Sample programs in $C^{++}$ for matrix computations in max plus algebra

Mihai Ivan and Gheorghe Ivan

Abstract. The main purpose of this paper is to propose five programs in $C^{++}$ for matrix computations and solving recurrent equations systems with entries in max plus algebra.

1 Introduction

Idempotent mathematics is based on replacing the usual arithmetic operations with a new set of basic operations, that is on replacing numerical fields by idempotent semirings. Exotic semirings such as the max-plus algebra $\mathbb{R}_{\max}$ have been introduced in connection with various fields: graph theory, Markov decision processes, discrete event systems theory, see [1], [3].

The paper is organized as follows. The semiring of matrices with entries in max plus algebra is presented in Section 2. In Section 3 we give five programs in language $C^{++}$ for matrix computations in max plus algebra.

2 Semirings. Matrices over max plus algebra

We start this section by recalling of some necessary backgrounds on semirings for our purposes (see [3],[2] and references therein for more details).

Let $S$ be a nonempty set endowed with two binary operations, addition (denoted with $\oplus$) and multiplication (denoted with $\otimes$). The algebraic structure $(S,\oplus,\otimes,\varepsilon,e)$ is a semiring, if it fulfills the following conditions:

1. $(S,\oplus,\varepsilon)$ is a commutative monoid with $\varepsilon$ as the neutral element for $\oplus$;
2. $(S,\otimes,e)$ is a monoid with $e$ as the identity element for $\otimes$;
3. $\otimes$ distributes over $\oplus$;
4. $\varepsilon$ is an absorbing element for $\otimes$, that is $a \otimes \varepsilon = \varepsilon \otimes a = \varepsilon$, $\forall a \in S$.

A semiring where addition is idempotent (that is, $a \oplus a = a$, $\forall a \in S$) is called an idempotent semiring. If $\otimes$ is commutative, we say that $S$ is a commutative semiring.

Let $(S,\oplus,\otimes,\varepsilon,e)$ be an (idempotent) semiring. For each pair of positive integer $(m,n)$, let $M_{m\times n}(S)$ be denote the set of $m \times n$ matrices with entries in $S$. The operations $\oplus$ and $\otimes$ on $S$ induce corresponding operations on $M_{m\times n}(S)$ in the obvious way. Indeed, if $A=(A_{ij}), B=(B_{ij}) \in M_{m\times n}(S)$ then we have:

$$A \oplus B = ((A \oplus B)_{ij}) \quad \text{where} \quad (A \oplus B)_{ij} := A_{ij} \oplus B_{ij}. \quad (2.1)$$

If $A=(A_{ij}) \in M_{m\times n}(S)$ and $B=(B_{jk}) \in M_{n\times p}(S)$ then we have:

---

$^{1}$AMS classification: 15A80, 68-04.
Key words and phrases: idempotent semiring, max plus algebra.
\[ A \otimes B = ((A \otimes B)_{ik}), \quad i = 1, m, \quad k = 1, p \quad \text{where} \quad (A \otimes B)_{ik} := \bigoplus_{j=1}^{n} A_{ij} \otimes B_{jk}. \quad (2.2) \]

The product of a matrix \( A = (A_{ij}) \in M_{m \times n}(S) \) with a scalar \( \alpha \in S \) is given by:

\[ \alpha \otimes A = ((\alpha \otimes A)_{ij}) \quad \text{where} \quad (\alpha \otimes A)_{ij} := \alpha \otimes A_{ij}. \quad (2.3) \]

The set \( M_{n \times n}(S) \) contains two special matrices with entries in \( S \), namely the zero matrix \( O_{\oplus n} \), which has all its entries equal to \( \varepsilon \), and the identity matrix \( I_{\otimes n} \), which has the diagonal entries equal to \( e \) and the other entries equal to \( \varepsilon \).

It is easy to check that the following proposition holds.

**Proposition 2.1.** \((M_{n \times n}(S), \oplus, \otimes, O_{\oplus n}, I_{\otimes n})\) is an idempotent semiring, where the operations \( \oplus \) and \( \otimes \) are given in (2.1) and (2.2). \( \Box \)

We call \((M_{n \times n}(S), \oplus, \otimes, O_{\oplus n}, I_{\otimes n})\) the semiring of \( n \times n \) matrices with entries in \( S \). In particular, if \( S := S_{\max} = (R \cup \{-\infty\}, \oplus := \max, \otimes := +, \varepsilon := -\infty, e := 0 \), is called the semiring of \( n \times n \) matrices over \( R_{\max} \).

When \( S = R_{\max} \), the operations \( \oplus \) and \( \otimes \) given in (2.1) and (2.2), becomes:

\[ (A \oplus B)_{ij} := \max\{A_{ij}, B_{ij}\} \quad \text{and} \quad (A \otimes B)_{ik} := \max_{1 \leq k \leq n}\{A_{ij} + B_{jk}\}. \quad (2.4) \]

The operation \( \otimes \) on \( M_{m \times n}(R_{\max}) \) given in (2.3) becomes:

\[ \alpha \otimes A = ((\alpha \otimes A)_{ij}) \quad \text{where} \quad (\alpha \otimes A)_{ij} := \alpha + A_{ij}. \quad (2.5) \]

### 3 Five programs in C++

In this section we give programs written in the language C++ for the basic operations with matrices over \( R_{\max} \) and for solving a recurrent linear system:

1. the sum of two matrices \( A, B \in M_{m \times n}(R_{\max}); \)
2. the product of two matrices \( A \in M_{m \times n}(R_{\max}) \) and \( B \in M_{n \times p}(R_{\max}); \)
3. the product of a matrix \( A \in M_{m \times n}(R_{\max}) \) with a scalar \( \alpha \in R_{\max}; \)
4. the power of a matrix \( A \in M_{n \times n}(R_{\max}); \)
5. the solving a linear system of the form:

\[ X(k + 1) = A \otimes X(k), \quad k \geq 0, \]

where \( A \in M_{n \times n}(R_{\max}) \) and \( X(k) \in M(n, 1; R_{\max}). \)

