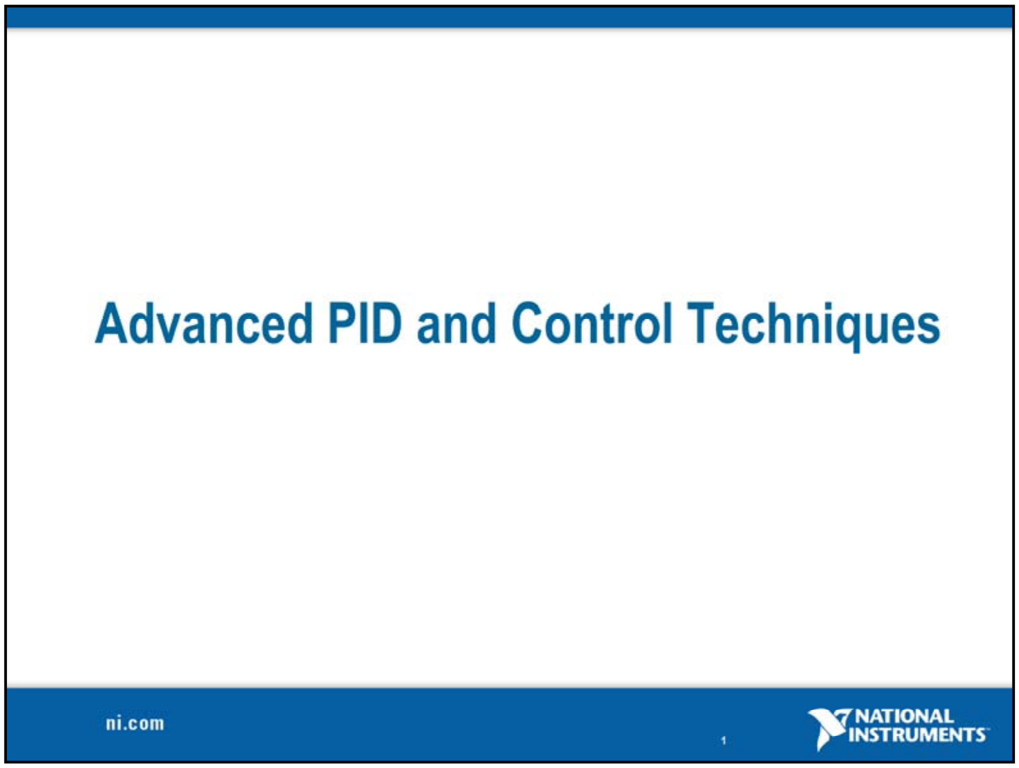


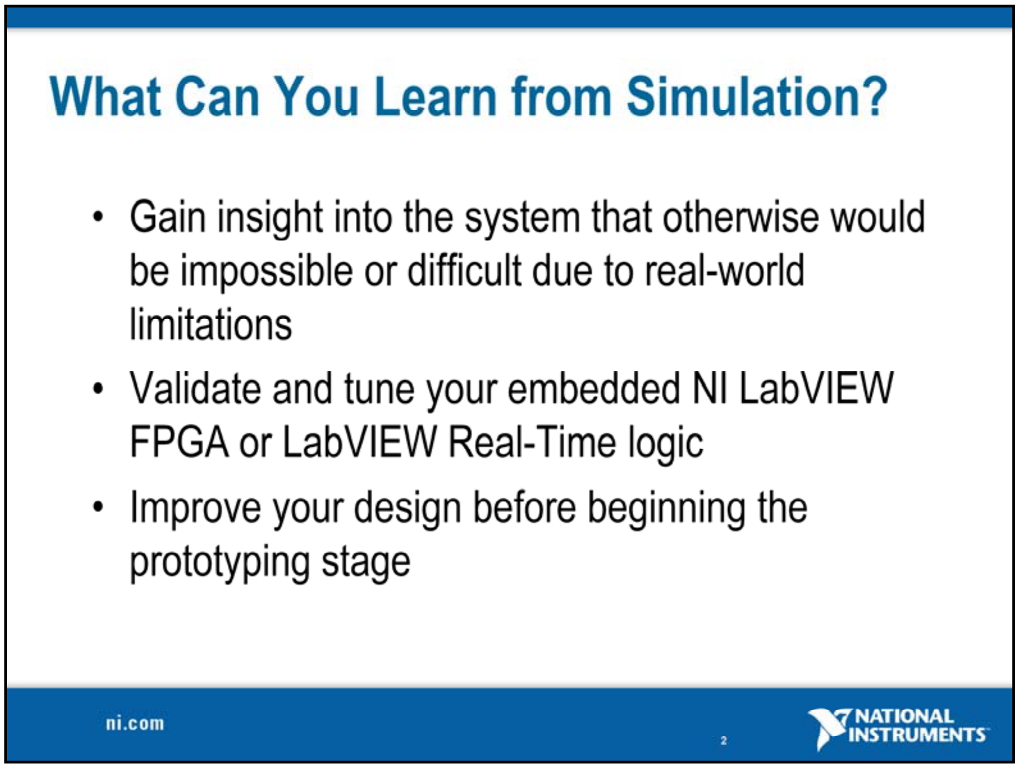
Advanced PID and Control Techniques

ni.com

1

Welcome to this session on how to design and implement control algorithms.

