficient matrix, a compressed coefficient matrix can be directly solved by dislocation Gaussian elimination. Compared with conventional methods, this method significantly improves the computing rate by solving matrices of different sizes. The higher the order and greater the sparsity of the coefficient matrix, the more evident the advantages of this compression method.

(4) By incorporating the compression method into the COMBUSS-3D software, it was found that the compressed storage module can significantly improve the computing rate of the simulation, by approximately 80%. Thus, the compressed storage method can be used to improve the computational efficiency of numerical simulation software, which is of great significance for efficiently solving engineering problems.

(5) For very large coefficient matrices, the significant advantages of the method cannot be proven compared with the iterative method; more studies will be presented in our further work.

Author Contributions: Conceptualization, Y.C., Y.Q. and T.Z.; Data curation, Y.C. and T.Z.; Project administration, Q.Q.; Software, Y.Q., T.Z. and Z.S.; Supervision, Q.Q.; Validation, Y.Z.; Visualization, Y.C. and T.Z.; Writing—original draft, Y.C.; Writing—review & editing, Y.C., Y.Z. and Q.Q. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported financially by the National Natural Science Foundation of China [grant number 52174188] and [grant number 52074156], and the youth projects of Science and Technology Innovation and Entrepreneurship Fund of China Coal Science and Industry Group [grant number 2022-QN001].

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data that support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

```
Public Sub Solve(A() As Double)
Dim M%, N%, i%, j%, k%, ii%, jj%
M = UBound(A, 1)
N = UBound(A, 2)
jj = (N − 1)/2
ii = M − jj
For i = 0 To ii
A(i, 0) = A(i, 0)/A(i, jj + 1)
For j = N To jj + 1 Step −1
A(i, j) = A(i, j)/A(i, jj + 1)
Next j
For k = 1 To jj
A(i + k, 0) = A(i + k, 0) − A(i + k, jj + 1 − k) × A(i, 0)
For j = N To jj + 1 Step −1
A(i + k, j − k) = A(i + k, j − k) − A(i + k, jj + 1 − k) × A(i, j)
Next j
Next k
Next i
For i = ii + 1 To M − 1
A(i, 0) = A(i, 0)/A(i, jj + 1)
For j = N To jj + 1 Step −1
A(i, j) = A(i, j)/A(i, jj + 1)
Next j
For k = 1 To M − i
A(i + k, 0) = A(i + k, 0) − A(i + k, jj + 1 − k) × A(i, 0)
For j = N − i + ii + 1 To jj + 1 Step −1
A(i + k, j − k) = A(i + k, j − k) − A(i + k, jj + 1 − k) × A(i, j)
Next j
Next k
Next i
A(M, 0) = A(M, 0)/A(M, jj + 1)
A(M, jj + 1) = 1
    "**********************
For i = M To jj Step −1
For k = 1 To jj
A(i − k, 0) = A(i − k, 0) − A(i − k, jj + 1 + k) × A(i, 0)
Next k
Next i
For i = jj − 1 To 1 Step −1
For k = 1 To i
A(i − k, 0) = A(i − k, 0) − A(i − k, jj + 1 + k) × A(i, 0)
Next k
```