The principal program is constituted from the following lines.

```c++
using System;
using System.Collections.Generic;
```
Sample programs in C++ for matrix computations in max plus algebra

using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
namespace Operations_with_matrices
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        public int[,] A5 = new int[50, 50];
        #region
        sum of two matrices in max plus algebra
        #endregion
        private void initMatrixA()
        {
            int column = 0;
            column = Convert.ToInt16(textColumn.Text);
            dataGridViewA.ColumnCount = column;
            dataGridViewA.AllowUserToOrderColumns = false;
            dataGridViewA.AllowUserToAddRows = false;
            dataGridViewA.Enabled = true;
            dataGridViewA.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.DisplayedCellsExceptHeaders;
            dataGridViewA.ColumnHeaderBorderStyle = DataGridViewHeaderBorderStyle.Raised;
            dataGridViewA.CellBorderStyle = DataGridViewCellBorderStyle.Single;
            dataGridViewA.GridColor = Color.DodgerBlue;
            dataGridViewA.ColumnHeadersVisible = false;
            dataGridViewA.RowHeadersVisible = false;
            dataGridViewA.BackgroundColor = Color.WhiteSmoke;
            dataGridViewA.BorderStyle = BorderStyle.None;
            dataGridViewA.AllowUserToResizeColumns = true;
            DataGridViewCellStyle style = new DataGridViewCellStyle();
            style.Format = "N0";
            style.NullValue = null;
            DataGridViewCellStyle columnHeaderStyle = new DataGridViewCellStyle();
            columnHeaderStyle.BackColor = Color.AntiqueWhite;
            columnHeaderStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
            dataGridViewA.ColumnHeadersDefaultCellStyle = columnHeaderStyle;
            DataGridViewTextBoxColumn CellR = new DataGridViewTextBoxColumn();
            dataGridViewA_vertically_organize_TheLabels.MyTextBox style = new DataGridViewTextBoxColumn();
            style.Format = "N0";
            style.NullValue = null;
            DataGridViewCellStyle columnHeaderStyle = new DataGridViewCellStyle();
            columnHeaderStyle.BackColor = Color.AntiqueWhite;
            columnHeaderStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
            dataGridViewA.ColumnHeadersDefaultCellStyle = columnHeaderStyle;
            dataGridViewA_vertically_organize_TheLabels.MyTextBox = new DataGridViewTextBoxColumn();
        }
Mihai Ivan, Gheorghe Ivan

CellR.Style.Alignment = DataGridViewContentAlignment.MiddleCenter;
DataGridViewCell CellL = new DataGridViewTextBoxCell();
CellL.Style.Alignment = DataGridViewContentAlignment.MiddleLeft;

int line = 0;
line = Convert.ToInt16(textLine.Text);
dataGridA.RowCount = line;
for (int i = 0; i < column; i++)
{
    dataGridA.Columns[i].Name = "C" + (i + 1);
    dataGridA.Columns[i].CellTemplate = CellR;
    dataGridA.Columns[i].Width = 30;
    dataGridA.Columns[i].DefaultCellStyle = style;
}

private void initMatrixB()
{
    int column = 0;
    column = Convert.ToInt16(textColumn.Text);
dataGridB.ColumnCount = column;
dataGridB.AllowUserToOrderColumns = false;
dataGridB.AllowUserToAddRows = false;
dataGridB.Enabled = true;
dataGridB.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.DisplayedCellsExceptHeaders;
dataGridB.ColumnHeaderBorderStyle = DataGridViewHeaderBorderStyle.Raised;
dataGridB.ColumnHeaderDefaultCellStyle = columnHeaderStyle;
DataGridViewCellStyle columnHeaderStyle = new DataGridViewCellStyle();
    columnHeaderStyle.BackColor = Color.AntiqueWhite;
    columnHeaderStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
dataGridB.ColumnHeaderDefaultCellStyle = columnHeaderStyle;

    DataGridViewCell CellR = new DataGridViewTextBoxCell();
    CellR.Style.Alignment = DataGridViewContentAlignment.MiddleRight;
    DataGridViewCell CellL = new DataGridViewTextBoxCell();
    CellL.Style.Alignment = DataGridViewContentAlignment.MiddleLeft;
Sample programs in C++ for matrix computations in max plus algebra

```csharp
int line = 0;
line = Convert.ToInt16(textLine.Text);
dataGridB.RowCount = line;
for (int i = 0; i < column; i++)
{
    dataGridB.Columns[i].Name = "C" + (i + 1);
dataGridB.Columns[i].CellTemplate = CellR;
dataGridB.Columns[i].Width = 30;
}

private void initMatrixOfAdditionResults()
{
    int column = 0;
    column = Convert.ToInt16(textColumn.Text);
dataGridResAddition.ColumnCount = column;
dataGridResAddition.AllowUserToOrderColumns = false;
dataGridResAddition.AllowUserToAddRows = false;
dataGridResAddition.Enabled = true;
dataGridResAddition.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.DisplayedCellsExceptHeaders;
dataGridResAddition.ColumnHeadersBorderStyle = DataGridViewHeaderBorderStyle.Raised;
dataGridResAddition.CellBorderStyle = DataGridViewCellBorderStyle.Single;
dataGridResAddition.GridColor = Color.DodgerBlue;
dataGridResAddition.ColumnHeadersVisible = false;
dataGridResAddition.RowHeadersVisible = false;
dataGridResAddition.BackgroundColor = Color.WhiteSmoke;
dataGridResAddition.BorderStyle = BorderStyle.None;
dataGridResAddition.AllowUserToResizeColumns = true;

DataGridViewCellStyle columnHeaderStyle = new DataGridViewCellStyle();
columnHeaderStyle.BackColor = Color.AntiqueWhite;
columnHeaderStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
dataGridResAddition.ColumnHeadersDefaultCellStyle = columnHeaderStyle;
DataGridViewCell CellR = new DataGridViewTextBoxCell();
CellR.Style.Alignment = DataGridViewContentAlignment.MiddleRight;
DataGridViewCell CellL = new DataGridViewTextBoxCell();
CellL.Style.Alignment = DataGridViewContentAlignment.MiddleLeft;
```

int line = 0;
line = Convert.ToInt16(textLine.Text);
dataGridResAddition.RowCount = line;
for (int i = 0; i < column; i++)

private void btGenerate_Click(object sender, EventArgs e)
{
    if (textLine.Text != "" && textColumn.Text != "")
    {
        initMatrixA();
        initMatrixB();
    }
    else
    {
        if (textLine.Text == "" && textColumn.Text == "")
            MessageBox.Show("Introduce number of lines and number of columns!");
        else if (textLine.Text == "")
            MessageBox.Show("Introduce number of lines!");
        else if (textColumns.Text == "")
            MessageBox.Show("Introduce number of columns!");
    }
}

private void btComputeSumClick(object sender, EventArgs e)
{
    initMatrixAdditionResult();

    int line = 0;
    int column = 0;
    column = Convert.ToInt16(textColumn.Text);
    line = Convert.ToInt16(textLine.Text);
    int[,] A = new int[line, column];
    int[,] B = new int[line, column];
    int[,] ResAddition = new int[line, column];

