

**Gaussian elimination**

School grade: K12

**Content**

[Definition 2](#_Toc125623445)

[The method of solving the method 3](#_Toc125623446)

[Implementation 5](#_Toc125623447)

[Algorithm 7](#_Toc125623448)

[The program c++ 9](#_Toc125623449)

[Exercise 1 11](#_Toc125623450)

[Exercise 2 12](#_Toc125623451)

[Exercise 3 15](#_Toc125623452)

[Exercise 4 (from BAC exams) 18](#_Toc125623453)

[Exercise 5 19](#_Toc125623454)

[Exercise 6 20](#_Toc125623455)

[Exercise 7 23](#_Toc125623456)

[Sources 25](#_Toc125623457)

# Definition

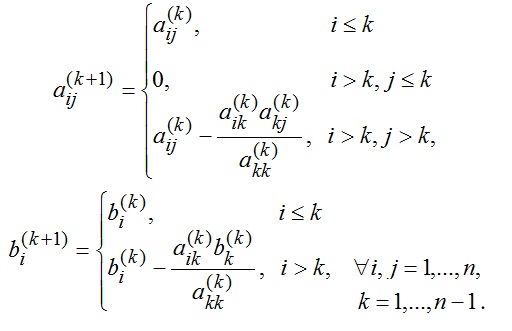
In mathematics, Gaussian elimination (also called row reduction) is a method used to solve systems of linear equations. It is named after Carl Friedrich Gauss, a famous German mathematician who wrote about this method but did not invent it.

Gaussian elimination is a technique for transforming the matrix A to upper triangular form. The transformation matrix T is a unitary lower triangular matrix obtained as a sequence (product) of elementary lower triangular transformations of the form T = Tn-1Tn-2 . . . T1, where the matrices Tp are lower triangular, order n in shape:

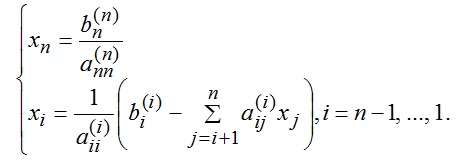
# ****The method of solving the method****

We note, initially, A(1)=A, b(1)=b, the superscript representing the stage.

The recurrence relations in the Gaussian elimination method are:



The system is solved by the inverse substitution method according to the relations:



Gaussian elimination is a method for solving matrix equations of the form Ax=b. The algorithm is not a complicated one, but it appears quite often in programming contests and has interesting applications.

Suppose we have the following system:

\begin{pmatrix}
a_{11} &  a_{12}  & \ldots & a_{1n}\\
a_{21} &  a_{22}  & \ldots & a_{2n}\\
\vdots & \vdots   & \ddots & \vdots\\
a_{n1} &  a_{n2} & \ldots  & a_{nn}
\end{pmatrix} * 
\begin{pmatrix}
x_{1} \\
x_{2} \\
\vdots \\
x_{n}
\end{pmatrix} = 
\begin{pmatrix}
b_{1} \\
b_{2} \\
\vdots \\
b_{n}
\end{pmatrix}

To solve the system we will transform all the elements below the main diagonal of the expanded matrix to 0 to be able to write each unknown only in terms of the unknowns with higher indices.


\begin{pmatrix}
a_{11} &  a_{12} & \ldots & a_{1n} & b_{1}\\
a_{21} &  a_{22} & \ldots & a_{2n} & b_{2}\\
a_{31} &  a_{32} & \ldots & a_{3n} & b_{3}\\
\vdots & \vdots  & \ddots & \vdots \\
a_{n1} &  a_{n2} & \ldots  & a_{nn} & b_{n}
\end{pmatrix}

\rightarrow

\begin{pmatrix}
a_{11} &  a_{12} & \ldots & a_{1n} & b_{1}\\
0      &  a'_{22} & \ldots & a'_{2n} & b'_{2}\\
0      &  0      & \ldots & a'_{3n} & b'_{3}\\
\vdots & \vdots  & \ddots & \vdots \\
0 &  0 & \ldots  & a'_{nn} & b'_{n}
\end{pmatrix}