In this session we are going to talk about how to use the most-common control algorithm (PID) and some tips and trics to improve the performance, tune, find loop rates, etc



What Can You Learn from Simulation?

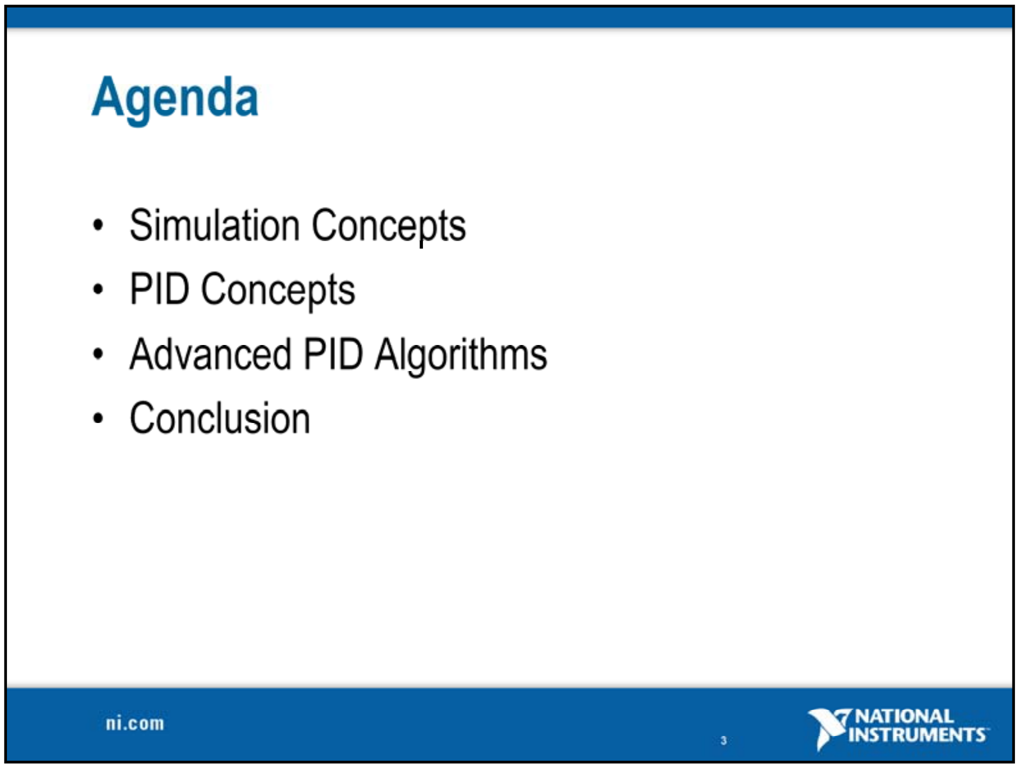
- Gain insight into the system that otherwise would be impossible or difficult due to real-world limitations
- Validate and tune your embedded NI LabVIEW FPGA or LabVIEW Real-Time logic
- Improve your design before beginning the prototyping stage

ni.com 2 NATIONAL INSTRUMENTS

Simulation can provide a lot of information about what the system is doing. We can add the effect of different element, like Encoder resolution, ADC sampling rate, etc and understand how does it affect the system. Another benefit is the possibility to slow down the process, to gain insight on what happens of fast systems or, vice versa, speed up slow process.

Because National Instruments is a unique company in the fact the we provide a full hardware solution couple with a full development platform, we can use this tools in very particular ways

For example, it is possible to combine the LabVIEW Control Design and Simulation Module along with the FPGA module to quickly prototype a controller. One of the main benefits of this approach is that there is no need to compile the FPGA code, which allows for a quick algorithm design turnaround and debugging.