    # region reading matrix A
    for (int i = 0; i < line; i++)
    {
        for (int j = 0; j < column; j++)
        {
            if (dataGridA.Rows[i].Cells[j].Value.ToString() == "E")
                A[i, j] = Int32.MinValue;
            else
            
...
Sample programs in C++ for matrix computations in max plus algebra

A[i, j] = Convert.ToInt16(dataGridA.Rows[i].Cells[j].Value.ToString());
if (dataGridB.Rows[i].Cells[j].Value.ToString() == "E")
    B[i, j] = Int32.MinValue;
else
    B[i, j] = Convert.ToInt16(dataGridB.Rows[i].Cells[j].Value.ToString());

# endregion
# region computation of sum
for (int i = 0; i < line; i++)
{
    for (int j = 0; j < column; j++)
    {
        if (A[i, j] > B[i, j])
            ResAddition[i, j] = A[i, j];
        else
            ResAddition[i, j] = B[i, j];
    }
}
# endregion
# region display sum
for (int i = 0; i < line; i++)
{
    for (int j = 0; j < column; j++)
    {
        if (ResAddition[i, j] == Int32.MinValue)
            dataGridResAddition.Rows[i].Cells[j].Value = "E";
        else
            dataGridResAddition.Rows[i].Cells[j].Value = ResAddition[i, j];
    }
}
#endregion

public void ResetValuesForAddition()
{
    textColumn.ResetText();
textLine.ResetText();
dataGridA.Rows.Clear();
dataGridB.Rows.Clear();
dataGridResAddition.Rows.Clear();
}

private void btReset_Click(object sender, EventArgs e)
private void initMatrixA2()
{
    int column = 0;
    column = Convert.ToInt16(textColumnA2.Text);
    dataGridA2.ColumnCount = column;
    dataGridA2.AllowUserToOrderColumns = false;
    dataGridA2.AllowUserToAddRows = false;
    dataGridA2.Enabled = true;
    dataGridA2.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.DisplayedCellsExceptHeaders;
    dataGridA2.ColumnHeaderBorderStyle = DataGridViewHeaderBorderStyle.Raised;
    dataGridA2.CellBorderStyle = DataGridViewCellBorderStyle.Single;
    dataGridA2.GridColor = Color.DodgerBlue;
    dataGridA2.ColumnHeaderVisible = false;
    dataGridA2.RowHeadersVisible = false;
    dataGridA2.BackgroundColor = Color.WhiteSmoke;
    dataGridA2.BorderStyle = BorderStyle.None;
    dataGridA2.AllowUserToResizeColumns = true;
    DataGridViewCellStyle columnHeaderStyle = new DataGridViewCellStyle();
    columnHeaderStyle.BackColor = Color.AntiqueWhite;
    columnHeaderStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridA2.ColumnHeadersDefaultCellStyle = columnHeaderStyle;
    DataGridViewTextBoxCell CellR = new DataGridViewTextBoxCell();
    CellR.Style.Alignment = DataGridViewContentAlignment.MiddleRight;
    DataGridViewTextBoxCell CellL = new DataGridViewTextBoxCell();
    CellL.Style.Alignment = DataGridViewContentAlignment.MiddleLeft;
    int line = 0;
    line = Convert.ToInt16(textLineA2.Text);
    dataGridA2.RowCount = line;
    for (int i = 0; i < column; i++)
    {
        dataGridA2.Columns[i].CellTemplate = CellR;
        dataGridA2.Columns[i].Width = 30;
    }
}
private void initMatrixB2()
{
    int column = 0;
    column = Convert.ToInt16(textcolumnB2.Text);
    dataGridViewB2.ColumnCount = column;
    dataGridViewB2.AllowUserToOrderColumns = false;
    dataGridViewB2.AllowUserToAddRows = false;
    dataGridViewB2.Enabled = true;
    dataGridViewB2.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.DisplayedCellsExceptHeaders;
    dataGridViewB2.ColumnHeadersBorderStyle = DataGridViewHeaderBorderStyle.Raised;
    dataGridViewB2.BackgroundColor = Color.WhiteSmoke;
    dataGridViewB2.GridColor = Color.DodgerBlue;
    dataGridViewB2.RowHeadersVisible = false;
    dataGridViewB2.RowHeadersVisible = false;
    dataGridViewB2.BackgroundColor = Color.WhiteSmoke;
    dataGridViewB2.BorderStyle = BorderStyle.None;
    dataGridViewB2.AllowUserToResizeColumns = true;
    DataGridViewCellStyle columnHeaderStyle = new DataGridViewCellStyle();
    columnHeaderStyle.BackColor = Color.AntiqueWhite;
    columnHeaderStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewB2.RowHeadersDefaultCellStyle = columnHeaderStyle;
    DataGridViewCell CellR = new DataGridViewTextBoxCell();
    CellR.Style.Alignment = DataGridViewContentAlignment.MiddleRight;
    DataGridViewCell CellL = new DataGridViewTextBoxCell();
    CellL.Style.Alignment = DataGridViewContentAlignment.MiddleLeft;
    line = 0;
    line = Convert.ToInt16(textLineB2.Text);
    dataGridViewB2.RowCount = line;
    for (int i = 0; i < column; i++)
    {
        dataGridViewB2.Columns[i].Name = "C" + (i + 1);
        dataGridViewB2.Columns[i].CellTemplate = CellR;
        dataGridViewB2.Columns[i].Width = 30;
    }
}