Having the matrix in this form we can easily find each unknown in the equation where the unknowns with lower indices have the coefficient 0:  
![
\:
x_{n}=\frac{b'_{n}}{a'_{nn}}](data:image/gif;base64,R0lGODlhRgAeAPMPAAAAABERESIiIjMzM0RERFVVVWZmZnd3d4iIiJmZmaqqqru7u8zMzN3d3e7u7v///yH5BAAAAAAALAAAAABGAB4AAATx8MlJq7046827/5cAjuSoGGWqSouQTMeyzmTQTAWte+3EvLtg5jBYJIDCpIUgewSUUIrowXhGo7mHAXGNKhQJ7kMRA3eVDAaBeugoFgcE4rA+Z8iPg8KxSfAXAQ4/HweFhodtJAcMeQuMGjcPCFlnAJaXmJmaAHUWBWJ2SgGPMmQJCwqUFYisI1MfCk8LACw3ZG0OtF0nJkUMcQuRDwWMqWcxQlMGqHxYQgxZegpBR3Avg6EzCCh509k6Ao+v3zNWDQQOTeQqR2BbEmSoQF8LcOsgaWu/EqcyBs333LTR04zbAIABNyxq9Igbk4QQI2aIAAA7)  
x_{n-1}=\frac{b'_{n-1}-a'_{n-1n}*x_{n}}{a'_{n-1n-1}}  
 \vdots  
 x_{i}=\frac{b'_{i}-\sum\limits_{j=i+1}^n a'_{ij}*x_{j}}{a'_{ii}}

Now that we know how to find the unknowns from the triangular form of the matrix, all that remains is to transform the matrix.

To transform the matrix into a triangular shape we will apply two operations:  
L_{i} \longleftrightarrow L_{j} : the interchange of two lines  
L_{j} \longleftarrow L_{j}+a*L_{i} where L_{i} is a row of the extended matrix.

For example:


\begin{pmatrix}
2  &  1 & -1 & 8 \
-3 & -1 & 2 & -11 \
-2 & 1 & 2 & -3
\end{pmatrix} \longrightarrow

\begin{pmatrix}
2 & 1 & -1 & 8 \
0 & \frac{1}{2} & \frac{1}{2} & 1 \
0 & 2 & 1 & 5
\end{pmatrix} \longrightarrow
\begin{pmatrix}
2 & 1 & -1 & 8 \
0 & \frac{1}{2} & \frac{1}{2} & 1 \
0 & 0 & -1 & 1
\end{pmatrix}


To get the second matrix I multiplied the first line by \frac{3}{2} and I added it to the second line, and then I added it to the last line (\frac{2}{2}=1). To get the last matrix I multiplied the second line by  -\frac{2}{\frac{1}{2}}=-4.

As can be seen from the example, at each step we build a column and a line from the final matrix, the column being filled with 0 below the fixed line. Suppose we want to convert all elements below row i on column j to 0. For every line k ( k>i ) we will multiply the line i by  -\frac{a_{kj}}{a_ij} and we will add it to line k thus the element on column j turning into 0. In case of a_{ij}=0 we have to look for a line k(k > i) such that a_{kj}\neq 0. If this line does not exist, the system has no solution. Applying these steps we will finally arrive at a triangular matrix from which we will find the unknowns. The complexity of the algorithm is O(N^3)

# Implementation

The code below also solves the case when we have more equations than unknowns.

void elim(int n,int m,double s[][]) {*//system with n unknown m equations*

for(int i=1,j=1,k;i<=n && j<=m;) {

for(k=i;k<=n; ++k)

if(s[k][j]!=0) break;*//* *we are looking for a line to use to form zeros on column j*

if(k>n) {*// I did not find any line for which s[i][j] is zero, so we move to the next column, line i not being the final one*

++j;

continue;

