https://www.halvorsen.blog

Simulation and Control with C# and WinForms

Hans-Petter Halvorsen

Contents

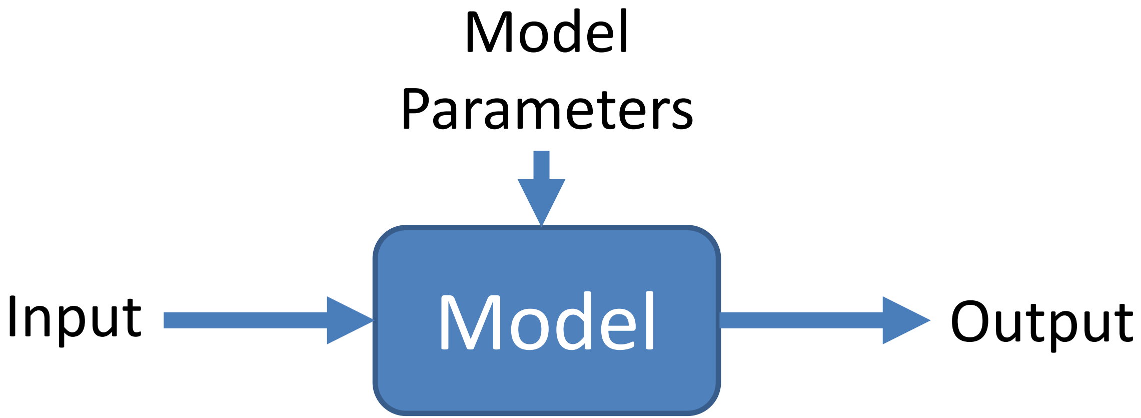
- What is a Model?
- C# WinForms Examples
- Timer
- Plotting
- Controller

Finally, we will end up with basic Control System, where we control a Dynamic System using a Mathematical Model

Audience

- This Tutorial is made for rookies making their first basic C# Win Forms Application
- You don’t need any experience in either Visual Studio or C#
- No skills in Automation or Control System is necessary

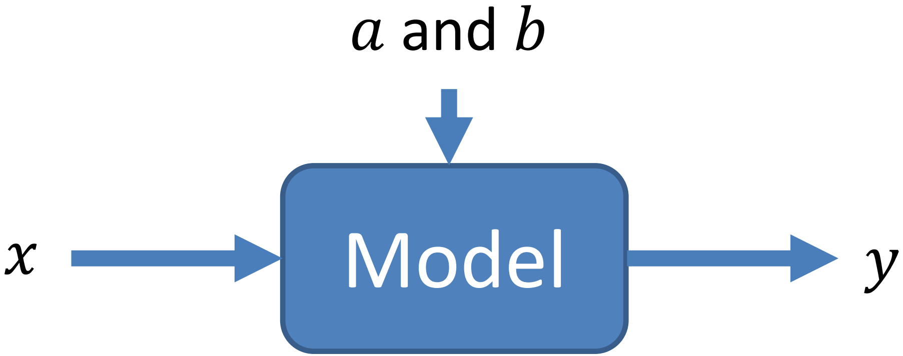
Model



Model Example

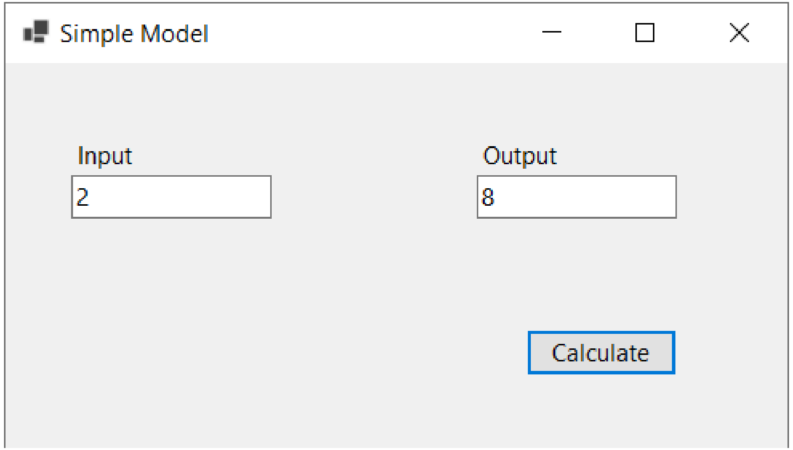
Simple Model: $y = ax + b$

This is 1. order linear model



WinForms App

$$y = ax + b$$



Input: 2

Output: 8

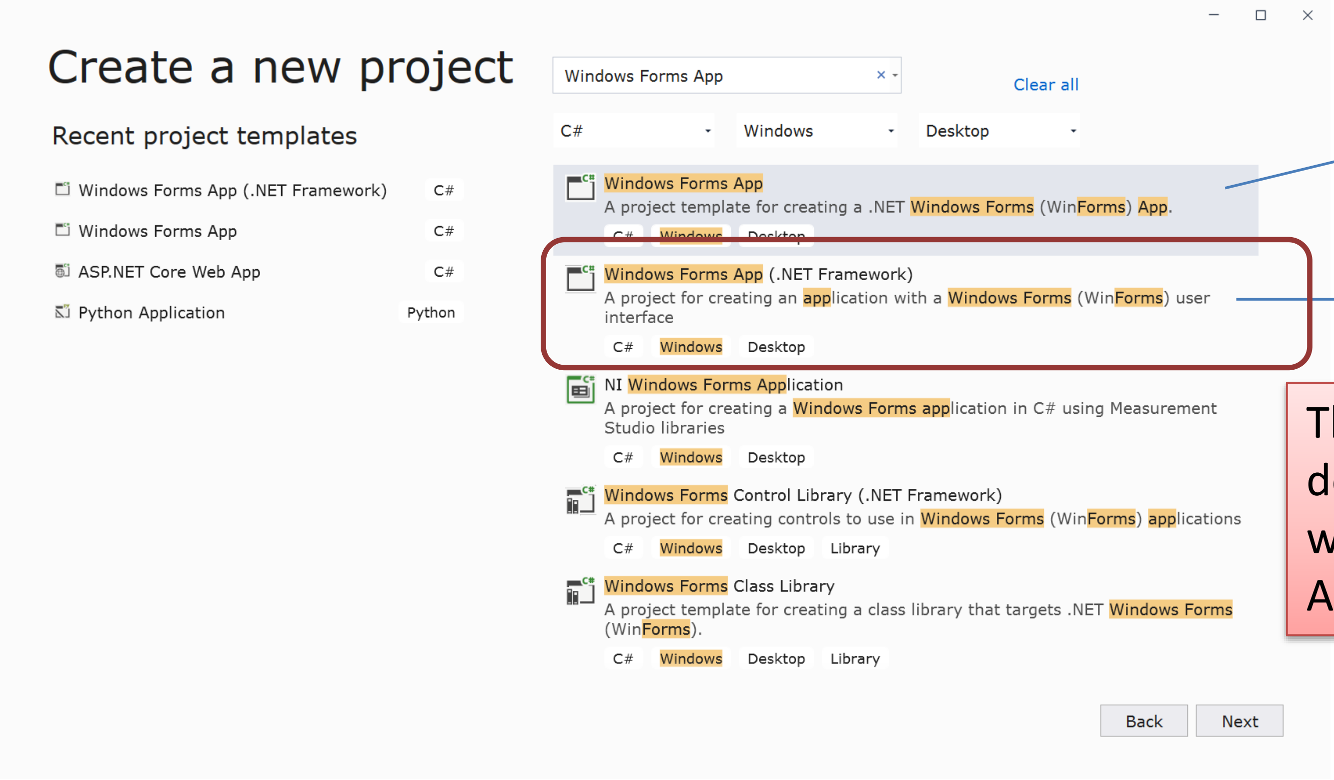
Example:

$$a = 2$$
$$b = 4$$

$$y = 2x + 4$$

$$y(2) = 2 \cdot 2 + 4 = 8$$

Create Project



Create a new project

Recent project templates

Windows Forms App (.NET Framework)	C#
Windows Forms App	C#
ASP.NET Core Web App	C#
Python Application	Python

