Exercise 11: Discretization (Solutions)

Two commonly used methods for discretization are as follows:

Euler forward method:

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

Euler backward method:

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

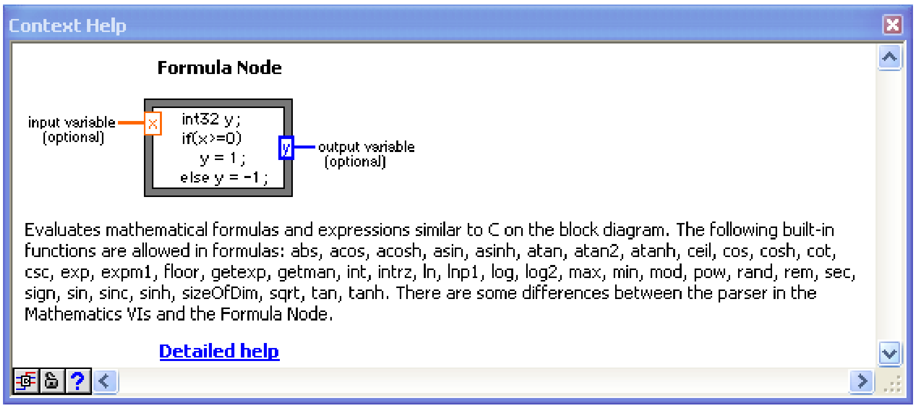
We will use these methods both in pen and paper exercises and in practical implementation in MathScript/LabVIEW along with built-in discretization methods.

Discretization in MathScript:

In MathScript we can use the function c2d() to convert from a continuous system to a discrete system.

Discretization in LabVIEW:

A Formula Node in LabVIEW evaluates mathematical formulas and expressions similar to C on the block diagram. In this way you may use existing C code directly inside your LabVIEW code. It is also useful when you have “complex” mathematical expressions.



Task 1: Discrete system

Given the following system:

$$\dot{x}_1 = K_p u - x_2$$

$$\dot{x}_2 = 0$$

$$y = x_1$$

Task 1.1

Find the discrete system and set it on state-space form (using "pen and paper).

Use Euler forward:

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

Solution:

The discrete version becomes:

$$\frac{x_1(k+1) - x_1(k)}{T_s} = K_p u(k) - x_2(k)$$

$$\frac{x_2(k+1) - x_2(k)}{T_s} = 0$$

$$y(k) = x_1(k)$$

This gives:

$$x_1(k+1) = x_1(k) + T_s K_p u(k) - T_s x_2(k)$$

$$x_2(k+1) = x_2(k)$$

$$y(k) = x_1(k)$$

You may also set the system on state-space form:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & -T_s \\ 0 & 1 \end{bmatrix}}_{A} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \underbrace{\begin{bmatrix} T_s K_p \\ 0 \end{bmatrix}}_{B} u(k)$$

$$y(k) = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{C} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \underbrace{[0]}_{\mathrm{D}} u(k)$$

Task 1.2

Define the continuous state-space model ("pen and paper") and then implement the continuous state-space model in MathScript. Then use MathScript to find the discrete state-space model. Compare the result from the previous task.

Set $K_p = 1$ and $T_s = 0.1$

Solution:

Continuous state-space model:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & -1 \\ 0 & 0 \end{bmatrix}}_{A} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \underbrace{\begin{bmatrix} K_p \\ 0 \end{bmatrix}}_{B} u$$

$$y = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{C} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \underbrace{[0]}_{\mathrm{D}} u$$

Which can be easily implemented in MathScript.

MathScript Code:

```
clear
clc

Kp=1;

A = [0 -1; 0 0];
B = [Kp 0]';
C = [1 0];
D = [0];

ssmodel = ss(A, B, C, D);


% Discrete System:
Ts = 0.1;
ssmodel_discete = c2d(ssmodel, Ts, 'forward')
```

Which gives the same result as we did with “pen and paper”:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & -T_s \\ 0 & 1 \end{bmatrix}}_{A} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \underbrace{\begin{bmatrix} T_s K_p \\ 0 \end{bmatrix}}_{B} u(k)$$

$$y(k) = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{C} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \underbrace{[0]}_{\mathrm{D}} u(k)$$

Setting $K_p = 1$ and $T_s = 0.1$ gives:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & -0.1 \\ 0 & 1 \end{bmatrix}}_{A} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \underbrace{\begin{bmatrix} 0.1 \\ 0 \end{bmatrix}}_{B} u(k)$$

$$y(k) = \underbrace{\begin{bmatrix} 1 & 0 \end{bmatrix}}_{C} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \underbrace{[0]}_{\mathrm{D}} u(k)$$

Note! We could also easily implemented this without using the built-in c2d() function, since the following yields in general:

A continuous state-space model:

$$\dot{x} = Ax + Bu$$

$$y = Cx + Du$$

Euler forward:

$$\dot{x} \approx \frac{x_{k+1} - x_k}{T_s}$$

Using this in general gives:

$$\frac{x_{k+1} - x_k}{T_s} = Ax_k + Bu_k$$

$$y_k = Cx_k + Du_k$$

This gives in general the following discrete stat-space model:

$$x_{k+1} = \underbrace{(I + T_s A)}_{A_d} x_k + \underbrace{T_s B}_{B_d} u_k$$

$$y_k = \underbrace{C}_{C_d} x_k + \underbrace{D}_{D_d} u_k$$

Where I is the identity matrix.