}

if(k!=i)for(int l=1; l<=m+1; ++l) swap(s[i][l],s[k][l]);*//we exchange the lines to have a null element on line i and column j*

for(k=i+1; k<=n; ++k)

for(int l=m+1; l>=j; --l)

s[k][l]-=((s[k][j]\*s[i][l])/s[i][j]);*//we apply the transformation for each line greater than i to have 0 on column j below line i*

++i; ++j;

}

*//we learn the unknown*

for(int i=n; i;--i)

for(int j=1; j<=m+1; ++j) if(fabs(s[i][j])>EPS) {

*//because it is possible to have more equations than unknowns*

*//we look for the first zero coefficient on each line, appearing from right to left*

if(j==m+1) {*//the line has no non-zero coefficients, so we have no solution*

g<<"Impossible";

exit(0);

}

x[j]=s[i][m+1];

for(int k=j+1; k<=m; ++k) x[j]-=s[i][k]\*x[k];

x[j]/=s[i][j];

break;*//we go to the previous line*

}

}

# Algorithm

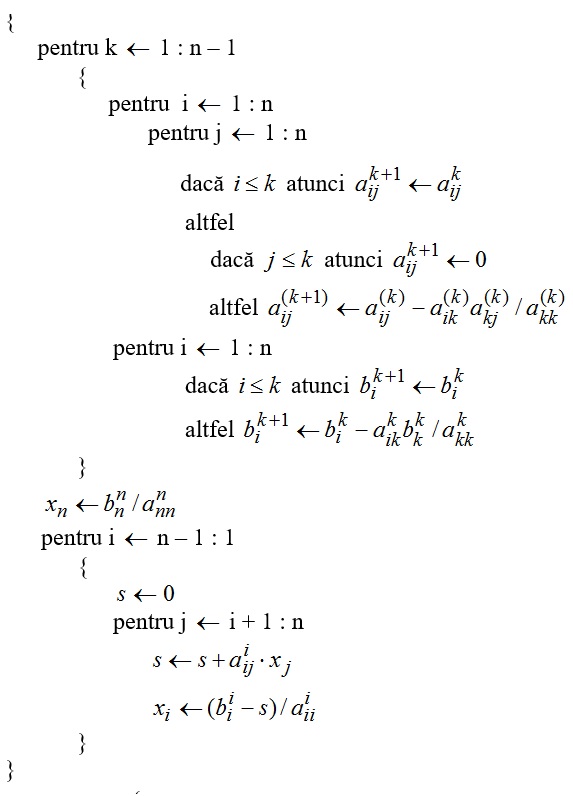
The algorithm associated with the Gauss elimination method is:

Inputs:

* n = the number of equations and unknowns of the system
* A = system matrix
* b = the vector of free terms

Outputs:

* x = the solution vector



# The program c++

#include "stdafx.h"

#include

using namespace std;

#define max 15

void main(void)

{

float s;

float a[max + 1][max + 1][max + 1], b[max + 1][max + 1], x[max + 1];

int n, i, j, k;

cout << "dati dimensiunea matricii" << endl; cin >> n;

cout << "dati matricea sistemului " << endl;

for (i = 1; i <= n; i++)

for (j = 1; j <= n; j++)

{

cout << "a[" << i << j << "]=";

cin >> a[i][j][1];

}

cout << endl;

cout << "dati vectorul termenilor liberi " << endl;

for (i = 1; i <= n; i++)

{

cout << "b[" << i << "]=";

cin >> b[i][1];

}

cout << endl;

for (k = 1; k <= n - 1; k++)

{

for (i = 1; i <= n; i++)

for (j = 1; j <= n; j++)

{

if (i <= k)a[i][j][k + 1] = a[i][j][k];

else if (j <= k)a[i][j][k + 1] = 0;

else a[i][j][k + 1] = a[i][j][k] - a[i][k][k] \* a[k][j][k] / a[k][k][k];

}

for (i = 1; i <= n; i++)

if (i <= k)b[i][k + 1] = b[i][k];

else b[i][k + 1] = b[i][k] - a[i][k][k] \* b[k][k] / a[k][k][k];

