https://www.halvorsen.blog

Machine Learning in Automation Systems

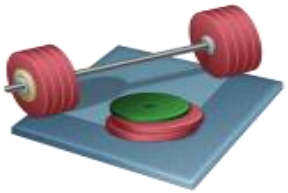
Hans-Petter Halvorsen

Table of Contents

Introduction

- Machine Learning is all about Data Analytics, complex Mathematical Models and Algorithms, used for Predictive Analytics.
- In this Assignment you will use more traditional and well known “Machine Learning” principles such as *System Identification*, *State Estimation with Kalman Filter* and *Model Predictive Control (MPC)*
- Previously you have been working with Neural Networks (not part of this assignment) which is also used in Machine Learning

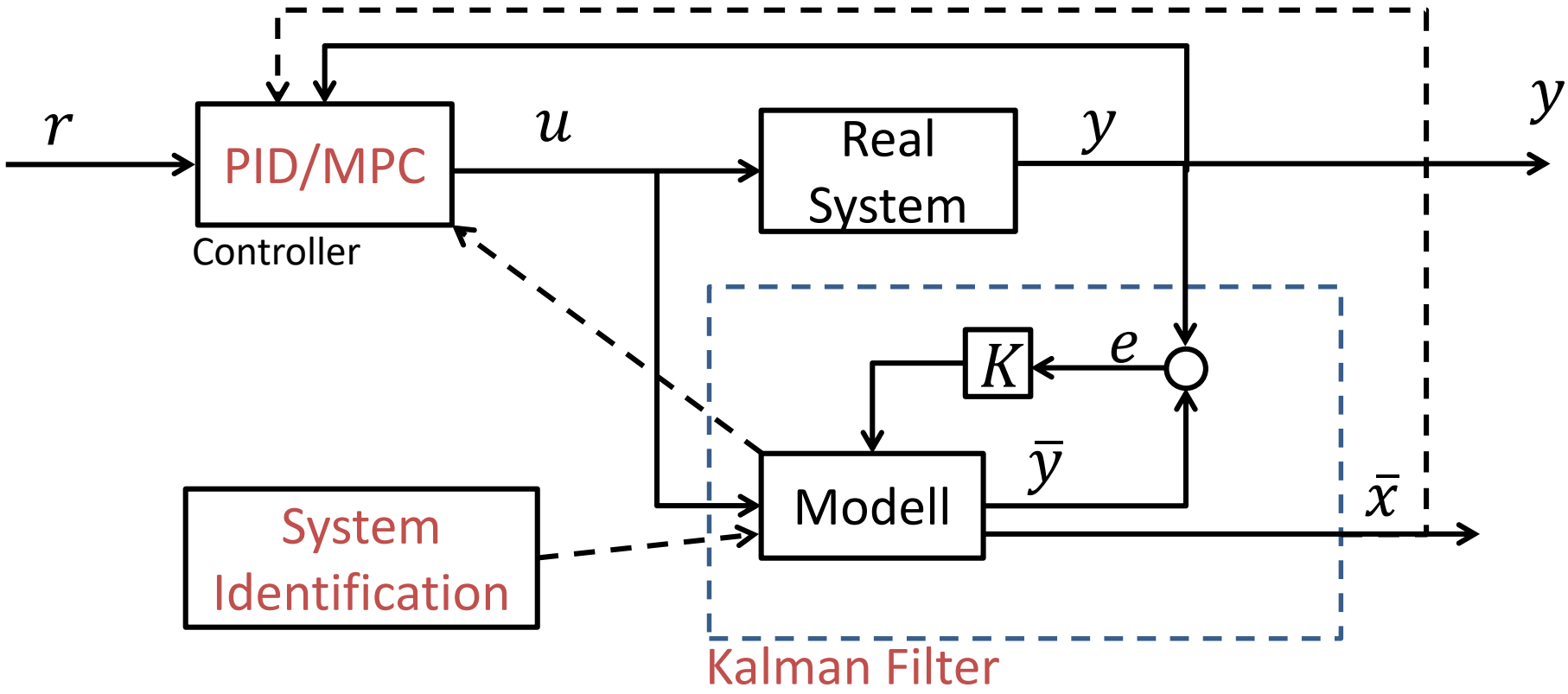
Lab Assignment Overview



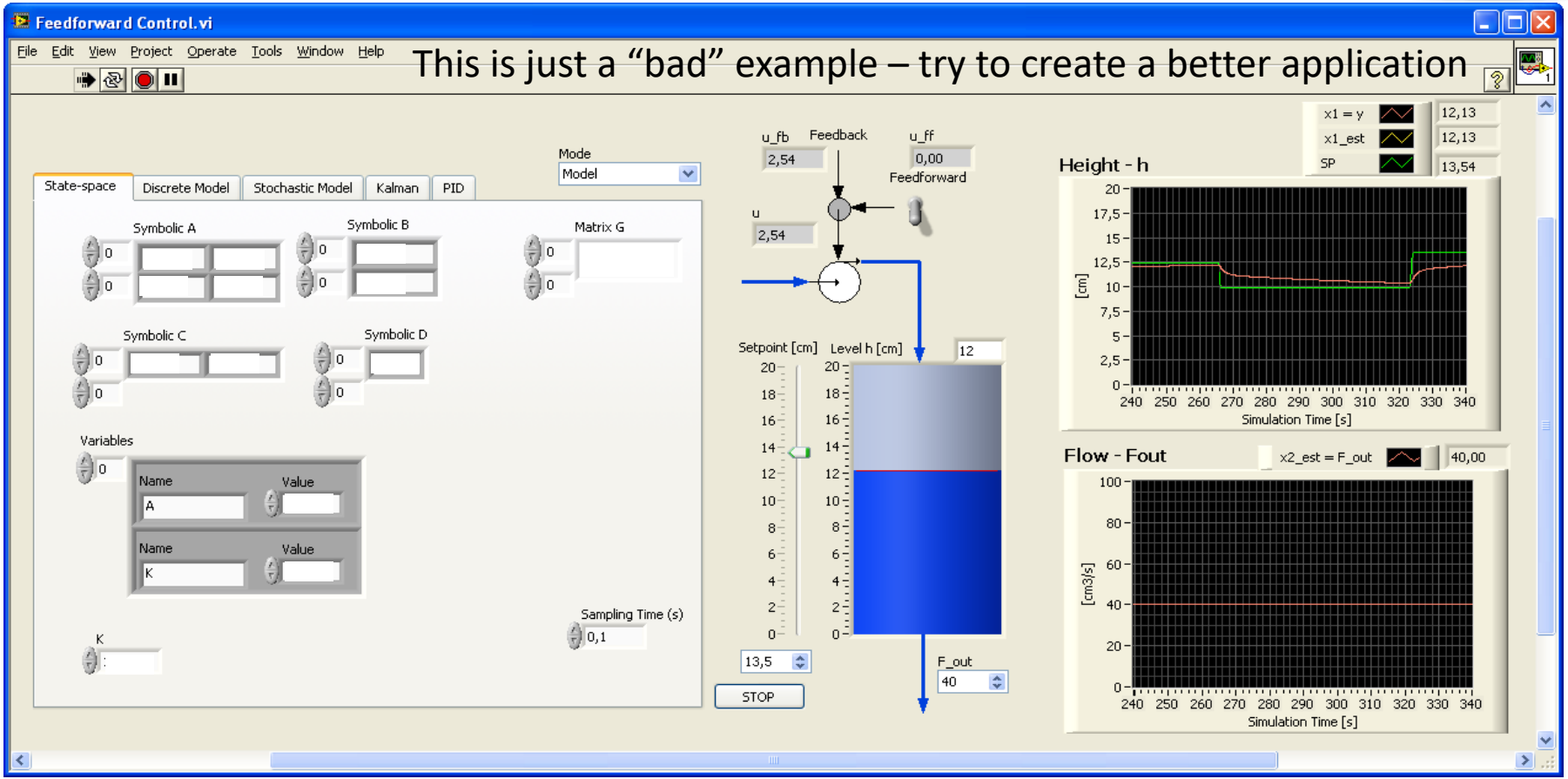
1. Modelling & System Identification
2. State Estimation with Kalman Filter
3. Feedback Control and Feedforward Control
4. Model Predictive Control (MPC)
   - Compare with traditional PID

See next slides for details...

System Overview



LabVIEW Example (PID + Kalman + FF)



Keywords

- Introduction to Machine Learning
- Modelling and Control Theory
- Simulation
- System Identification
- State Estimation using Kalman Filter
- Feedback Control (PID)
- Feedforward Control
- Model Predictive Control (MPC)
- Programming/LabVIEW

Learning Goals

- Learn more Programming (LabVIEW/MATLAB)
- Learn Practical Implementation of Control Theory, such as:
  - System Identification
  - State Estimation with Kalman Filter
  - Feedback and Feedforward Control
  - Model Predictive Control (MPC)
- Learn more about Modelling and Simulation of Dynamic Systems
- Learn practical Machine Learning (ML) Implementation

Machine Learning

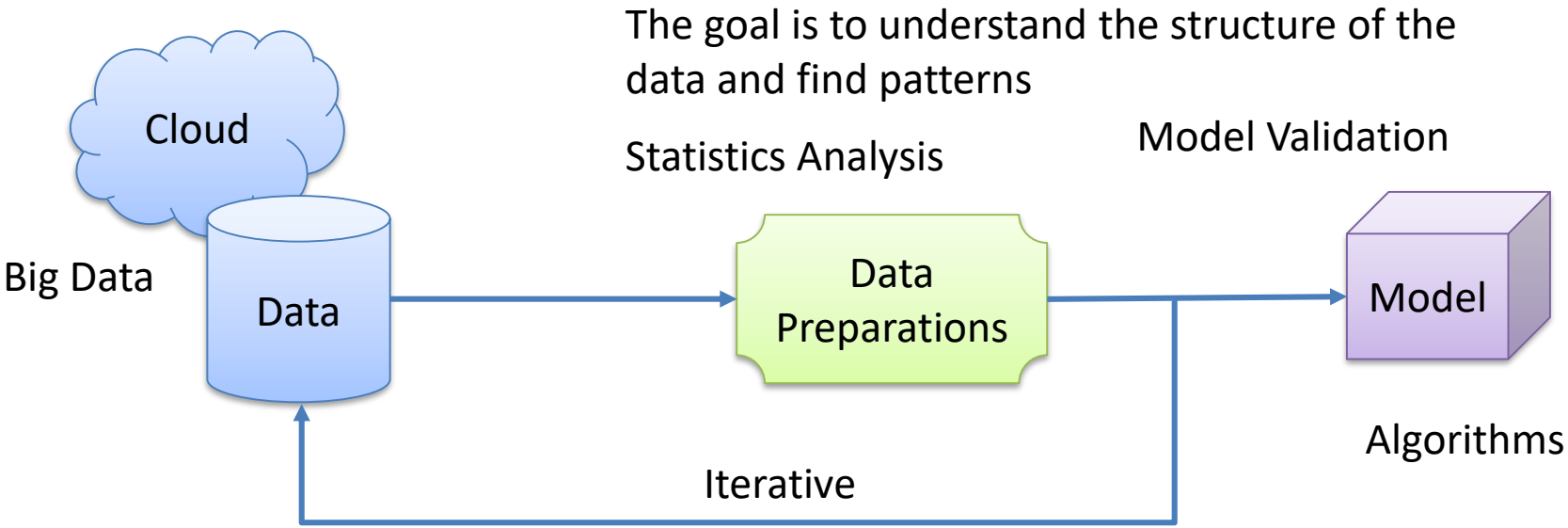
- Machine Learning (ML) is all about Data Analytics, complex Mathematical Models and Algorithms used for Predictive Analytics.
- Machine Learning is closely related to (and often overlaps with) computational statistics, which also focuses on prediction-making through the use of computers. It has strong ties to mathematical optimization
- In System Identification, State Estimation (Kalman Filter) and Model Predictive Control (MPC) all these things apply

Machine Learning

- Machine Learning is about examine large amount of data (“Big Data”) looking for Patterns.
- It applies statistical techniques to large amounts of data, looking for the best pattern to solve your problem. This pattern can be referred to as a data model.
- The machine learning process starts with raw data and ends up with a model derived from that data.
- The machine learning algorithm is run on prepared data, and the result is referred to as a model.
- The knowledge gained is then used for Predictions, i.e., Predict the Future
- Machine Learning is an iterative process, which continuously updates the model when new data/knowledge arrives.