This mean we can find A_d and B_d in MathScript like this:

```
…
I = eye(2);

Ad = I + Ts*A
Bd = Ts*B
```

Which should give the same results.

Task 2: Discrete Controller

A controller is given by the following transfer function:

$$h_r(s) = \frac{u(s)}{e(s)} = \frac{2(s + 0.5)}{s + 1}$$

Task 2.1

Find the continuous differential equation.

Solutions:

We have:

$$h_r(s) = \frac{u(s)}{e(s)} = \frac{2(s + 0.5)}{s + 1}$$

This gives:

$$u(s)\,[1 + s] = [2s + 1]\,e(s)$$

Inverse Laplace gives:

$$u + \dot{u} = e + 2\dot{e}$$

Task 2.2

Find the discrete difference equation. Use the Euler <u>backward</u> method.

Solutions:

Using the Euler <u>backward</u> method gives:

$$u_k + \frac{u_k - u_{k-1}}{T_S} = e_k + \frac{2(e_k - e_{k-1})}{T_S}$$

Further:

$$\frac{T_S}{T_S} u_k + \frac{u_k - u_{k-1}}{T_S} = \frac{T_S}{T_S} e_k + \frac{2(e_k - e_{k-1})}{T_S}$$

And:

$$\frac{T_S}{T_S} u_k + \frac{1}{T_S} u_k - \frac{u_{k-1}}{T_S} = \frac{T_S}{T_S} e_k + \frac{2e_k}{T_S} - \frac{2\,e_{k-1}}{T_S}$$

And:

$$\frac{(T_S + 1)}{T_S} u_k = \frac{1}{T_S} u_{k-1} + \frac{T_S}{T_S} e_k + \frac{2}{T_S} e_k - \frac{2}{T_S} e_{k-1}$$

And:

$$u_k = \frac{u_{k-1}}{T_S + 1} + \frac{T_S}{(T_S + 1)} e_k + \frac{2}{(T_S + 1)} e_k - \frac{2}{(T_S + 1)} e_{k-1}$$

Finally:

$$\underline{\underline{u_k = \frac{u_{k-1}}{T_S + 1} + \frac{[T_S + 2]e_k}{T_S + 1} - \frac{2e_{k-1}}{T_S + 1}}}$$

Task 3: Discrete State-space model

Given the following system:

$$\dot{x}_1 = -a_1x_1 - a_2x_2 + bu$$

$$\dot{x}_2 = -x_2 + u$$

$$y = x_1 + cx_2$$

Task 3.1

Find the discrete state-space model.

Solution:

We use Euler forward:

$$\frac{x_1(k+1) - x_1(k)}{T_s} = -a_1x_1(k) - a_2x_2(k) + bu(k)$$

$$\frac{x_2(k+1) - x_2(k)}{T_s} = -x_2(k) + u(k)$$

$$y(k) = x_1(k) + cx_2(k)$$

This gives:

$$x_1(k+1) = x_1(k) - T_sa_1x_1(k) - T_sa_2x_2(k) + T_sbu(k)$$

$$x_2(k+1) = x_2(k) - T_sx_2(k) + T_su(k)$$

$$y(k) = x_1(k) + cx_2(k)$$

Futher:

$$x_1(k+1) = (1 - T_sa_1)x_1(k) - T_sa_2x_2(k) + T_sbu(k)$$

$$x_2(k+1) = (1 - T_s)x_2(k) + T_su(k)$$

$$y(k) = x_1(k) + cx_2(k)$$

This gives the following discrete state-space model:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \underbrace{\begin{bmatrix} (1 - T_sa_1) & -T_sa_2 \\ 0 & (1 - T_s) \end{bmatrix}}_{A} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \underbrace{\begin{bmatrix} T_sb \\ T_s \end{bmatrix}}_{B} u(k)$$

$$y(k) = \underbrace{\begin{bmatrix} 1 & \mathrm{c} \end{bmatrix}}_{C} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \underbrace{\begin{bmatrix} 0 \end{bmatrix}}_{\mathrm{D}} u(k)$$

With values $a_1 = 5, a_2 = 2, b = 1, c = 1$ and $T_s = 0.1$:

$$\begin{bmatrix} x_1(k+1) \\ x_2(k+1) \end{bmatrix} = \underbrace{\begin{bmatrix} 0.5 & -0.2 \\ 0 & 0.9 \end{bmatrix}}_{A} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \underbrace{\begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}}_{B} u(k)$$

$$y(k) = \underbrace{\begin{bmatrix} 1 & 1 \end{bmatrix}}_{C} \begin{bmatrix} x_1(k) \\ x_2(k) \end{bmatrix} + \underbrace{[0]}_{\mathrm{D}} u(k)$$

Task 3.2

Define the continuous state-space model ("pen and paper") and then implement the continuous state-space model in MathScript. Then use MathScript to find the discrete state-space model. Compare the result from the previous subtask.

Use values $a_1 = 5, a_2 = 2, b = 1, c = 1$ and $T_s = 0.1$.

Solution:

A continuous state-space model is given by:

$$\dot{x} = Ax + Bu$$

$$y = Cx$$

This gives:

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} -a_1 & -a_2 \\ 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} b \\ 1 \end{bmatrix} u$$

$$y = \begin{bmatrix} 1 & c \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

MathScript Code:

```
clear, clc

a1 = 5;
a2 = 2;
b = 1;
c = 1;

A = [-a1 -a2; 0 -1];
B = [b 1]';
C = [1 c];
D = [0];

ssmodel = ss(A, B, C, D);


% Discrete System:
Ts = 0.1;
ssmodel_discete = c2d(ssmodel, Ts, 'forward')
```

→ We get the same answer in MathScript as with "pen and paper".