Agenda

- Simulation Concepts
- PID Concepts
- Advanced PID Algorithms
- Conclusion

ni.com

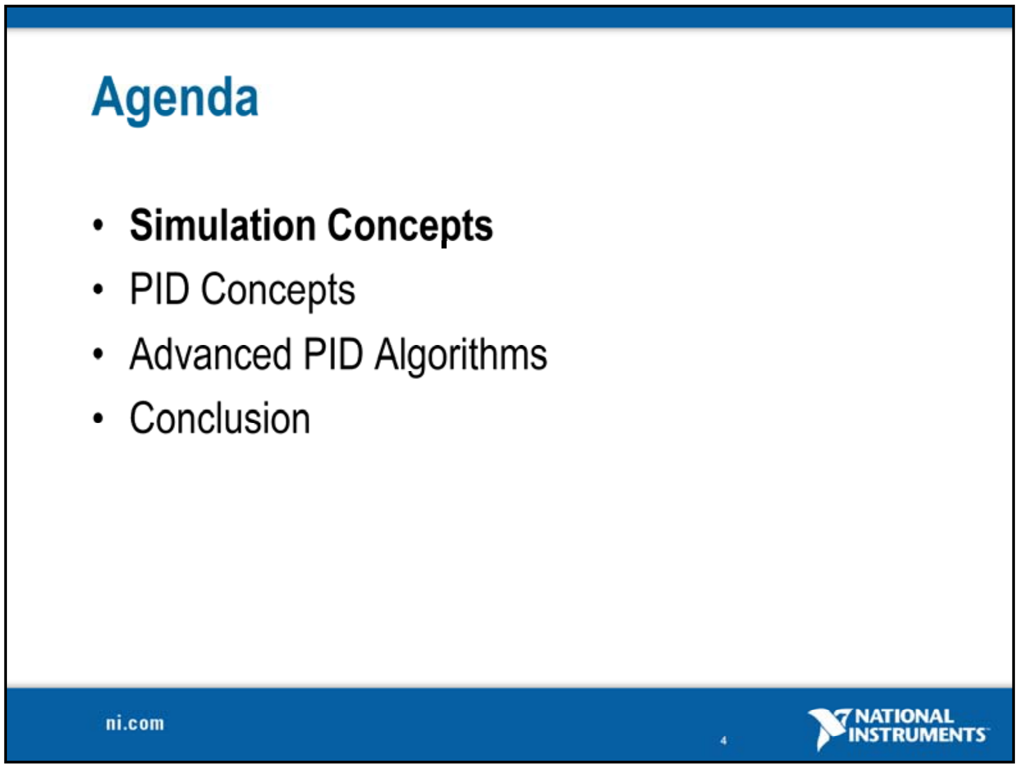
3

Here is the flow of the presentation I'm going to follow. First I'm going to talk about how to use simulation. I know, this might be a scary topic for you but don't worry; you don't need a PhD in control to follow the presentation

Second, we are going to briefly introduce the PID algorithms, how to use it within the NI software platform and some tips about tuning and performance

In the third section of the presentation I'm going to talk on how to improve the PID algorithm and some other control algorithms present in our platform.

We will finish the presentation with some conclusions and a location to find more information



Agenda

- **Simulation Concepts**
- PID Concepts
- Advanced PID Algorithms
- Conclusion

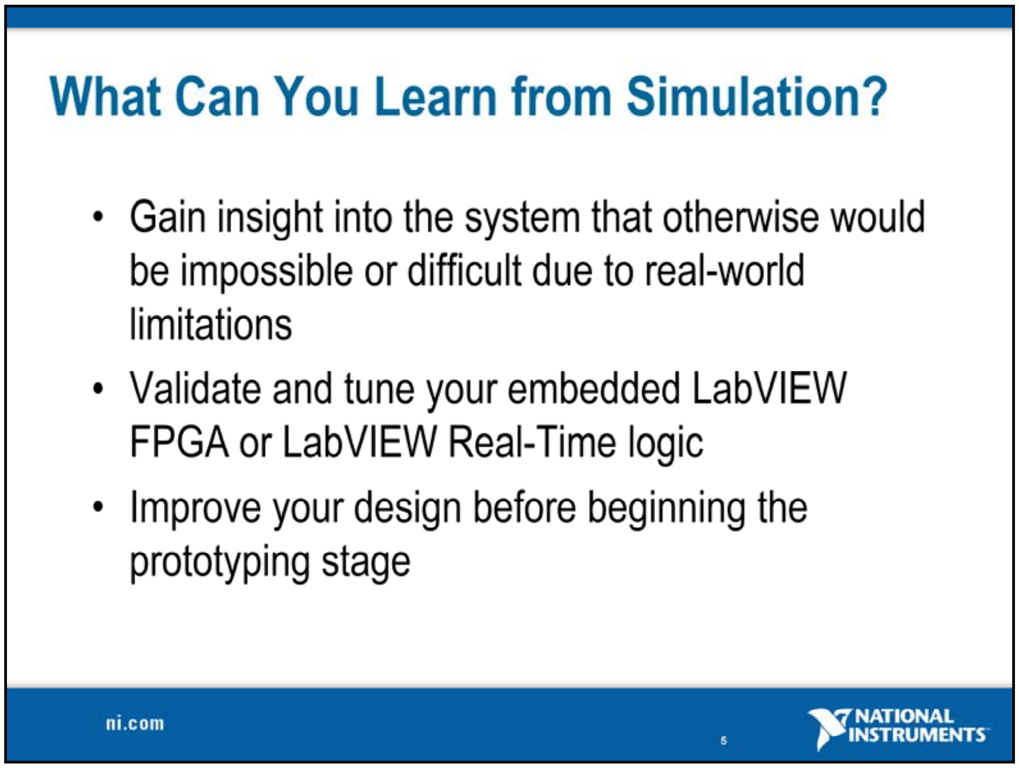
ni.com 4

Here is the flow of the presentation I'm going to follow. First I'm going to talk about how to use simulation. I know, this might be a scary topic for you but don't worry; you won't need a PhD in control to follow the presentation

Second, we are going to briefly introduce the PID algorithms, how to use it within the NI software platform and some tips about tuning and performance

In the third section of the presentation I'm going to talk on how to improve the PID algorithm and some other control algorithms present in our platform.

We will finish the presentation with some conclusions and a location to find more information



What Can You Learn from Simulation?

