In the name of God

(GAUSS ELIMINATION)

INTRODUCTION, DEFINITIONS AND EXAMPLES

By

Reza Nopour

A part of:

Finite Element Method Course

Professor M. R. Eslami

Mechanical Engineering Department

Amirkabir University of Technology

03/03/03 Solar Hijri

Problem Description

Consider Gaussian elimination method for solving systems of linear equations. Write associate code (*C++/Python*) and explain some examples.

First some introduction and definition will be introduced, then the code will be explained followed by some examples.

Introduction

In mathematics, *Gaussian elimination*, also known as row reduction, is an algorithm for solving systems of linear equations. It consists of a sequence of row-wise operations performed on the corresponding matrix of coefficients. This method can also be used to compute the rank of a matrix, the determinant of a square matrix, and the inverse of an invertible matrix. The method is named after Carl Friedrich Gauss (1777–1855). To perform row reduction on a matrix, one uses a sequence of elementary row operations to modify the matrix until the lower left-hand corner of the matrix is filled with zeros, as much as possible. There are three types of elementary row operations:

- Swapping two rows,
- Multiplying a row by a nonzero number,
- Adding a multiple of one row to another row.

Using these operations, a matrix can always be transformed into an upper triangular matrix, and in fact one that is in row *echelon* form.

History

The method of Gaussian elimination appears – albeit without proof – in the Chinese mathematical text Chapter Eight: *Rectangular Arrays of The Nine Chapters on the Mathematical Art*. Its use is illustrated in eighteen problems, with two to five equations. The first reference to the book by this title is dated to 179 AD, but parts of it were written as early as approximately 150 BC. It was commented on by Liu Hui in the 3rd century.

The method in Europe stems from the notes of Isaac Newton. In 1670, he wrote that all the algebra books known to him lacked a lesson for solving simultaneous equations, which Newton then supplied. Cambridge University eventually published the notes as Arithmetica Universalis in 1707 long after Newton had left academic life. The notes were widely imitated, which made (what is now called) Gaussian elimination a standard lesson in algebra textbooks by the end of the 18th century. Carl Friedrich Gauss in 1810 devised a notation for symmetric elimination that was adopted in the 19th century by professional hand computers to solve the normal equations of least-squares problems.

Some authors use the term Gaussian elimination to refer only to the procedure until the matrix is in echelon form, and use the term Gauss–Jordan elimination to refer to the procedure which ends in reduced echelon form. The name is used because it is a variation of Gaussian elimination as described by Wilhelm Jordan in 1888. However, the method also appears in an article by Clasen published in the same year. Jordan and Clasen probably discovered Gauss–Jordan elimination independently.

Definitions

This part is from *Finite Elements Methods in Mechanics* book by Prof. M.R. Eslami.

The Gauss elimination method is widely used to solve systems of equations obtained through finite element modeling. The standard method of Gauss elimination is developed to solve a system of equations with a constant coefficient matrix of $n \times n$ size, or a square matrix. This method is partially modified to handle the rectangular banded matrices, which are naturally obtained for finite element problems. Gauss elimination is a direct method of solution of the system of equations. An alternative to this method is the iteration and relaxation algorithm, which is sometimes employed to solve a system of equations obtained by finite element modeling.

To describe the method, consider the general system of three linear equations as follows,

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= c_1 \\ a_{21}x_1 + a_{22}x_2 + a_{23}x_3 &= c_2 \\ a_{31}x_1 + a_{32}x_2 + a_{33}x_3 &= c_3 \end{aligned} \tag{1}$$

As the first step, the first equation is multiplied through by $-a_{21}/a_{11}$ and added to the second equation to replace the second equation. Similarly, the first equation is multiplied through $-a_{31}/a_{11}$ and added to the third equation to replace the third equation. The final set of equations become

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= c_1 \\ a'_{22}x_2 + a'_{23}x_3 &= c'_2 \\ a'_{32}x_2 + a'_{33}x_3 &= c'_3 \end{aligned} \tag{2}$$

where a' and c' are the new coefficients resulting from the multiplications and additions. The result of the first step operation was elimination of the variable x_1 from the second and third equations. Now, as the second step, we try to eliminate x_2 from the third equation. Multiplying through the second equation by $-a'_{32}/a'_{22}$ and adding it to the third equation yields