Task 4: Discrete Low-pass Filter

Transfer function for a first-order low-pass filter may be written:

$$H(s) = \frac{1}{T_f s + 1}$$

Where T_f is the time-constant of the filter.

Task 4.1

Create the discrete low-pass filter algorithm using "pen and paper".

Use the **Euler Backward** method.

$$\dot{x} = \frac{x_k - x_{k-1}}{T_s}$$

Solutions:

Given:

$$\frac{y}{u} = \frac{1}{T_f s + 1}$$

This gives:

$$(T_f s + 1)y = u$$

$$T_f s y + y = u$$

Inverse Laplace gives:

$$T_f \dot{y} + y = u$$

We use the Euler Backward discretization method, $\dot{x} \approx \frac{x_k - x_{k-1}}{T_s}$, which gives:

$$T_f \frac{y_k - y_{k-1}}{T_s} + y_k = u_k$$

Then we get:

$$T_f(y_k - y_{k-1}) + y_k T_s = u_k T_s$$

Further:

$$T_f y_k - T_f y_{k-1} + y_k T_s = u_k T_s$$

Further:

$$y_k(T_f + T_s) = \ T_f y_{k-1} + u_k T_s$$

This gives:

$$y_k = \frac{T_f}{T_f + T_s} y_{k-1} + \frac{T_s}{T_f + T_s} u_k$$

For simplicity we set:

$$\frac{T_s}{T_f + T_s} \equiv a$$

This gives:

$$y_k = (1 - a) y_{k-1} + a u_k$$

where:

$$a = \frac{T_s}{T_f + T_s}$$

This algorithm can easily be implemented in a Formula Node in LabVIEW.

Task 4.2

Create a discrete low-pass filter in **LabVIEW** using the **Formula Node** in LabVIEW.

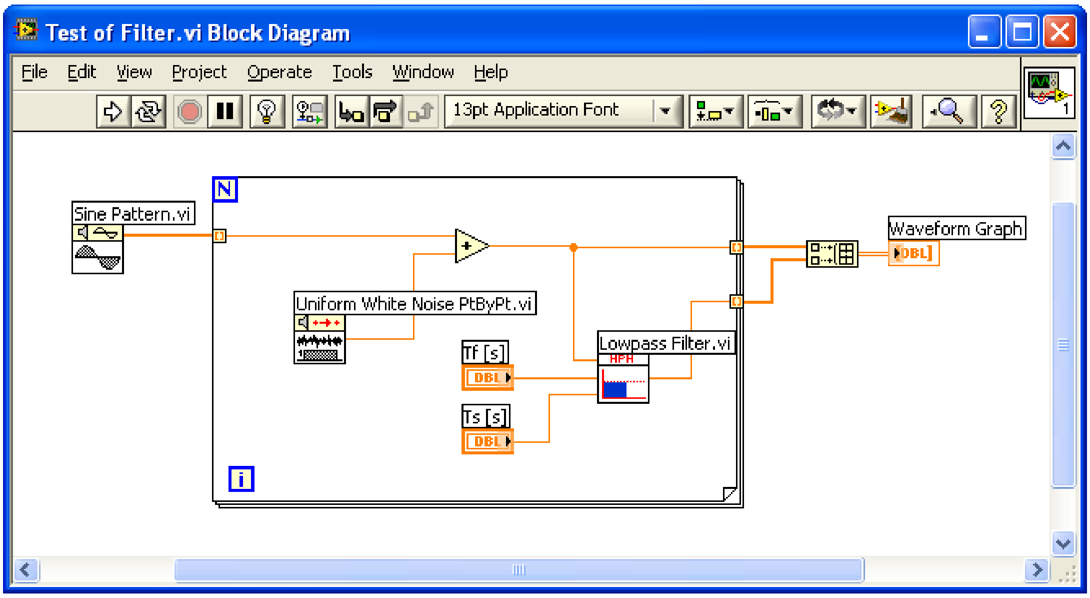
Create a **SubVI** of the code. You will use this subVI in your project later. The user needs to be able to set the time constant of the filter T_f from the outside, i.e., it should be an input to the SubVI. The simulation Time-step T_s needs also to be set from the outside.

Test and make sure your filter works!

Note! A golden rule is that:

$$T_s \leq \frac{T_f}{5}$$

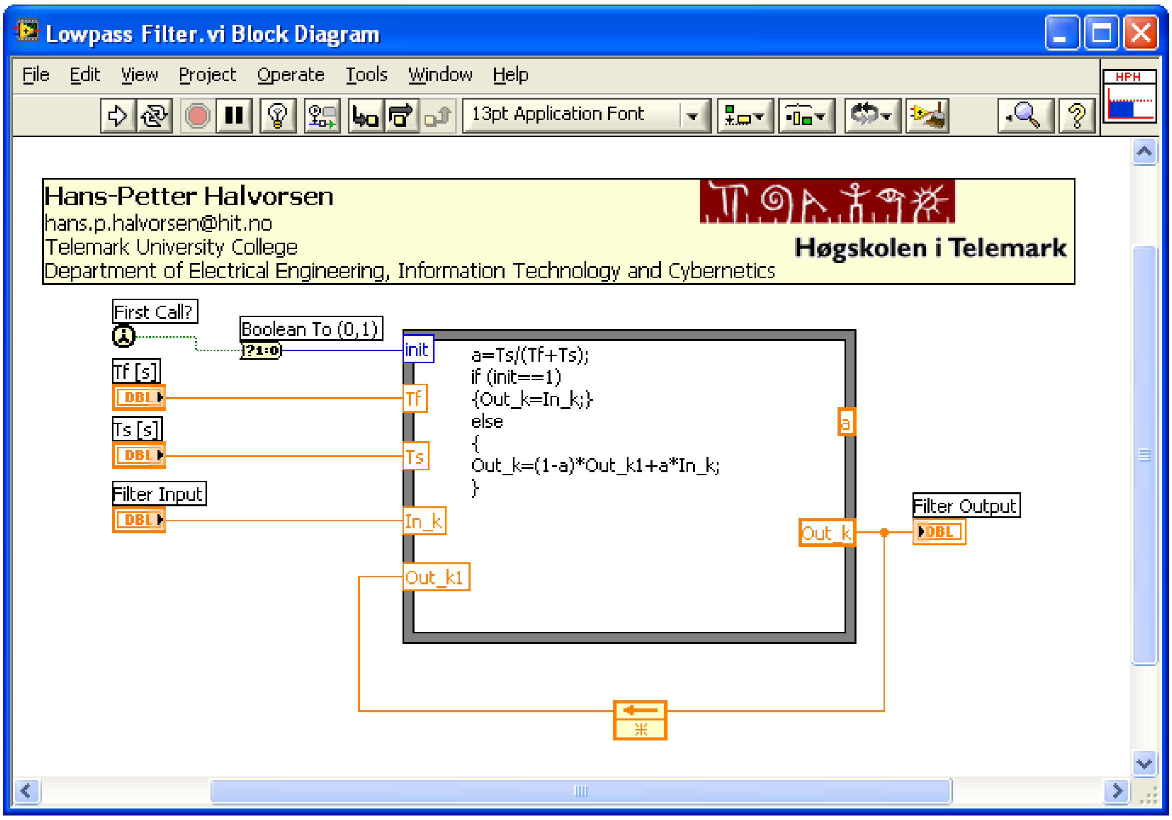
You may, e.g., use the "**Uniform White Noise PtByPt.vi**". Example:



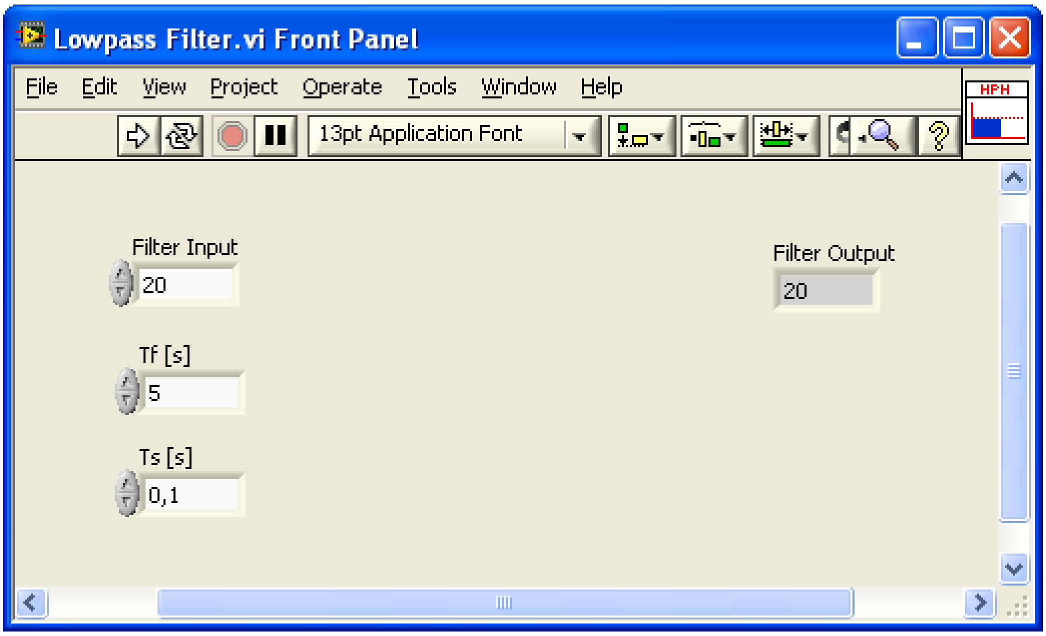
Solutions:

LabVIEW Program:

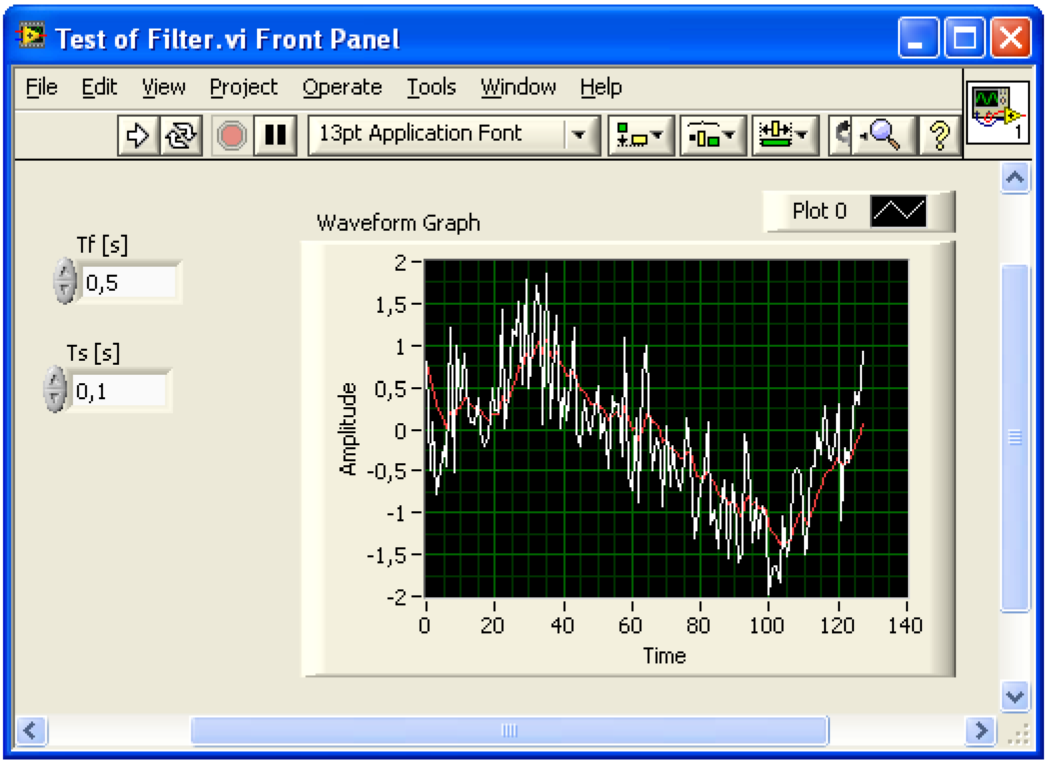
Block Diagram:



Front Panel:



We check if the filter works as expected:



Task 5: Discrete PI controller

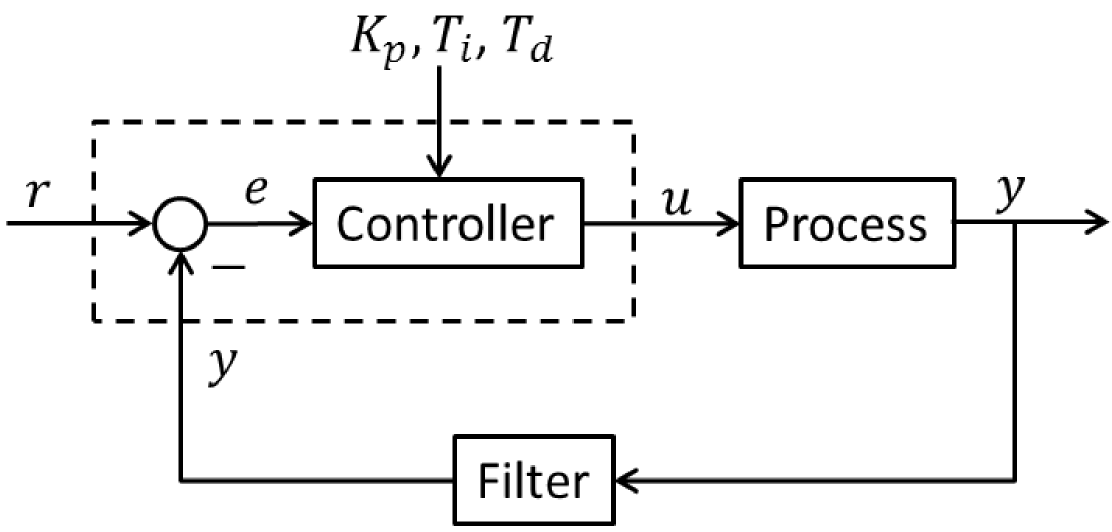
A continuous-time PI controller may be written:

$$u(t) = u_0 + K_p e(t) + \frac{K_p}{T_i}\int_0^t e d\tau$$

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

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

Below we see a block diagram of a simple control system:



Task 5.1

Create the discrete PI Controller algorithm using “pen and paper”. Use the Euler Backward method.

$$\dot{x} = \frac{x_k - x_{k-1}}{T_s}$$

Solutions:

We start with:

$$u(t) = u_0 + K_p e(t) + \frac{K_p}{T_i}\int_0^t e d\tau$$

In order to make a discrete version using, e.g., Euler, we can derive both sides of the equation:

$$\dot{u} = \dot{u}_0 + K_p \dot{e} + \frac{K_p}{T_i} e$$

If we use Euler Forward we get:

$$\frac{u_k - u_{k-1}}{T_s} = \frac{u_{0,k} - u_{0,k-1}}{T_s} + K_p \frac{e_k - e_{k-1}}{T_s} + \frac{K_p}{T_i} e_k$$

Then we get:

$$\boxed{\boldsymbol{u_k = u_{k-1} + u_{0,k} - u_{0,k-1} + K_p(e_k - e_{k-1}) + \frac{K_p}{T_i} T_s e_k}}$$