Windows Forms App

Clear all

C# · Windows · Desktop

Windows Forms App
A project template for creating a .NET Windows Forms (WinForms) App.
C# Windows Desktop

Windows Forms App (.NET Framework)
A project for creating an application with a Windows Forms (WinForms) user interface
C# Windows Desktop

NI Windows Forms Application
A project for creating a Windows Forms application in C# using Measurement Studio libraries
C# Windows Desktop

Windows Forms Control Library (.NET Framework)
A project for creating controls to use in Windows Forms (WinForms) applications
C# Windows Desktop Library

Windows Forms Class Library
A project template for creating a class library that targets .NET Windows Forms (WinForms).
C# Windows Desktop Library

New .NET 5

.NET Framework 4.x

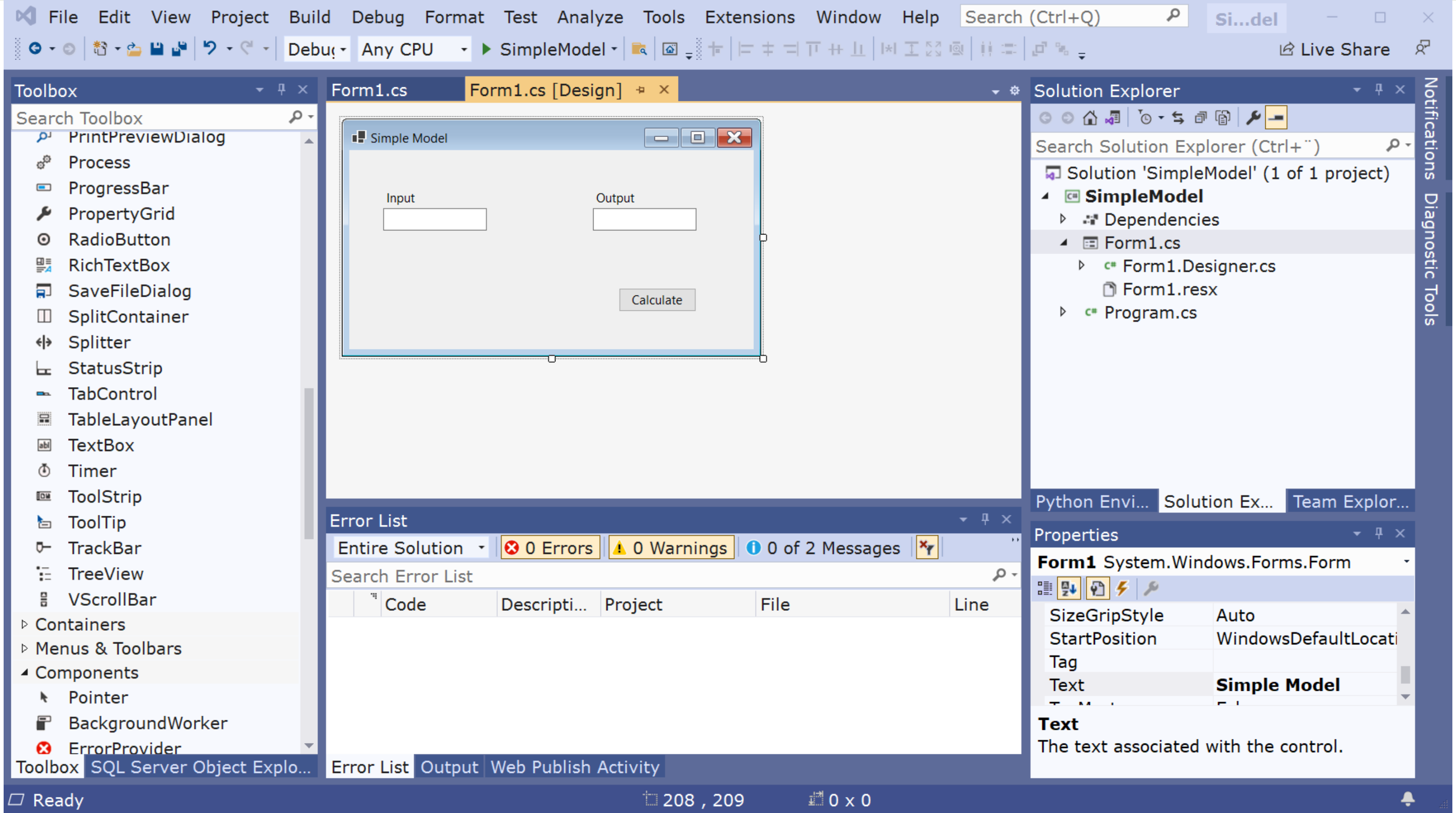
The Chart component so far does not exist for .NET 5, so we select " Windows Forms App (.NET Framework)"

C# Examples

Note!

- The examples provided can be considered as a “proof of concept”
- The sample code is very simplified for clarity and doesn't necessarily represent best practices.

Visual Studio Project



SizeGripStyle	Auto
StartPosition	WindowsDefaultLocati
Tag	
Text	**Simple Model**

Text

The text associated with the control.

C# Code

```
using System;
using System.Windows.Forms;

namespace SimpleModel
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void btnCalculate_Click(object sender, EventArgs e)
        {
            double x, y;

            x = Convert.ToDouble(txtInput.Text);

            y = LinearModel(x);
            txtOutput.Text = y.ToString();
        }

        double LinearModel(double x)
        {
            double a = 2;
            double b = 4;
            double y;

            y = a * x + b;
            return y;
        }
    }
}
```

C# Code

```
double LinearModel(double x)
{
    double a = 2;
    double b = 4;
    double y;

    y = a * x + b;
    return y;
}
```

C# Code

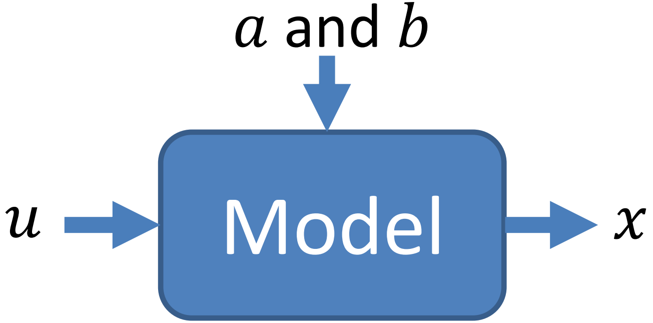
```
private void btnCalculate_Click(object sender, EventArgs e)
{
          double x, y;

          x = Convert.ToDouble(txtInput.Text);

          y = LinearModel(x);

          txtOutput.Text = y.ToString();
}
```

