Matrix calculator program

This program is written to implement computational algorithms in academic textbooks and tries to make the algorithms as similar to the mathematical form as the usual form of algorithms; Therefore :

1. The method chosen to calculate each parameter is not necessarily the best method

2. The method of implementing the selected algorithm is not necessarily the best algorithm

In the following, we will describe the written code;

The program consists of three main classes:

1. Matrix class

2. Function class

3. Form class

Matrix class:

This class is written to simulate a matrix in C #. The reason for using this class instead of a two-dimensional array is to have the advantages of memory over a dynamic array, and in some special cases the superiority of time complexity. This is how matrix elements are taken from the user and stored in a list through an address converter method, and by defining an Indexor, the elements of a matrix can be accessed like a two-dimensional array. In the following, we will review the methods used in this class.

1. public Matrix (int Row, int Col):

This method is a class builder and prepares the list as needed according to the parameters received from the user

2. public bool Init (int Row, int Col):

This method, like the constructor method, prepares the list in the class according to the received parameters, with the difference that there is no need to create a new object, and this is one of the advantages of using the written class.

3. public void Diagonal\_Maker ():

Using this function, the constructed object can be converted to a diagonal matrix.

4. private int AC (int Row, int Col):

This function receives the address received from the user that belongs to a matrix and converts it to a usable address for the list.

5. public int Zero\_count ():

This function counts the zeros in the list.

6. public int Element\_Count ():

This function returns the number of objects in the matrix.

7. public bool Swap\_Rows (int A, int B):

Using this function, two rows of the matrix can be moved, which is also one of the preliminary row operations.

8. public bool Swap\_Cols (int A, int B):

Using this function, two matrix columns can be moved, which in some cases may be used to calculate determinants.

9. public bool Square ():

Using this function, you can check that the matrix is square

10. public bool Up\_Triangle ():

This function above checks that the matrix is triangular.

11. public bool Down\_Triangle ():

This lower function checks that the matrix is triangular.

12. public int Zero\_Rows ():

Using this function, you can find the number of rows that are completely zero

13. public void Select\_Row (int Row, List <double> Output):

Using this function, you can select a specific row of the matrix and return it to the user as a list.

14. public void Select\_Col (int Col, List <double> Output):

Using this function, you can select a specific column of the matrix and return it to the user as a list.

15. public void Row\_Sum (int Row, double Value, int S\_Row):

The function of this function is to add one row of the matrix with a coefficient of another row, which is one of the preliminary row operations.

16. public void Row\_Multiplier (int Row, double Value):

This function multiplies a row of matrices by a number, which is one of the basic row operations.

17. public bool Element\_Exist (int row, int col):

This function checks if the given number is in the matrix

18. public void Row\_Remove (int Row):

This function removes the desired row from the matrix.

19. public void Col\_Remove (int Col):

This function removes the desired column from the matrix.

20. public void Scaler\_multipy (double Input):

This function multiplies the matrix by a number.

21. public bool Equal (Matrix Input):

This function checks the equality of the two matrices by checking them side by side.

Function class:

In this class, there are all the functions required to calculate the capabilities provided in the program, and the functions generally receive their values by reference through the function arguments, then we will examine the available methods.

1. private int Pivot (Matrix Input):

The function of this function is to check the original diameter of the matrix for it not to be zero, if it is zero it will move with its rows below which are not zero, and if all the rows below it are zero, there is really no need to do anything.

2. private void Gauss (Matrix Input):

This function is a simulation of Gauss's method for making a matrix triangular and its algorithm is tried to be similar to its mathematical algorithm.

3. private void Gaus\_Jordan (Matrix Input):

This function is a simulation of Gauss-Jordan's method for triangulating the matrix; The difference between this method and Gauss is that in this method the main diameter of the matrix is also 1.

4. public int Rank (Matrix Input):

The function of this function is to calculate the rank for the matrix and uses the method of linear independent rows.

5. public double Determinant (Matrix Input):

In this function, the determinant of the matrix is calculated by Gaussian method and multiplied by the principal diameters.

6. public void Transpose (Matrix Input, Matrix Output):

In this function, the transatrix of the matrix is calculated and placed in the Matrix Output.

7. public bool Addition (Matrix A, Matrix B, Matrix Output):

Using this function, two matrices can be added together

8. public bool Subtraction (Matrix A, Matrix B, Matrix Output):

Using these two functions, two matrices can be subtracted

9. public bool Multipy (Matrix A, Matrix B, Matrix Output):

The task of this function is to perform matrix multiplication operations

10. public bool QR\_Factor (Matrix Input, Matrix Q, Matrix R):

The function of this function is to calculate the QR decomposition of the matrix; This function uses Household Holder transformations to find matrix decomposition.

11. public bool LU\_Factor (Matrix Input, Matrix L, Matrix U):

The task of this function is to find the LU decomposition of the matrix, which we use to find the triangular top of the Gauss-Jordan method and the bottom triangle to find the coefficients used in the Gaussian method; Of course, separate coefficients are calculated to avoid increasing the complexity of memory.

12. public bool invers (Matrix Input, Matrix Output):

The function of this function is to find the inverse of the matrix; In this function, the method of finding matches is used, but of course, the added matrix can also be used.

13. public bool Eigenvalue (Matrix Input, Matrix Output):

The task of this function is to find the eigenvalues of the matrix, which is from the method of shifting the QR decomposition to RQ to the top of the triangle, and the objects on the main diameter are the eigenvalues of the matrix. This is inaccurate and may show the wrong value to the user if there is a duplicate answer.

14. public double Trace (Matrix Input):

The task of this function is to find the sum of the elements on the original diameter

15. public bool SM (Matrix Input):

The function of this function is to detect the symmetry of the input matrix.

Form class:

In this class, information is taken from the user and the information is displayed to the user; The TextBox is used to capture information and display information, and the Button is used to record information.

Data input is in the form of CSV, which means that each row must be entered in a new line to separate the columns from each other, and to separate the rows from each other. To convert the data into a list, split is used, which is almost a simple task due to the fact that the letter separator is clear, and to prevent incorrect entry of information, two steps of the check are embedded as follows:

1. First check that there is no letter in the input information

2. Then using Tryparse if the value can not be converted to a number, an error message will be printed.