Where

$$e_k = r_k - y_k$$

We can also split the equation above in 2 different pars by setting:

$$\Delta u_k = u_k - u_{k-1}$$

This gives the following PI control algorithm:

$$\boldsymbol{e_k = r_k - y_k}$$

$$\boldsymbol{\Delta u_k = u_{0,k} - u_{0,k-1} + K_p(e_k - e_{k-1}) + \frac{K_p}{T_i} T_s e_k}$$

$$\boldsymbol{u_k = u_{k-1} + \Delta u_k}$$

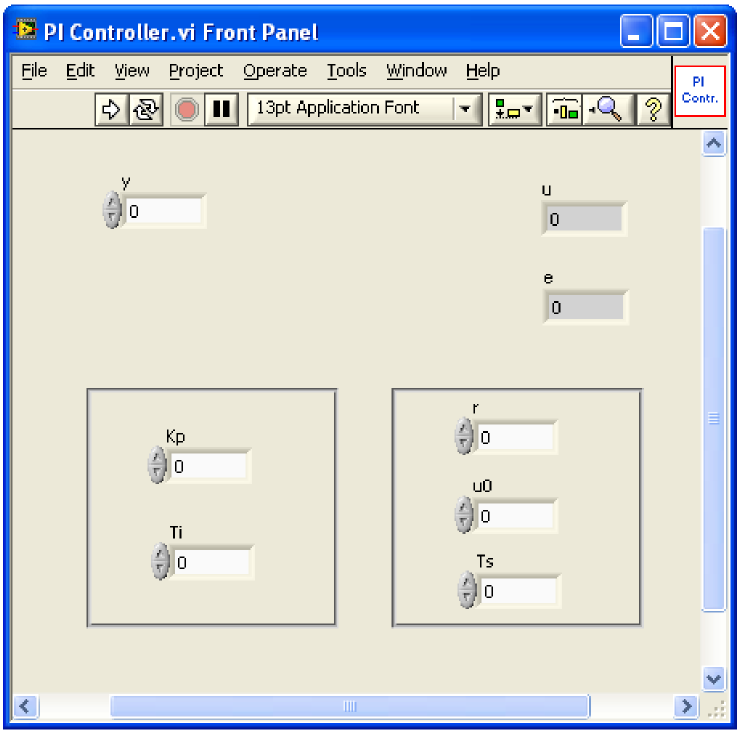
This algorithm can easily be implemented in a Formula Node in LabVIEW.

Task 5.2

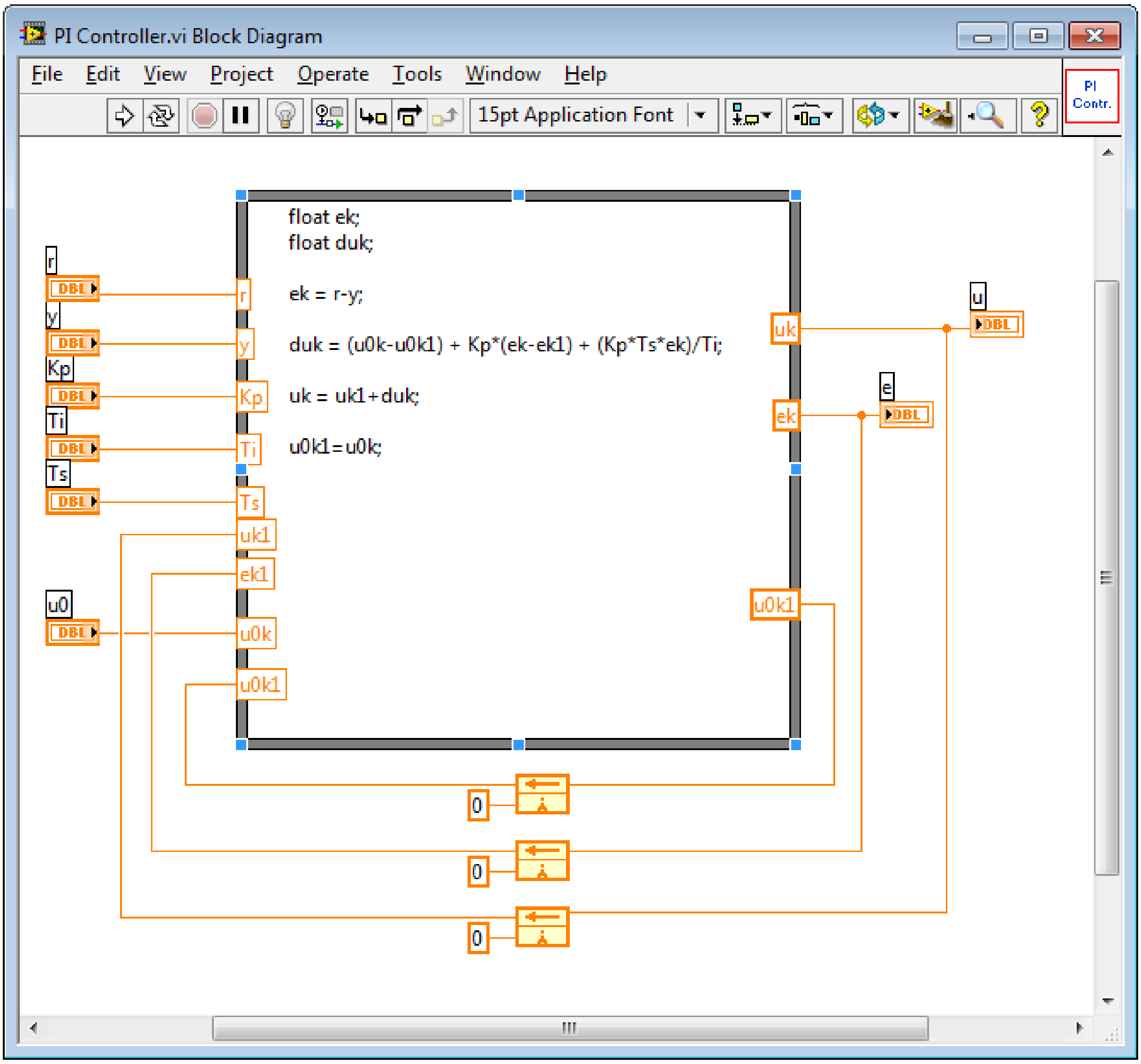
Create a discrete PI controller in LabVIEW using the **Formula Node**. Create a **SubVI** of the code. You will use this subVI in your project later.