private void initMatrixProduct()
{
    int column = 0;
    column = Convert.ToInt16(textcolumnB2.Text);
dataGridProduct.ColumnCount = column;
dataGridProduct.AllowUserToOrderColumns = false;
dataGridProduct.AllowUserToAddRows = false;
dataGridProduct.Enabled = true;
dataGridProduct.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.DisplayedCellsExceptHeaders;
dataGridProduct.ColumnHeaderBorderStyle = DataGridViewHeaderBorderStyle.Raised;
dataGridProduct.CellBorderStyle = DataGridViewCellBorderStyle.Single;
dataGridProduct.GridColumnColor = Color.DodgerBlue;
dataGridProduct.ColumnHeaderVisible = false;
dataGridProduct.RowHeadersVisible = false;
dataGridProduct.BackgroundColor = Color.WhiteSmoke;
dataGridProduct.BorderStyle = BorderStyle.None;
dataGridProduct.AllowUserToResizeColumns = true;
DataGridViewCellStyle columnHeaderStyle = new DataGridViewCellStyle();
columnHeaderStyle.BackColor = Color.AntiqueWhite;
columnHeaderStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
dataGridProduct.ColumnHeaderDefaultCellStyle = columnHeaderStyle;
DataGridViewCell CellR = new DataGridViewTextBoxCell();
CellR.Style.Alignment = DataGridViewContentAlignment.MiddleRight;
DataGridViewCell CellL = new DataGridViewTextBoxCell();
CellL.Style.Alignment = DataGridViewContentAlignment.MiddleLeft;
int line = 0;
line = Convert.ToInt16(textLineA2.Text);
dataGridProduct.RowCount = line;
for (int i = 0; i < column; i++)
{
    dataGridViewColumn[i].Name = "C" + (i + 1);
dataGridProduct.Columns[i].CellTemplate = CellR;
dataGridProduct.Columns[i].Width = 30;
}
private void textColumnA2_Leave(object sender, EventArgs e)
{
    textLineB2.Text = textColumnA2.Text;
}
private void btGenerare2_Click(object sender, EventArgs e)
{
    if (textLineA2.Text != "" && textColumnA2.Text != "" && textLineB2.Text != "" && textcolumnB2.Text != "")
{ initMatrixA2(); 
    initMatrixB2(); 
}
else
{
    if (textLineA2.Text == "" && textColumnA2.Text == "" && textLineB2.Text == "" && textColumnB2.Text == "")
        MessageBox.Show("Introduce number of lines and number of columns!");
    else
    {
        if (textLineA2.Text == "" || textLineB2.Text == "")
            MessageBox.Show("Introduce number of lines for matrix A!");
        else if (textColumnA2.Text == "")
            MessageBox.Show("Introduce number of columns for matrix A!");
        else if (textColumnB2.Text == "")
            MessageBox.Show("Introduce number of columns for matrix B!");
    }
}
private void btComputationProduct_Click(object sender, EventArgs e)
{
    initMatrixProduct();
    int lineA = 0; int lineB = 0;
    int columnA = 0; int columnB = 0;
    columnA = Convert.ToInt16(textColumnA2.Text);
    lineA = Convert.ToInt16(textLineA2.Text);
    columnB = Convert.ToInt16(textColumnB2.Text);
    lineB = Convert.ToInt16(textLineB2.Text);
    int[,] A2 = new int[lineA, columnA];
    int[,] B2 = new int[lineB, columnB];
    int[,] Product = new int[lineA, columnB];
    int[,] Sum = new int[lineA, columnB];
    int k;
    # region reading matrices A and B
    for (int i = 0; i < lineA; i++)
    {
        for (int j = 0; j < columnA; j++)
        {
            if (dataGridA2.Rows[i].Cells[j].Value.ToString() == "E")
                A2[i, j] = Int32.MinValue;
        }
    }
}
else
    A2[i, j] = Convert.ToInt16(dataGridA2.Rows[i].Cells[j].Value.ToString());
}

for (int i = 0; i < lineB; i++)
{
    for (int j = 0; j < columnB; j++)
    {
        if (dataGridB2.Rows[i].Cells[j].Value.ToString() == "E")
            B2[i, j] = Int32.MinValue;
        else
            B2[i, j] = Convert.ToInt16(dataGridB2.Rows[i].Cells[j].Value.ToString());
    }
}

# endregion
# region computation product
for (int i = 0; i < lineA; i++)
{
    for (int j = 0; j < columnB; j++)
    {
        Product[i, j] = Int32.MinValue;
        for (k = 0; k < lineB; k++)
        {
            if (A2[i, k] == Int32.MinValue || B2[k, j] == Int32.MinValue)
                Sum[i, j] = Int32.MinValue;
            else
                Sum[i, j] = A2[i, k] + B2[k, j];
        }
    }
}

#endregion
# region display product
for (int i = 0; i < lineA; i++)
{
    for (int j = 0; j < columnB; j++)
    {
        if (Product[i, j] == Int32.MinValue)
            dataGridProduct.Rows[i].Cells[j].Value = "E";
    }
}
else
    dataGridProduct.Rows[i].Cells[j].Value = Product[i, j];
}

# endregion
}

public void Reset_Values_for_Multiplication()
{
    textColumnA2.ResetText();
    textLineA2.ResetText();
textcolumnB2.ResetText();
textLineB2.ResetText();
dataGridA2.Rows.Clear();
dataGridB2.Rows.Clear();
dataGridProduct.Rows.Clear();
}

private void btResetMultiplication_Click(object sender, EventArgs e)
{
    Reset_Values_for_Multiplication();
}

#region scalar product in max plus algebra

private void initMatrixA3()
{
    int column = 0;
    column = Convert.ToInt16(textcolumnA3.Text);
dataGridA3.ColumnCount = column;
dataGridA3.AllowUserToOrderColumns = false;
dataGridA3.AllowUserToAddRows = false;
dataGridA3.Enabled = true;
dataGridA3.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.DisplayedCellsExceptHeaders;
dataGridA3.ColumnHeaderBorderStyle = DataGridViewHeaderBorderStyle.Raised;
dataGridA3.CellBorderStyle = DataGridViewCellBorderStyle.Single;
dataGridA3.BackgroundColor = Color.DodgerBlue;
dataGridA3.RowHeadersVisible = false;
dataGridA3.BorderStyle = BorderStyle.None;
dataGridA3.AllowUserToResizeColumns = true;
DataGridViewCellStyle columnHeaderStyle = new DataGridViewCellStyle();

