

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ
КАФЕДРА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

ЗВІТ ДО ЛАБОРАТОРНОЇ РОБОТИ №1
З ДИСЦИПЛІНИ «Математичне моделювання та чисельні методи»
НА ТЕМУ «Обчислення функцій та побудова графіків у середовищі MATLAB.
Елементи моделювання в Simulink.»

Виконав:
студент гр. ПЗП-17-1
Кириченко О. В.
Перевірив:
Матвеев Д. І.

Мета роботи: Навчитись використовувати математичне середовище MATLAB та середовище для математичного моделювання Simulink. Відтворити досліджену поведінку математичного середовища за допомогою мови програмування.

Задля локанічності нижче наведено загальні методи на мові C#, які будуть використані у завданнях:

```
public (int height, int width) GetMatrixSize<T>(T[,] matrix)
{
    return (matrix.GetLength(0), matrix.GetLength(1));
}
public void PrintMatrix<T>(T[,] matrix)
{
    var size = GetMatrixSize(matrix);
    for(int i = 0; i < size.height; ++i)
    {
        for (int j = 0; j < size.width; ++j)
            Console.Write($"{matrix[i, j]} ");
        Console.WriteLine();
    }
}
public void PrintVector<T>(T[] vector, string name)
{
    Console.WriteLine($"{name} = {string.Join(" ", vector)}");
}
public double[,] ConvertToDouble(int[,] matrix)
{
    var size = GetMatrixSize(matrix);
    double[,] result = new double[size.height, size.width];
    for (int i = 0; i < size.height; ++i)
        for (int j = 0; j < size.width; ++j)
            result[i, j] = (double)matrix[i, j];
    return result;
}
public double[] ConvertToDouble(int[] vector)
{
    return vector.Select(item => (double)item).ToArray();
}
public T[,] VectorToMatrix<T>(T[] vector, bool inline)
{
    if(inline)
    {
        T[,] result = new T[1, vector.Length];
        for (int i = 0; i < vector.Length; ++i)
            result[0, i] = vector[i];
    }
}
```

```

        return result;
    }
    else
    {
        T[,] result = new T[vector.Length, 1];
        for (int i = 0; i < vector.Length; ++i)
            result[i, 0] = vector[i];
        return result;
    }
}

public double[,] Multiply(double[,] a, double[,] b)
{
    var sizea = GetMatrixSize(a);
    var sizeb = GetMatrixSize(b);

    if (sizea.width != sizeb.height)
        throw new ArgumentException();

    double[,] result = new double[sizea.height, sizeb.width];

    for(int i = 0; i < sizea.height; ++i)
        for(int j = 0; j < sizeb.width; ++j)
            for(int k = 0; k < sizea.width; ++k)
                result[i, j] += a[i, k] * b[k, j];

    return result;
}

public T[] MatrixToVector<T>(T[,] matrix)
{
    var size = GetMatrixSize(matrix);
    if(size.height == 1)
    {
        T[] vector = new T[size.width];
        for (int i = 0; i < size.width; ++i)
            vector[i] = matrix[0, i];
        return vector;
    }
    else if(size.width == 1)
    {
        T[] vector = new T[size.height];
        for (int i = 0; i < size.height; ++i)
            vector[i] = matrix[i, 0];
        return vector;
    }
}

```

```

        throw new ArgumentException();
    }
    public T[,] GetMinor<T>(T[,] matrix, int row, int column)
    {
        var size = GetMatrixSize(matrix);
        T[,] submatrix = new T[size.height - 1, size.width - 1];

        int di = 0;
        for(int i = 0; i < size.height; ++i)
        {
            if(i == row)
            {
                di = -1;
                continue;
            }

            int dj = 0;
            for(int j = 0; j < size.width; ++j)
            {
                if(j == column)
                {
                    dj = -1;
                    continue;
                }

                submatrix[i + di, j + dj] = matrix[i, j];
            }
        }

        return submatrix;
    }
    public double Det(double[,] matrix)
    {
        var size = GetMatrixSize(matrix);

