3/ Gauss Elimination with Partial Pivoting

Grzegorz Studzinski / 306504 / IS / gr. 7

Date: 27.03.2020

Task source: http://home.agh.edu.pl/~byrska/src/MN_2020/4_Gauss_Elimination_Partial_Pivoting.pdf

$$a_{00}^{(2)}x_0 + a_{01}^{(2)}x_1 + \ldots + a_{0n}^{(2)}x_n = b_0^{(2)}$$

$$a_{11}^{(n)}x_1 + \ldots + a_{1n}^{(n)}x_n = b_1^{(n)}$$

$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$

$$a_{nn}^{(n)}x_n = b_n^{(n)}$$

1. Source code

1.1 Collecting user input

Gauss.txt is to be filled with data in the following order:

1st line: number of equations

Next lines: a[0][0] a[0][1] ……. b[0] and so on

Problems and solutions:

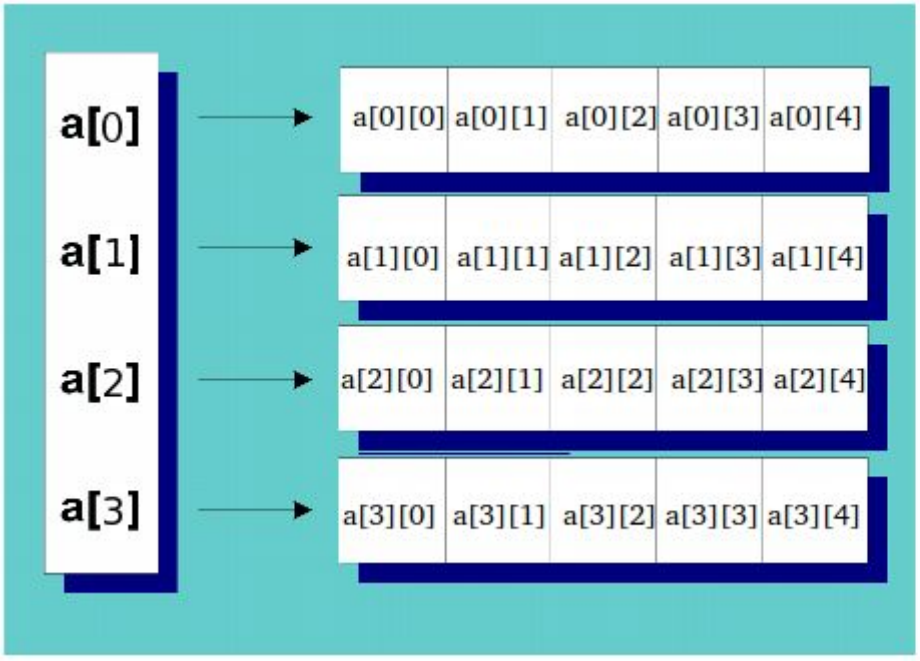
a) Using a dynamic 2D array and not getting an error.

Solution:

A dynamic 2D array is basically an array of *pointers to arrays*. You can initialize it using a loop, like this:

```
int** a = new int*[rowCount]; for(int i = 0; i < rowCount; ++i) a[i] = new int[colCount];
```

The above, for `colCount= 5` and `rowCount = 4`, would produce the following:



```
ifstream fin("gauss.txt");
fin >> n;


a = new float*[n];
for (int i = 0; i < n; ++i)
    a[i] = new float[n+1];


for (int i = 0; i < n; i++) {
    for (int j = 0; j < n+1; j++) {
        fin >> a[i][j];
        if (j > n-1) cout << "| ";
        cout << a[i][j] << "    ";
    }
    cout << "\n";
}
```

```
Microsoft Visual Studio Debug Console
3    2    -4    | 3
2    3    3    | 15
5    -3    1    | 14
```

Also - displays the matrix in console window.

1.2 Partial pivotisation

Comparing elements vertically and swapping them if the pivot element is smaller.

```
for ( int i=0;i<n;i++)
    for (int k=i;k<n;k++)
        if (a[i][i]<a[k][i])
            for (int j = 0; j <= n; j++) {
                double temp = a[i][j];
                a[i][j] = a[k][j];
                a[k][j] = temp;
            }
```

1.3 Gauss elimination

```
//gauss elimination
for (int i =0; i<n-1;i++) // not going to the last row
    for (int k = i + 1; k < n; k++)
    {
        double mult = a[k][i] / a[i][i];
        for (int j = 0; j <= n; j++)
            a[k][j] = a[k][j] - mult * a[i][j]; //make the elements below the pivot equal to zero.
    }
```

1.4 Back substitution.

```
for (int i = n - 1; i >= 0; i--)
{
    x[i] = a[i][n];                //x[i] is now the right side of equation in line [i]
    for (int j = i + 1; j < n; j++)
        if (j != i)            //substract all left side numbers except the coefficient of the variable whose value is being calculated
            x[i] = x[i] - a[i][j] * x[j];
    x[i] = x[i] / a[i][i];            //divide the right side of equation by the coefficient of the variable to be calculated
}
```

1.5 Results.

```
cout << "\nThe values of the variables are as follows:\n";
for (int i = 0; i < n; i++)
    cout <<"x("<<i<<")= "<< x[i] << endl;
```

2. Examples

Program displays the matrix after every operation as shown on the screen below:

Matrix 3x3 example:

```
Microsoft Visual Studio Debug Console
The matrix before pivotisation: (from file)
3   2   -4   | 3
2   3   3   | 15
5   -3   1   | 14

The matrix after pivotisation:
5   -3   1   | 14
2   3   3   | 15
3   2   -4   | 3

The matrix after gauss elimination:
5   -3   1   | 14
0   4.2   2.6   | 9.4
0   0   -6.95238   | -13.9048

The values of the variables are as follows:
x(0)= 3
x(1)= 1
x(2)= 2
```

Matrix 4x4 example:

```
Microsoft Visual Studio Debug Console
The matrix before pivotisation: (from file)
0   2   -4   1   | 2
2   3   3   3   | 15
5   -3   1   1   | 14
2   -1   0   1   | 12

The matrix after pivotisation:
5   -3   1   1   | 14
2   3   3   3   | 15
2   -1   0   1   | 12
0   2   -4   1   | 2

The matrix after gauss elimination:
5   -3   1   1   | 14
0   4.2   2.6   2.6   | 9.4
0   0   -0.52381   0.47619   | 5.95238
0   0   0   -5   | -62

The values of the variables are as follows:
x(0)= -2.89091
x(1)= -5.38182
x(2)= -0.0909091
x(3)= 12.4
```