DataGridViewAutoSizeRowsMode.MinValue

}
columnHeaderStyle.BackColor = Color.AntiqueWhite;
columnHeaderStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
dataGridViewA3.ColumnHeadersDefaultCellStyle = columnHeaderStyle;
DataGridViewCell CellR = new DataGridViewTextBoxCell();
CellR.Style.Alignment = DataGridViewContentAlignment.MiddleRight;
DataGridViewCell CellL = new DataGridViewTextBoxCell();
CellL.Style.Alignment = DataGridViewContentAlignment.MiddleLeft;
int line = 0;
line = Convert.ToInt16(textLineA3.Text);
dataGridViewA3.RowCount = line;
for (int i = 0; i < column; i++)
{
    dataGridViewA3.Columns[i].Name = "C" + (i + 1);
    dataGridViewA3.Columns[i].CellTemplate = CellR;
    dataGridViewA3.Columns[i].Width = 30;
}
private void initMatrixScalarProduct()
{
    int column = 0;
    column = Convert.ToInt16(textColumnA3.Text);
dataGridViewScalarProduct.ColumnCount = column;
dataGridViewScalarProduct.AllowUserToOrderColumns = false;
dataGridViewScalarProduct.AllowUserToAddRows = false;
dataGridViewScalarProduct.Enabled = true;
dataGridViewScalarProduct.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.DisplayedCellsExceptHeaders;
dataGridViewScalarProduct.ColumnHeadersBorderStyle = DataGridViewColumnHeaderBorderStyle.Raised;
dataGridViewScalarProduct.CellBorderStyle = DataGridViewCellBorderStyle.Single;
dataGridViewScalarProduct.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
dataGridViewScalarProduct.DefaultCellStyle.BackColor = Color.AntiqueWhite;
dataGridViewScalarProduct.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
dataGridViewScalarProduct.DefaultCellStyle.BackColor = Color.AntiqueWhite;
}
Sample programs in C++ for matrix computations in max plus algebra

DataGridViewCell CellR = new DataGridViewTextBoxCell();
CellR.Style.Alignment = DataGridViewContentAlignment.MiddleRight;
DataGridViewCell CellL = new DataGridViewTextBoxCell();
CellL.Style.Alignment = DataGridViewContentAlignment.MiddleLeft;

int line = 0;
line = Convert.ToInt16(textLineA3.Text);
dataGridViewScalarProduct.RowCount = line;

for (int i = 0; i < column; i++)
{
    dataGridViewScalarProduct.Columns[i].Name = "C" + (i + 1);
    dataGridViewScalarProduct.Columns[i].CellTemplate = CellR;
    dataGridViewScalarProduct.Columns[i].Width = 30;
}

private void GeneratingButons3_Click(object sender, EventArgs e)
{
    if (textLineA3.Text != "")
    {
        initMatrixA3();
    }
    else
        MessageBox.Show("Introduce number of lines and number of columns!");
}

private void btScalarProduct_Click(object sender, EventArgs e) {
    initMatrixScalarProduct();
    if (textScalar.Text != "")
    {
        int lineA = 0;
        int columnA = 0;
        columnA = Convert.ToInt16(textColumnA3.Text);
        lineA = Convert.ToInt16(textLineA3.Text);
        int[,] A3 = new int[lineA, columnA];
        int[,] ScalarProduct = new int[lineA, columnA];
        int a;

        #region reading of scalar a and matrix A
        a = Convert.ToInt16(textScalar.Text);
        for (int i = 0; i < lineA; i++)
        {
            for (int j = 0; j < columnA; j++)
            {
                if (dataGridViewA3.Rows[i].Cells[j].Value.ToString() == "E")
                {
                    // Code Snippet
                }
            }
        }
        // Continue the code
    }

    // More code
}
A3[i, j] = Int32.MinValue;
else
    A3[i, j] = Convert.ToInt16(dataGridA3.Rows[i].Cells[j].Value.ToString());
}

#region computation of scalar product
for (int i = 0; i < lineA; i++)
{
    for (int j = 0; j < columnA; j++)
    {
        if (A3[i, j] == Int32.MinValue)
            Scalar Product[i, j] = Int32.MinValue;
        else
            Scalar Product[i, j] = (A3[i, j] + a);
    }
}
#endregion

#region display scalar product
for (int i = 0; i < lineA; i++)
{
    for (int j = 0; j < columnA; j++)
    {
        if (ScalarProduct[i, j] == Int32.MinValue)
            dataGridScalarProduct.Rows[i].Cells[j].Value = "E";
        else
            dataGridScalarProduct.Rows[i].Cells[j].Value = ScalarProduct[i, j];
    }
}
#endregion

public void Reset_Values_for_Scalar_Multiplication()
{
    textcolumnA3.ResetText();
    textScalar.ResetText();
    textLineA3.ResetText();
    dataGridA3.Rows.Clear();
    dataGridScalarProduct.Rows.Clear();
}

private void btReset_Scalar_Product_Click(object sender, EventArgs e)
Sample programs in C++ for matrix computations in max plus algebra

```csharp
{
    Reset_Values_for_Scalar_Multiplication();
}
#endregion

#region Power of a matrix in max plus algebra
private void initMatrixA4() {
    int column = 0;
    column = Convert.ToInt16(textlineA4.Text);
    dataGridViewA4.ColumnCount = column;
    dataGridViewA4.AllowUserToOrderColumns = false;
    dataGridViewA4.AllowUserToAddRows = false;
    dataGridViewA4.Enabled = true;
    dataGridViewA4.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.DisplayedCellsExceptHeaders;
    dataGridViewA4.ColumnHeaderBorderStyle = DataGridViewHeaderBorderStyle.Raised;
    dataGridViewA4.CellBorderStyle = DataGridViewCellBorderStyle.Single;
    dataGridViewA4.GridColor = Color.DodgerBlue;
    dataGridViewA4.ColumnHeaderVisible = false;
    dataGridViewA4.RowHeadersVisible = false;
    dataGridViewA4.BackgroundColor = Color.WhiteSmoke;
    dataGridViewA4.BorderStyle = BorderStyle.None;
    dataGridViewA4.AllowUserToResizeColumns = true;

    DataGridViewCellStyle columnHeaderStyle = new DataGridViewCellStyle();
    columnHeaderStyle.BackColor = Color.AntiqueWhite;
    columnHeaderStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewA4.ColumnHeaderDefaultCellStyle = columnHeaderStyle;

    dataGridViewA4[col].Width = 30;
    dataGridViewA4.Rows[0][col].Value = "C" + (col + 1);
}
```
private void initPowerMatrix ()
{
    int column = 0;
    column = Convert.ToInt16(textlineA4.Text);
    dataGridViewMatrix_at_power_n.ColumnCount = column;
    dataGridViewMatrix_at_power_n.AllowUserToOrderColumns = false;
    dataGridViewMatrix_at_power_n.AllowUserToAddRows = false;
    dataGridViewMatrix_at_power_n.Enabled = true;
    dataGridViewMatrix_at_power_n.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.DisplayedCellsExceptHeaders;
    dataGridViewMatrix_at_power_n.ColumnHeaderBorderStyle = DataGridViewHeaderBorderStyle.Raised;
    dataGridViewMatrix_at_power_n.CellBorderStyle = DataGridViewCellBorderStyle.Single;
    dataGridViewMatrix_at_power_n.GridColor = Color.DodgerBlue;
    dataGridViewMatrix_at_power_n.ColumnHeadersVisible = false;
    dataGridViewMatrix_at_power_n.RowHeadersVisible = false;
    dataGridViewMatrix_at_power_n.BackgroundColor = Color.WhiteSmoke;
    dataGridViewMatrix_at_power_n.BorderStyle = BorderStyle.None;
    dataGridViewMatrix_at_power_n.AllowUserToResizeColumns = true;
    dataGridViewMatrix_at_power_n.AllowUserToOrderColumns = false;