Simulations of Dynamic Systems

Dynamic Model

In this example we will use the following 1. order differential equation:

$$\dot{x} = -ax + bu$$



Note that $\dot{x} = \frac{dx}{dt} = x'(t)$

Different notation is used in different textbooks and examples

In order to simulate such model with C#, we need to find a **discrete** version

Discretization

- In order to simulate this system, we typically need to find the discrete differential equation (difference equation)
- We can use e.g., the **Euler** Approximation:

$$\dot{x} \approx \frac{x(k+1) - x(k)}{T_s}$$

Where T_s is the Sampling Time

Discrete Model

We have the continuous differential equation: $\dot{x} = -ax + bu$

We apply Euler: $\dot{x} \approx \frac{x(k+1)-x(k)}{T_s}$

Then we get:

$$\frac{x(k+1) - x(k)}{T_s} = -ax(k) + bu(k)$$

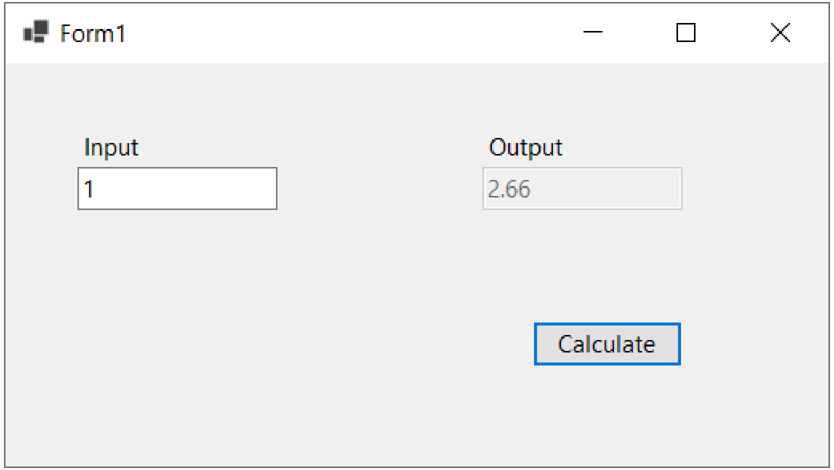
This gives the following discrete differential equation (difference equation):

$$x(k+1) = (1 - T_s a)x(k) + T_s bu(k)$$

This equation can easily be implemented in any text-based programming language

WinForms App

$$x(k+1) = (1 - T_s a)x(k) + T_s b u(k)$$



Input: 1

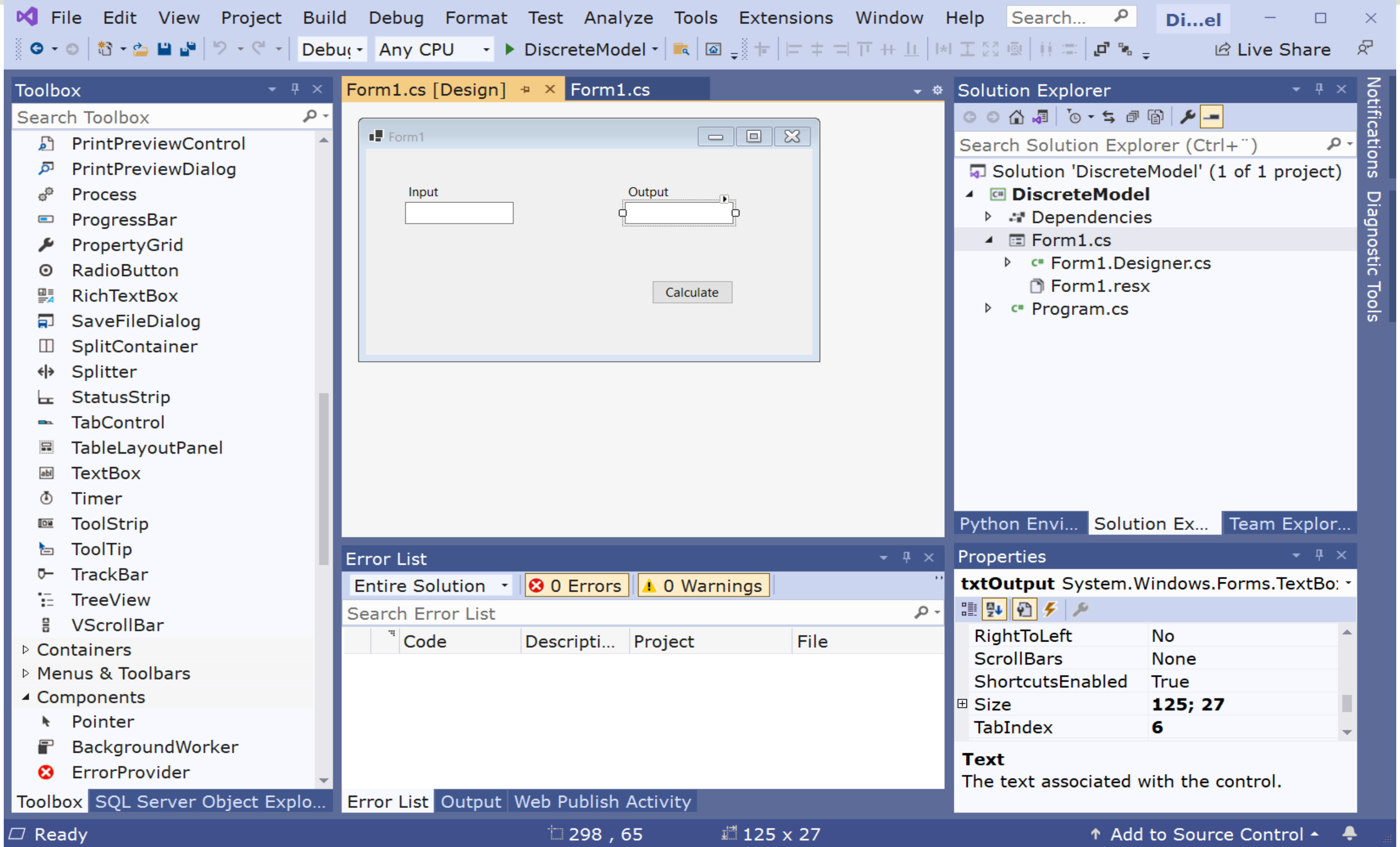
Output: 2.66

Every time we click “Calculate”, a new updated value of $x(k+1)$ is calculated

Example:

$$a = 0.25$$
$$b = 2$$

Visual Studio Project



txtOutput System.Windows.Forms.TextBo:

RightToLeft	No
ScrollBars	None
ShortcutsEnabled	True
⊞ Size	**125; 27**
TabIndex	**6**

Text
The text associated with the control.

C# Code

```
using System;
using System.Windows.Forms;

namespace DiscreteModel
{
    public partial class Form1 : Form
    {
        double x = 0;
        double u=1;
        double Ts = 0.1;

        public Form1()
        {
            InitializeComponent();
            txtInput.Text = "1";
        }

        private void btnCalculate_Click(object sender, EventArgs e)
        {
            u = Convert.ToDouble(txtInput.Text);
            x = DiscreteModel(x, u);
            txtOutput.Text = x.ToString("0.##");
        }

        double DiscreteModel(double xk, double u)
        {
            double a = 0.25;
            double b = 2;
            double xk1;
            xk1 = (1-Ts*a) * xk + Ts*b*u;
            return xk1;
        }
    }
}
```

Discrete Model - C# Code

$$x(k+1) = (1 - T_s a)x(k) + T_s b u(k)$$

```
double DiscreteModel(double xk, double u)
{
     double a = 0.25;
     double b = 2;
     double xk1;

     xk1 = (1-Ts*a) * xk + Ts*b*u;
     return xk1;
}
```