```

```

        if (size.height != size.width)
            throw new ArgumentException();

        if (size.width == 2 && size.height == 2)
            return matrix[0, 0] * matrix[1, 1] - matrix[0, 1] * matrix[1, 0];

        return Enumerable.Range(0, size.width).Sum(
            i => Math.Pow(-1, i) * matrix[0, i] * Det(GetMinor(matrix, 0, i))
        );
    }

    public T[, ] Transpose<T>(T[, ] matrix)
    {
        var size = GetMatrixSize(matrix);
        T[, ] transposed = new T[size.width, size.height];
        for (int i = 0; i < size.height; ++i)
            for (int j = 0; j < size.width; ++j)
                transposed[j, i] = matrix[i, j];
        return transposed;
    }

    public void Multiply(double[, ] matrix, double value)
    {
        var size = GetMatrixSize(matrix);
        for (int i = 0; i < size.height; ++i)
            for (int j = 0; j < size.width; ++j)
                matrix[i, j] *= value;
    }

    public double[, ] Invert(double[, ] matrix)
    {
        double det = Det(matrix);

        if (det == 0)
            throw new ArgumentException();

        var size = GetMatrixSize(matrix);
        double[, ] transposed = Transpose(matrix);
        double[, ] complements = new double[size.height, size.width];
    }

```

```

        for (int i = 0; i < size.height; ++i)
            for (int j = 0; j < size.width; ++j)
                complements[i, j] = Math.Pow(-1, i + j) *
                    Det(GetMinor(transposed, i, j));

        Multiply(complements, 1 / det);

        return complements;
    }
    public double[] PolyFit(double[] x, double[] y, int k)
    {
        if (x.Length != y.Length)
            throw new ArgumentException();

        int n = x.Length;

        double[] powersums = new double[2 * k + 1];
        for (int i = 0; i <= 2 * k; ++i)
            powersums[i] = Enumerable.Range(0, n).Sum(
                j => Math.Pow(x[j], i)
            );

        double[,] xm = new double[k + 1, k + 1];
        double[] ym = new double[k + 1];
        for (int i = 0; i <= k; ++i)
        {
            for (int j = 0; j <= k; ++j)
                xm[i, j] = powersums[i + j];
            ym[i] = Enumerable.Range(0, n).Sum(
                j => Math.Pow(x[j], i) * y[j]
            );
        }
        return MatrixToVector(
            Multiply(Invert(xm), VectorToMatrix(ym, false))
        );
    }
}

```

```

public Func<double, double> Poly(double[] p)
{
    return x => p.Select((a, i) => a * Math.Pow(x, i)).Sum();
}

public double[] PolyVal(double[] p, double[] x)
{
    Func<double, double> poly = Poly(p);
    return x.Select(xi => poly(xi)).ToArray();
}

public void OutPolyFitVal(
    double[] x, double[] y, int n, Plot plot, Plot plotSmooth,
    double smooth, Color color, double thickness = 1
) {
    double[] pfn = PolyFit(x, y, n);
    double[] pvn = PolyVal(pfn, x);

    PrintVector(pfn, $"\\nQ{n}");
    PrintVector(pvn, $"V{n}");

    plot.PlotSignalXY(x, pvn, color, thickness);

    double min = x.Min();
    double spread = x.Max() - min;
    int count = (int)(spread / smooth);
    double step = spread / count;

    double[] xsmooth = Enumerable.Range(0, count).Select(
        i => i * step + min
    ).ToArray();
    double[] ysmooth = PolyVal(pfn, xsmooth);
    plotSmooth.PlotSignalXY(xsmooth, ysmooth, color, thickness);
}

public void ShowPlotForm(params Bitmap[] renderedPlots)
{
    Form plotForm = new Form();

```

```

plotForm.AutoScroll = true;
plotForm.Size = new Size(
    renderedPlots.Take(2).Sum(plot => plot.Width),
    renderedPlots.Max(plot => plot.Height)
);
Point location = new Point(0, 0);
foreach(Bitmap plot in renderedPlots)
{
    PictureBox picture = new PictureBox();

    picture.Size = plot.Size;
    picture.Location = location;
    picture.Image = plot;

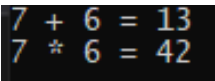
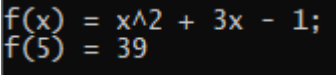
    plotForm.Controls.Add(picture);
    location.Offset(plot.Width, 0);
}

plotForm.ShowDialog();
}

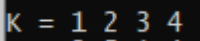
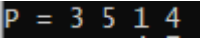
public static class Range
{
    public static double[] Make(double start, double end, double delta)
    {
        int count = (int)Math.Ceiling((end - start) / delta);
        return Enumerable.Range(0, count).Select(
            i => i * delta + start
        ).ToArray();
    }
}