}

x[n] = b[n][n] / a[n][n][n];

for (i = n - 1; i >= 1; i--)

{

s = 0;

for (j = i + 1; j <= n; j++)

s = s + a[i][j][i] \* x[j];

x[i] = (b[i][i] - s) / a[i][i][i];

}

cout << "The approximate solution is:" << endl;

for (i = 1; i <= n; i++)

cout << x[i] << ' ';

cout << endl;

system("pause");

}

# Exercise 1

The following system is considered:



The coefficients are written in tabular form, and on the right in a separate column - free members. The column with free members is separated for convenience. The array that includes this column is called extended.

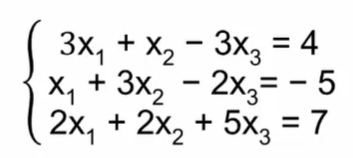


In addition, the main coefficient matrix must be reduced to upper triangular form. This is the main point of solving the system by the Gaussian method. Simply, after some manipulations, the matrix should look like this, so that there are only zeros in its lower left part:

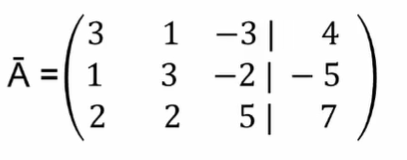


# Exercise 2

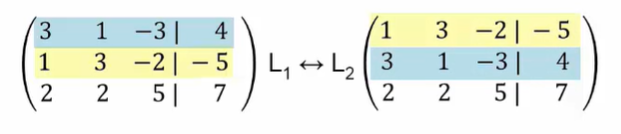
The following system is considered:

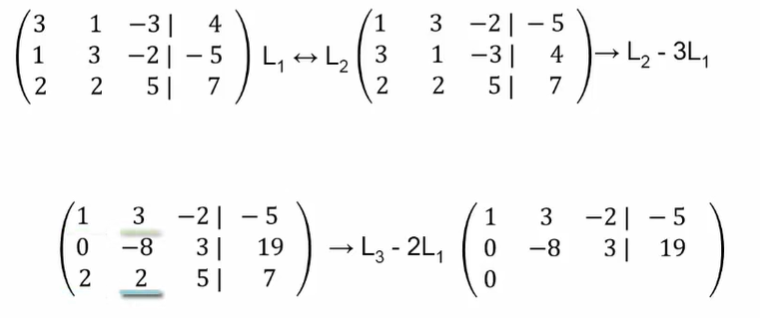


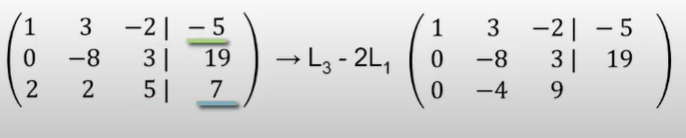
The extended matrix associated with the system is:



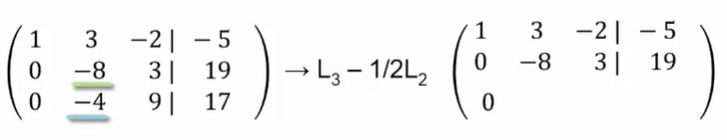
We were changing the line L1  with the line L2 to have the lowest value on the first line in the first position