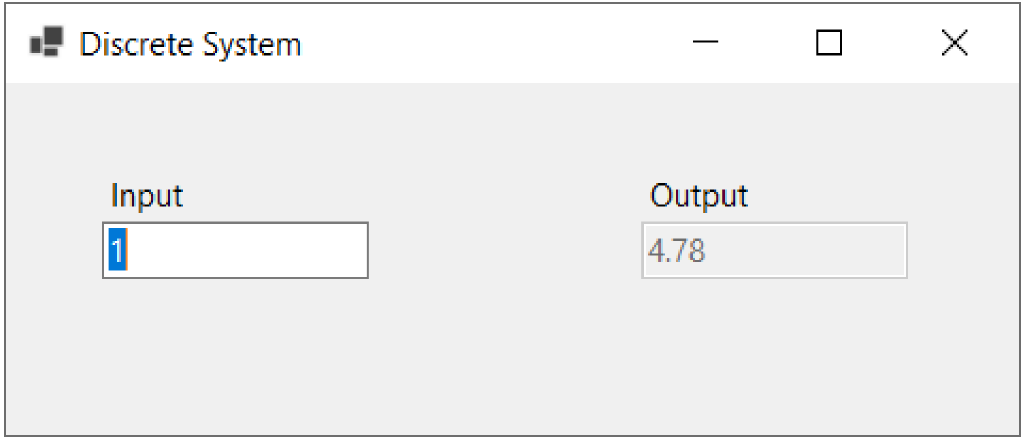
C# Code

```
using System;
using System.Windows.Forms;

namespace DiscreteModel
{
    public partial class Form1 : Form
    {
        double x = 0;
        double u = 1;
        double Ts = 0.1;

        public Form1()
        {
            InitializeComponent();
            txtInput.Text = "1";
        }

        private void btnCalculate_Click(object sender, EventArgs e)
        {
            u = Convert.ToDouble(txtInput.Text);
            x = DiscreteModel(x, u);
            txtOutput.Text = x.ToString("0.##");
        }

        double DiscreteModel(double xk, double u)
        {
            double a = 0.25;
            double b = 2;
            double xk1;
            xk1 = (1-Ts*a) * xk + Ts*b*u;
            return xk1;
        }
    }
}
```

Using a Timer

Timer

So far, we needed to push the “Calculate” Button in order to calculate a new updated value. We need to find a better solution, so we introduce and using a Timer



Discrete System

Input: 1

Output: 4.78

Here we have removed the button and using a Timer instead. A Timer is like a While Loop

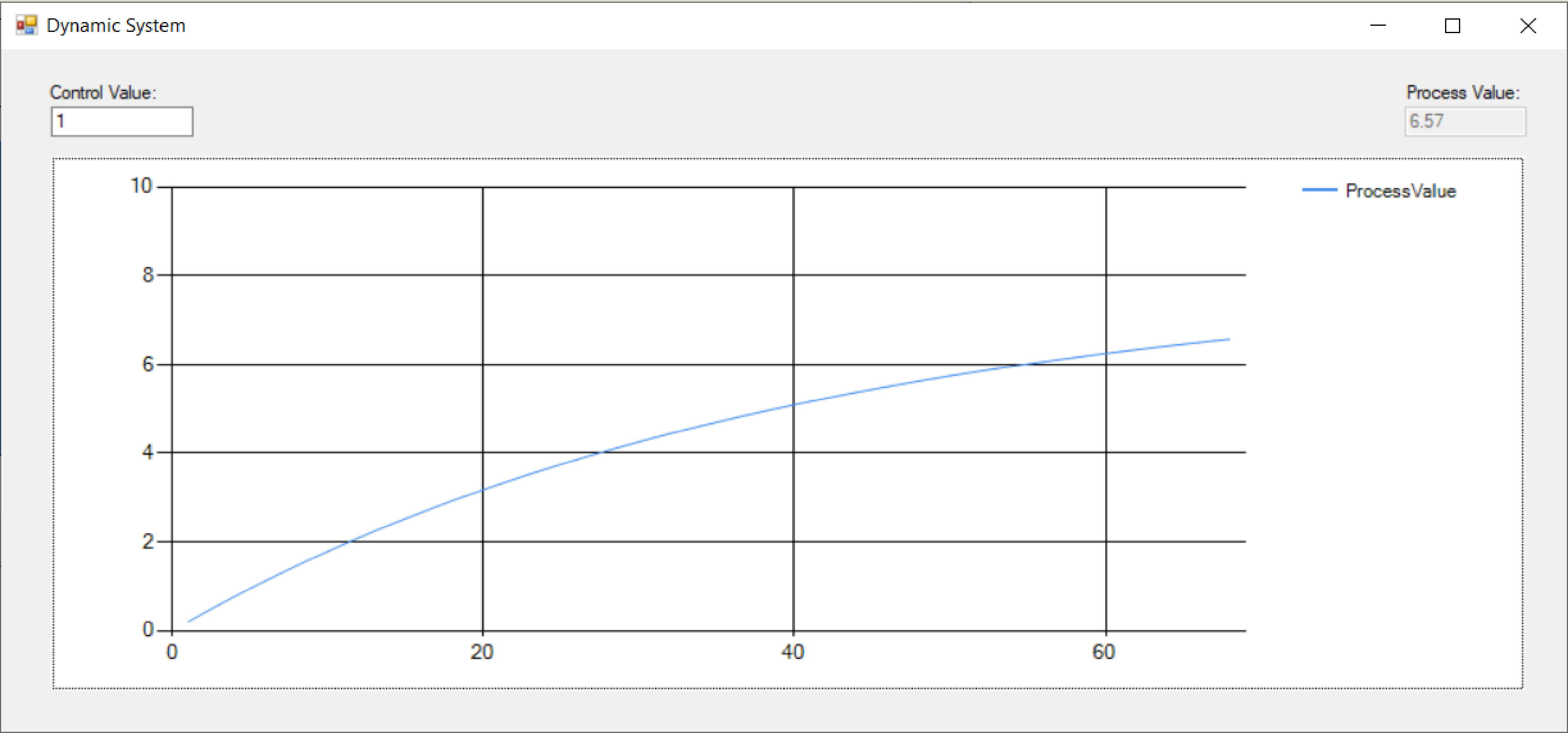
Timer - C# Code

We move the code from the Button Event Handler to the Timer Event Handler:

```
private void timerSimulationLoop_Tick(object sender, EventArgs e)
{
      u = Convert.ToDouble(txtInput.Text);

      x = DiscreteModel(x, u);

      txtOutput.Text = x.ToString("0.##");
}
```