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 &= c_1 \\ a'_{22}x_2 + a'_{23}x_3 &= c'_2 \\ a''_{33}x_3 &= c''_3 \end{aligned} \tag{3}$$

where a''_{33} and c''_3 are obtained from the arithmetic operations. The system of equations (2) is now triangularized and is ready to be solved. The solution begins from the last equation, in which

$$x_3 = c''_3/a''_{33} \tag{4}$$

The value of x_3 from Eq. (4) is substituted into the second of Eq. (3) and is used to solve for x_2. Finally, the values of x_3 and x_2 are substituted into the first of Eq. (3) and x_1 is obtained. The solution procedure will begin once the system of equations is triangularized, as seen in Eq. (3). The operation of triangularization may be followed through a series of matrix multiplications. Consider the following matrix

$$\begin{aligned} [D] = [A|C] &= \begin{bmatrix} a_{11} & a_{12} & a_{13} & c_1 \\ a_{21} & a_{22} & a_{23} & c_2 \\ a_{31} & a_{32} & a_{33} & c_3 \end{bmatrix} \\ [S_3][S_2][S_1][D] &= \begin{bmatrix} a_{11} & a_{12} & a_{13} & c_1 \\ 0 & a'_{22} & a'_{23} & c'_2 \\ 0 & 0 & a''_{33} & c''_3 \end{bmatrix} \end{aligned} \tag{5}$$

where the broken line indicates the matrix partitioning. Defining the matrices

$$[S_1] = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{a_{21}}{a_{11}} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, [S_2] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -\frac{a_{31}}{a_{11}} & 0 & 1 \end{bmatrix}$$

$$[S_3] = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{a'_{32}}{a'_{22}} & 1 \end{bmatrix} \quad (6)$$

Once the matrix of coefficients is triangularized, the solution follows from the last equation, and proceeds upward by proper back substitution. The algorithm is as follows,

$$d_{11} = k_{11}$$

for $j = 2, \cdots n$

$$g_{m_j,j} = k_{m_j,j}$$

$$g_{ij} = k_{ij} - \sum_{r=m_m}^{i-1} l_{ri} g_{rj} \qquad i = m_j + 1, \ldots j - 1.$$

where

$$l_{ij} = \frac{g_{ij}}{d_{ii}} \qquad i = m_j, \ldots j - 1$$

$$d_{jj} = k_{jj} - \sum_{r=m_j}^{j-1} l_{rj} g_{rj}.$$

C++ program

The C++ program is provided in **Table 1**.

Table 1 C++ Program of Gauss Elimination Method.

```
#include <iostream>
#include <iomanip>
#include <math.h>
#include <stdlib.h>
#define SIZE 15

using namespace std;

int main() {
    float a[SIZE][SIZE], x[SIZE], ratio;
    int i, j, k, n;

    // Input: Number of unknowns
    cout << "Enter the number of unknowns: ";
    cin >> n;
```

```
    // Input: Coefficients of Augmented Matrix
    cout << "Enter the coefficients of the Augmented Matrix:" <<
endl;
    for (i = 1; i <= n; i++) {
        for (j = 1; j <= n + 1; j++) {
            cout << "a[" << i << "][" << j << "] = ";
            cin >> a[i][j];
        }
    }

    // Applying Gauss Elimination
    for (i = 1; i <= n - 1; i++) {
        if (a[i][i] == 0.0) {
            cout << "Mathematical Error!";
            exit(0);
        }
        for (j = i + 1; j <= n; j++) {
            ratio = a[j][i] / a[i][i];
            for (k = 1; k <= n + 1; k++) {
                a[j][k] = a[j][k] - ratio * a[i][k];
            }
        }
    }

    // Obtaining Solution by Back Substitution Method
    x[n] = a[n][n + 1] / a[n][n];
    for (i = n - 1; i >= 1; i--) {
        x[i] = a[i][n + 1];
        for (j = i + 1; j <= n; j++) {
            x[i] = x[i] - a[i][j] * x[j];
        }
        x[i] = x[i] / a[i][i];
    }

    // Displaying Solution
    cout << endl << "Solution:" << endl;
    for (i = 1; i <= n; i++) {
        cout << "x[" << i << "] = " << x[i] << endl;
    }

    return 0;
}
```

After running this code, a window will be open as follows,

The number of unknowns should be determined. After that, $[A|C]$ can be set. It will be clarified in *examples* section.