Machine Learning

A simplified sketch of the Machine Learning process:



Machine Learning Applications

- Create complex Weather models from a large amount of collected weather data. The weather models are then used to predict the weather in the future (short or long termed)
- Transportation: Self driving cars, ships or so-called Autonomous vehicles
- Marketing and sales, e.g., Online recommendation offers such as those from Amazon and Netflix
- Apple Siri (intelligent personal assistant) and similar services
- Financial services, such as Stock market, etc.
- ... hundreds of other examples

Software

NATIONAL INSTRUMENTS LabVIEW™

LabVIEW Control Design and Simulation Module

MATLAB

If you prefer, you may use MATLAB for some of the Tasks

LabVIEW has built-in features for System Identification and Estimation

Hardware

Your Personal Computer

USB-6008 or similar DAQ device

Level Tank

Only available in the Laboratory!

Online students: You can do 95% of the Assignment without this Hardware using Simulators and a provided “Black Box Model”

The teacher have not done all the Tasks in detail, so he may not have all the answers! That's how it is in real life also!

Very often it works on one computer but not on another. You may have other versions of the software, you may have installed it in the wrong order, etc...
In these cases Google is your best friend!

The Teacher don’t have all the answers (very few actually ☹)!! Sometimes you just need to “Google” in order to solve your problems, Collaborate with other Students, etc. That’s how you Learn!

Troubleshooting & Debugging

Visual Studio

Use the **Debugging Tools** in your Programming IDE.
Visual Studio, LabVIEW, etc. have great Debugging Tools! Use them!!

"Google It"!

You probably will find the answer on the Internet

Google

Another person in the world probably had a similar problem

Use available Resources such as User Guides, Datasheets, Text Books, Tutorials, Examples, Tips & Tricks, etc.

My System is not Working??

Multimeter, etc.

Check your electric circuit, electrical cables, DAQ device, etc. Check if the wires from/to the DAQ device is correct. Are you using the same I/O Channel in your Software as the wiring suggest? etc.

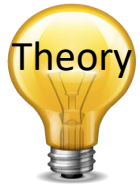
Theory

Hans-Petter Halvorsen, M.Sc.

Background Theory

- System Identification
- Kalman Filter
- Feedforward Control
- Discretization
- Skogestad PID Tuning
- Model Predictive Control (MPC)

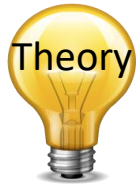
System Identification and Estimation



- **System Identification:** System Identification uses statistical methods to build mathematical models of dynamical systems from measured data
- **State Estimation:** Use of mathematical models in order to estimate the internal states of a process

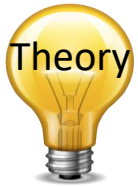
LabVIEW has built-in functionality for both System Identification and State Estimation

System Identification Categories



We have 2 main categories of System Identification:

- **Parameter Estimation** based on that we have developed a mathematical model using the laws of physics (Mechanistic Models) and you want to find the unknown model parameters. Here we will use least squares method as an example. The unknown parameters are then found from experimental data.
- **Black-box / Subspace methods**: System Identification based on that you do not have a mathematical model available. The models (Empirical Models) are found from experimental data only using advanced algorithms.



System Identification

Datalogging from Real System (Experimental Data)

Physical Knowledge

Mechanistic Models

Empirical Models

PLS/PCR, Black-box, DSR/Subspace, Wavelet, etc.

Finding mathematical model(s) using the laws of physics/first principles

The model is found from experimental data

Empirical modelling refers to any kind of (computer) modelling based on empirical observations rather than on mathematically describable relationships of the system modelled.

Datalogging from Real System (Experimental Data)

Parameter Estimation

The unknown Parameters within the model(s) needs to be found

Example of unknown Parameters: Pump gain, Valve constants, etc.

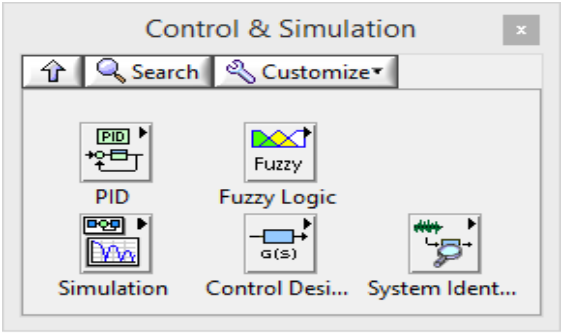
Trial and Error, Step Response, Least Square Methods, etc.

Some of these can be found in data sheets, etc., while others is not so easy to find. Then Parameter Estimation is a good method to find these.

System Identification & Estimation in LabVIEW

- “LabVIEW Control Design and Simulation Module” has built-in features for Control, Simulation, System Identification and Estimation, which we shall use in this Assignment
- In addition we shall also create some features from scratch in order to get a deeper understanding of the theory behind (and for comparison)

Control & Simulation Palette in LabVIEW installed with LabVIEW Control Design and Simulation Module

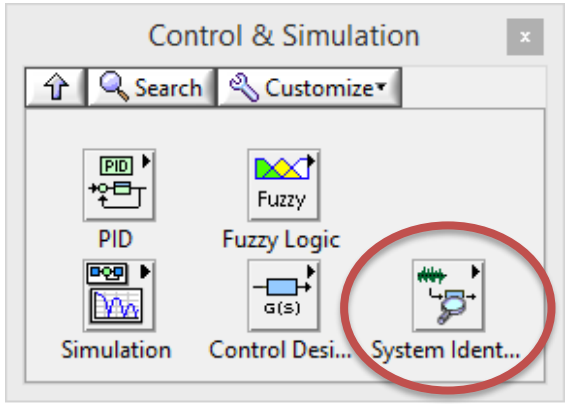


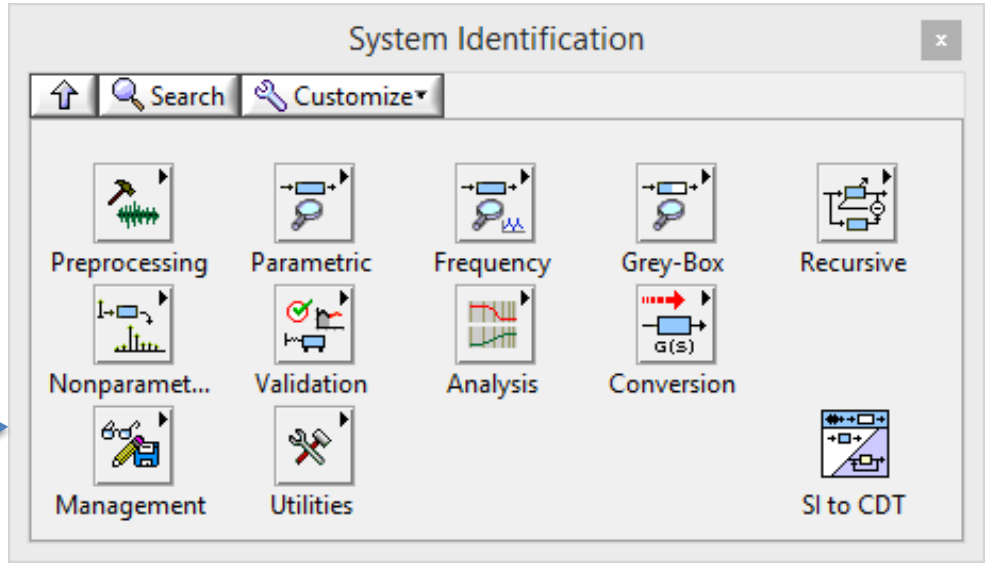
Modelling and System Identification

Hans-Petter Halvorsen, M.Sc.

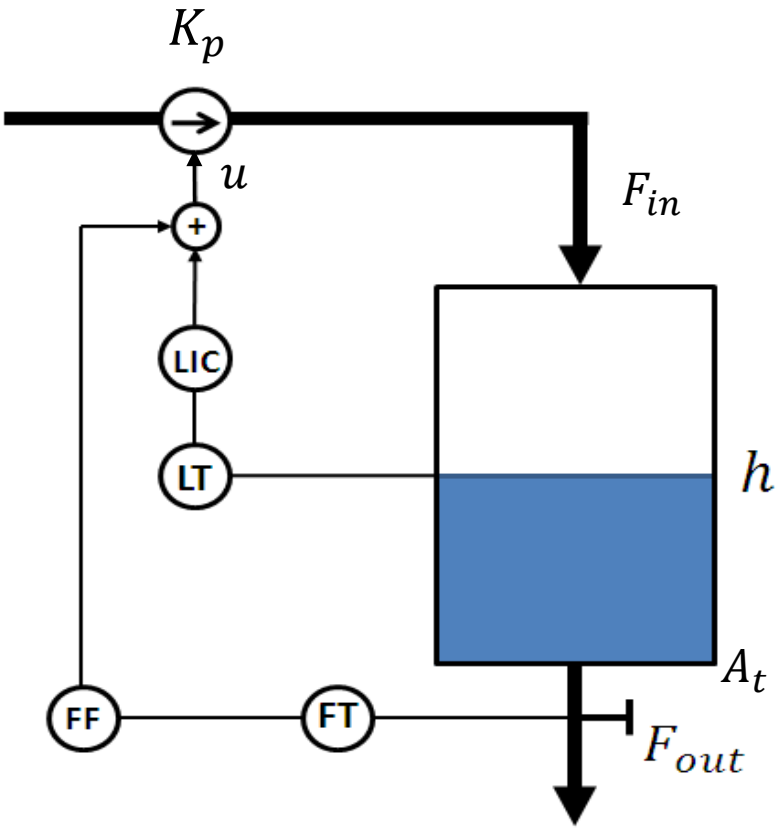
System Identification in LabVIEW

"LabVIEW Control Design and Simulation Module" has built-in features for System Identification





Level Tank



$$A_t \frac{dh}{dt} = F_{in} - F_{out}$$

or:

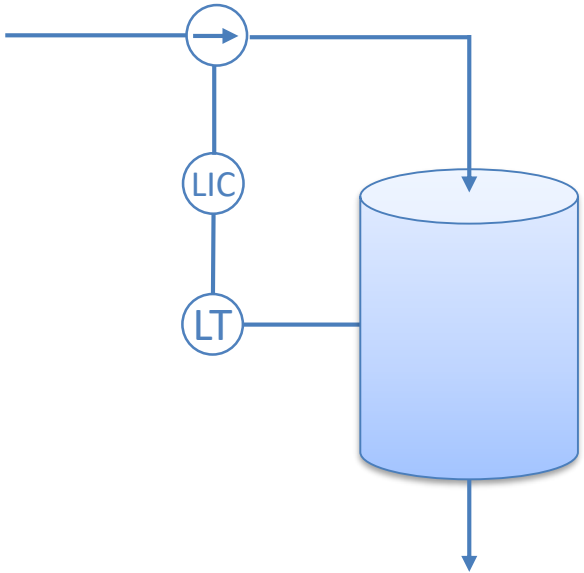
$$\dot{h} = \frac{1}{A_t}(K_p u - F_{out})$$

Where:

- F_{in} - flow into the tank , F_{in}= $K_p u$
- F_{out} - flow out of the tank
- A_t is the cross-sectional area of the tank

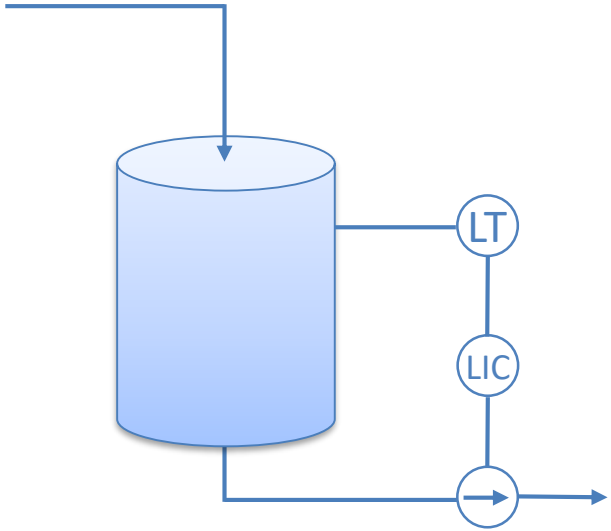
PID Controller – Reverse or Direct Mode?

Reverse Action Mode



Reverse action mode: If the controller must decrease the control signal to bring the increased process measurement back to the setpoint, the controller shall have Reverse Action Mode.

Direct Action Mode



Direct action mode: If the controller must increase the control signal to bring the increased process measurement back to the setpoint, the controller shall have Direct Action Mode.