Plotting

WinForms App

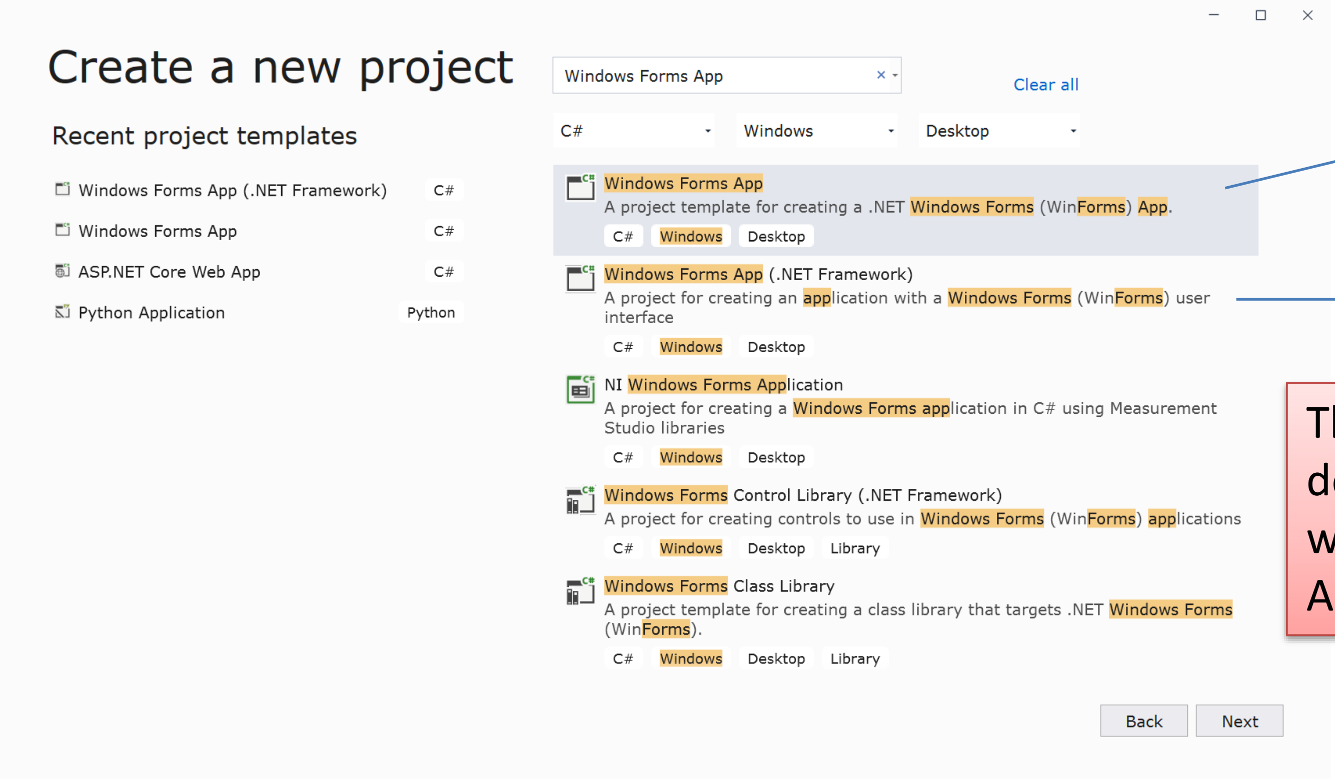


Dynamic System

Control Value: 1

Process Value: 6.57

Create Project



Create a new project

Windows Forms App

Clear all

C# | Windows | Desktop

Recent project templates

Windows Forms App (.NET Framework)	C#
Windows Forms App	C#
ASP.NET Core Web App	C#
Python Application	Python

Windows Forms App
A project template for creating a .NET Windows Forms (WinForms) App.
C# Windows Desktop

Windows Forms App (.NET Framework)
A project for creating an application with a Windows Forms (WinForms) user interface
C# Windows Desktop

NI Windows Forms Application
A project for creating a Windows Forms application in C# using Measurement Studio libraries
C# Windows Desktop

Windows Forms Control Library (.NET Framework)
A project for creating controls to use in Windows Forms (WinForms) applications
C# Windows Desktop Library

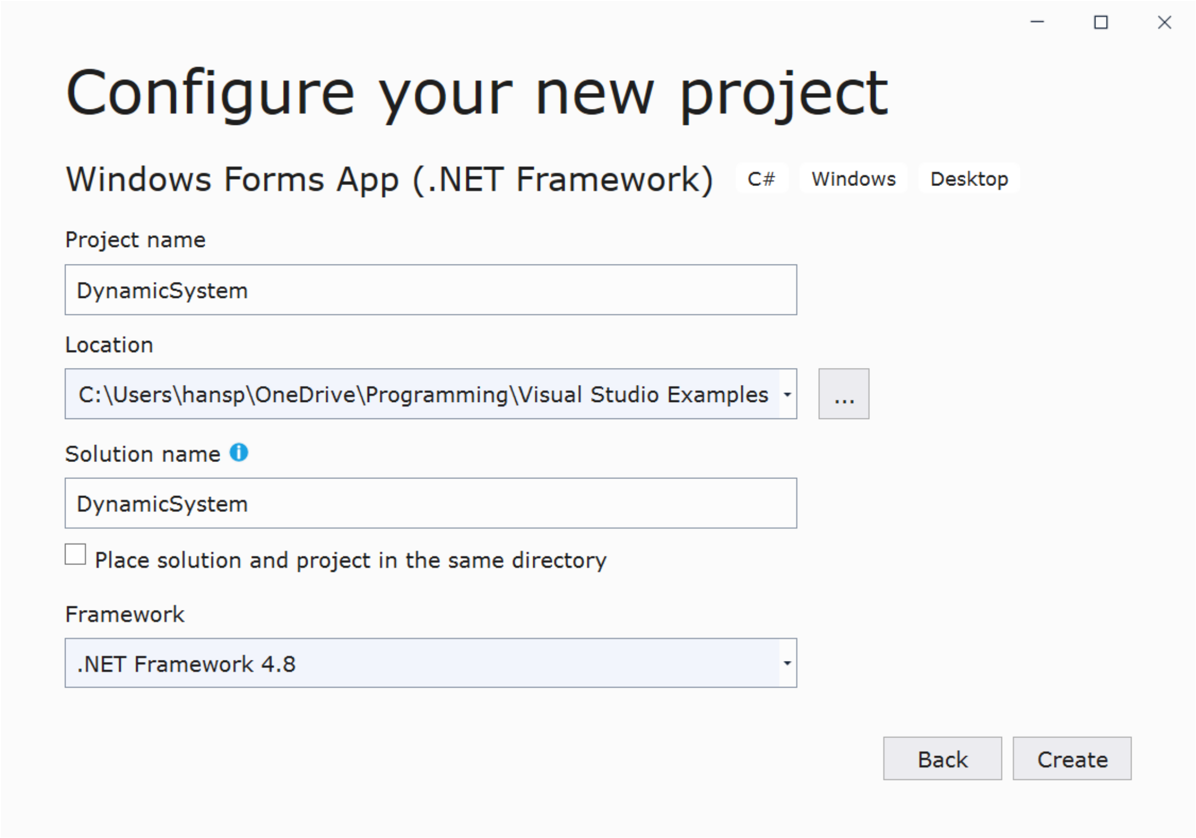
Windows Forms Class Library
A project template for creating a class library that targets .NET Windows Forms (WinForms).
C# Windows Desktop Library

New .NET 5

.NET Framework 4.x

The Chart component so far does not exist for .NET 5, so we select “ Windows Forms App (.NET Framework)”

Create Project



Configure your new project

Windows Forms App (.NET Framework)