Python program

Python program can be found in **Table 2**. In this code a function is written and then in #input matrix (*line 24, bolded here*) the matrix in question can be specified. `{val:.6f}` section evaluate the solution number decimal up to 6 places.

Table 2 Python Program of Gauss Elimination Method.

```
def gaussian_elimination(a):
    n = len(a)

    # Applying Gaussian elimination
    for i in range(n):
        if a[i][i] == 0.0:
            print("Mathematical Error!")
            return None
        for j in range(i + 1, n):
            ratio = a[j][i] / a[i][i]
            for k in range(n + 1):
                a[j][k] -= ratio * a[i][k]

    # Obtaining solution by back substitution method
    x = [0] * n
    for i in range(n - 1, -1, -1):
        x[i] = a[i][n]
        for j in range(i + 1, n):
            x[i] -= a[i][j] * x[j]
        x[i] /= a[i][i]

    return x

#input matrix
augmented_matrix = [
    [2, 5, -3, 1,6],
    [5, 1, -2, -5,15],
    [-3, -2, 6, 2, 12],
    [1, -5, 2, 3, 7]
]

solution = gaussian_elimination(augmented_matrix)

# Displaying Solution

if solution:
    print("Solution:")
    for i, val in enumerate(solution, start=1):
        print(f"x[{i}] = {val:.6f}")
```

Program General Description

Input:

1- The user specifies the number of unknowns (variables) in the system of linear equations.

2- The coefficients of the augmented matrix (including the right-hand side constants) are provided by the user.

Gaussian Elimination:

1- The program transforms the augmented matrix into an upper triangular form using Gaussian elimination.

2- It starts by checking if the diagonal element (pivot) is zero. If so, it exits with an error message.

3- For each row below the current row, it calculates a ratio and subtracts a multiple of the current row from the subsequent rows to make the elements below the pivot zero.

Back Substitution:

1- After obtaining the upper triangular matrix, the program solves for the unknowns using back substitution.

2- Starting from the last row, it calculates the value of each unknown by subtracting the known values of the other variables.

Output:

1- The solution (values of the unknowns) is displayed after performing the calculations.

Examples

First example is as same as that provided in page 138 of *Finite Elements Methods in Mechanics* book by Prof. M.R. Eslami.

$$
\begin{aligned}
2x_1 + 5x_2 - 3x_3 + x_4 &= 6 \\
5x_1 + x_2 - 2x_3 - 5x_4 &= 15 \\
-3x_1 - 2x_2 + 6x_3 + 2x_4 &= 12 \\
x_1 - 5x_2 + 2x_3 + 3x_4 &= 7
\end{aligned}
$$

Solution: Following the Gauss elimination method, the first equation is multiplied by $-5/2$ and the result is added to the second equation, then multiplied by 3/2 and added to the third equation, and finally, multiplied by $-1/2$ and added to the fourth equation, to give

$$
\begin{aligned}
2x_1 + 5x_2 - 3x_3 + x_4 &= 6 \\
0 - \frac{23}{2}x_2 + \frac{11}{2}x_3 - \frac{15}{2}x_4 &= 0 \\
0 + \frac{11}{2}x_2 + \frac{3}{2}x_3 + \frac{7}{2}x_4 &= 21 \\
0 - \frac{15}{2}x_2 + \frac{7}{2}x_3 + \frac{5}{2}x_4 &= 4.
\end{aligned}
$$

Now, the second equation is multiplied by 11/23 and is added to the third equation, and then multiplied by $-15/23$ and added to the fourth equation, to give

$$
\begin{aligned}
2x_1 + 5x_2 - 3x_3 + x_4 &= 6 \\
-\frac{23}{2}x_2 + \frac{11}{2}x_3 - \frac{15}{2}x_4 &= 0 \\
+\frac{95}{23}x_3 - \frac{2}{23}x_4 &= 21 \\
-\frac{2}{23}x_3 + \frac{170}{23}x_4 &= 4.
\end{aligned}
$$

Finally, the third equation is multiplied by 2/95 and is added to the fourth Equation, yielding

$$
\begin{aligned}
2x_1 + 5x_2 - 3x_3 + x_4 &= 6 \\
-\frac{23}{2}x_2 + \frac{11}{2}x_3 - \frac{15}{2}x_4 &= 0 \\
+\frac{95}{23}x_3 - \frac{2}{23}x_4 &= 21 \\
+\frac{16146}{2185}x_4 &= \frac{422}{95}.
\end{aligned}
$$