```


1. Арифметичні обчислення у MATLAB:

Завдання	MATLAB	C#
Обчислити суму, добуток будь-яких двох чисел:	<pre>>> 6+7 ans = 13 >> 6*7 ans = 42</pre>	<pre>Console.WriteLine(\$"7 + 6 = {7 + 6}"); Console.WriteLine(\$"7 * 6 = {7 * 6}");</pre> 
Обчислити значення функції $y(x) = x^2 + 3x - 1$ у точці $x = 5$	<pre>editor - C:\Users\Олер\Documents\MATLAB\y.m y.m x + function [output] = y(input) output = input ^ 2 + 3 * input - 1; end >> y(5) ans = 39</pre>	<pre>Func<double, double> f = new Func<double, double>(x => x * x + 3 * x - 1); Console.WriteLine(\$"\\nf(x) = x^2 + 3x - 1;\\nf(5) = {f(5)}"</pre> 

2. Обчислення з векторами та матрицями у MATLAB:

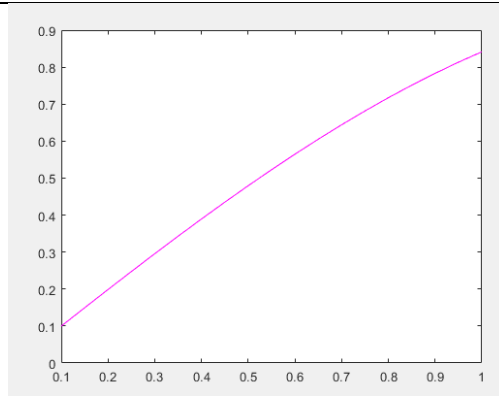
Завдання	MATLAB	C#
Задати вектор-рядок К, як послідовність чисел від 1 до 4 із кроком 1	<pre>>> K = 1:4 K = 1 2 3 4</pre>	<pre>int count = 4; int[] k = Enumerable.Range(1, count).ToArray(); PrintVector(k, "\\nK");</pre> 
Задати вектор-стовпець, як послідовність із 4 чисел	<pre>>> P=[3;5;1;4] P = 3 5 1 4</pre>	<pre>int[] p = new int[] { 3, 5, 1, 4 }; PrintVector(p, "P");</pre> 

Обчисліть вектор суми K та P, їх скалярний добуток	<pre>>> K + P' ans = 4 7 4 8 >> K * P ans = 32</pre>	<pre>int[] sum = Enumerable.Range(0, count).Select(i => k[i] + p[i]).ToArray(); PrintVector(sum, "K + P"); int mult = Enumerable.Range(0, count).Aggregate(0, (a, i) => a + k[i] * p[i]); Console.WriteLine(\$"K * P = {mult}\n");</pre> <pre>K + P = 4 7 4 8 K * P = 32</pre>
Створити матрицю M цілих чисел 4x4	<pre>>> M = [1 3 2 7; 5 6 3 2; 8 9 11 1; -1 7 3 0] M = 1 3 2 7 5 6 3 2 8 9 11 1 -1 7 3 0</pre>	<pre>int[,] m = new int[4, 4] { { 1, 3, 2, 7 }, { 5, 6, 3, 2 }, { 8, 9, 11, 1 }, { -1, 7, 3, 0 } }; Console.WriteLine("M = "); PrintMatrix(m);</pre> <pre>M = 1 3 2 7 5 6 3 2 8 9 11 1 -1 7 3 0</pre>
Обчислити добуток матриці M на вектор-стовпець P	<pre>>> M * P ans = 48 56 84 35</pre>	<pre>double[,] result = Multiply(ConvertToDouble(m), ConvertToDouble(VectorToMatrix(p, false))); Console.WriteLine("\nM * P = "); PrintMatrix(result);</pre> <pre>M * P = 48 56 84 35</pre>

3. Апроксимація та побудова графіків у MATLAB.