Project name

DynamicSystem

Location

C:\Users\hansp\OneDrive\Programming\Visual Studio Examples

Solution name

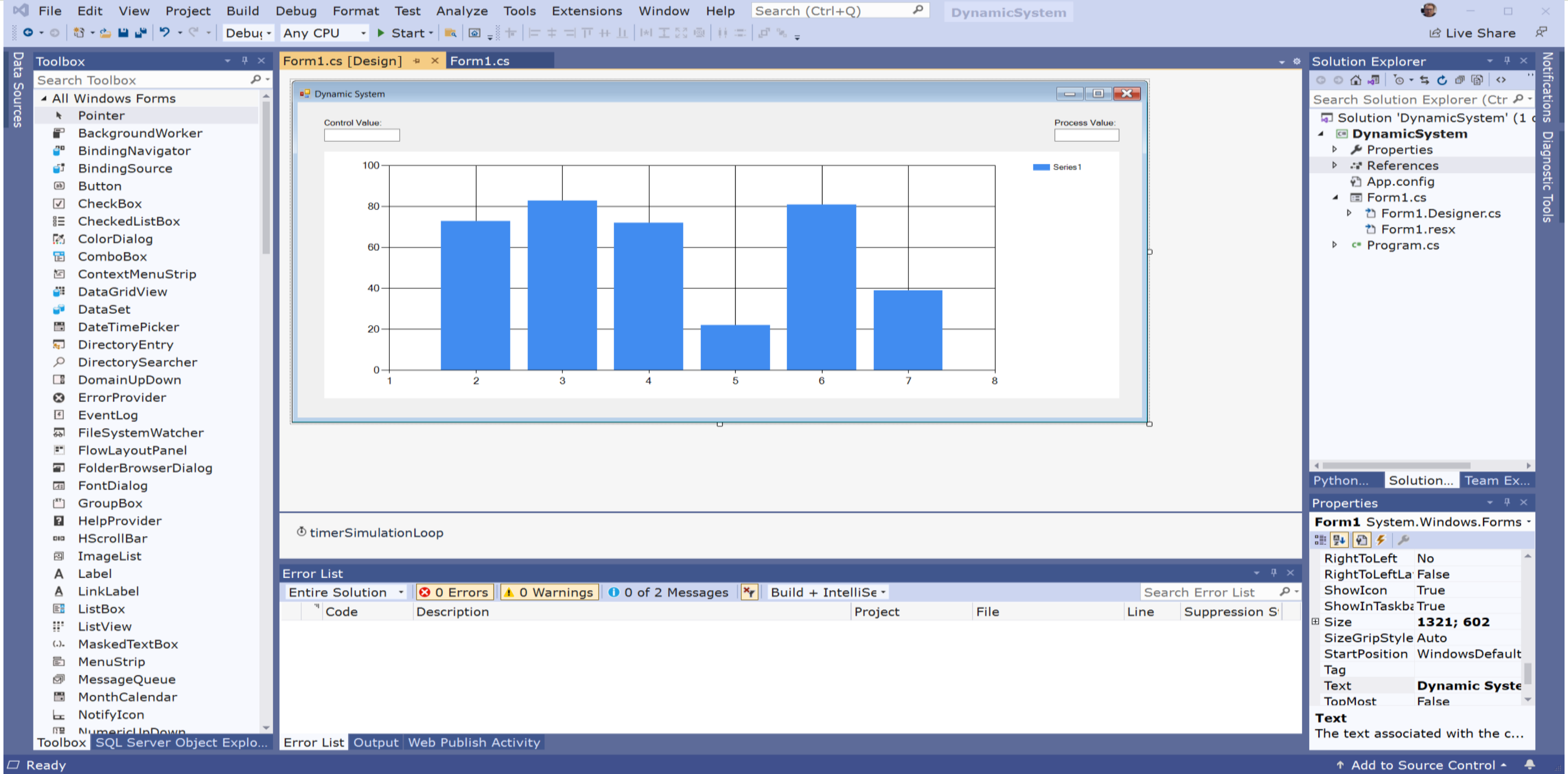
DynamicSystem

□ Place solution and project in the same directory

Framework

.NET Framework 4.8

Visual Studio Project



Form1 System.Windows.Forms

RightToLeft	No
RightToLeftLa	False
ShowIcon	True
ShowInTaskba	True
Size	**1321; 602**
SizeGripStyle	Auto
StartPosition	WindowsDefault
Tag	
Text	**Dynamic Syste**
TopMost	False

Text
The text associated with the c...

C# Code

```
using System;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;

namespace DynamicSystem
{
    public partial class Form1 : Form
    {
        double processValue = 0; double controlValue = 1; double Ts = 0.1;
        public Form1()
        {
            InitializeComponent();
            txtControlValue.Text = "1";

            chartMeasurementData.Series.Clear();
            chartMeasurementData.Series.Add("ProcessValue");
            chartMeasurementData.Series["ProcessValue"].ChartType = SeriesChartType.Line;
            ChartArea area1 = chartMeasurementData.ChartAreas[0];
            area1.AxisY.Minimum = 0;
            area1.AxisY.Maximum = 10;

            timerSimulationLoop.Interval = 1000;
            timerSimulationLoop.Start();
        }

        double DiscreteModel(double xk, double u)
        {
            double a = 0.25; double b = 2; double xk1;
            xk1 = (1 - Ts * a) * xk + Ts * b * u;
            return xk1;
        }

        private void timerSimulationLoop_Tick(object sender, EventArgs e)
        {
            controlValue = Convert.ToDouble(txtControlValue.Text);
            processValue = DiscreteModel(processValue, controlValue);
            txtProcessValue.Text = processValue.ToString("0.##");
            chartMeasurementData.Series["ProcessValue"].Points.AddY(processValue);
        }
    }
}
```

Improvements

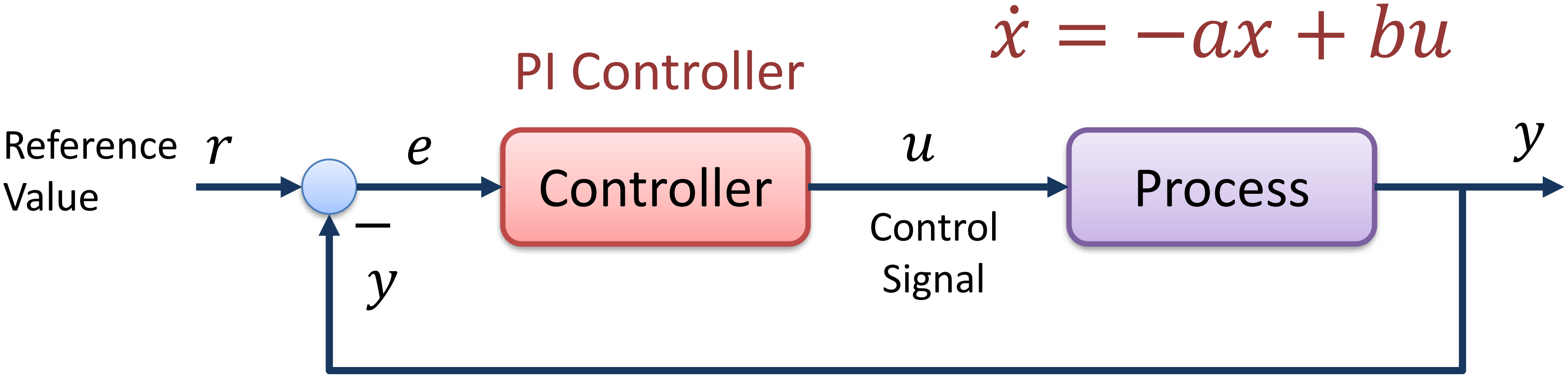
- Improve GUI
- Add Units
- Create and use a PID Controller
- Etc.

Controller

Control System

- We have created and used a Mathematical Model
- Next step is to create a Control System
- We will implement a PI Controller

Control System



PI Controller

The PI Controller is given by:

$$u(t) = K_p e(t) + \frac{K_p}{T_i}\int_0^t e d\tau$$

Where u is the controller output and e is the control error:

$$e(t) = r(t) - y(t)$$

Where r is the reference signal (setpoint)

Or PI Controller on Transfer Function form (we use Laplace):

$$u(s) = K_p e(s) + \frac{K_p}{T_i s} e(s)$$

In order to implement the PI controller in our C# program, we need to make a discrete version

PI Controller

The PI Controller is given by:

$$u(s) = K_p e(s) + \frac{K_p}{T_i s} e(s)$$

We set $z = \frac{1}{s} e \Rightarrow sz = e \Rightarrow \dot{z} = e$

This gives:

$$\dot{z} = e$$

$$u = K_p e + \frac{K_p}{T_i} z$$

This is the PI controller on state-space form

Discrete PI Controller

Using Euler:

$$\dot{z} \approx \frac{z_{k+1} - z_k}{T_s}$$

Where T_s is the Sampling Time.