The backward solution from the fourth equation for x_4 is

$$x_4 = \frac{211}{351} = 0.601.$$

This value of x_4 is set in the third equation to obtain x_3 as

$$\frac{95}{23} - \frac{2}{23}\frac{211}{23} = 21 \rightarrow x_3 = 5.097$$

Substituting x_4 and x_3 in the second equation, x_2 is obtained as

$$-\frac{23}{2}x_2 + 107.8 - 99.9 = 0 \rightarrow x_2 = 2.046$$

And, finally, substituting x_4, x_3, and x_2 in the first equation, x_1 is calculated as

$$2x_1 + 5(2.046) - 3(5.097) + 0.601 = 6 \rightarrow x_1 = 5.23$$

For C++ code, the results are as follows,

```
Enter the number of unknowns: 4
Enter the coefficients of the Augmented Matrix:
a[1][1] = 2
a[1][2] = 5
a[1][3] = -3
a[1][4] = 1
a[1][5] = 6
a[2][1] = 5
a[2][2] = 1
a[2][3] = -2
a[2][4] = -5
a[2][5] = 15
a[3][1] = -3
a[3][2] = -2
a[3][3] = 6
a[3][4] = 2
a[3][5] = 12
a[4][1] = 1
a[4][2] = -5
a[4][3] = 2
a[4][4] = 3
a[4][5] = 7

Solution:
x[1] = 5.23077
x[2] = 2.04558
x[3] = 5.09687
x[4] = 0.60114
```

It can be seen that the results are as same as results presented in page 139 of *Finite Elements Methods in Mechanics* book by Prof. M.R. Eslami.

Now, for python we have the following procedure. The matrix of the problem is as follows,

```
#input matrix
augmented_matrix = [
    [2, 5, -3, 1,6],
    [5, 1, -2, -5,15],
    [-3, -2, 6, 2, 12],
    [1, -5, 2, 3, 7]
]
```

The results can be obtained as,

```
Solution:
x[1] = 5.230769
x[2] = 2.045584
x[3] = 5.096866
x[4] = 0.601140
```

which demonstrates its efficiency.

The ***second*** example is

$$
\begin{aligned}
&-2x_1 + 5x_2 - 3x_3 + x_4 - 9x_5 + 5x_6 = 5\\
&4x_1 + x_2 - 2x_3 - 5x_4 + 6x_5 + 3x_6 = 6\\
&4x_1 + 4x_2 + 6x_3 + 2x_4 - x_5 + x_6 = 12\\
&\frac{1}{2}x_1 + 5x_2 + 2x_3 + 3x_4 + x_5 + 5x_6 = 1\\
&6x_1 + 2x_2 + 4x_3 + 5x_4 - 6x_5 - 8x_6 = 13\\
&x_1 - 3x_2 + 8x_3 - 3x_4 - x_5 + 2x_6 = 5
\end{aligned}
$$

For C++ program we have,

```
Enter the number of unknowns: 6
Enter the coefficients of the Augmented Matrix:
a[1][1] = -2
a[1][2] = 5
a[1][3] = -3
a[1][4] = 1
a[1][5] = -9
a[1][6] = 5
a[1][7] = 5
a[2][1] = 4
a[2][2] = 1
a[2][3] = -2
```

a[2][4] = -5
a[2][5] = 6
a[2][6] = 3
a[2][7] = 6
a[3][1] = 4
a[3][2] = 4
a[3][3] = 6
a[3][4] = 2
a[3][5] = -1
a[3][6] = 1
a[3][7] = 12
a[4][1] = 0.5
a[4][2] = 5
a[4][3] = 2
a[4][4] = 3
a[4][5] = 1
a[4][6] = 5
a[4][7] = 1
a[5][1] = 6
a[5][2] = 2
a[5][3] = 4
a[5][4] = 5
a[5][5] = -6
a[5][6] = -8
a[5][7] = 13
a[6][1] = 1
a[6][2] = -3
a[6][3] = 8
a[6][4] = -3
a[6][5] = -1
a[6][6] = 2
a[6][7] = 5