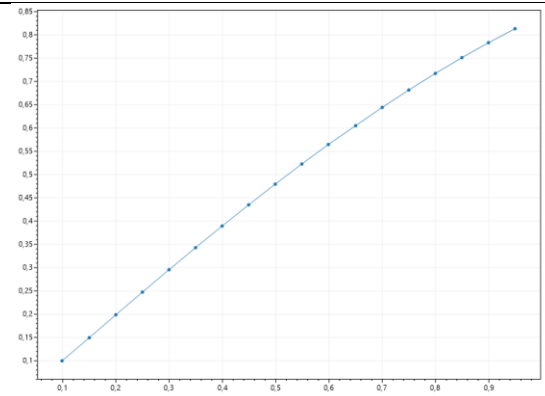
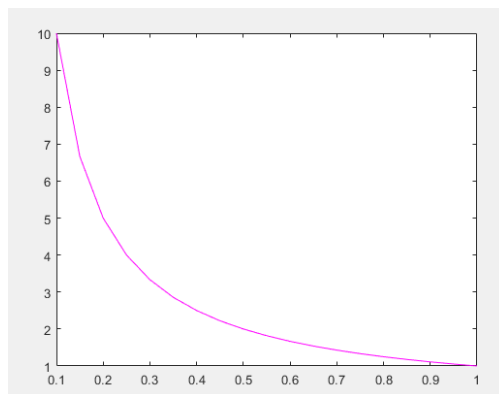
Завдання	MATLAB	C#
Побудувати графіки функцій $y_1(x) = \sin(x)$ $y_2(x) = \frac{1}{x}$	<pre>>> x = 0.1:0.05:1 >> y1 = sin(x) >> plot(x, y1, '-m') ...</pre>	<pre>double[] xs = Range.Make(0.1, 1, 0.05); Func<double, double> y1 = xv => Math.Sin(xv); double[] y1s = xs.Select(xv => y1(xv)).ToArray(); Plot plotY1 = new Plot(); plotY1.PlotSignalXY(xs, y1s);</pre>

де x змінюється
від 0.1 до 1 із
кроком 0.05



```
>> y2 = 1 ./ x
```

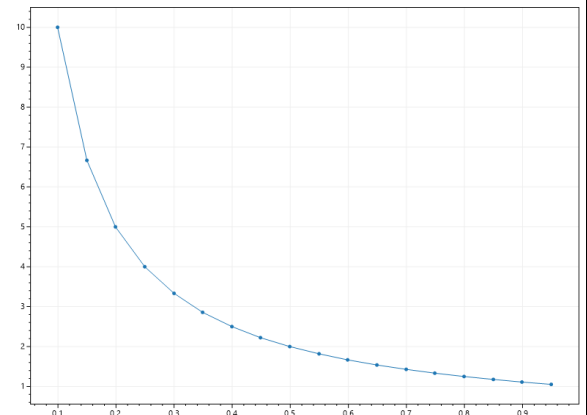
```
>> plot(x, y2, '-m')
```



```
Func<double, double> y2 = xv => 1.0 /xv;  
double[] y2s = xs.Select(  
    xv => y2(xv)  
)>.ToArray();
```

```
Plot plotY2 = new Plot();  
plotY2.PlotSignalXY(xs, y2s);
```

```
ShowPlotForm(  
    plotY1.Render(),  
    plotY2.Render()  
);
```



Знайти
коефіцієнти
апроксимуючог
о поліному $g(x)$
2-го, 3-го та 5-го
степенів для
наданих
векторів;

```
>> X = [ -2.1 -1 0.5 1 2.1 3.1 ]
```

```
>> Y = [ 0.6 0.1 -0.5 0.1 0.7 0.9]
```

```
>> pf2 = polyfit(X, Y, 2)
```

```
pf2 =  
    0.1404    -0.0458    -0.1398
```

```
>> pf3 = polyfit(X, Y, 3)
```

```
pf3 =  
    -0.0309    0.1908    0.0991    -0.2464
```

```
>> pf5 = polyfit(X, Y, 5)
```

```
pf5 =  
    0.0521    -0.1787    -0.2747    1.1282    0.2227    -0.8495
```

```
Plot plotPoints = new Plot();  
Plot plotSmooth = new Plot();
```

```
double[] x = new double[] {  
    -2.1, -1, 0.5, 1, 2.1, 3.1  
};  
double[] y = new double[] {  
    0.6, 0.1, -0.5, 0.1, 0.7, 0.9  
};
```