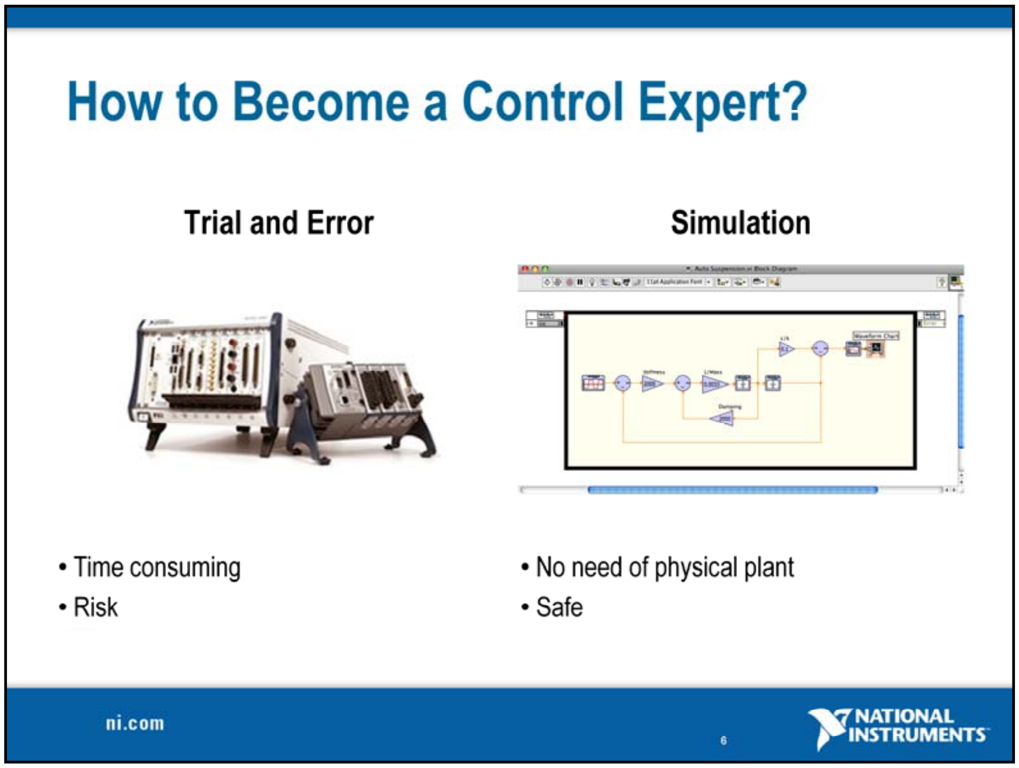
- Gain insight into the system that otherwise would be impossible or difficult due to real-world limitations
- Validate and tune your embedded LabVIEW FPGA or LabVIEW Real-Time logic
- Improve your design before beginning the prototyping stage

ni.com 5 NATIONAL INSTRUMENTS

Simulation can provide a lot of information about what the system is doing. We can add the efect of different element, like Encoder resolution, ADC sampling rate, etc and understand how does it affect the system. Another benefit is the posibility to slow down the process, to gain insight on what happens of fast systems or, vice versa, speed up slow process.

Because National Instruments is a unique company in the fact the we provide a full hardware solution couple with a full development platform, we can use this tools in very particular ways

For example, it is possible to combine the LabVIEW Control Design and Simulation Module along with the FPGA module to quickly prototype a controller. One of the main benefits of this approach is that there is no need to compile the FPGA code, which allows for a quick algorithm design turnaround and debugging.

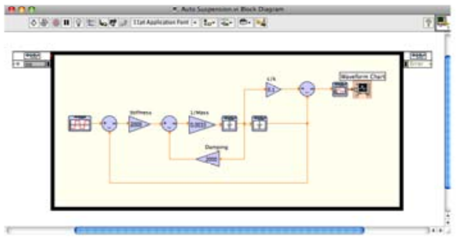


How to Become a Control Expert?