Solutions:

Front Panel:



Block Diagram:



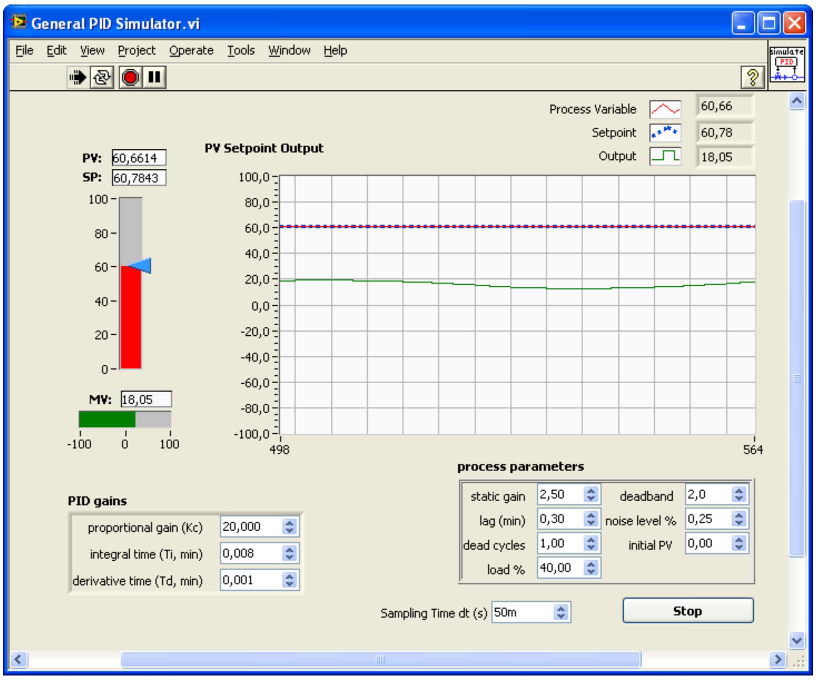
Task 6: Simulation

Implement your PI controller and Low-pass filter in a simulation. You may, e.g., use the example "**General PID Simulator.vi**" as a base for your simulation. Use the "NI Example Finder" (Help → Find Examples…) in order to find the VI in LabVIEW.

Note! You will need the "LabVIEW PID and Fuzzy Logic Toolkit" in order to fulfill this Task.

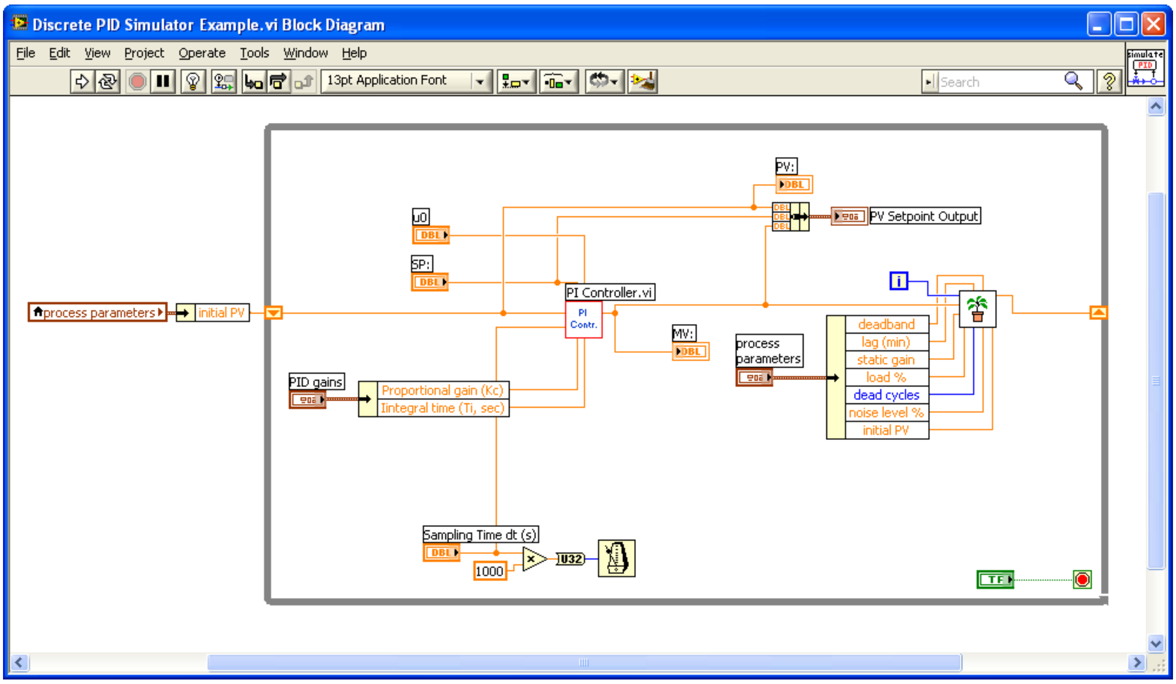
Run the example and see how it is implemented and how it works.