LEVEL TANK
LM-900 LEVEL CONTROL SYSTEM
INSTRUTEK
LARVIK - NORWAY
CONTROLLER
LIC
PURGE-
METER
FLOWCHART
OPERATION
INTERLOCKING
h
A_t
F_{out}
RESERVOIR
4104.5.50
Level Tank

LM-900 Level System

Can be manually adjusted

- For real system: a handle on the red tank
- For Simulator: A Numeric control on the Front Panel (HMI)

The level is measured

$$\dot{h} = \frac{1}{A_t}\left[K_p(u - u_0) - F_{out}\right]$$

We need to find the unknown model parameter(s) using System Identification methods

(A_t can be found by measuring the radius of the tank)

$$A_t \approx 78.5\ cm$$

Level Tank model – Integrator Model

$$\dot{h} = \frac{1}{A_t}\left[K_p(u - u_0) - F_{out}\right]$$

- K_p $[cm^3/s)/V]$ is the pump gain
- F_{out}[cm^3/s] is is the outflow through the valve
- A_t $[cm^2]$ is the cross-sectional area of the tank
- u $[V]$ is the control signal to the pump

You should use this model in your linear Kalman filter algorithm

Level Tank model - 1.order linear system

A more accurate model may, e.g., be:

$$\dot{h} = \frac{1}{A_t}\left[K_p(u - u_0) - K_v h\right]$$

where K_v is the valve gain on the outflow.

It is more normal to put it like this:

$$\dot{h} = -\frac{K_v}{A_t}h + \frac{K_p}{A_t}u \qquad \text{(The general term is } \dot{x} = ax + bu\text{)}$$

The model above is a so-called Time-constant system (1.order linear system).

Level Tank model - 1.order Nonlinear Model

The following model is even more accurate:

$$\dot{h} = \frac{1}{A_t}\left[K_p(u - u_0) - K_v\sqrt{\rho g h}\right]$$

This is a so-called 1.order nonlinear model

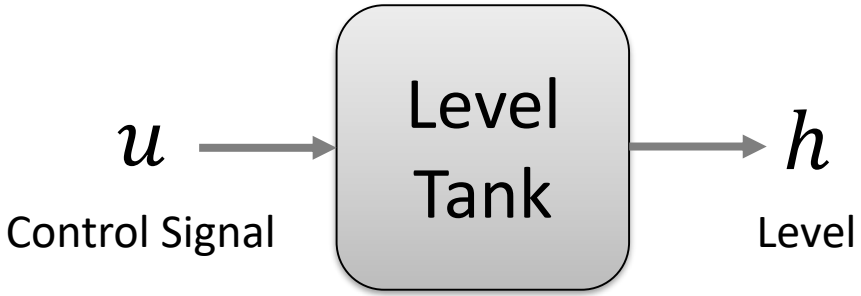
- $h\ [cm]$ is the level
- $u\ [V]$ is the pump control signal to the pump
- u_0 is the bias voltage needed to get any flow (with u less than u_0 there is no flow into the tank)
- $A_t[cm^2]$ is the cross-sectional area of the tank
- $K_p[(cm3/s)/V]$ is the pump gain
- K_v is the valve constant. It depends on the opening of the valve, but if the opening is constant, K_v is constant
- ρ is the is the density of the liquid (water: $1\ kg/m^3$)
- g is the is the gravity constant, $9.81\ \mathrm{m}/s^2$

You may find K_p and K_v using, e.g., the Least Square method

“Real Process” → “Black Box Model”

- The Real Level Tank is only available in the Laboratory
- A “Real” Level Tank will we provided as a “black box”. Actually, it is just a LabVIEW SubVI where the Block Diagram and the Process Parameters are hidden for you.
- Useful for Online Students and when you are working with the Assignment outside the Laboratory

“Real Process” → “Black Box Model”

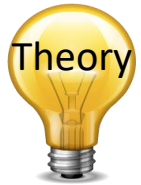


You can assume that the following model is a good representation of the Black Box Model:

$$\dot{h} = \frac{1}{A_t}\left[K_p u - F_{out}\right]$$

This means you need to unknown parameters using some kind of system identification method

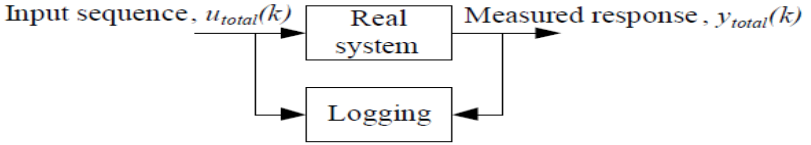
System Identification



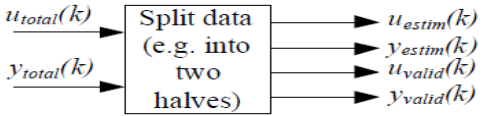
In general, System Identification consists of the following steps:

Make sure to include all these steps in your solution.

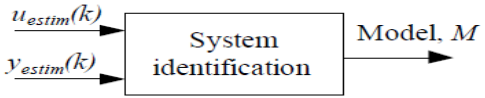
1. Excite the real system, and log input and output:



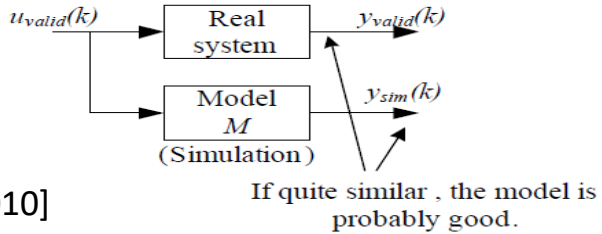
2. Split data, for estimation and for validation :



3. Estimate model:

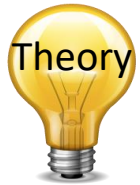


4. Check (validate) model using e.g. simulation:



[Figure: F. Haugen, Advanced Dynamics and Control: TechTeach, 2010]

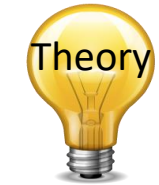
System Identification



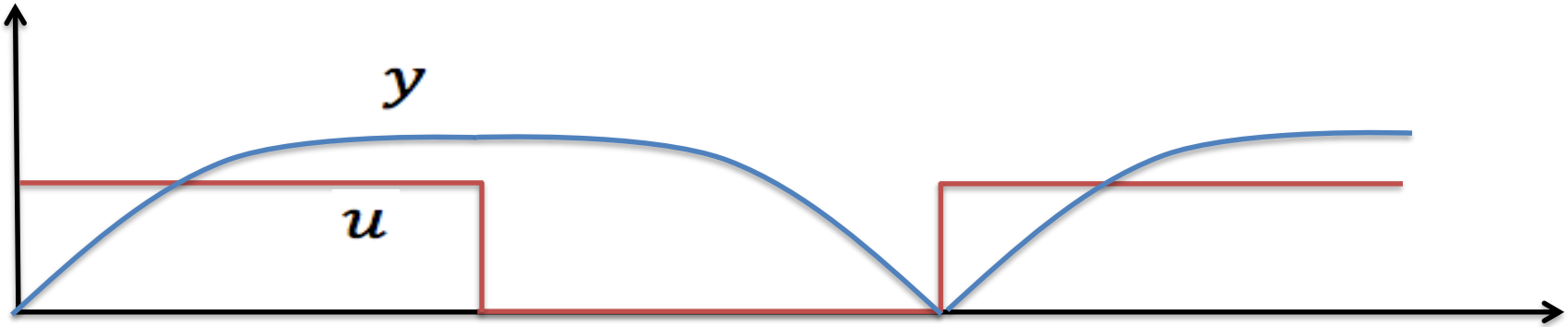
Suggestions: Find the Model Parameters using:

- The **Least Square Method** $\theta_{LS} = (\Phi^{T}\Phi)^{-1}\Phi^{T}Y$
- Then adjust and fine-tune the Model Parameters using the "**Trial and Error**" method if necessary
- It is advised that you use at least 2 different methods for comparison.
- Other relevant methods may be: "Step response method", Sub-space methods, DSR, built-in methods in LabVIEW/MATLAB, etc.

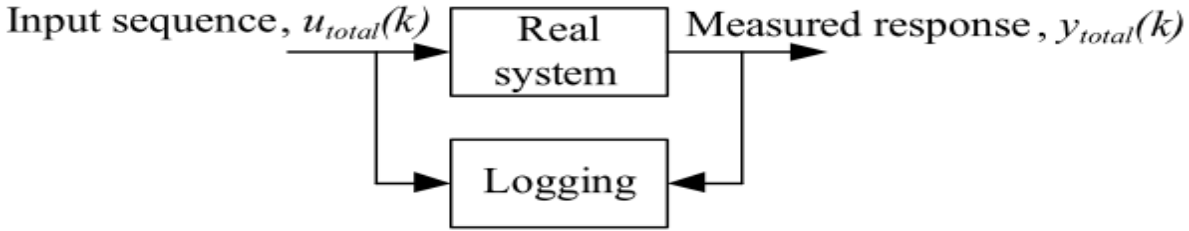
Data Logging



1. Exite the Real System, e.g.:



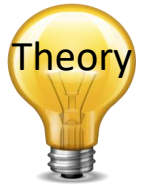
2. Log Data to File



[Figure: F. Haugen, Advanced Dynamics and Control: TechTeach, 2010]

3. Use the Logged Data to find the model or the model parameters

Least Square Example



Given:

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

We want to find the unknown *a* and *b*.

This gives:

$$\underbrace{\dot{x}}_{y} = \underbrace{[x \quad u]}_{\varphi} \underbrace{\begin{bmatrix} a \\ b \end{bmatrix}}_{\theta}$$

i.e.,:

$$\theta = \begin{bmatrix} a \\ b \end{bmatrix}$$

Then we need to discretize:

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

This gives:

$$\underbrace{\frac{x_{k+1} - x_k}{T_s}}_{y} = \underbrace{[x_k \quad u_k]}_{\varphi} \underbrace{\begin{bmatrix} a \\ b \end{bmatrix}}_{\theta}$$

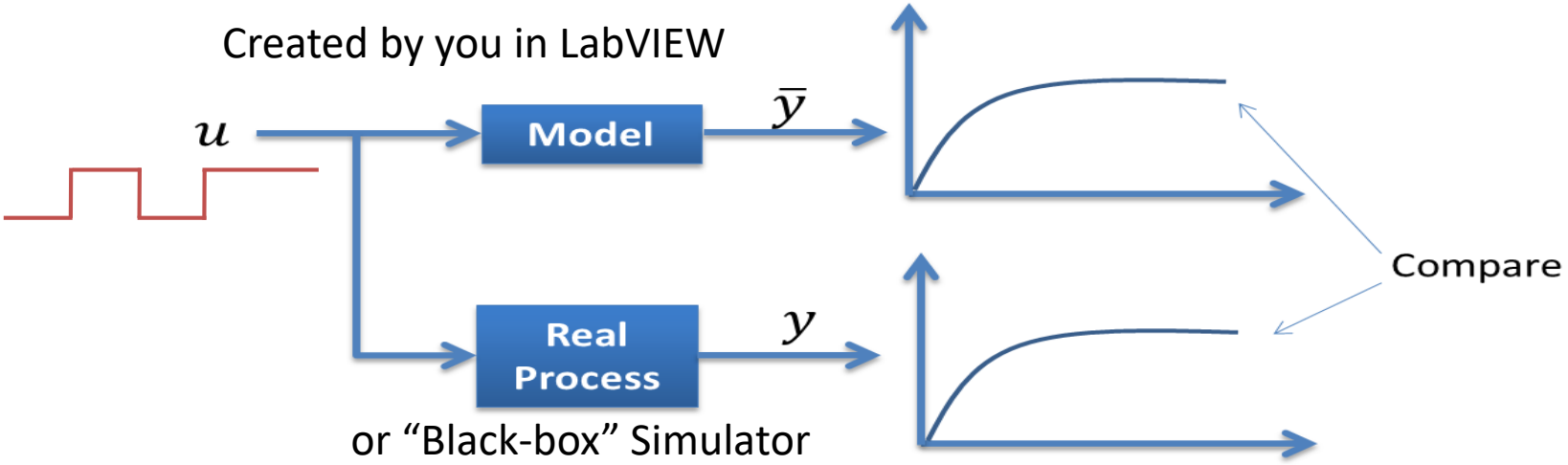
Based on logged data we get:

$$\underbrace{\begin{bmatrix} \vdots \\ \frac{x_{k-1} - x_{k-2}}{T_s} \\ \frac{x_k - x_{k-1}}{T_s} \\ \frac{x_{k+1} - x_k}{T_s} \end{bmatrix}}_{Y} = \underbrace{\begin{bmatrix} \vdots & \vdots \\ x_{k-2} & u_{k-2} \\ x_{k-1} & u_{k-1} \\ x_k & u_k \end{bmatrix}}_{\Phi} \underbrace{\begin{bmatrix} a \\ b \end{bmatrix}}_{\theta}$$

The we find the uknows a and b using LS:

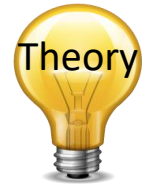
$$\theta_{LS} = (\Phi^T \Phi)^{-1} \Phi^T Y$$