Trial and Error

- Time consuming
- Risk

Simulation



- No need of physical plant
- Safe

ni.com

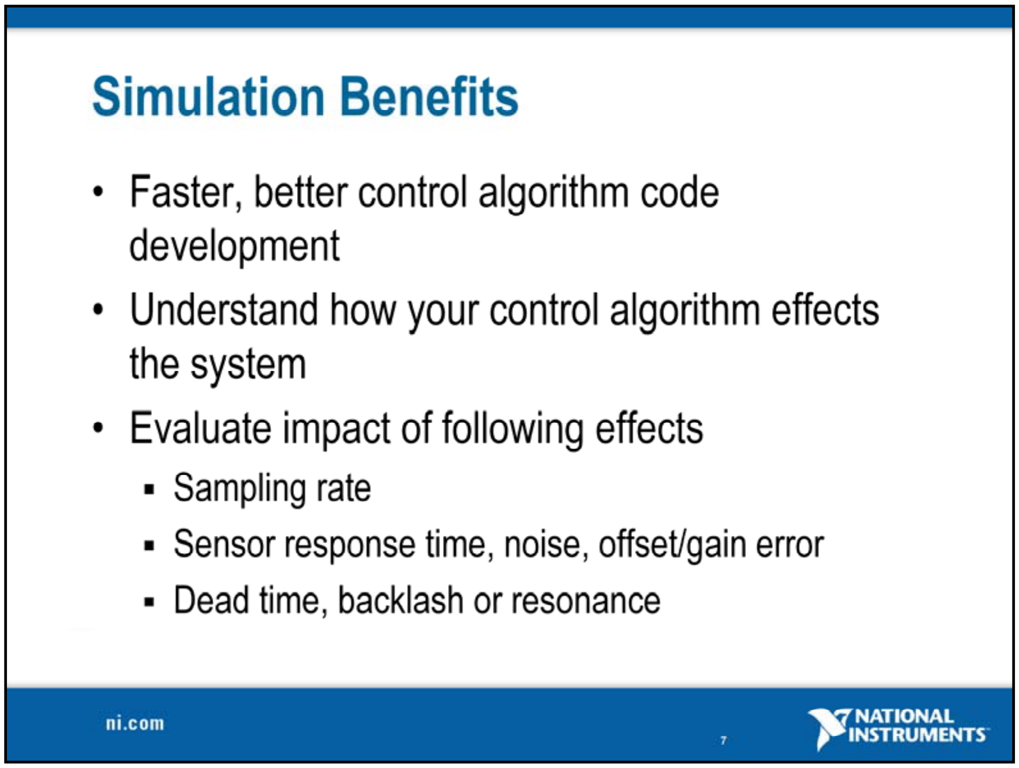
6

How often have you been waiting for a piece of component to start programming your control algorithm? You can always start with other areas in your application, like communication, dataloging and such, but there is often a piece that is missing and that is delaying the whole project.

Typically, PID implementation and tuning requires a lot of trial and error. This approach has several drawback, like the need of an expert to correctly tune the control system or the risk of putting the system under unstable operations and damage it

How many of you have tuned a controller? How often has the control done something unexpectedly? I now that, back at corporate, when our motion experts tune a system, they always have a hand in the E-button just in case.

All these topics can be solved or, at least, attenuated using simulation. Fortunately, we can use the same programming environment that we are already using to program our controllers, usually on and embedded controller like compactRIO and the user interface to simulate the dynamics of the system we want to control.



Simulation Benefits

- Faster, better control algorithm code development
- Understand how your control algorithm effects the system
- Evaluate impact of following effects
  - Sampling rate
  - Sensor response time, noise, offset/gain error
  - Dead time, backlash or resonance

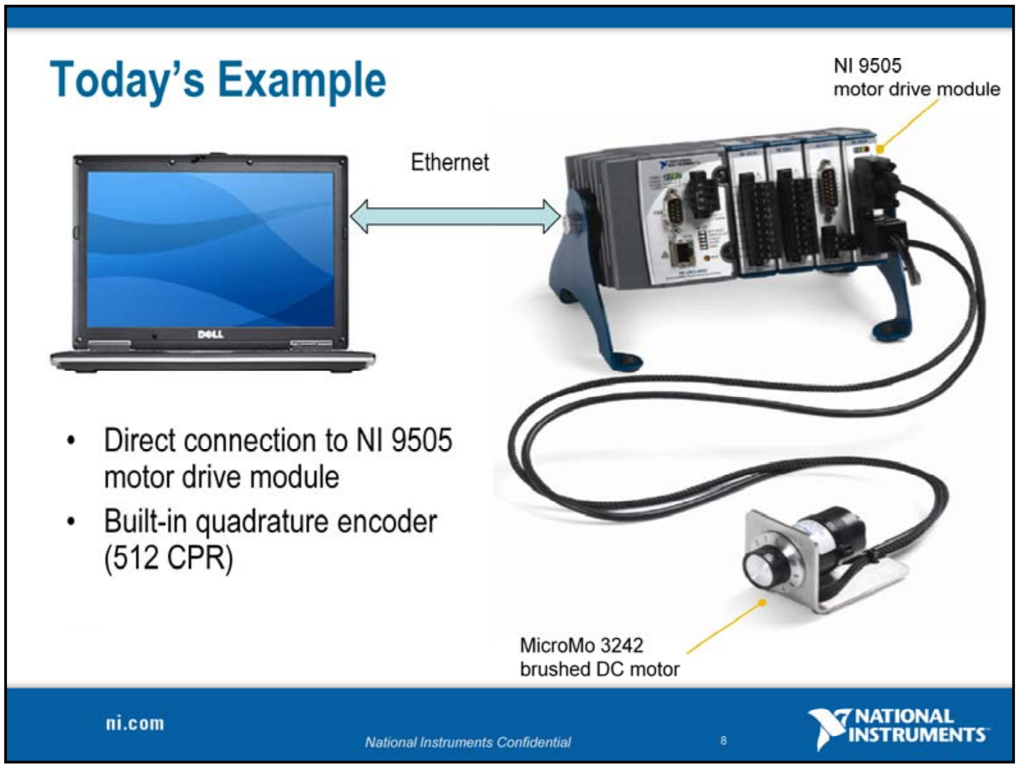
ni.com 7 NATIONAL INSTRUMENTS

As mentioned before, using simulation when building your application have a number of benefits including

-Faster algorithm code development: When working with software tools it is easy to quickly iterate between different controller configuration and gains while testing on the virtual plant

- Effects of the control algorithm on the system. LabVIEW interactive capabilities can be use to quickly and intuitively understand what is the effect of the control algorithms in the plant behavior

- Effect impact: not only is important to understand how the control algorithm works, but also to take into consideration implementation details like the controller sampling rate, the effect of sensor parameter and non-linear effect that are present on a system like dead times, backlash or resonance.