Note! Save the VI with a new name and replace the controller used in the example with the controller you created in the previous task.

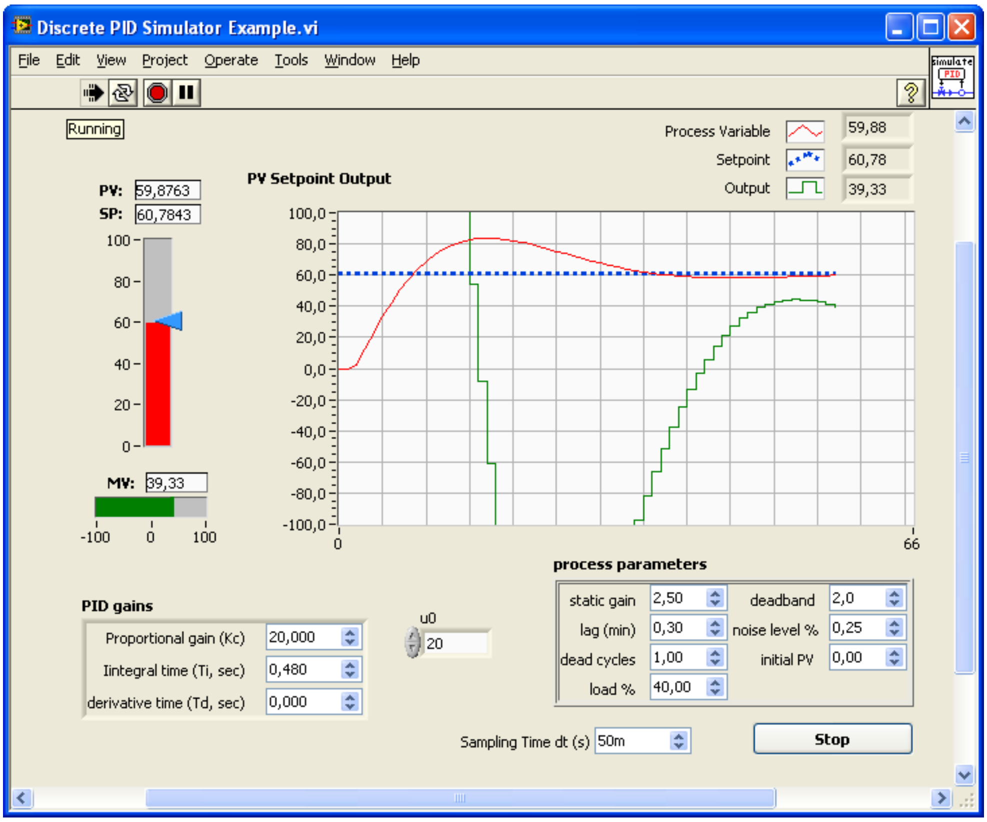


Solutions

Block Diagram:

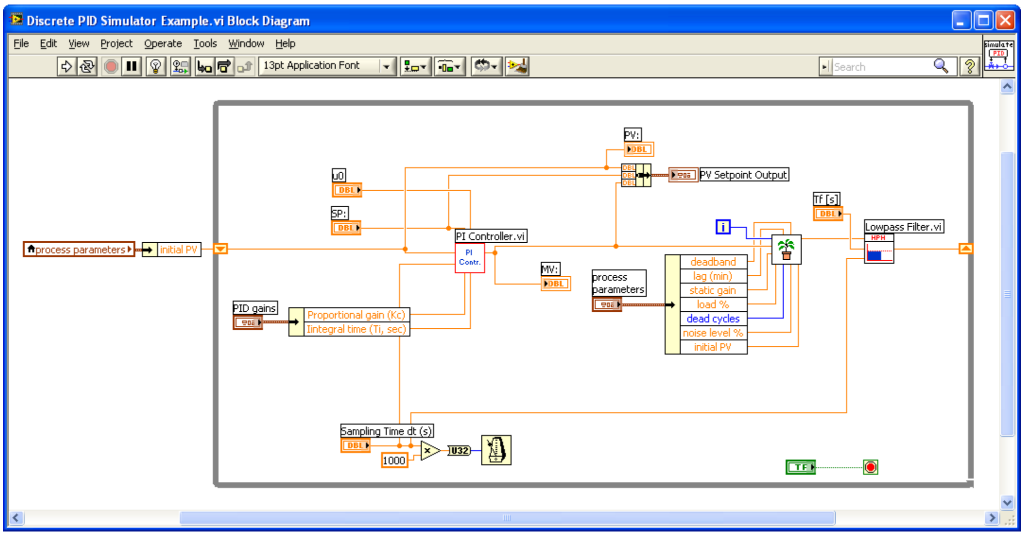


Front Panel:



With Noise and Low-pass filter:

Block Diagram:



Additional Resources

- http://home.hit.no/~hansha/?lab=discretization

Here you will find tutorials, additional exercises, etc.