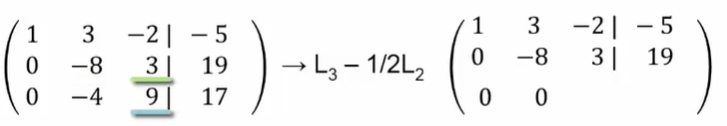


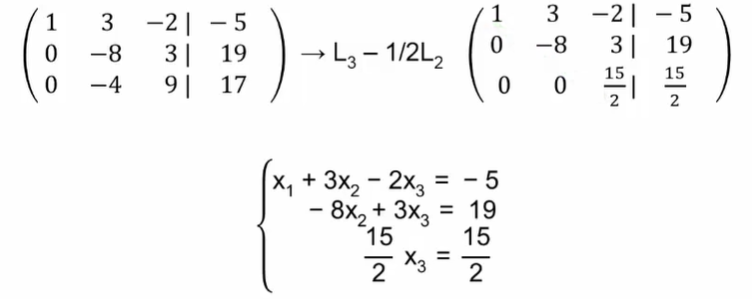


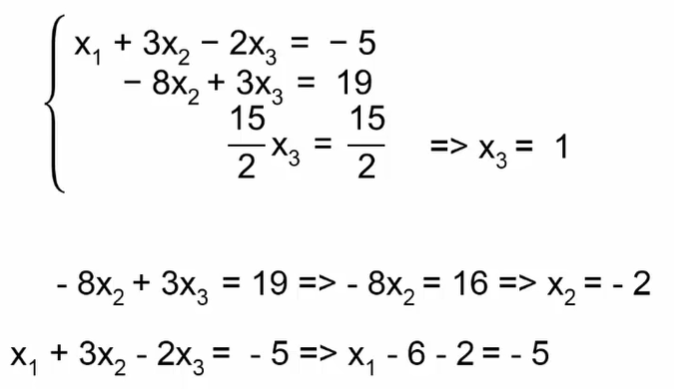












# Exercise 3

Apply the Gaussian elimination method to solve the system:



**Solving**:

The matrix A associated to the system (in step 1) and the vector of free terms b are:





The solutions are:

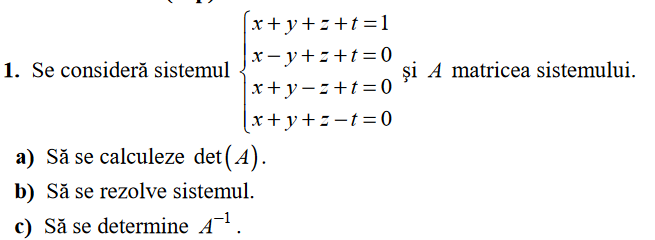


that is, the solution of the system is:

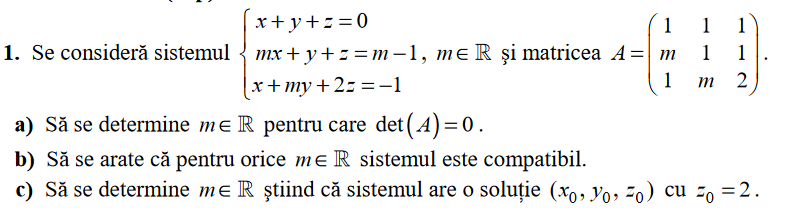


# Exercise 4 (from BAC exams)

BAC exam 2018



BAC exam 2017



# Exercise 5

The following system is considered:



The coefficients are written in tabular form, and on the right in a separate column - free members. The column with free members is separated for convenience. The array that includes this column is called extended.



In addition, the main coefficient matrix must be reduced to upper triangular form. This is the main point of solving the system by the Gaussian method. Simply, after some manipulation, the matrix should look like this, so that there are only zeros in its lower left part:



# 

# Exercise 6

The following system is considered:

Diagram

Description automatically generated with low confidence

The extended matrix associated with the system is:

A picture containing text, clock

Description automatically generated

We were changing the line L1  with the line L2 to have the lowest value on the first line in the first position

Diagram

Description automatically generated

Calendar

Description automatically generated with medium confidence

A picture containing text, clock, gauge

Description automatically generated



A picture containing logo

Description automatically generated

Logo

Description automatically generated

Diagram

Description automatically generated

Text, letter

Description automatically generated

# Exercise 7

Apply the Gaussian elimination method to solve the system:



**Solving**:

The matrix A associated with the system (in step 1) and the vector of free terms b are:





The solutions are:



that is, the solution of the system is:



# Sources

<https://en.wikipedia.org/wiki/Regular_polygon>

<https://www.youtube.com/watch?v=qetSusATv2w>