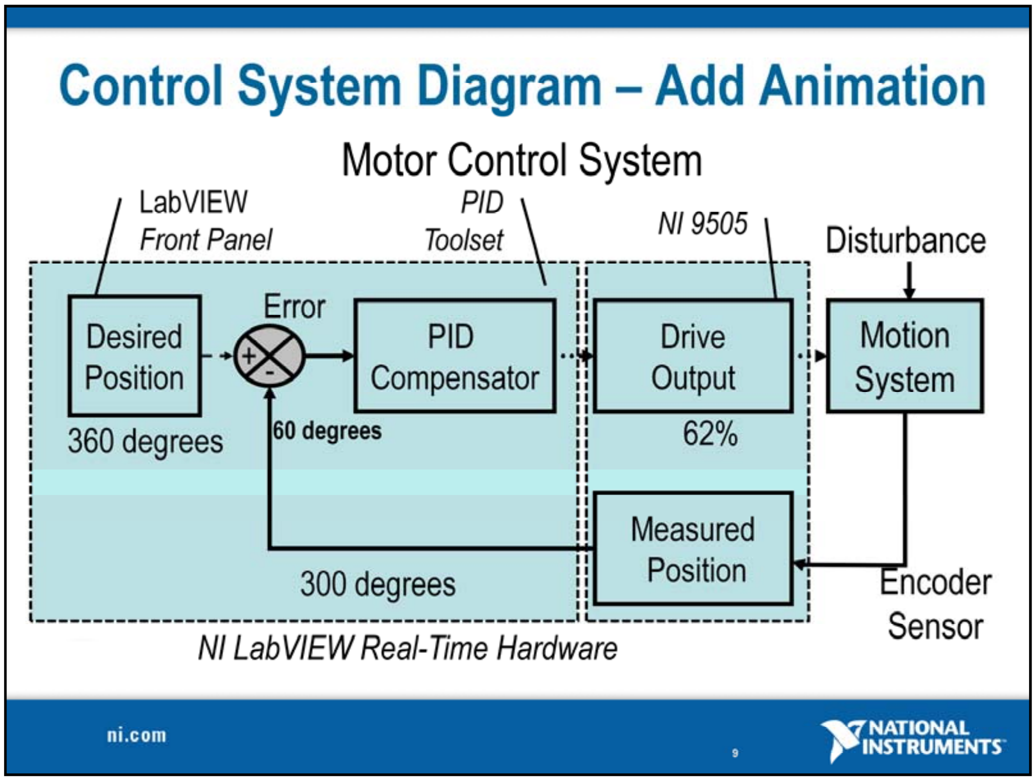
Today's Example

NI 9505
motor drive module
Ethernet
MicroMo 3242
brushed DC motor
ni.com
National Instruments Confidential
8
NATIONAL INSTRUMENTS

- Direct connection to NI 9505 motor drive module
- Built-in quadrature encoder (512 CPR)

Since we don't have hardware present for the demonstration, we are going to simulate the following setup.

The objective of the controller is to precisely control the position of a brush DC motor. To do so, we are going to use a NI 9505 Servo Drive module. This module is responsible for generating the PWM signal that will power the DC motor while reading the quadrature encoder to read the position. The NI 9505 Servo Module will be embedded on a compactRIO embedded controller where our PID control algorithm is going to run on a Real Time operating System. Typically and computer (either laptop or desktop) is connected using an Ethernet cable to provide a user interface