This gives:

$$\frac{z_{k+1} - z_k}{T_s} = e_k$$

$$u_k = K_p e_k + \frac{K_p}{T_i} z_k$$

Finally, we get the following discrete PI controller:

$$e_k = r_k - y_k$$

$$u_k = K_p e_k + \frac{K_p}{T_i} z_k$$

$$z_{k+1} = z_k + T_s e_k$$

This algorithm can easily be implemented in C#.

Control System Implementation

PI Controller

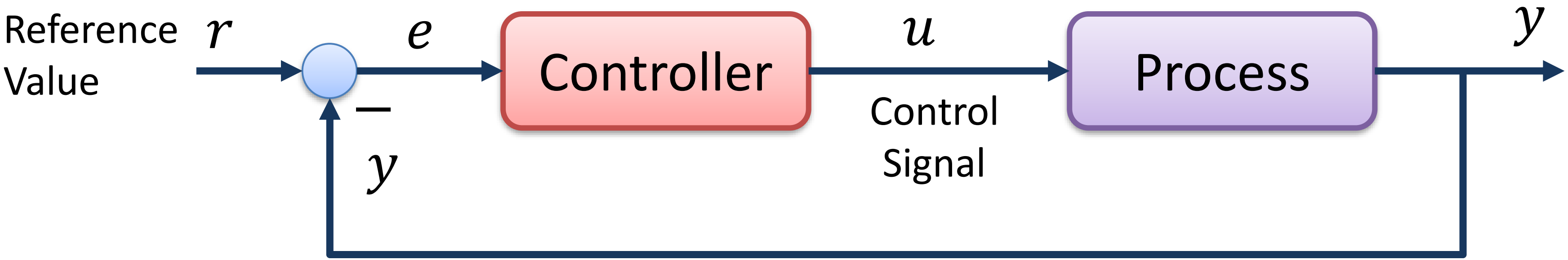
$$e_k = r_k - y_k$$

$$u_k = K_p e_k + \frac{K_p}{T_i} z_k$$

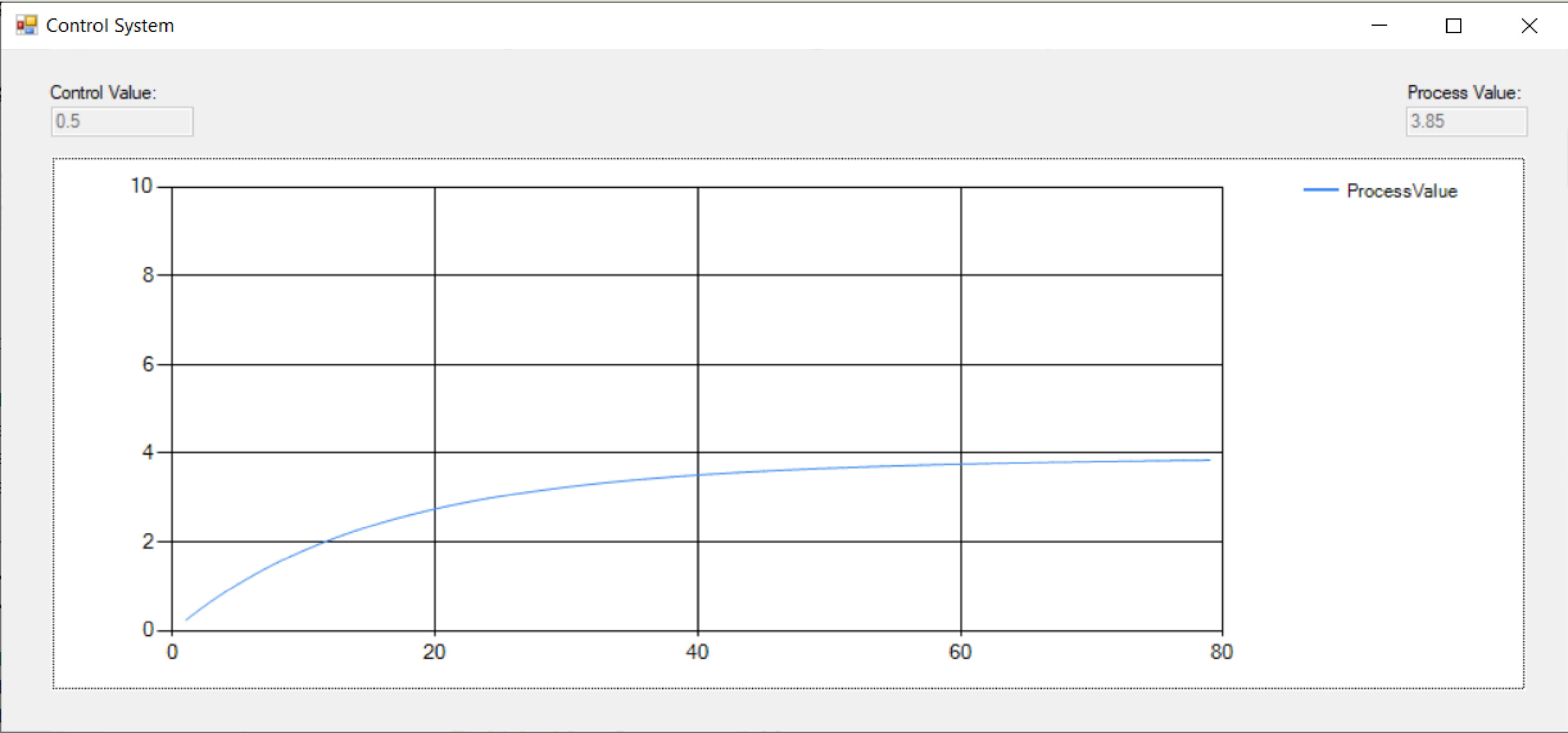
$$z_{k+1} = z_k + T_s e_k$$

Discrete Mathematical Model

$$x(k+1) = (1 - T_s a)x(k) + T_s bu(k)$$



WinForms App



Control Value:
0.5

Process Value:
3.85

C# Code

```
using System;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;

namespace DynamicSystem
{
    public partial class Form1 : Form
    {
        double processValue = 0; controlValue = 0;
        double Ts = 0.1;
        double Kp = 0.3; Ti = 5; r = 4;
        double u=0, z = 0;

        public Form1()
        {
            InitializeComponent();

            chartMeasurementData.Series.Clear();
            chartMeasurementData.Series.Add("ProcessValue");
            chartMeasurementData.Series["ProcessValue"].ChartType = SeriesChartType.Line;
            ChartArea area1 = chartMeasurementData.ChartAreas[0];
            area1.AxisY.Minimum = 0; area1.AxisY.Maximum = 10;

            timerSimulationLoop.Interval = 1000;
            timerSimulationLoop.Start();
        }

        private void timerSimulationLoop_Tick(object sender, EventArgs e)
        {
            controlValue = PiController(processValue);
            processValue = DiscreteModel(processValue, controlValue);

            txtProcessValue.Text = processValue.ToString("0.##");
            txtControlValue.Text = controlValue.ToString("0.##");

            chartMeasurementData.Series["ProcessValue"].Points.AddY(processValue);
        }

        double DiscreteModel(double xk, double u)
        {
            double a = 0.25; double b = 2; double xk1;
            xk1 = (1 - Ts * a) * xk + Ts * b * u;
            return xk1;
        }

        double PiController(double y)
        {
            double e  = r - y;
            u = Kp * e + (Kp / Ti) * z;
            z = z + Ts * e;
            return u;
        }
    }
}
```