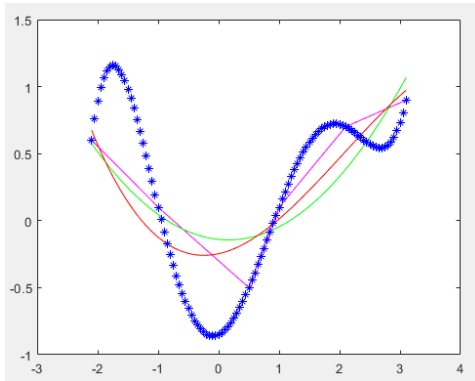
```
PrintVector(x, $"X");  
PrintVector(y, $"Y");
```

```
plotPoints.PlotSignalXY(  
    x, y, Color.Violet, 5  
);  
plotSmooth.PlotSignalXY(  
    x, y, Color.Violet, 5  
);
```

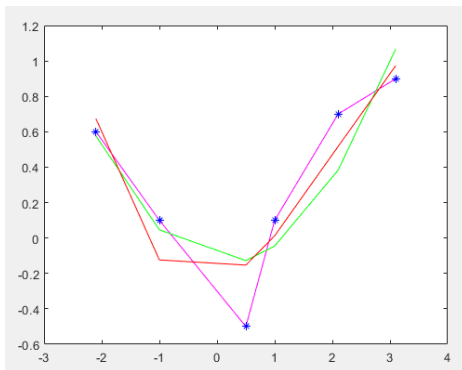
побудувати
графіки заданої
функції та
апроксимуючих
функцій для
значень x , що
змінюються від
-2.1 до 3.1 з
кроком 0.05

```
xp = -2.1:0.05:3.1
```

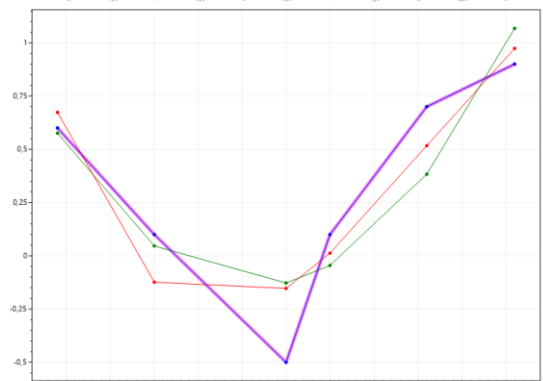
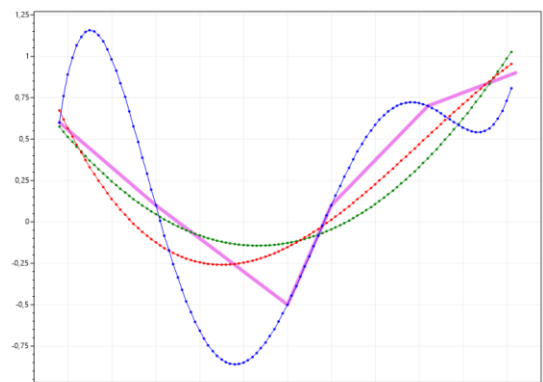
```
>> plot(  
    X, Y, '-m',  
    xp, polyval(pf2, xp), '-g',  
    xp, polyval(pf3, xp), '-r',  
    xp, polyval(pf5, xp), '*b'  
)
```



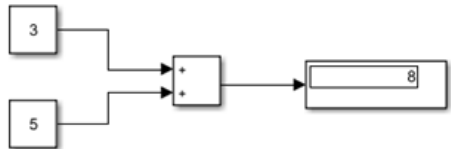



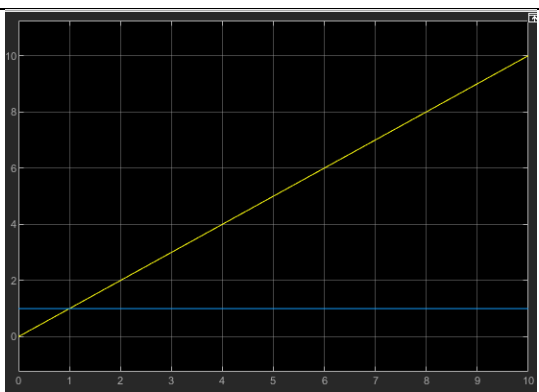
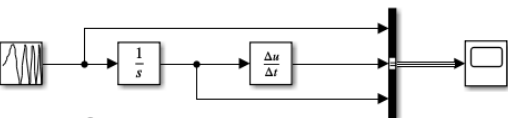
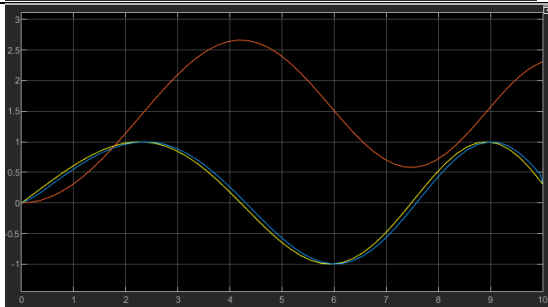
```
>> plot(  
    X, Y, '-m',  
    X, polyval(pf2, X), '-g',  
    X, polyval(pf3, X), '-r',  
    X, polyval(pf5, X), '*b'  
)
```