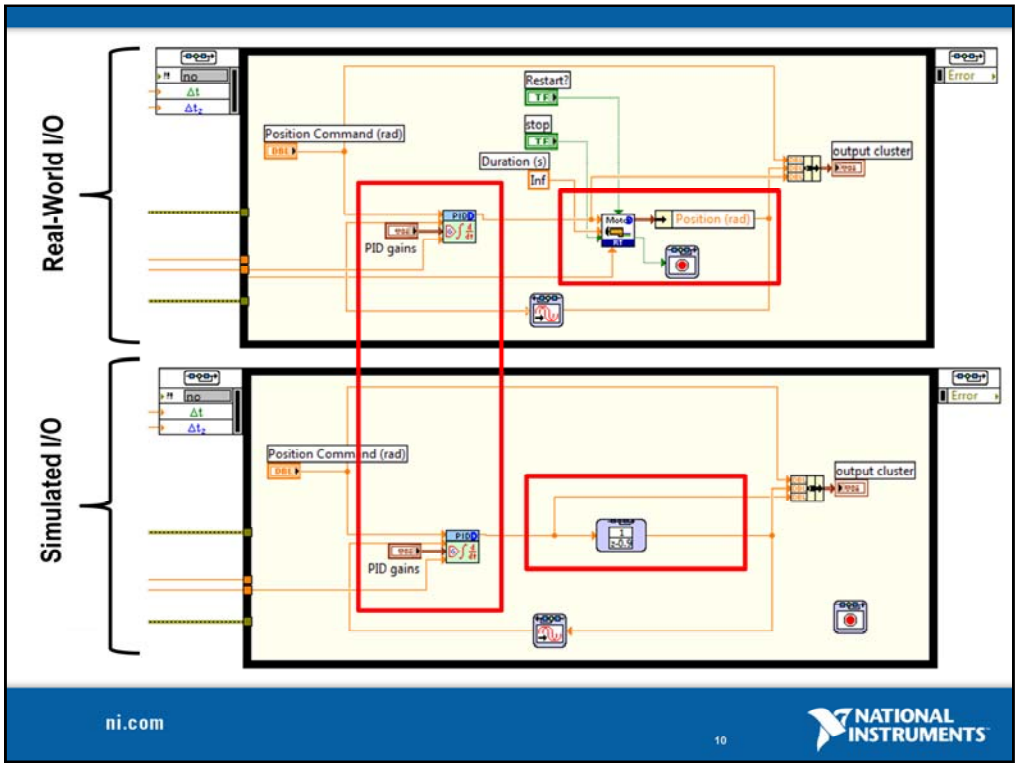

Control System Diagram – Add Animation

In this slide we are showing a diagram of our full system

The DC Motor is the system to be controlled. It's being act by the PWM signal being generated by the 9505 and the position feedback is measure with a quadrature encoder.

The final objective of the controller is to move the motor to a desired position. To accomplish that we will compute the difference (or error) between out desired position (or setpoint) and whe current position measured through a quadrature encoder. Once we have that error, we will feed that information to our PID compensator that will generate the neccesary output to be used on our motion system.

As mentioned with the previous slide, the 9505 is the responsible of generating the PWM output based on the reference signal and also to decode the encoder signal from the motor. The setpoint, PID calculation and error are computed in the cRIO embedded controller.

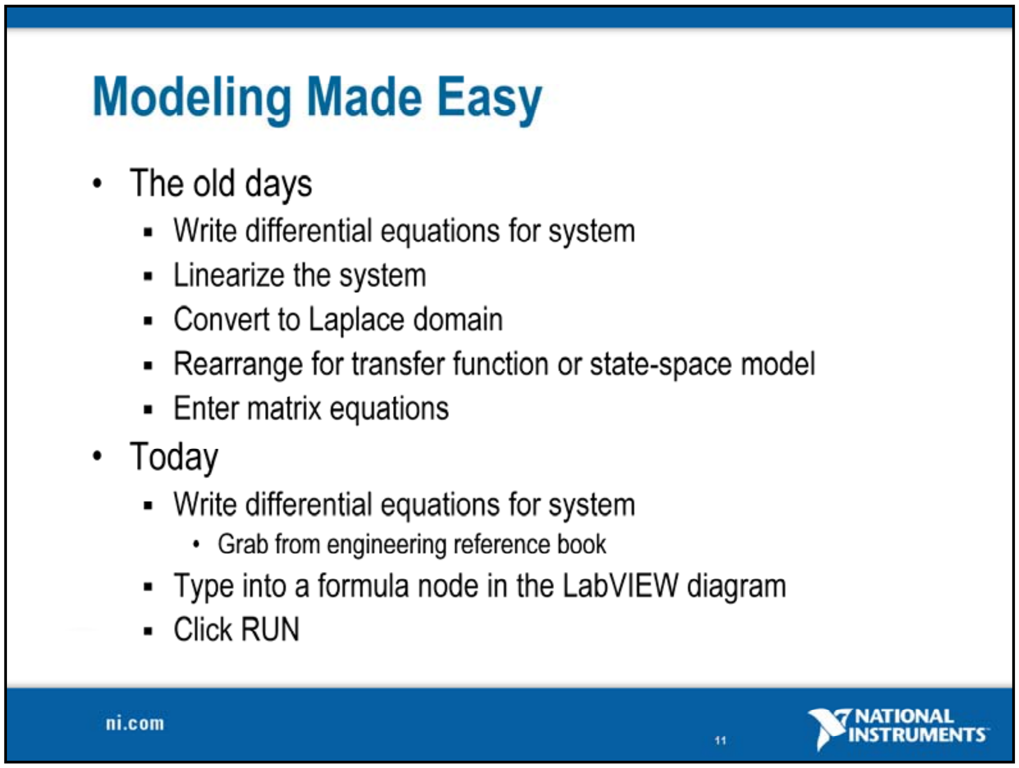


In the slide you can see where the PID is implemented and where the I/O is implemented. To simplify the code, . I've put all he 9505 programming under a VI.

If you now look at the "Simulated" version of the code, you'll realized the code is exactly the same, instead we had replaced the I/O with a VI that simulates the behaviour of our system (DC motor in our case)

As you can see, above we have the "real implementation" and you can check how the elements are very similar to the diagram shown before. I've hide all the 9505 programming under a VI.