    DataGridViewCellStyle columnHeaderStyle = new DataGridViewCellStyle();
    columnHeaderStyle.BackColor = Color.AntiqueWhite;
    columnHeaderStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewMatrix_at_power_n.ColumnHeaderDefaultCellStyle = columnHeaderStyle;

    DataGridViewCell CellR = new DataGridViewTextBoxCell();
    CellR.Style.Alignment = DataGridViewContentAlignment.MiddleRight;
    DataGridViewCell CellL = new DataGridViewTextBoxCell();
    CellL.Style.Alignment = DataGridViewContentAlignment.MiddleLeft;

    int line = 0;
    line = Convert.ToInt16(textlineA4.Text);
    dataGridViewMatrix_at_power_n.RowCount = line;
    for (int i = 0; i < column; i++)
    {
        dataGridViewMatrix_at_power_n.Columns[i].Name = "C" + (i + 1);
        dataGridViewMatrix_at_power_n.Columns[i].CellTemplate = CellR;
        dataGridViewMatrix_at_power_n.Columns[i].Width = 30;
    }
}

private void btGenerareA4_Click(object sender, EventArgs e)
{
    if (textlineA4.Text != "")
    {
        // Code to generate A4 matrix based on the text in textlineA4
    }
}
Sample programs in C++ for matrix computations in max plus algebra

```cpp
{
    initMatrixA4();
}
else
    MessageBox.Show("Introduce number of lines and columns!");

private void btComputPower_Click(object sender, EventArgs e)
{
    initMatrixPower();
    labelPower.Text = textPower.Text;
    if (textPower.Text != "")
    {
        int lineA = 0;
        int columnA = 0;
        columnA = Convert.ToInt16(textlineA4.Text);
        lineA = Convert.ToInt16(textlineA4.Text);
        int[,] A4 = new int[lineA, columnA];
        int[,] B = new int[lineA, columnA];
        int[,] Power_n = new int[lineA, columnA];
        int[,] sum = new int[lineA, columnA];
        int a;
        a = Convert.ToInt16(textPower.Text);
        #region reading matrix A
        for (int i = 0; i < lineA; i++)
        {
            for (int j = 0; j < columnA; j++)
            {
                if (dataGridA4.Rows[i].Cells[j].Value.ToString() == "E")
                    A4[i, j] = Int32.MinValue;
                else
                    A4[i, j] = Convert.ToInt16(dataGridA4.Rows[i].Cells[j].Value.ToString());
                B[i, j] = A4[i, j];
            }
        }
        #endregion
        #region PowerMatrix A
        for (int p = 2; p <= a; p++)
        {
            for (int i = 0; i < lineA; i++)
            {
                for (int j = 0; j < columnA; j++)
                {
                    Power_n[i, j] = Int32.MinValue;
                }
            }
            for (int i = 0; i < lineA; i++)
            {
                for (int j = 0; j < columnA; j++)
                {
                    Power_n[i, j] = Int32.MinValue;
                }
            }
            for (int i = 0; i < lineA; i++)
            {
                for (int j = 0; j < columnA; j++)
                {
                    Power_n[i, j] = Int32.MinValue;
                }
            }
```
```
for (int k = 0; k < lineA; k++)
{
    if (A4[i, k] == Int32.MinValue —— B[k, j] == Int32.MinValue)
        sum[i, j] = Int32.MinValue;
    else
        sum[i, j] = A4[i, k] + B[k, j];
    if (Power_n[i, j] > sum[i, j])
        Power_n[i, j] = Power_n[i, j];
    else
        Power_n[i, j] = sum[i, j];
}
}
for (int i = 0; i < lineA; i++)
{
    for (int j = 0; j < columnA; j++)
    {
        B[i, j] = Power_n[i, j];
    }
}
#endregion
#region display matrix
for (int i = 0; i < lineA; i++)
{
    for (int j = 0; j < columnA; j++)
    {
        if (Power_n[i, j] == Int32.MinValue)
            dataGridMatrix_at_power_n.Rows[i].Cells[j].Value = "E";
        else
            dataGridMatrix_at_power_n.Rows[i].Cells[j].Value = Power_n[i, j];
    }
}
Sample programs in C++ for matrix computations in max plus algebra