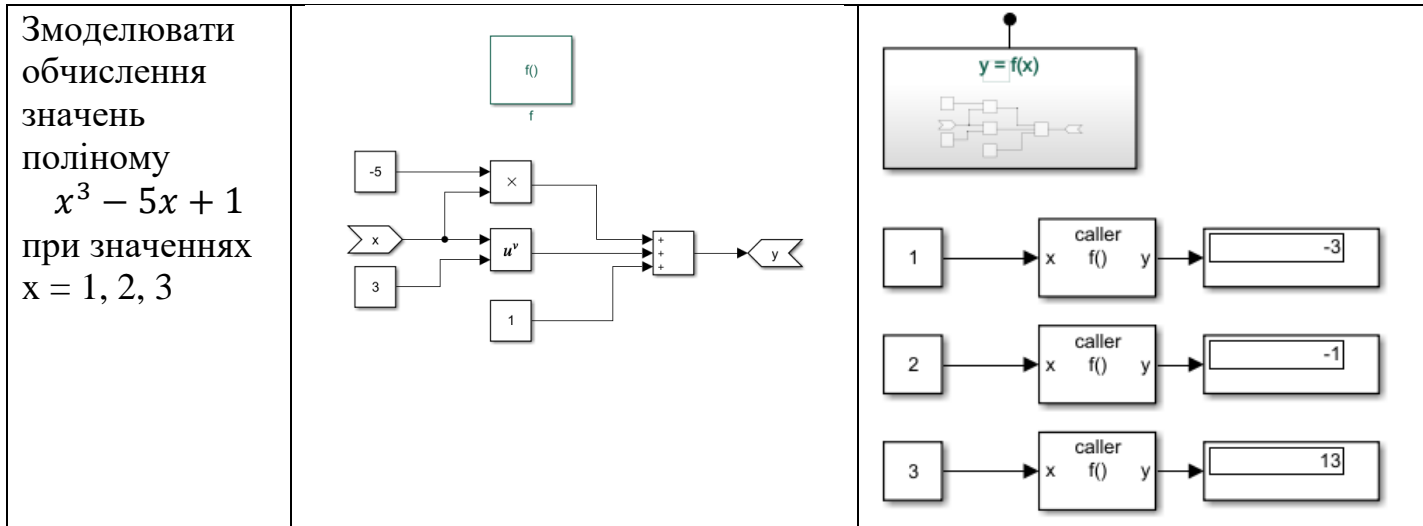


```
OutPolyFitVal(  
    x, y, 2, plotPoints, plotSmooth,  
    0.05, Color.Green  
);  
OutPolyFitVal(  
    x, y, 3, plotPoints, plotSmooth,  
    0.05, Color.Red  
);  
OutPolyFitVal(  
    x, y, 5, plotPoints, plotSmooth,  
    0.05, Color.Blue  
);  
ShowPlotForm(  
    plotPoints.Render(),  
    plotSmooth.Render()  
);
```



4. Побудова моделей у Simulink.

Завдання	Модель Simulink	Результат виконання
Змоделювати суму і добуток двох чисел		
Змоделювати обчислення значення $\cos(2)$		
Змоделювати обчислення функції e^x при $x = 2$		
Змоделювати інтегрування константи		
Змоделювати інтегрування пилоподібного сигналу із наступним його диференціюванням		



Висновки: в ході роботи було здобуто навички використання математичного середовища MATLAB та середовища для математичного моделювання Simulink, проведено ряд простих та матричних обчислень, побудовано графіки функцій, апроксимовано вектори. Досліджену поведінку математичного середовища було відтворено за допомогою мови програмування C#.