Trial & Error Method



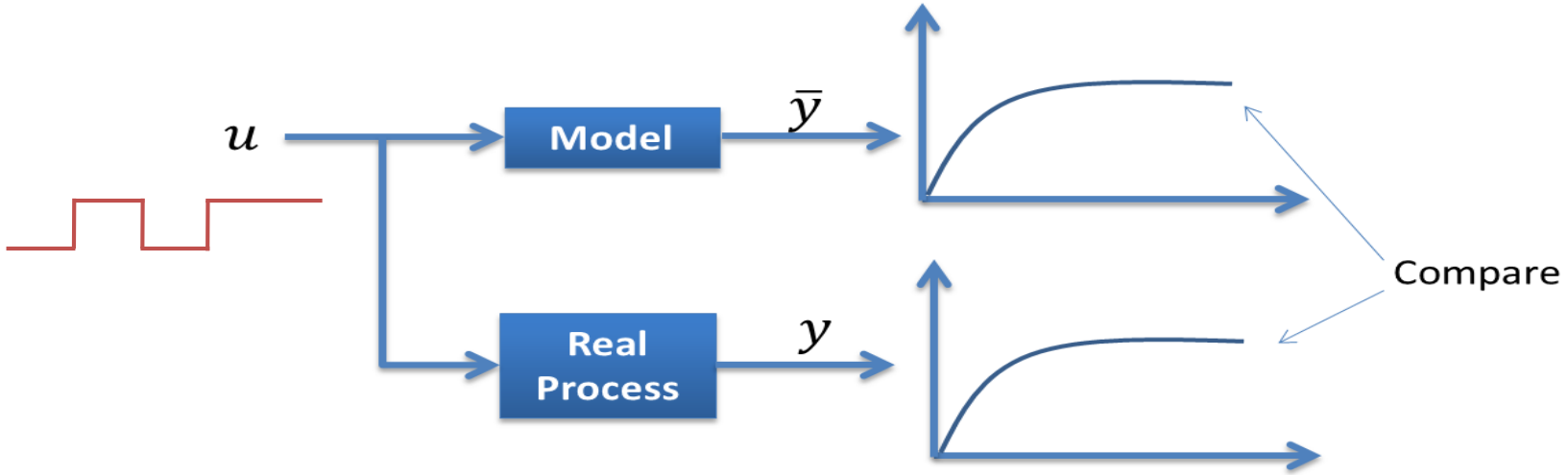
Adjust model parameters and then compare the response from the real system with the simulated model. If they are “equal”, you have probably found a good model (at least in that working area)

Model Validation



Make sure to validate that your model works as expected

Example of simple model validation:



Model Values

$$\dot{h} = \frac{1}{A_t}[K_p u - F_{out}]$$

If you don't have the red Level Tank nearby, you may use the following values as a starting point for your simulations in the rest of the Assignment:

$$A_t = 78.5\ cm$$

$$K_p = 16.5\ cm^3/s$$

F_{out} should be adjustable from your Front Panel

The range for F_{out} could, eg., be $0 \leq F_{out} \leq 40 cm^3/s$

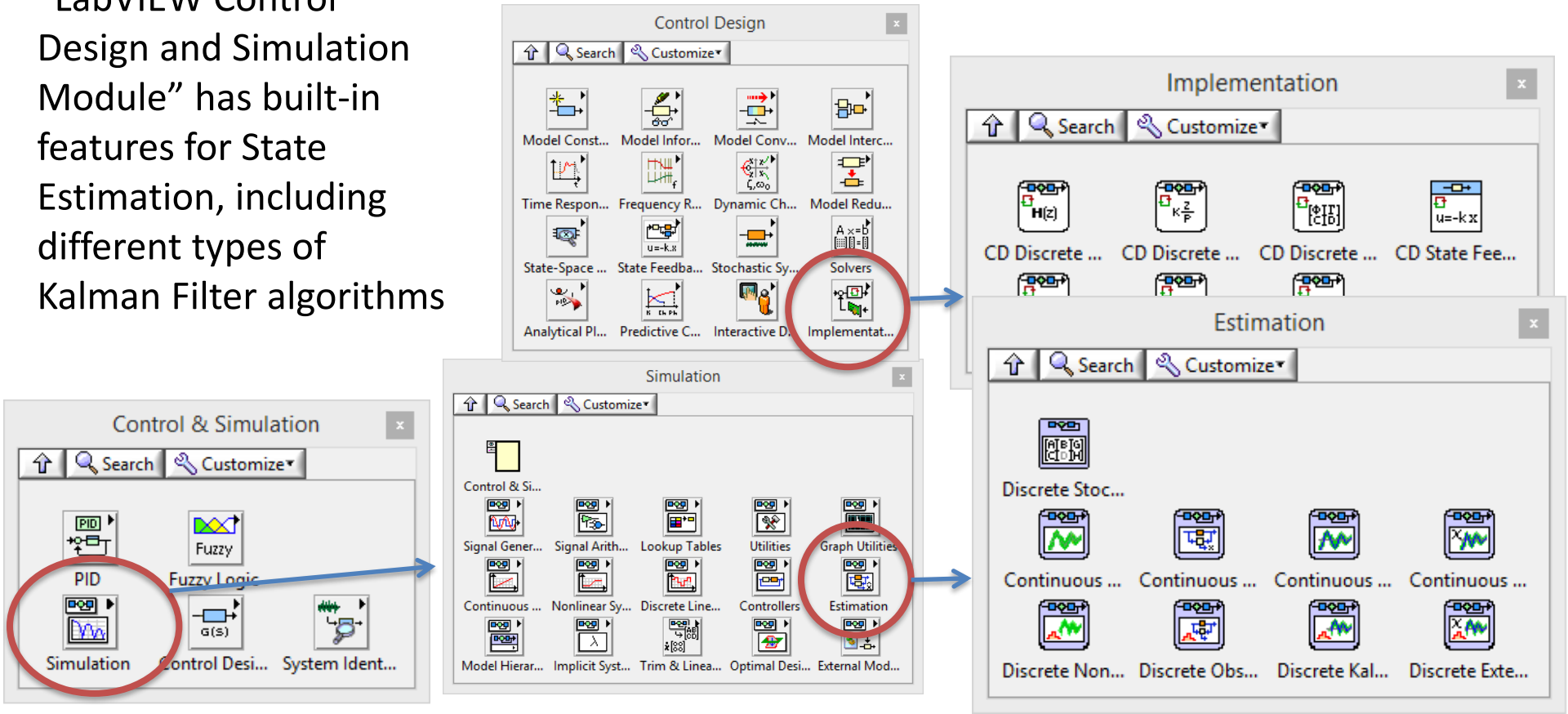
Congratulations! - You are finished with the Task

State Estimation and Kalman Filter

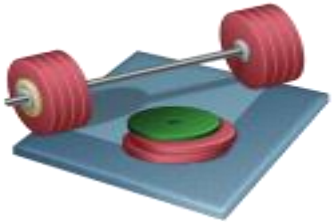
Hans-Petter Halvorsen, M.Sc.

State Estimation in LabVIEW

"LabVIEW Control Design and Simulation Module" has built-in features for State Estimation, including different types of Kalman Filter algorithms



State Estimation with Kalman Filter



- The Kalman Filter is a commonly used method to estimate the values of state variables of a dynamic system that is excited by stochastic (random) disturbances and stochastic (random) measurement noise.
- We will estimate the process variable(s) using a Kalman Filter.
- You should use one of the built-in Kalman Filter algorithms in addition to create your own algorithm from scratch. Compare the results.

State-space Model

$$\dot{h} = \frac{1}{A_t}\left[K_p u - F_{out}\right]$$

For the real system, only the level (*h*) is measured, so we want to use a Kalman Filter for estimating the outflow (F_{out}) of the tank (Which we will use in a Feedforward control later).

We need to find a state-space model that we can use in the Kalman Filter.
We need to extend our existing model of the water tank (shown above) with a new state for F_{out} - We can use the following approach:

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} h \\ F_{out} \end{bmatrix}$$

Set

$$x_1 = h$$

And

$$x_2 = F_{out}$$

Then assume that F_{out} is constant (which means that $\dot{F}_{out} = 0$)
This mean we can set our system on the following general state-space form:

$$\dot{x} = Ax + Bu$$
$$y = Cx + Du$$

Discrete State-space Model

Next, find the discrete state-space model for the system as well (both pen and paper and LabVIEW):

$$x_{k+1} = Ax_k + Bu_k$$
$$y_k = Cx_k + Du_k$$

Tip! Pen and Paper: Use Euler forward on each of the differential equations. Then put these discrete equations on a discrete state-space form

Use the Euler forward method:

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

The discrete state-space model can then be used in a Kalman Filter algorithm.

How does the computer find the discrete State-space model?

Given a continuous State-space model:

$$\dot{x} = A_c x + B_c u$$
$$y = C_c x + D_c u$$

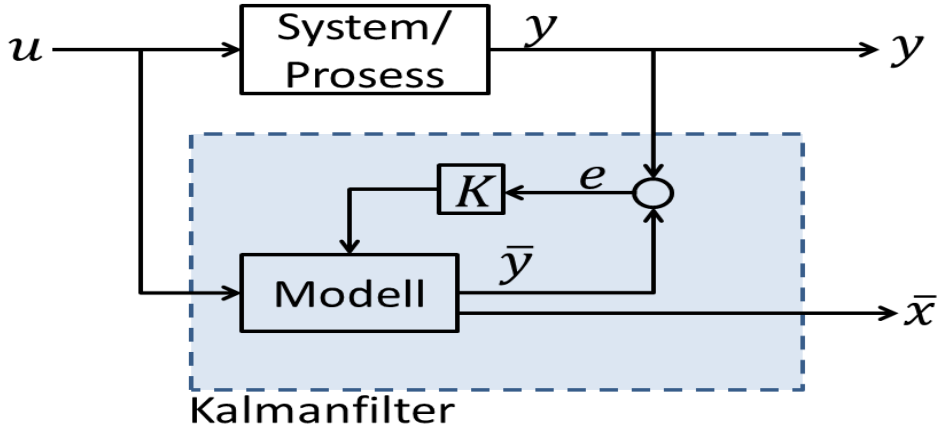
When using a computer we can use the matrix formula:

$$x_{k+1} = \underbrace{(I + T_s A_c)}_{A} x_k + \underbrace{T_s B_c}_{B} u_k$$
$$y_k = C x_k + D u_k$$

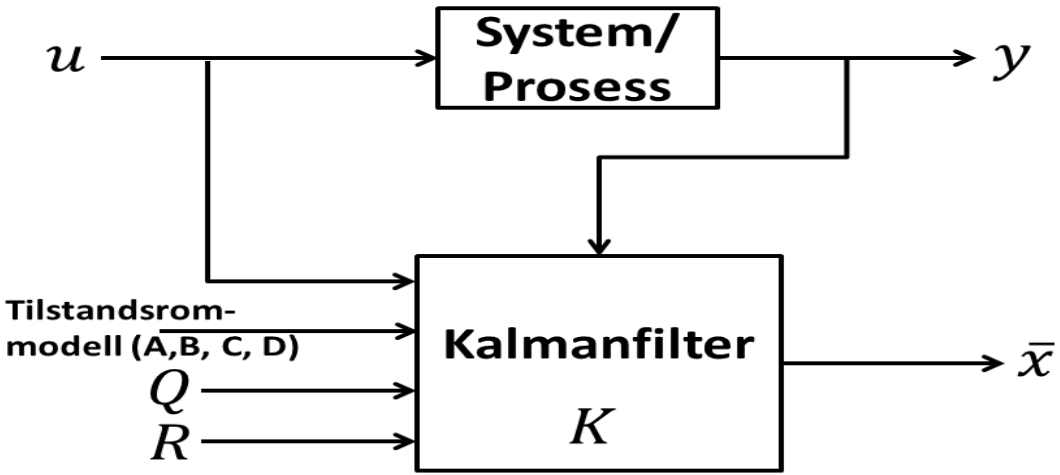
We can easily implement this equation in LabVIEW/MATLAB or use built-in c2d functions

This equation is derived using the Euler forward method on a general state-space model.