```csharp
dataGridMatrix_.at_.Power_.n.Rows.Clear();
private void btReset_lifting_.at_.power_.Click(object sender, EventArgs e)
{
    Reset_Values_for_Lifting_.at_.Power();
}
#endregion
#region Solving equations system in max plus algebra
private void initMatrixA5()
{
    int column = 0;
    column = Convert.ToInt16(textlineA6.Text);
    dataGridViewA5.ColumnCount = column;
    dataGridViewA5.AllowUserToOrderColumns = false;
    dataGridViewA5.AllowUserToAddRows = false;
    dataGridViewA5.Enabled = true;
    dataGridViewA5.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.DisplayedCellsExceptHeaders;
    dataGridViewA5.ColumnHeadersBorderStyle = DataGridViewHeaderBorderStyle.Raised;
    dataGridViewA5.CellBorderStyle = DataGridViewCellBorderStyle.Single;
    dataGridViewA5.GridColor = Color.DodgerBlue;
    dataGridViewA5.ColumnHeadersVisible = false;
    dataGridViewA5.RowHeadersVisible = false;
    dataGridViewA5.BackgroundColor = Color.WhiteSmoke;
    dataGridViewA5.BorderStyle = BorderStyle.None;
    dataGridViewA5.AllowUserToResizeColumns = true;
    DataGridViewCellStyle columnHeaderStyle = new DataGridViewCellStyle();
    columnHeaderStyle.BackColor = Color.AntiqueWhite;
    columnHeaderStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewA5.ColumnHeadersDefaultCellStyle = columnHeaderStyle;
    DataGridViewCell CellR = new DataGridViewTextBoxCell();
    CellR.Style.Alignment = DataGridViewContentAlignment.MiddleRight;
    DataGridViewCell CellL = new DataGridViewTextBoxCell();
    CellL.Style.Alignment = DataGridViewContentAlignment.MiddleLeft;
    int line = 0;
    line = Convert.ToInt16(textlineA5.Text);
    dataGridViewA5.RowCount = line;
    for (int i = 0; i < column; i++)
    {
        dataGridViewA5.Columns[i].Name = "C" + (i + 1);
    }
    dataGridViewA5.Columns[0].Name = "A1";
   .dataGridViewA5.Columns[1].Name = "A2";
    dataGridViewA5.Columns[2].Name = "A3";
    dataGridViewA5.Columns[3].Name = "A4";
    dataGridViewA5.Columns[4].Name = "A5";
    dataGridViewA5.Columns[5].Name = "A6";
    dataGridViewA5.Columns[6].Name = "A7";
    dataGridViewA5.Columns[7].Name = "A8";
```