Solution:
x[1] = 0.181972
x[2] = 2.69366
x[3] = 1.02178
x[4] = -2.35383
x[5] = -0.485615
x[6] = -1.41115

For Python we have,

```
#input matrix
augmented_matrix = [
    [ -2, 5, -3, 1, -9, 5, 5],
    [4, 1, -2, -5,6, 3, 6],
    [4, 4, 6, 2, -1, 1, 12],
    [0.5, 5, 2, 3, 1, 5, 1],
    [6, 2, 4, 5, -6, -8, 13],
    [1, -3, 8, -3, -1, 2, 5],
]
```

The results can be obtained as,

Solution:

x[1] = 0.181972

x[2] = 2.693662

x[3] = 1.021776

x[4] = -2.353829

x[5] = -0.485615

x[6] = -1.411149

which demonstrates its efficiency as results are the same as those obtained from C++ program.

The ***third*** example is ten equations with ten unknowns.

$$-2x_1 + 5x_2 - 3x_3 + x_4 - 9x_5 + 5x_6 + x_7 + 2x_8 + 3x_9 + 4x_{10} = 16$$
$$4x_1 + x_2 - 2x_3 - 5x_4 + 6x_5 + 3x_6 - x_7 - x_8 - 3x_9 + x_{10} = 20$$
$$4x_1 + 4x_2 + 6x_3 + 2x_4 - x_5 + x_6 + 3x_7 - x_8 + 3x_9 + 5x_{10} = 24$$
$$\frac{1}{2}x_1 + 5x_2 + 2x_3 + 3x_4 + x_5 + 5x_6 + x_7 + 3x_8 + x_9 - 5x_{10} = 28$$
$$6x_1 + 2x_2 + 4x_3 + 5x_4 - 6x_5 - 8x_6 + 6x_7 + x_8 + 3x_9 + 4x_{10} = 32$$
$$x_1 - 3x_2 + 8x_3 - 3x_4 - x_5 + 2x_6 + x_7 - 8x_8 + 8x_9 - 3x_{10} = 36$$
$$2x_1 + 6x_2 + 2x_3 + 3x_4 - 6x_5 + 6x_6 + 1.5x_7 - 3x_8 + 4x_9 - x_{10} = 15$$
$$3x_1 + 6x_2 + 2x_3 - x_4 + x_5 + 5x_6 + 4x_7 + 2x_8 + x_9 + 3x_{10} = 18$$
$$4x_1 + 6x_2 + 10x_3 - 3x_4 + 2.5x_5 + 2.5x_6 + 1.75x_7 - 0.5x_8 + 6x_9 + 2.5x_{10} = 20.25$$
$$5x_1 - 3.75x_2 + 10x_3 + 3x_4 + x_5 - 2x_6 + 6x_7 - 3.5x_8 + 7x_9 - 4x_{10} = 30$$

Enter the number of unknowns: 10
Enter the coefficients of the Augmented Matrix:
a[1][1] = -2
a[1][2] = 5
a[1][3] = -3
a[1][4] = 1
a[1][5] = -9
a[1][6] = 5
a[1][7] = 1
a[1][8] = 2
a[1][9] = 3
a[1][10] = 4
a[1][11] = 16
a[2][1] = 4
a[2][2] = 1
a[2][3] = -2
a[2][4] = -5
a[2][5] = 6
a[2][6] = 3
a[2][7] = -1
a[2][8] = -1
a[2][9] = -3
a[2][10] = 1
a[2][11] = 20