Kalman Filter

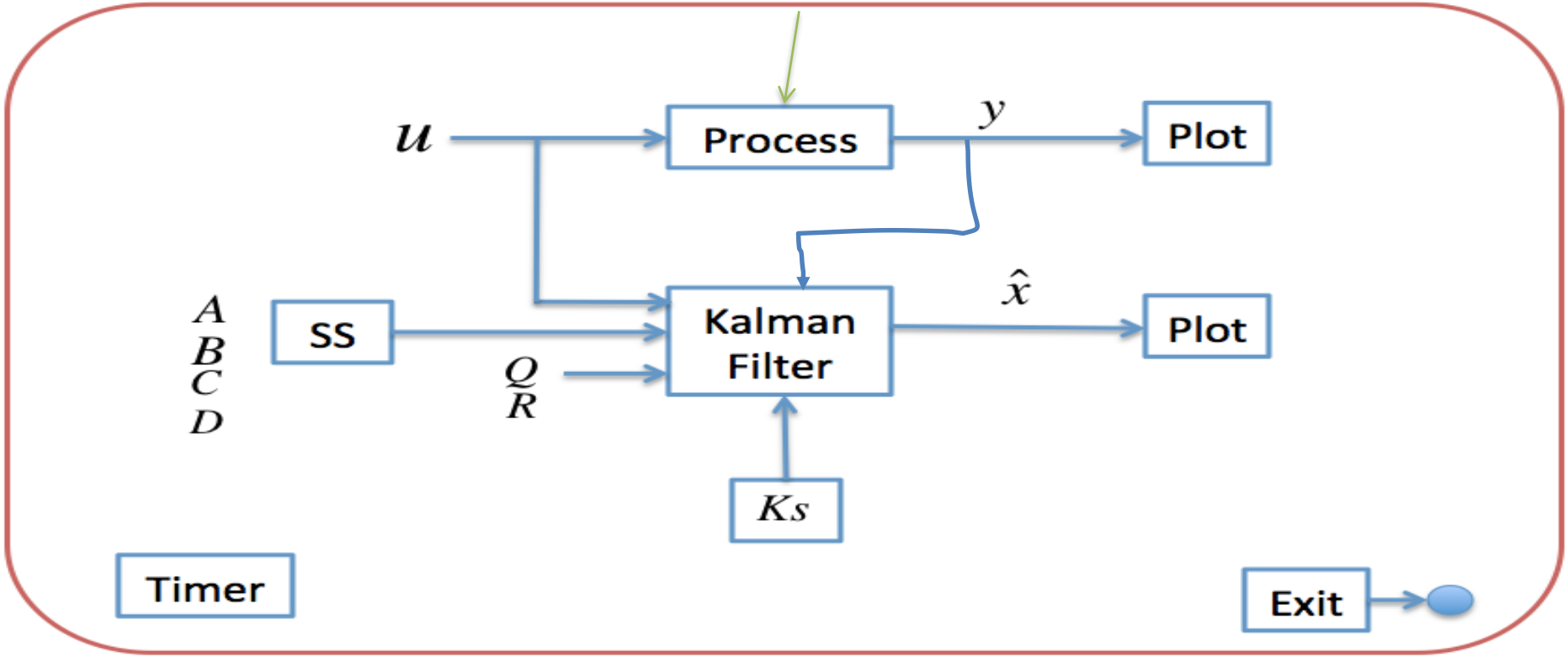


Start using a simulator (model).
When the simulator is working, switch to the real process



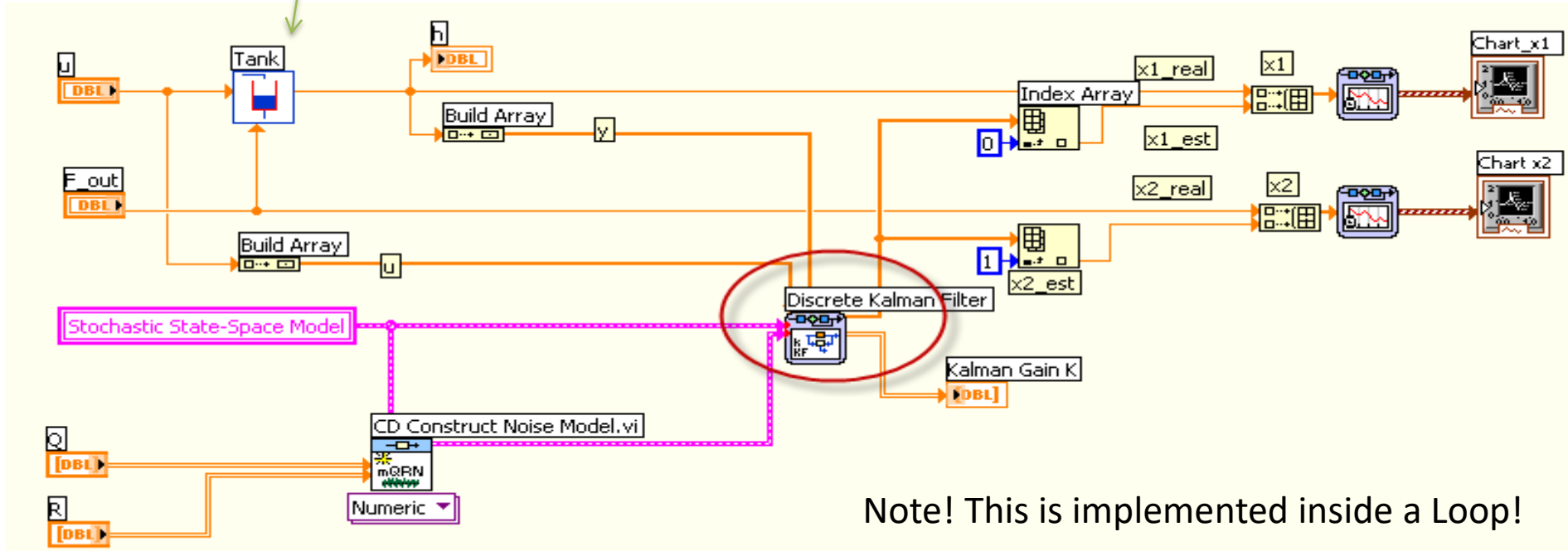
Sketch of Kalman Filter in LabVIEW

Start using a simulator (model). When the simulator is working, switch to the real process



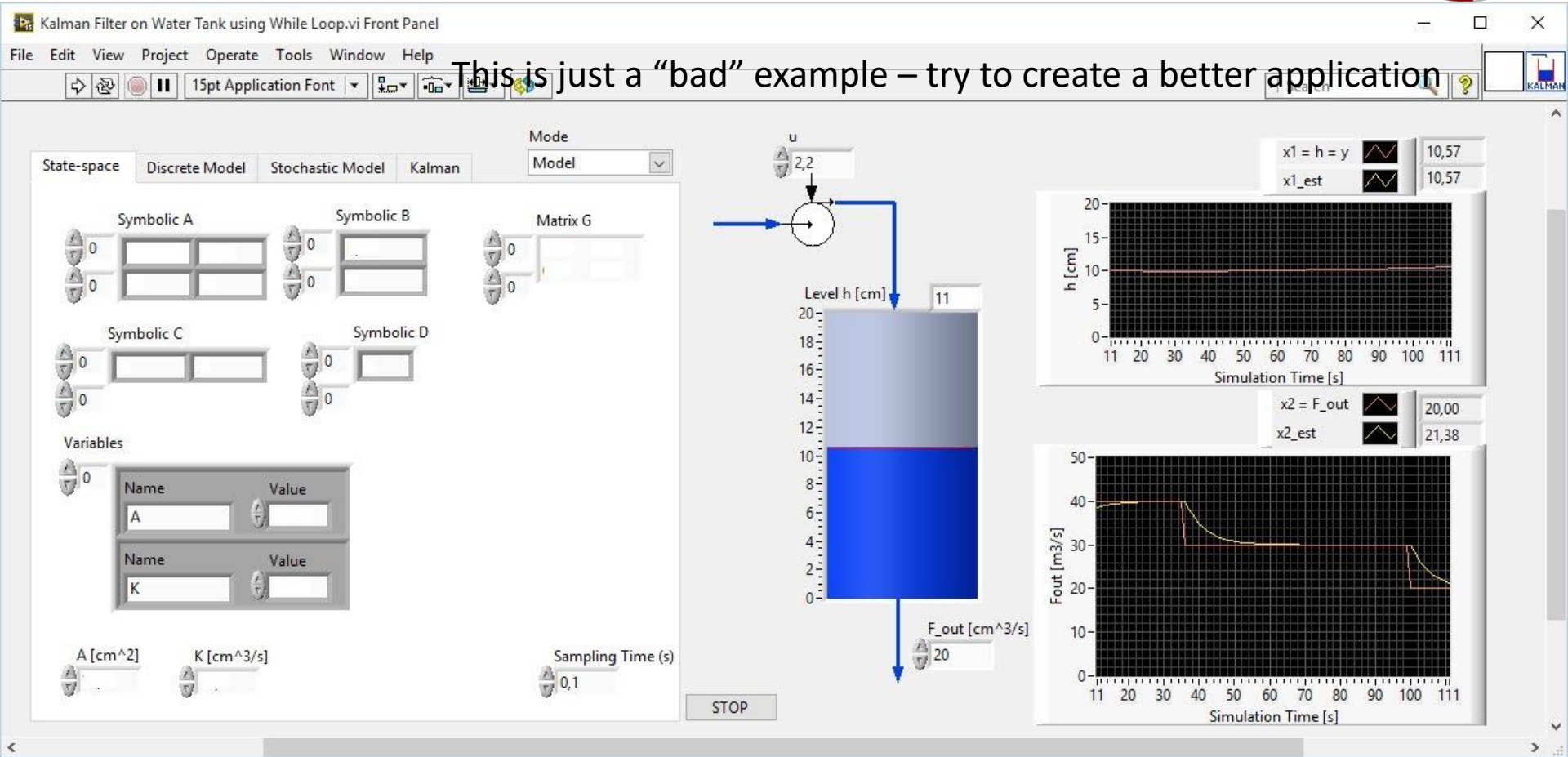
Kalman Filter in LabVIEW

Start using a simulator (model). When the simulator is working properly, switch to the real process. You may also add some noise to your model to make it more realistic.



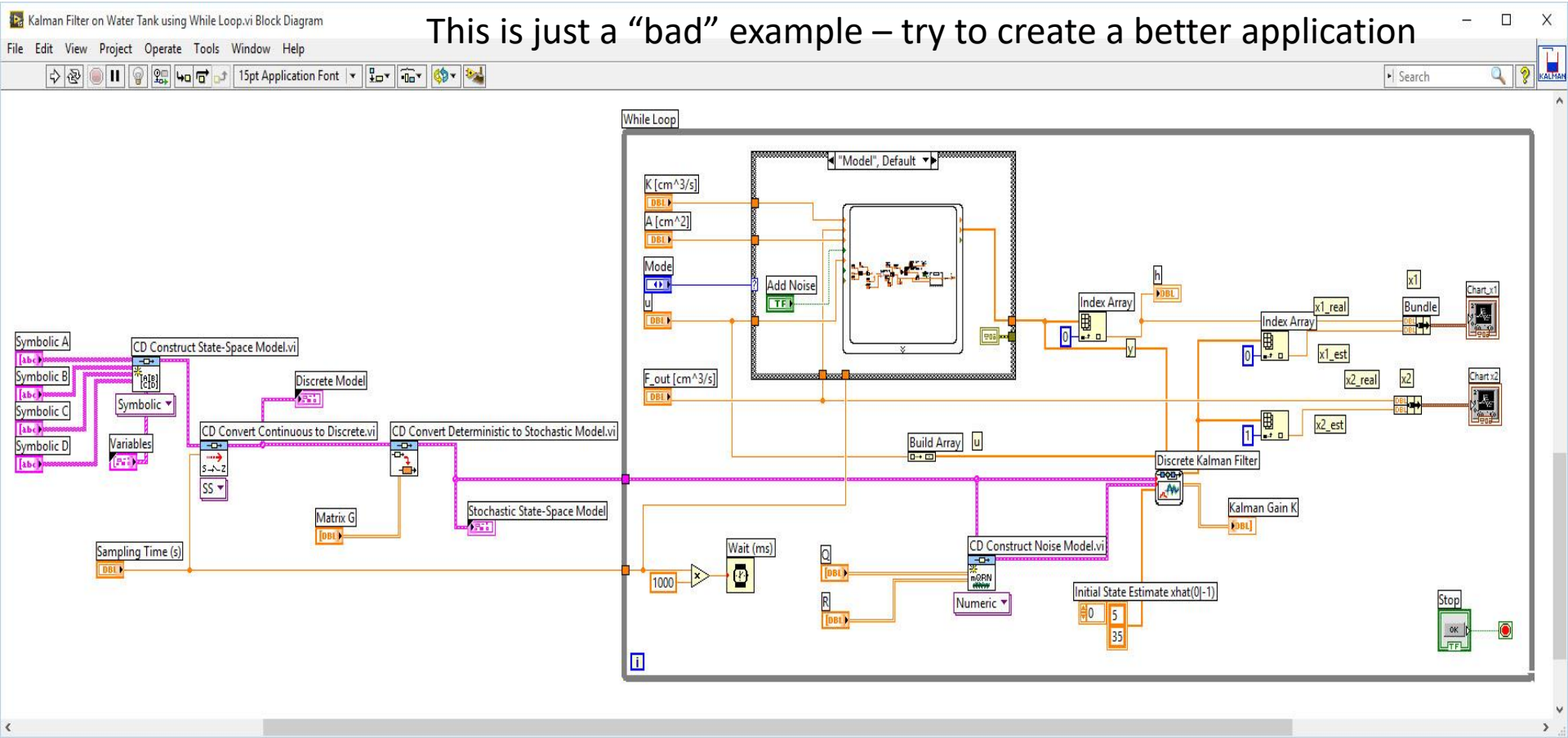
Note! This is implemented inside a Loop!