C# Code

```
...

namespace DynamicSystem
{
    public partial class Form1 : Form
    {
        double processValue = 0; controlValue = 0;
        double Ts = 0.1;
        double Kp = 0.3; Ti = 5; r = 4;
        double u=0, z = 0;

        ...

        double PiController(double y)
        {
            double e   = r - y;
            u = Kp * e + (Kp / Ti) * z;
            z = z + Ts * e;
            return u;
        }
    }
}
```

$$e_k = r_k - y_k$$

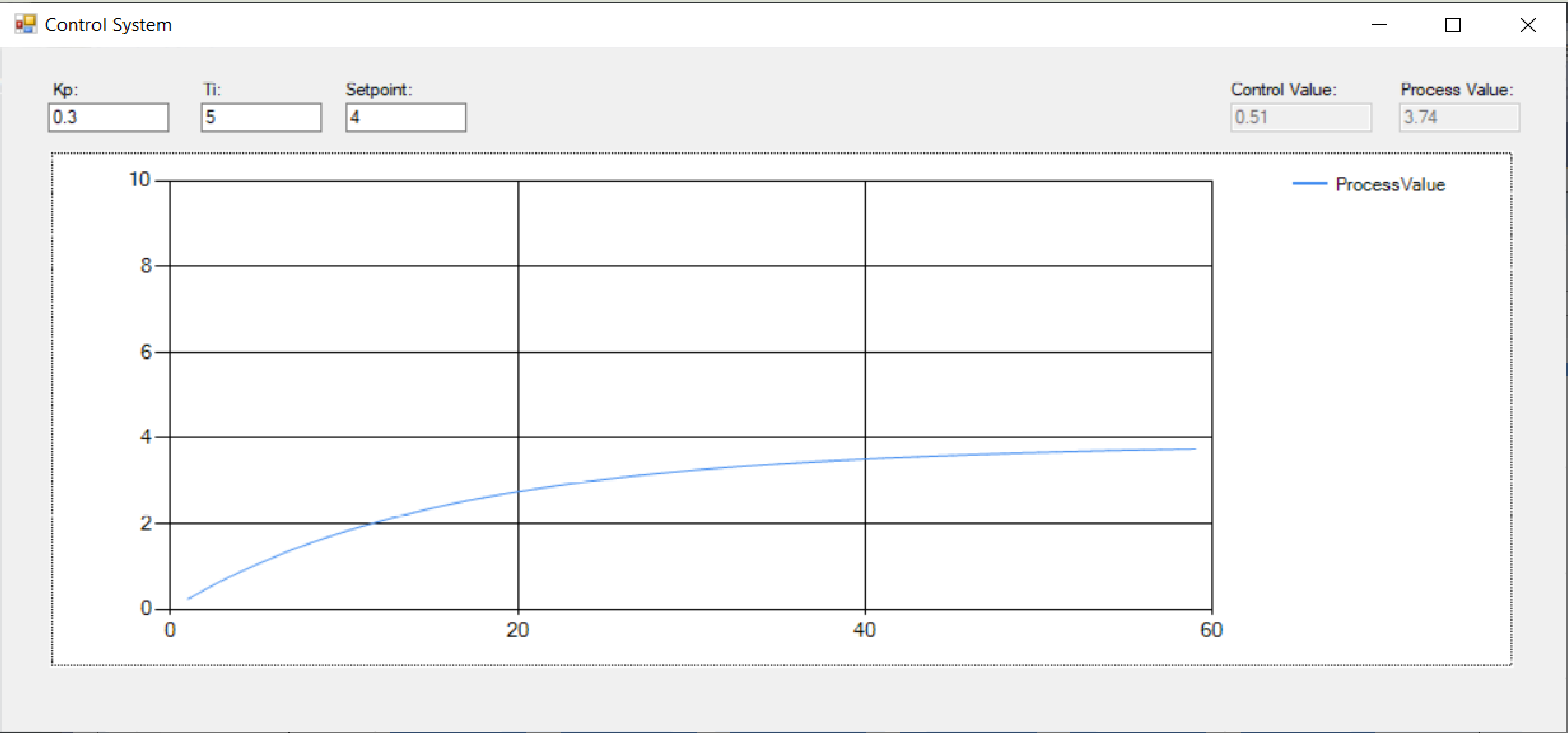
$$u_k = K_p e_k + \frac{K_p}{T_i} z_k$$

$$z_{k+1} = z_k + T_s e_k$$

Improvement

- Possible to set Reference Value (r) from GUI
- Possible to set K_p and T_i from GUI
- Plot Control Value
- Add Units
- Improve GUI in general
- Add separate Classes for Controller, etc.
- Etc.

WinForms App



Control System

Kp: 0.3

Ti: 5

Setpoint: 4

Control Value: 0.51

Process Value: 3.74

C# Code

```
using System;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization.Charting;

namespace DynamicSystem
{
    public partial class Form1 : Form
    {
        double processValue = 0;
        double controlValue = 0;
        double Ts = 0.1;

        //PI Controller
        double Kp = 0.3;
        double Ti = 5;
        double r = 4;
        double u=0, z = 0;

        public Form1()
        {
            InitializeComponent();

            // Default Values
            txtKp.Text = Kp.ToString();
            txtTi.Text = Ti.ToString();
            txtR.Text = r.ToString();


            //Chart Initialization
            chartMeasurementData.Series.Clear();
            chartMeasurementData.Series.Add("ProcessValue");
            chartMeasurementData.Series["ProcessValue"].ChartType = SeriesChartType.Line;

            ChartArea area1 = chartMeasurementData.ChartAreas[0];
            area1.AxisY.Minimum = 0;
            area1.AxisY.Maximum = 10;

            //Timer Initialization
            timerSimulationLoop.Interval = 1000;
            timerSimulationLoop.Start();
        }

        private void timerSimulationLoop_Tick(object sender, EventArgs e)
        {
            //Control System
            controlValue = PiController(processValue);
            processValue = DiscreteModel(processValue, controlValue);

            //Update GUI
            txtProcessValue.Text = processValue.ToString("0.##");
            txtControlValue.Text = controlValue.ToString("0.##");

            //Plot Data
            chartMeasurementData.Series["ProcessValue"].Points.AddY(processValue);
        }
```

```
        double DiscreteModel(double xk, double u)
        {
            double a = 0.25; double b = 2; double xk1;

            xk1 = (1 - Ts * a) * xk + Ts * b * u;

            return xk1;
        }

        double PiController(double y)
        {
            double e  = r - y;
            u = Kp * e + (Kp / Ti) * z;
            z = z + Ts * e;

            return u;
        }

        private void txtKp_TextChanged(object sender, EventArgs e)
        {
            Kp = Convert.ToDouble(txtKp.Text);
        }

        private void txtTi_TextChanged(object sender, EventArgs e)
        {
            Ti = Convert.ToDouble(txtTi.Text);
        }

        private void txtR_TextChanged(object sender, EventArgs e)
        {
            r = Convert.ToDouble(txtR.Text);
        }

    }
}
```

Summary

- We started to implement a Process Model
- Then we added a Timer and a Chart
- Then we added a PI Controller and a Control System
- There are still lots of Improvements to make, including improvements with Code, GUI and tuning the Control System, etc.
- The examples provided can be considered as a “proof of concept”
- The sample code is very simplified for clarity and doesn't necessarily represent best practices.

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

E-mail: hans.p.halvorsen@usn.no

Web: https://www.halvorsen.blog