a[3][1] = 4
a[3][2] = 4
a[3][3] = 6
a[3][4] = 2
a[3][5] = -1
a[3][6] = 1
a[3][7] = 3
a[3][8] = -1
a[3][9] = 3
a[3][10] = 5
a[3][11] = 24
a[4][1] = 0.5
a[4][2] = 5
a[4][3] = 2
a[4][4] = 3
a[4][5] = 1
a[4][6] = 5
a[4][7] = 1
a[4][8] = 3
a[4][9] = 1
a[4][10] = -5
a[4][11] = 28
a[5][1] = 6
a[5][2] = 2
a[5][3] = 4
a[5][4] = 5
a[5][5] = -6
a[5][6] = -8
a[5][7] = 6
a[5][8] = 1
a[5][9] = 3
a[5][10] = 4
a[5][11] = 32
a[6][1] = 1
a[6][2] = -3
a[6][3] = 8
a[6][4] = -3
a[6][5] = -1
a[6][6] = 2
a[6][7] = 1
a[6][8] = -8
a[6][9] = 8
a[6][10] = -3
a[6][11] = 36
a[7][1] = 2
a[7][2] = 6
a[7][3] = 2
a[7][4] = 3
a[7][5] = -6
a[7][6] = 6
a[7][7] = 1.5
a[7][8] = -3
a[7][9] = 4

```
a[7][10] = -1
a[7][11] = 15
a[8][1] = 3
a[8][2] = 6
a[8][3] = 2
a[8][4] = -1
a[8][5] = 1
a[8][6] = 5
a[8][7] = 4
a[8][8] = 2
a[8][9] = 1
a[8][10] = 3
a[8][11] = 18
a[9][1] = 4
a[9][2] = 6
a[9][3] = 10
a[9][4] = -3
a[9][5] = 2.5
a[9][6] = 2.5
a[9][7] = 1.75
a[9][8] = -0.5
a[9][9] = 6
a[9][10] = 2.5
a[9][11] = 20.25
a[10][1] = 5
a[10][2] = -3.75
a[10][3] = 10
a[10][4] = 3
a[10][5] = 1
a[10][6] = -2
a[10][7] = 6
a[10][8] = -3.5
a[10][9] = 7
a[10][10] = -4
a[10][11] = 30

Solution:
x[1] = 53.7500
x[2] = -75.9600
x[3] = 152.6579
x[4] = -78.2219
x[5] = -146.5661
x[6] = 61.2794
x[7] = -19.0426
x[8] = 66.0540
x[9] = -190.0005
x[10] = -32.2651
```

For Python we have,

```python
#input matrix
augmented_matrix = [
    [ -2, 5, -3, 1, -9, 5, 1, 2, 3, 4, 16],
    [4, 1, -2, -5, 6, 3, -1, -1, -3, 1, 20],
```

```
    [4, 4, 6, 2, -1, 1, 3, -1, 3, 5, 24],
    [0.5, 5, 2, 3, 1, 5, 1, 3, 1, -5, 28],
    [6, 2, 4, 5, -6, -8, 6, 1, 3, 4, 32],
    [1, -3, 8, -3, -1, 2, 1, -8, 8, -3, 36],
    [2, 6, 2, 3, -6, 6, 1.5, -3, 4, -1, 15],
    [3, 6, 2, -1, 1, 5, 4,2, 1, 3, 18],
    [4, 6, 10, -3, 2.5, 2.5, 1.75, -0.5, 6, 2.5, 20.25],
    [5, -3.75, 10, 3, 1, -2, 6, -3.5, 7, -4, 30],
]
```

The results can be obtained as,

Solution:

x[1] = 53.749955

x[2] = -75.960002

x[3] = 152.657882

x[4] = -78.221895

x[5] = -146.566125

x[6] = 61.279368

x[7] = -19.042616

x[8] = 66.053997

x[9] = -190.000525

x[10] = -32.265077

which demonstrates its efficiency as results are the same as those obtained from C++ program.

References

- *Atkinson, Kendall A. (1989), An Introduction to Numerical Analysis (2nd ed.), New York: John Wiley & Sons, ISBN 978-0471624899*.
- *Bolch, Gunter; Greiner, Stefan; de Meer, Hermann; Trivedi, Kishor S. (2006), Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications (2nd ed.), Wiley-Interscience, ISBN 978-0-471-79156-0*.
- *Calinger, Ronald (1999), A Contextual History of Mathematics, Prentice Hall, ISBN 978-0-02-318285-3*.
- Carnahan, B., Luther, H. A., & Wilkes, J. O. (1969). *Applied numerical methods (Vol. 2). New York: Wiley.*
- **Eslami, M. R. (2014).** ***Finite elements methods in mechanics. Switzerland: Springer International Publishing.***
- *Farebrother, R.W. (1988), Linear Least Squares Computations, STATISTICS: Textbooks and Monographs, Marcel Dekker, ISBN 978-0-8247-7661-9*.
- *Lauritzen, Niels, Undergraduate Convexity: From Fourier and Motzkin to Kuhn and Tucker*.
- *Golub, Gene H.; Van Loan, Charles F. (1996), Matrix Computations (3rd ed.), Johns Hopkins, ISBN 978-0-8018-5414-9*.