LabVIEW Example (Kalman Filter)



LabVIEW Example (Kalman Filter)

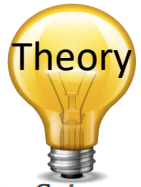
This is just a “bad” example – try to create a better application



Testing the Kalman Filter

- As with every model-based algorithm you should test your Kalman Filter with a simulated process before applying it to the real system.
- You can implement a simulator in LabVIEW since you already have a model (the Kalman Filter is model-based).
- In the testing, you can start with testing the Kalman Filter with the model in the simulator (without noise).
- Then you can introduce some noise in your simulator.
- You could also introduce some reasonable model errors by making the simulator model somewhat different from the Kalman Filter model, and check if the Kalman Filter still produces usable estimates.

Kalman Filter Algorithm



You may want to use this algorithm when you are creating your own Kalman Filter algorithm in LabVIEW

Pre Step: Find the steady state Kalman Gain K

K is time-varying, but you normally implement the steady state version of Kalman Gain K. Use the "CD Kalman Gain.vi" in LabVIEW or one of the functions kalman, kalman_d or lqe in MathScript.

Init Step: Set the initial Apriori (Predicted) state estimate

$$\bar{x}_0 = x_0$$

Step 1: Find Measurement model update

$$\bar{y}_k = g(\bar{x}_k, u_k)$$

For Linear State-space model:

$$\bar{y}_k = C\bar{x}_k + Du_k$$

Step 2: Find the Estimator Error

$$e_k = y_k - \bar{y}_k$$

Step 3: Find the Aposteriori (Corrected) state estimate

$$\hat{x}_k = \bar{x}_k + Ke_k$$

Where K is the Kalman Filter Gain. Use the steady state Kalman Gain or calculate the time-varying Kalman Gain.

Step 4: Find the Apriori (Predicted) state estimate update

$$\bar{x}_{k+1} = f(\hat{x}_k, u_k)$$

For Linear State-space model:

$$\bar{x}_{k+1} = A\hat{x}_k + Bu_k$$

Step 1-4 goes inside a loop in your program.

Note! Different notation is used in different literature:

Apriori (or Predicted) state estimate: $\bar{x}$ or x_p

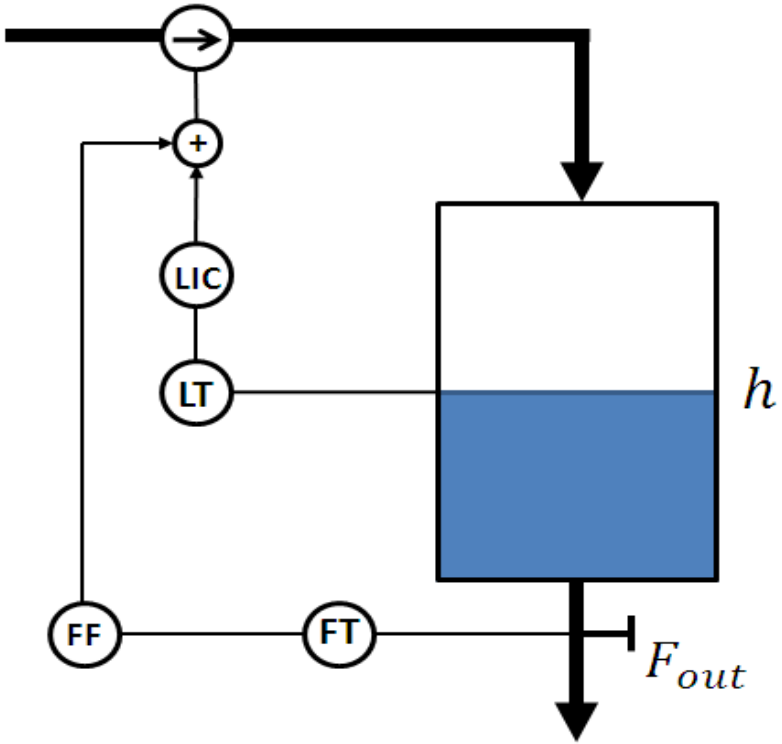
Aposteriori (or Corrected) state estimate: $\hat{x}$ or x_c

Congratulations! - You are finished with the Task

Control System

Hans-Petter Halvorsen

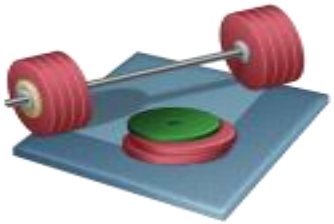
System Overview



PID Control

Hans-Petter Halvorsen, M.Sc.

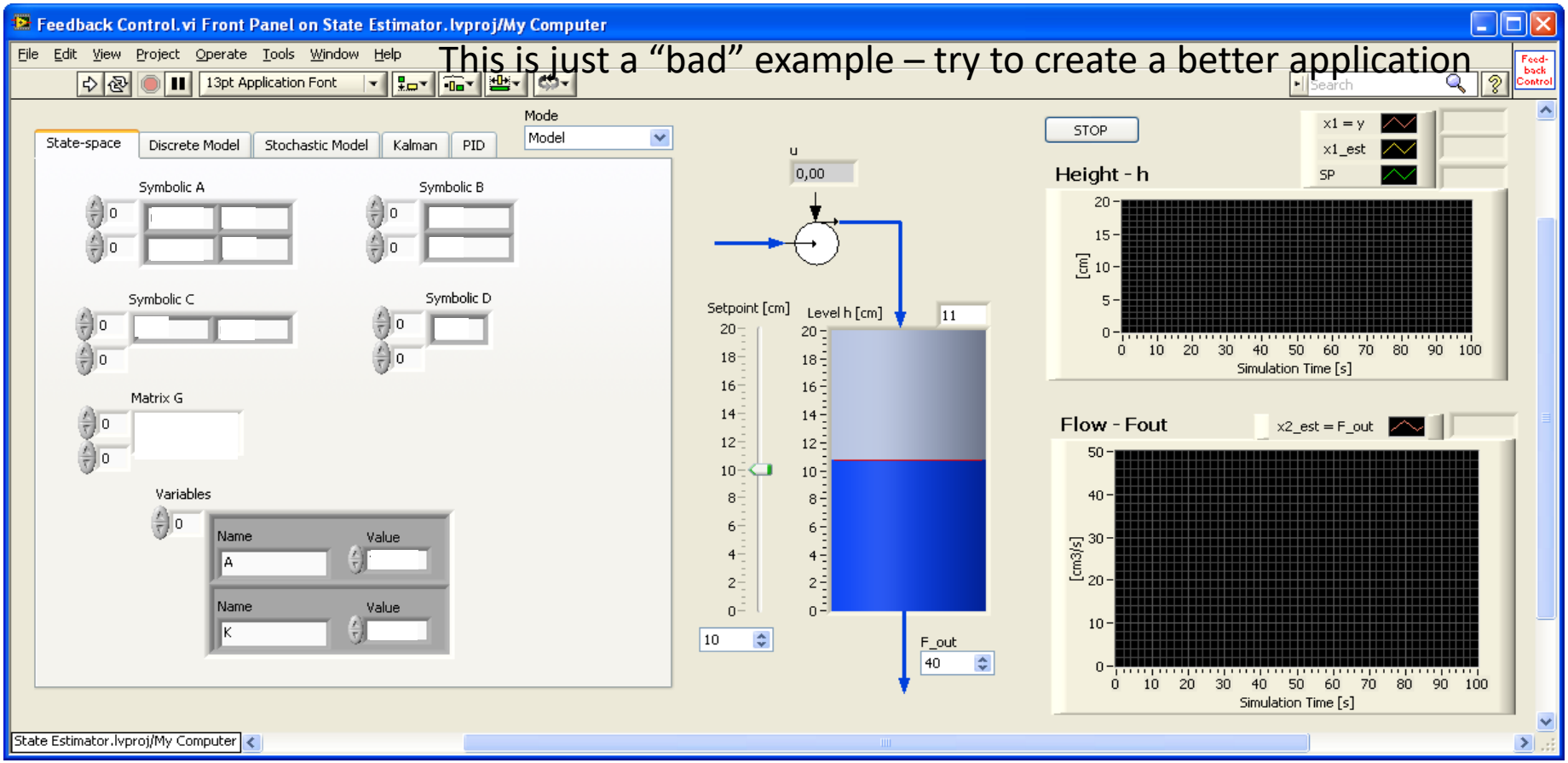
PID Control



- Control the model (and the real process) using standard PI(D) control
- Create proper GUI
- It should be possible to easily switch between the model and the real process

LabVIEW Example (PID + Kalman)

This is just a “bad” example – try to create a better application

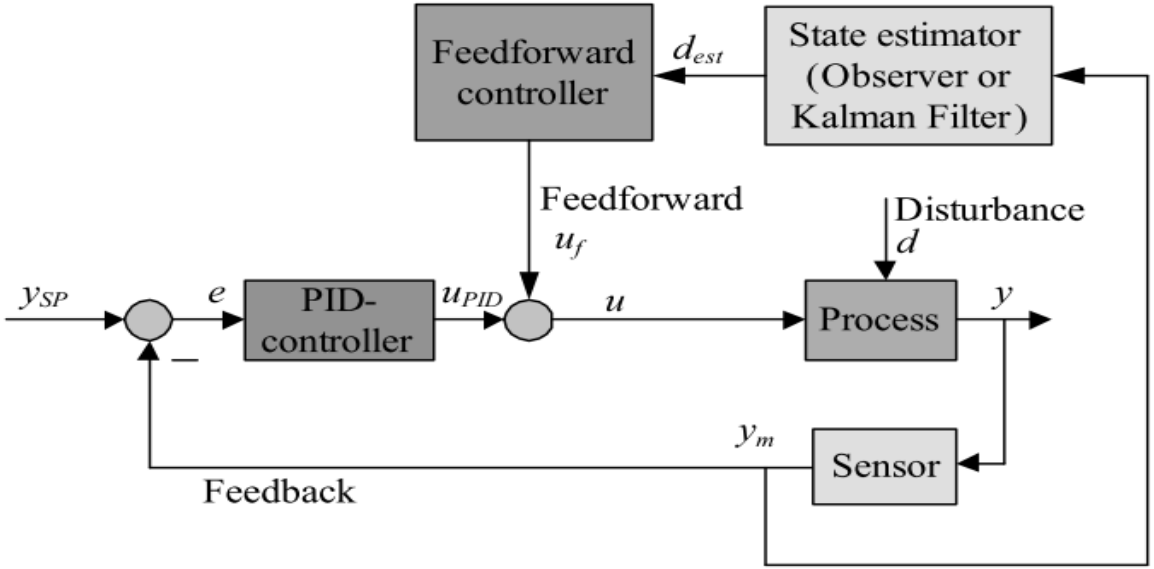


Congratulations! - You are finished with the Task

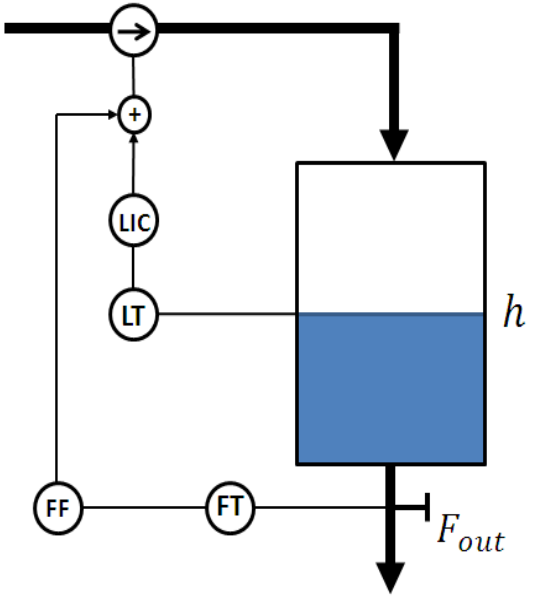
Feedforward Control

Hans-Petter Halvorsen, M.Sc.

Feedforward Control



[Figure: F. Haugen, Advanced Dynamics and Control: TechTeach, 2010]



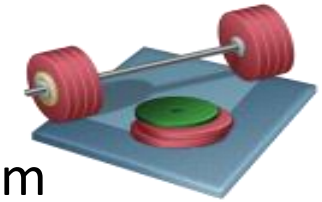
$$\dot{h} = \frac{1}{A_t}\left[K_p u - F_{out}\right]$$

Feedforward Control

- In this model is *Fout* a noise signal/disturbance that we want to remove by using Feedforward.
- We want to design the Feedforward controller so that *Fout* is eliminated.
  - Solve for the control variable *u*, and substituting the process output variable *h* by its setpoint h_{sp}.
  - *Fout* is not measured, so you need to use the estimated value instead. Assume that the setpoint is constant.