Next i
End Sub

References

1. Alonso, P.; Delgado, J.; Gallego, R.; Peña, J.M. A collection of examples where Neville elimination outperforms Gaussian elimination. *Appl. Math. Comput.* **2010**, *216*, 2525–2533. [CrossRef]
2. Gilbert, A.; Indyk, P. Sparse Recovery Using Sparse Matrices. *Proc. IEEE* **2010**, *98*, 937–947. [CrossRef]
3. Pan, V.Y.; Zhao, L. Numerically safe Gaussian elimination with no pivoting. *Linear Algebra Its Appl.* **2017**, *527*, 349–383. [CrossRef]
4. Davis, T.A.; Hu, Y. The university of Florida sparse matrix collection. *ACM Trans. Math. Softw.* **2011**, *38*, 1–25. [CrossRef]
5. Tiskin, A. Communication-efficient parallel generic pairwise elimination. *Future Gener. Comput. Syst.* **2007**, *23*, 179–188. [CrossRef]
6. Alanelli, M.; Hadjidimos, A. Block Gauss elimination followed by a classical iterative method for the solution of linear systems. *J. Comput. Appl. Math.* **2004**, *163*, 381–400. [CrossRef]
7. Ji, J. Gauss–Jordan elimination methods for the Moore–Penrose inverse of a matrix. *Linear Algebra Its Appl.* **2012**, *437*, 1835–1844. [CrossRef]
8. Misawa, M.; Sekiya, T.; Oba, M. Improved Solution of Equations by Regularizing Ill-Conditioned Coefficient Matrix for System Identification. *AIAA J.* **2013**, *51*, 2076–2085. [CrossRef]
9. Vuduc, R.; Demmel, J.W.; Yelick, K.A. OSKI: A library of automatically tuned sparse matrix kernels. *J. Phys. Conf. Ser.* **2005**, *16*, 521–530. [CrossRef]
10. Wambui Mutoru, J.; Firoozabadi, A. Form of multicomponent Fickian diffusion coefficients matrix. *J. Chem. Thermodyn.* **2011**, *43*, 1192–1203. [CrossRef]
11. Yu, Y.; Zha, X.W.; Li, W. A Criterion for Maximally Six-Qubit Entangled States via Coefficient Matrix. *Int. J. Theor. Phys.* **2016**, *56*, 931–941. [CrossRef]
12. Rostami, M.W. New Algorithms for Computing the Real Structured Pseudospectral Abscissa and the Real Stability Radius of Large and Sparse Matrices. *SIAM J. Sci. Comput.* **2015**, *37*, S447–S471. [CrossRef]
13. D'Azevedo, E.F.; Fahey, M.R.; Mills, R.T. Vectorized Sparse Matrix Multiply for Compressed Row Storage Format. In *Lecture Notes in Computer Science*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 99–106. [CrossRef]
14. Hsieh, S.-H.; Yang, Y.-S.; Hsu, P.-Y. Integration of General Sparse Matrix and Parallel Computing Technologies for Large-Scale Structural Analysis. *Comput.-Aided Civ. Infrastruct. Eng.* **2002**, *17*, 423–438. [CrossRef]
15. Stabrowski, M.M. Product or sum with transposed matrix: What is best for unsymmetric sparse matrix compression. *Adv. Eng. Softw.* **2004**, *35*, 223–229. [CrossRef]
16. Chun-Yuan, L.; Yeh-Ching, C.; Jen-Shiuh, L. Efficient data compression methods for multi-dimensional sparse array operations. In Proceedings of the First International Symposium on Cyber Worlds, Tokyo, Japan, 6–8 November 2002.
17. Im, E.-J.; Yelick, K.; Vuduc, R. Sparsity: Optimization Framework for Sparse Matrix Kernels. *Int. J. High Perform. Comput. Appl.* **2004**, *18*, 135–158. [CrossRef]
18. Li, Y.; Dong, W.; Peng, Y. Study on matrix compressive storage method based on 0-1 property-matrix. *Comput. Eng. Appl.* **2003**, *39*, 82–84.
19. Cheng, G.; Zhang, B. Compression Storage and Solution of Large and Sparse Matrix in Traveltime Tomography of Reflection Seismic Data. *Prog. Geophys.* **2008**, *23*, 674–680.
20. Yang, H.; Fang, H.; Zhang, C. Large image reconstruction based on sparse-banded matrix. *Comput. Eng. Appl.* **2013**, *10*, 184–187.
21. Peña, J.M. Eigenvalue localization and pivoting strategies for Gaussian elimination. *Appl. Math. Comput.* **2013**, *219*, 7725–7729. [CrossRef]
22. Xiao, Y.; Gao, P.; Lu, Y. Improved Parallel Gaussian Elimination Algorithm in Magnetotelluric Occam's Inversion. In *Intelligent Computing Theories and Application*; Springer International Publishing: Berlin/Heidelberg, Germany, 2016; pp. 591–600. [CrossRef]
23. Druinsky, A.; Toledo, S. Factoring matrices with a tree-structured sparsity pattern. *Linear Algebra Its Appl.* **2011**, *435*, 1099–1110. [CrossRef]
24. Wang, Y.; Wu, Y.; Zhang, J. Study of Simulation and Development of Goaf Nitrogen Injecting Software. *Coal Min. Technol.* **2018**, *23*, 6–11.
25. Wu, Q.; Qin, Y.; Guo, L. Calculation of the Heat Emitting from the Wall Rock at Drifting Face with Finite Element Method. *China Saf. Sci. J.* **2020**, *12*, 33–36.
26. Qin, Y.; Song, H.; Wu, J.; Bai, Y.; Dong, Z.; Ye, F. Analysis of surrounding rock heat dissipation for trapezoid roadway by finite-volume method. *J. Liaoning Tech. Univ. Nat. Sci.* **2015**, *43*, 898–904.
27. Qin, Y.; Song, H.; Wu, J.; Dong, Z.-y. Numerical analysis of temperature field of surrounding rock under periodic boundary using Finite Volume Method. *J. China Coal Soc.* **2015**, *40*, 1541–1549.
28. Qin, Y.; Liu, H.; Zhu, C. Numerical Simulation of Goaf Hot Blast on Coal Mining Face with High Temperature. *Saf. Coal Mines* **2011**, *42*, 11–14.
29. Liu, W.; Qin, Y. Multi-physics coupling model of coal spontaneous combustion in longwall gob area based on moving coordinates. *Fuel* **2017**, *188*, 553–566. [CrossRef]
30. Qin, Y.; Sun, Q.; Liu, W. Three finite volume schemes for elastic mechanics. *J. Liaoning Tech. Univ.* **2012**, *31*, 349–353.

31. Qin, Y.P.; Sun, Q.; Yang, X.B.; Zhang, G.Y. Analysis of Four Finite Volume Schemes for Plane Stress Problems. *Appl. Mech. Mater.* **2012**, *204–208*, 4635–4642. [CrossRef]
32. Qin, Y.-P.; Liu, W.; Yang, X.-B.; Luo, W.; Hao, Y.-J. Numerical simulation of impact of non-Darcy seepage on spontaneous combustion in goaf. *J. China Coal Soc.* **2012**, *37*, 1177–1183.