```csharp
```
private void initMatrixX0()
{
    int column = 0;
    column = 1;
    dataGridViewX0.ColumnCount = column;
    dataGridViewX0.AllowUserToOrderColumns = false;
    dataGridViewX0.AllowUserToAddRows = false;
    dataGridViewX0.Enabled = true;
    dataGridViewX0.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.DisplayedCellsExceptHeaders;
    dataGridViewX0.ColumnHeaderBorderStyle = DataGridViewHeaderBorderStyle.Raised;
    dataGridViewX0.ColumnHeaderStyle = DataGridViewCellStyle.Single;
    dataGridViewX0.DefaultCellStyle = columnHeaderStyle;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleLeft;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewContentAlignment.MiddleRight;
    dataGridViewX0.DefaultCellStyle.Alignment =DataGridViewCellStyle.RowHeadersModeewater.
    dataGridViewX0.DefaultCellStyle.Alignment = DataGridViewCellStyle.RowHeadersMode.
private void initMatrixSolutionSystem()
{
    int column = 0;
    column = 1;
    dataGridViewSolutionXk.ColumnCount = column;
    dataGridViewSolutionXk.AllowUserToOrderColumns = false;
    dataGridViewSolutionXk.AllowUserToAddRows = false;
    dataGridViewSolutionXk.Enabled = true;
    dataGridViewSolutionXk.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.DisplayedCellsExceptHeaders;
    dataGridViewSolutionXk.RowHeadersVisible = false;
    dataGridViewSolutionXk.BackgroundColor = Color.WhiteSmoke;
    dataGridViewSolutionXk.BorderStyle = BorderStyle.None;
    dataGridViewSolutionXk.AllowUserToResizeColumns = true;

    DataGridViewCellBorderStyle columnHeaderStyle = new DataGridViewCellStyle();
    columnHeaderStyle.BackColor = Color.AntiqueWhite;
    columnHeaderStyle.Alignment = DataGridViewContentAlignment.MiddleCenter;
    dataGridViewSolutionXk.ColumnHeadersDefaultCellStyle = columnHeaderStyle;

    dataGridViewSolutionXk.Columns[i].Name = "C" + (i + 1);
    dataGridViewSolutionXk.Columns[i].CellTemplate = CellR;
    dataGridViewSolutionXk.Columns[i].Width = 30;
}

private void btGenerateMatrices_Click(object sender, EventArgs e)
{
    if (textlineA5.Text != "")
private void btComputSystem_Click(object sender, EventArgs e)
{
    initMatrixSolutionSystem();
    label_k.Text = textk.Text;
    if (textk.Text != "")
    {
        int lineA = 0;
        int columnA = 0;
        int linex0 = 0;
        columnA = Convert.ToInt16(textlinieA5.Text);
        lineA = Convert.ToInt16(textlinieA5.Text);
        linex0 = Convert.ToInt16(textlineA5.Text);
        int[,] A5 = new int[lineA, columnA];
        int[,] X0 = new int[linex0, 1];
        int[,] Xk = new int[lineA, 1];
        int[,] B = new int[lineA, columnA];
        int[,] Power_n = new int[lineA, columnA];
        int[,] sum = new int[lineA, columnA];
        int[,] sum2 = new int[lineA, 1];
        int k;
        k = Convert.ToInt16(textk.Text);
        #region readings matrix A and matrix X0
        for (int i = 0; i < lineA; i++)
        {
            for (int j = 0; j < coloanaA; j++)
            {
                if (dataGridViewA5.Rows[i].Cells[j].Value.ToString() == "E")
                    A5[i, j] = Int32.MinValue;
                else
                    A5[i, j] = Convert.ToInt16(dataGridViewA5.Rows[i].Cells[j].Value.ToString());
                    B[i, j] = A5[i, j];
            }
        }
        for (int i = 0; i < linex0; i++)
        {
            initMatrixA5();
            initMatrixX0();
        }
        else
            MessageBox.Show("Introduce number of lines and columns!");
    }
if (dataGridX0.Rows[i].Cells[0].Value.ToString() == "E")
    X0[i, 0] = Int32.MinValue;
else
    X0[i, 0] = Convert.ToInt16(dataGridX0.Rows[i].Cells[0].Value.ToString());
}
#endregion
#region lifting at power k of matrix A
if (k == 1)
{
    for (int i = 0; i < lineA; i++)
    {
        for (int j = 0; j < columnA; j++)
        {
            Power_n[i, j] = A5[i, j];
        }
    }
}
else
{
    for (int p = 2; p <= k; p++)
    {
        for (int i = 0; i < lineA; i++)
        {
            for (int j = 0; j < columnA; j++)
            {
                Power_n[i, j] = Int32.MinValue;
                for (int h = 0; h < lineA; h++)
                {
                    if (A5[i, h] == Int32.MinValue || B[h, j] == Int32.MinValue)
                        sum[i, j] = Int32.MinValue;
                    else
                        sum[i, j] = A5[i, h] + B[h, j];
                    if (Power_n[i, j] > sum[i, j])
                        Power_n[i, j] = Power_n[i, j];
                    else
                        Power_n[i, j] = sum[i, j];
                }
            }
        }
    }
}
for (int i = 0; i < lineA; i++)
{
    for (int j = 0; j < columnA; j++)
{  
  B[i, j] = Power_n[i, j];  
}

{  
}

region computation of matrix X(k)
for (int i = 0; i < lineA; i++)  
{  
  for (int j = 0; j < 1; j++)  
  {  
    Xk[i, j] = Int32.MinValue;  
    for (int h = 0; h < lineA; h++)  
    {  
      if (Power_n[i, h] == Int32.MinValue || X0[h, j] == Int32.MinValue)  
        sum2[i, j] = Int32.MinValue;  
      else  
        sum2[i, j] = Power_n[i, h] + X0[h, j];  
      if (Xk[i, j] > sum2[i,j])  
        Xk[i, j] = Xk[i,j];  
      else  
        Xk[i, j] = sum2[i,j];  
    }  
  }  
}

region display of matrix X(k)
for (int i = 0; i < lineA; i++)  
{  
  if (Xk[i, 0] == Int32.MinValue)  
    dataGridViewSolutionXk.Rows[i].Cells[0].Value = "E";
  else  
    dataGridViewSolutionXk.Rows[i].Cells[0].Value = Xk[i,0];
}

public void Reset_Values_for_Computation_system()
{
    textlineA5.ResetText();
}
Sample programs in $C^{++}$ for matrix computations in max plus algebra

textk.ResetText();
label_k.Text = "k";
dataGridSolutionXk.Rows.Clear();
dataGridA5.Rows.Clear();
dataGridX0.Rows.Clear();
}
private void btResetSystem_Click(object sender, EventArgs e)
{
    Reset_Values_for_Computation_System();
}
#endregion

We illustrate the utilization of the above programs in the following cases.

1. **Sum of two matrices $A$ and $B$ in $\mathbb{R}_{\text{max}}$**

| Inputs data | Outputs data |
|-------------|--------------|
| Number of lines | $3$ |
| Number of columns | $4$ |
| Matrix $A$ | Matrix $B$ | Matrix $A \oplus B$ |
| 3 | 9 | 9 | 8 | 9 | 9 | 8 | 2 |
| 6 | 0 | 4 | 9 | 2 | 1 | 6 | 3 |
| $E$ | 5 | $-7$ | 1 | 1 | 2 | 4 | $-5$ |

2. **Product of two matrices $A$ and $B$ in $\mathbb{R}_{\text{max}}$**

| Inputs data |
|-------------|
| Number of lines of $A$! | $3$ |
| Number of columns of $A$! | $4$ |
| Number of lines of $B$! | $4$ |
| Number of columns of matrix $B$! | $3$ |

| Matrix $A$ | Matrix $B$ | Matrix $A \otimes B$ |
|------------|------------|---------------------|
| 2 | 1 | -1 | 4 | 5 | 0 | 1 |
| $E$ | 0 | 5 | 3 | 7 | 4 | $E$ |
| $-4$ | $-2$ | $E$ | 6 | $-5$ | 9 | 2 |
| | | | | 8 | $-6$ | 1 |
3. Multiplication with scalar $a$ of a matrix $A$ in $\mathbb{R}_{max}$

| Inputs data | Outputs data |
|-------------|--------------|
| Number of lines for matrix $A$! | 3 |
| Number of columns for matrix $A$! | 5 |
| Scalar $a$ | -4 |

Matrix $A$

\[
\begin{bmatrix}
4 & -7 & 8 & 2 & E \\
5 & E & 0 & E & 8 \\
9 & 2 & E & 3 & 1 \\
\end{bmatrix}
\]

Matrix $a \otimes A$

\[
\begin{bmatrix}
0 & -11 & 4 & -2 & E \\
1 & E & -4 & E & 4 \\
5 & -2 & E & -1 & -3 \\
\end{bmatrix}
\]

4. Power of a matrix $A$ in $\mathbb{R}_{max}$

| Inputs data | Outputs data |
|-------------|--------------|
| Number of lines and columns for matrix $A$! | 5 |
| Power of matrix $A$! | 9 |

Matrix $A$

\[
\begin{bmatrix}
1 & 0 & -2 & E & 3 \\
0 & 2 & E & 4 & 1 \\
1 & -1 & -4 & 5 & 3 \\
7 & 9 & 4 & 3 & 0 \\
8 & 0 & -2 & 0 & E \\
\end{bmatrix}
\]

Matrix $A^{(9)}$

\[
\begin{bmatrix}
50 & 52 & 47 & 46 & 47 \\
53 & 55 & 50 & 56 & 53 \\
54 & 56 & 51 & 57 & 54 \\
59 & 61 & 56 & 55 & 52 \\
52 & 52 & 47 & 52 & 49 \\
\end{bmatrix}
\]

5. Solving equations system in $\mathbb{R}_{max}$

| Inputs data | Outputs data |
|-------------|--------------|
| Number of lines and columns for matrix $A$! | 4 |
| Value for $k$! | 10 |

Matrix $A$

\[
\begin{bmatrix}
3 & -5 & -9 & 2 \\
4 & 8 & 7 & 4 \\
-6 & E & 0 & E \\
1 & 1 & E & 2 \\
\end{bmatrix}
\]

Matrix $X(0)$

\[
\begin{bmatrix}
4 \\
3 \\
2 \\
1 \\
\end{bmatrix}
\]

Matrix $X(10)$

\[
\begin{bmatrix}
70 \\
83 \\
56 \\
76 \\
\end{bmatrix}
\]
References

[1] W. Kuich and A. Salomaa, *Semirings, Automata, Languages*, EACTS Monographs on Theoretical Computer Science, 5, Springer-Verlag, 1986.

[2] G. L. Litvinov, *The Maslov dequantization, idempotent and tropical mathematics: a brief introduction*, Journal of Mathematical Sciences, 140 (2007), no. 3, 426–444.

[3] M. Mohri, *Semirings, frameworks and algorithms for shortest-distance problems*, Journal of Automata, Languages and Combinatorics, 7 (2002), no. 3, 321–350.

Author’s addresses

West University of Timișoara,
Department of Mathematics, Bd. V. Pârvan, no. 4, 300223, Timișoara, Romania

E-mail: mihai31ro@yahoo.com; ivan@math.uvt.ro