We will use Feedforward Control in order to improve the control, compared to ordinary Feedback Control.

Feedforward Control

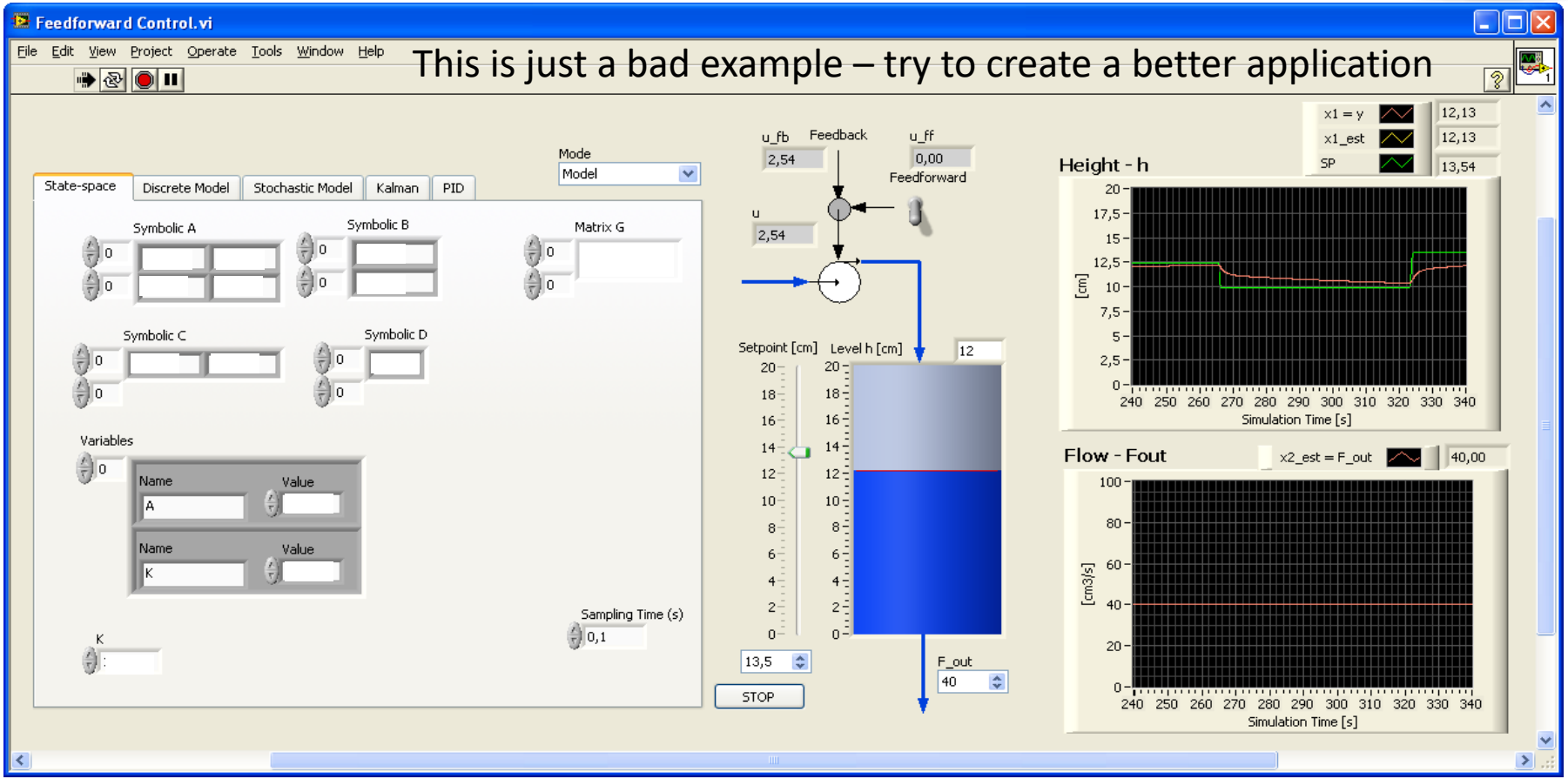


- You should first test it on the simulator then on the real system afterwards. Start by using the simulator and then extend the program to make it easy to switch between the real process and the simulator.
- Does the feedforward control improve the level control (compare with not using feedforward control, but only feedback control)?
- You should make it possible to turn the feedforward controller on/off from the Front Panel so it is easy to see the difference.
- Without feedforward control the control signal range of the PID controller is normally *[0, 5]*. With feedforward the output signal from the PID controller can be set to have the range *[-5, +5]*, so the contribution *upid* from the PID controller can be negative. If *upid* cannot be negative, the total signal *u=upid+uf* may not be small enough value to give proper control when the outflow is small. The signal to the DAQ device still needs to be limited to *0-5V* as before.

Final Control System with Kalman Filter

- It should be possible to easily switch between the (1) model and the (2) real process in your GUI
- It should be possible to easily switch between (3) Feedback Control and Feedback + (4) Feedforward Control in your GUI

LabVIEW Example (PID + Kalman + FF)



Congratulations! - You are finished with the Task

Model Predictive Control

Theory

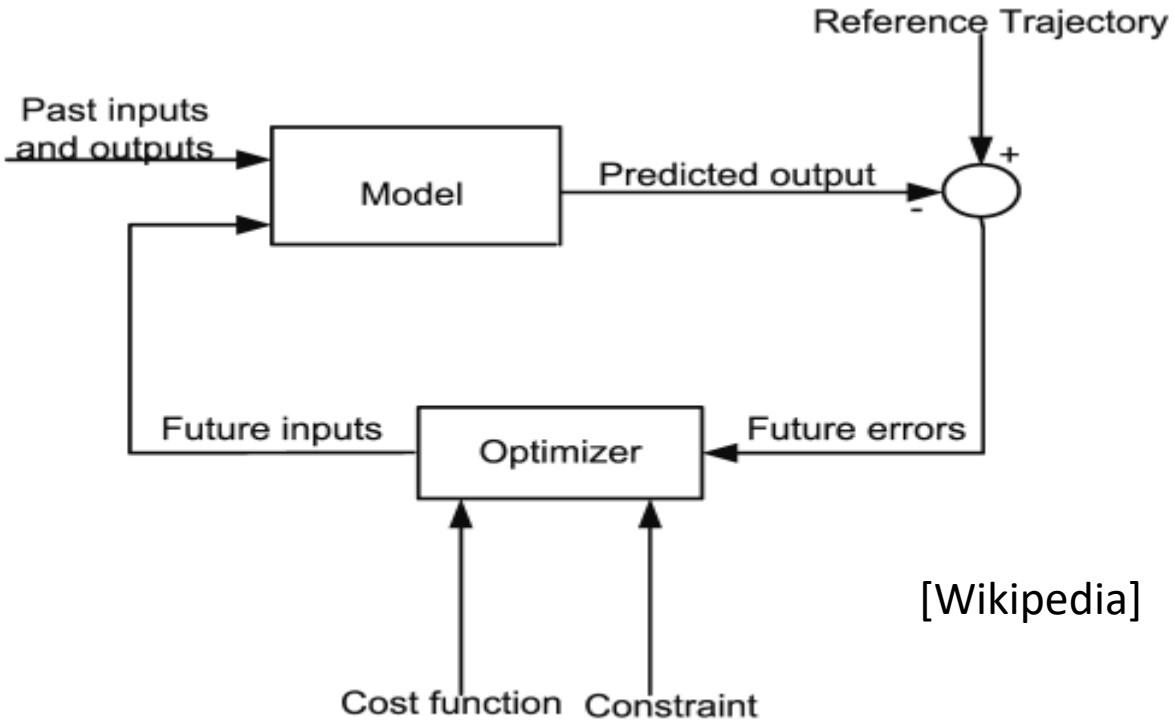
Hans-Petter Halvorsen, M.Sc.

Model Predictive Control (MPC)

- Model predictive control (MPC) is an advanced method of process control that has been in use in the process industries since the 1980s.
- Model Predictive Control (MPC) is a multivariable control algorithm.
- Model predictive controllers rely on dynamic models of the process, most often linear empirical models obtained by system identification.
- MPC is based on iterative, finite-horizon optimization of a plant model.
- This is achieved by optimizing a finite time-horizon, but only implementing the current timeslot. MPC has the ability to anticipate future events and can take control actions accordingly.

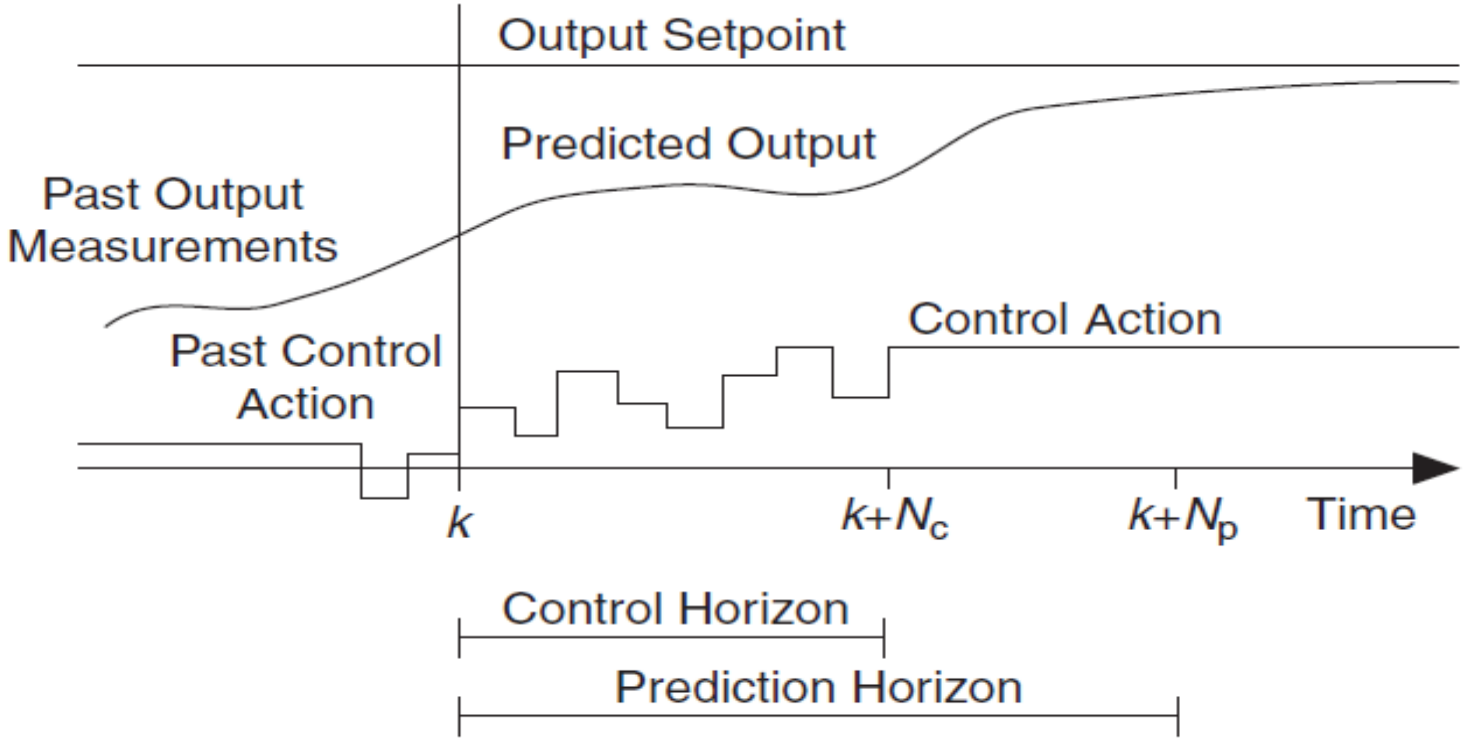
https://en.wikipedia.org/wiki/Model_predictive_control

Model Predictive Control (MPC)



[Wikipedia]

Model Predictive Control (MPC)



[Figure: National Instruments, LabVIEW Control Design user Manual, 2008.
Available: http://www.ni.com/pdf/manuals/371057f.pdf]

When constructing an MPC controller, you must provide the following information:

- Prediction horizon (N_p)—The number of samples in the future during which the MPC controller predicts the plant output.
- Control horizon (N_c) —The number of samples within the prediction horizon during which the MPC controller can affect the control action.
- Note!

$$N_c < N_p$$

Model Predictive Control (MPC)

The cost function often used in MPC is like this (a linear quadratic function):

$$J = \sum_{k=0}^{N_p} (\hat{y} - r)^T Q\, (\hat{y} - r) + \sum_{k=0}^{N_c} \Delta u^T R\, \Delta u$$

Where:

N_p – Prediction horizon, N_c – Control horizon

r – Set-point

$\hat{y}$ – Predicted process output

Δu – Predicted change in control value, $\Delta u_k = u_k - u_{k-1}$

Q – Output error weight matrix

R – Control weight matrix

So the basic problem is to solve:

$$\frac{\partial J}{\partial u} = 0$$

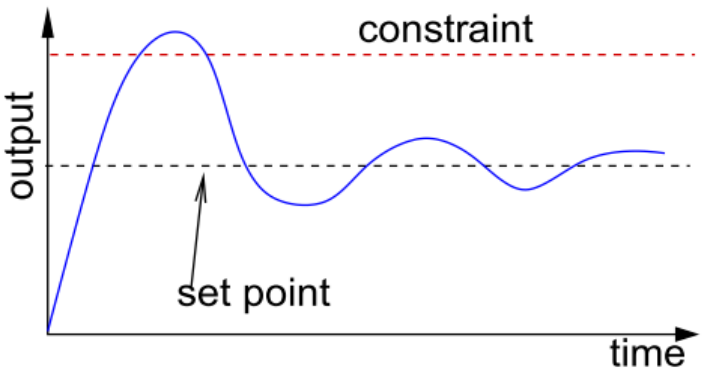
[National Instruments, LabVIEW Control Design user Manual, 2008. Available: http://www.ni.com/pdf/manuals/371057f.pdf]

PID vs. MPC

- MPC is often used in addition to traditional control like PID – not as a replacement.
- In large plants MPC is not a replacement for traditional PID, but used in addition to PID controllers.
- PID controllers are used as single-loop controllers, while MPC is used as an overall system.
- PID handles only a single input and a single output (SISO systems), while MPC is a more advanced method of process control used for MIMO systems (Multiple Inputs, multiple Outputs).

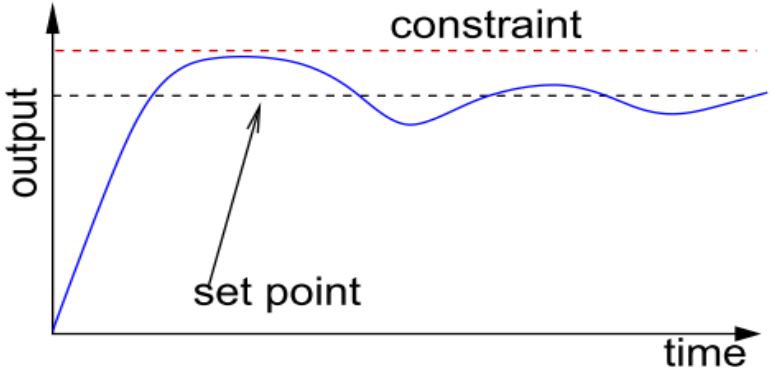
PID vs. MPC

Traditional Control (PID)



- No knowledge about constraints
- Set-point far from constraints
- Not optimal process operation
- SISO systems
- A mathematical model is not needed

MPC



- Constraints included in the design
- Set-point can be closer to constraints
- Improved process operation
- MIMO systems
- A mathematical model is needed

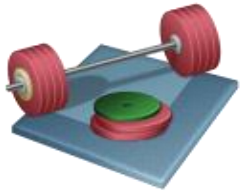
Another advantage of MPC is that cross coupling in multiple input and multiple output (MIMO) systems are taken into consideration in an optimal way. MPC is a simple method for controlling MIMO systems.

MPC in LabVIEW

Model Predictive Control

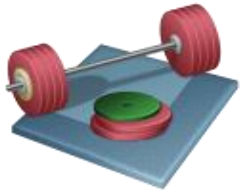
Hans-Petter Halvorsen

MPC Implementation in LabVIEW



- Implement MPC for the Level Tank System in LabVIEW using the built-in MPC functionality in LabVIEW.
- If you prefer, you can instead use the MPC Toolbox in MATLAB combined with a “MATLAB Script Node in LabVIEW”

PID vs. MPC

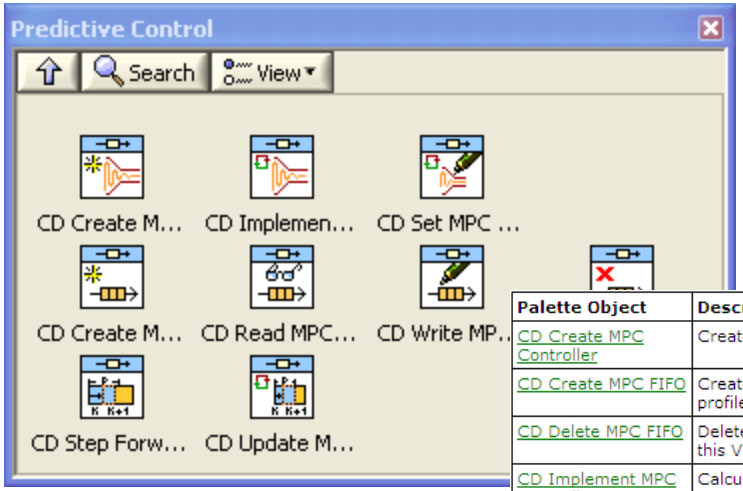


Compare your Control Systems using PID and MPC respectively:

- Ease of Implementation
- Complexity
- Behavior
- Performance
- Advantages and Disadvantages
- ...

MPC in LabVIEW

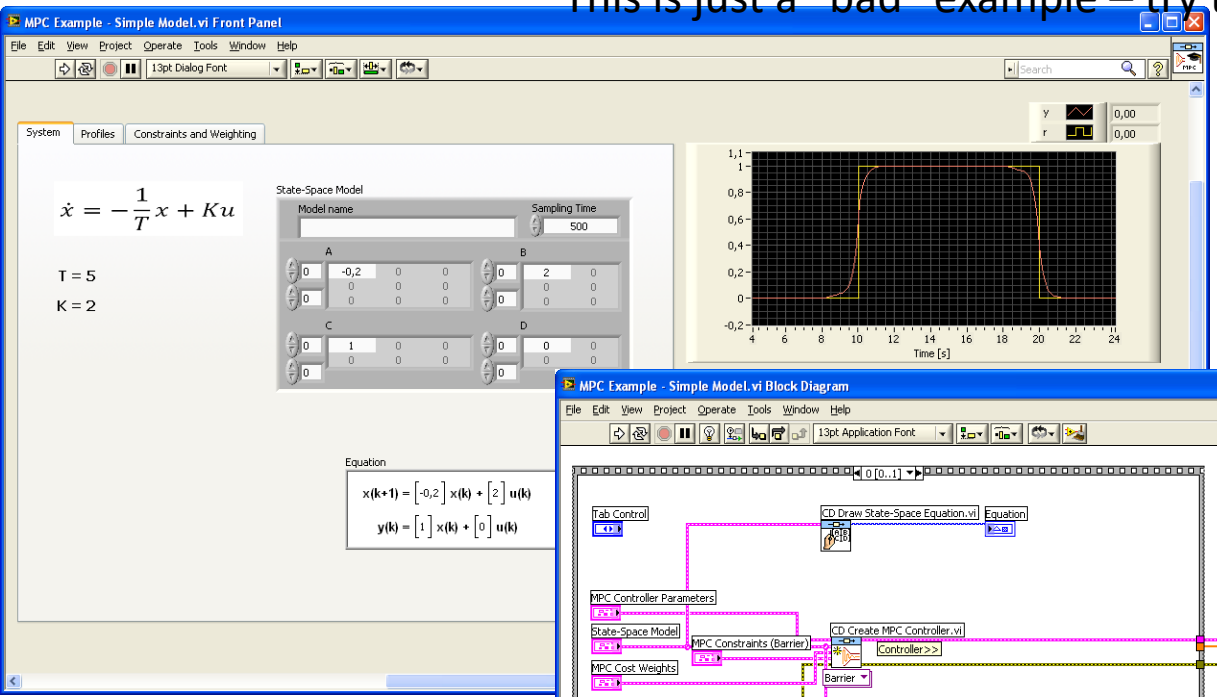
In LabVIEW you have the following Predictive Control palette:

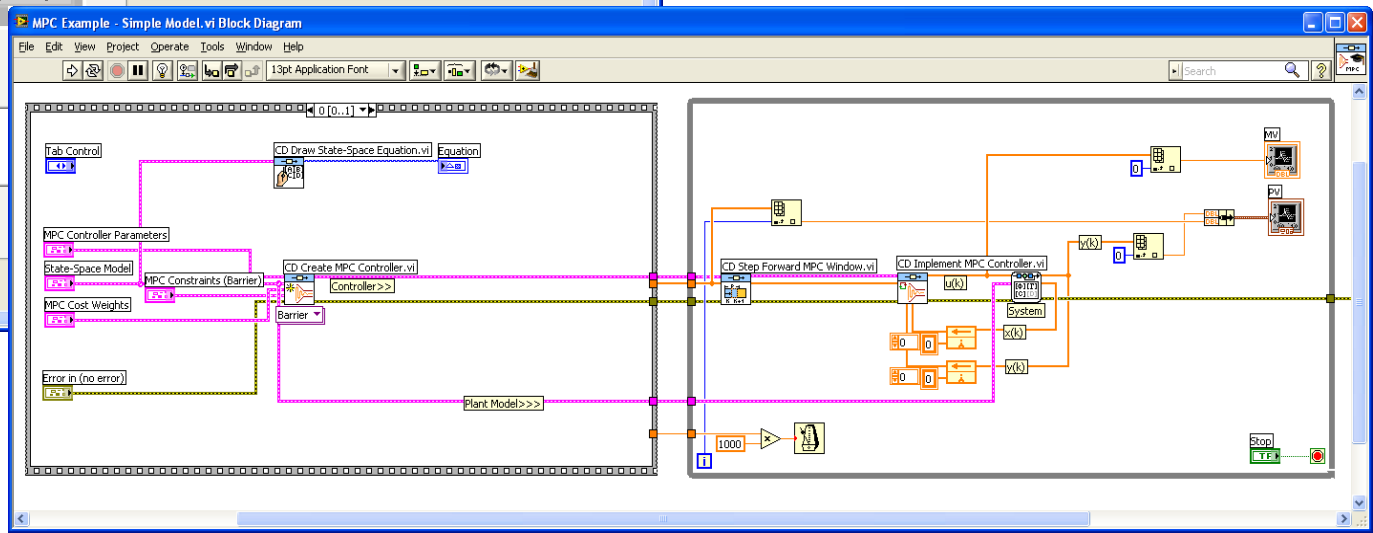


Palette Object	Description
CD Create MPC Controller	Creates a model predictive control (MPC) controller for a state-space model. You must manually select the polymorphic instance to use.
CD Create MPC FIFO	Creates a queue or real-time (RT) FIFO for an MPC controller. You use this queue or RT FIFO to update setpoint and/or disturbance profiles dynamically.
CD Delete MPC FIFO	Deletes the **MPC FIFO**. After you delete this FIFO, the CD Write MPC FIFO VI stops writing data to the FIFO and the loop that contains this VI terminates.
CD Implement MPC Controller	Calculates the **Control Action u(k)** to apply to the plant. This VI uses the **Output Reference Window**, **Disturbance Window**, and **Control Action Reference Window** parameters to calculate the control action along the control horizon at time *k*.
CD Read MPC FIFO	Reads a portion, or window, of profile values from the **MPC FIFO**.
CD Set MPC Controller	Updates specified parameters of a model predictive control (MPC) controller for a state-space model. You must manually select the polymorphic instance to use.
CD Step Forward MPC Window	Calculates the appropriate portion, or window, of the setpoint and/or disturbance profiles. You wire these windows to the appropriate input(s) of the CD Implement MPC Controller VI.
CD Update MPC Window	Calculates the appropriate portion, or window, of the setpoint or disturbance profile of a signal from time *k* to time *k* + **Prediction Horizon**. You wire these windows to the appropriate input(s) of the CD Implement MPC Controller VI.
CD Write MPC FIFO	Writes a control action setpoint, output setpoint, or disturbance profile window to the **MPC FIFO**. You then use the CD Read MPC FIFO VI to read values from this MPC FIFO.

MPC Example in LabVIEW

This is just a “bad” example – try to create a better application





MPC - Tips and Tricks

- You can use the given examples as a starting point or build entirely from scratch.
- You must anyway have understanding, both in terms of implementation in LabVIEW and basic principles regarding basic MPC theory.
- Play and Explore: It is important that you ”Add Value" to your code compared to the given examples.

Congratulations! - You are finished with the Task

Congratulations! - You are finished with all the Tasks in the Assignment!

Hans-Petter Halvorsen

University of South-Eastern Norway

www.usn.no

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

Web: https://www.halvorsen.blog