In the diagram below is the same code using a simulated version of the motor.



Modeling Made Easy

- The old days
  - Write differential equations for system
  - Linearize the system
  - Convert to Laplace domain
  - Rearrange for transfer function or state-space model
  - Enter matrix equations
- Today
  - Write differential equations for system
    - Grab from engineering reference book
  - Type into a formula node in the LabVIEW diagram
  - Click RUN

ni.com 11 NATIONAL INSTRUMENTS

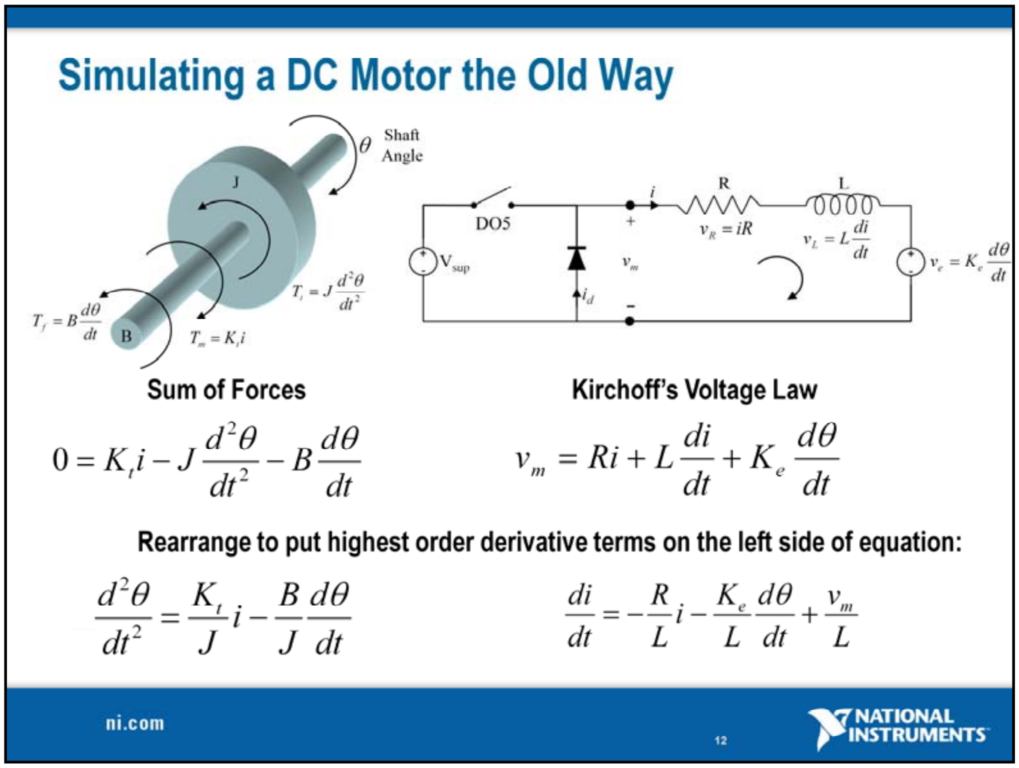
Now that we have talked about the benefits of simulation, let’s talk about the scary part. How to model

Hipotetically we are in a situation were we don’t have the hardware with us, and decide to move forward using simulation. How can we tackel the model of the plant or, in our case, the motor. There are basically two approaches.

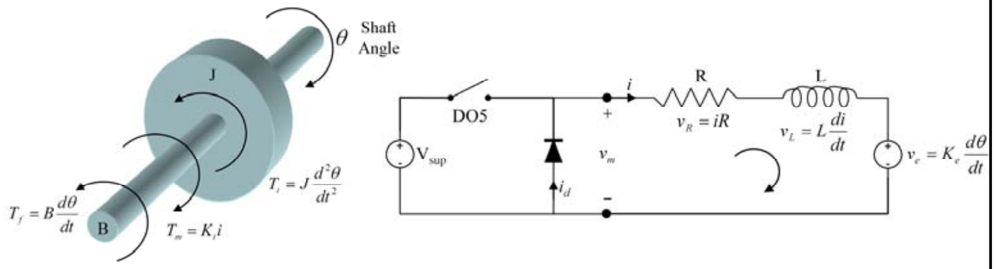
The first one is the one we learn in college: we identify the phisical phenomena that describes the dynamics of the system, find the differential equations, we then make some assumptions on the conditions of the system so we can linearized it, we convert to the Laplace domain and, after some algebra find the transfer function of the system.

OR, another approach, is that we can write the differential equations and write them directly using text-based math.

Let’s see how to model the DC motor with both approaches



Simulating a DC Motor the Old Way



Sum of Forces

$$0 = K_t i - J\frac{d^2\theta}{dt^2} - B\frac{d\theta}{dt}$$

Kirchoff's Voltage Law

$$v_m = Ri + L\frac{di}{dt} + K_e\frac{d\theta}{dt}$$

Rearrange to put highest order derivative terms on the left side of equation:

$$\frac{d^2\theta}{dt^2} = \frac{K_t}{J}i - \frac{B}{J}\frac{d\theta}{dt}$$

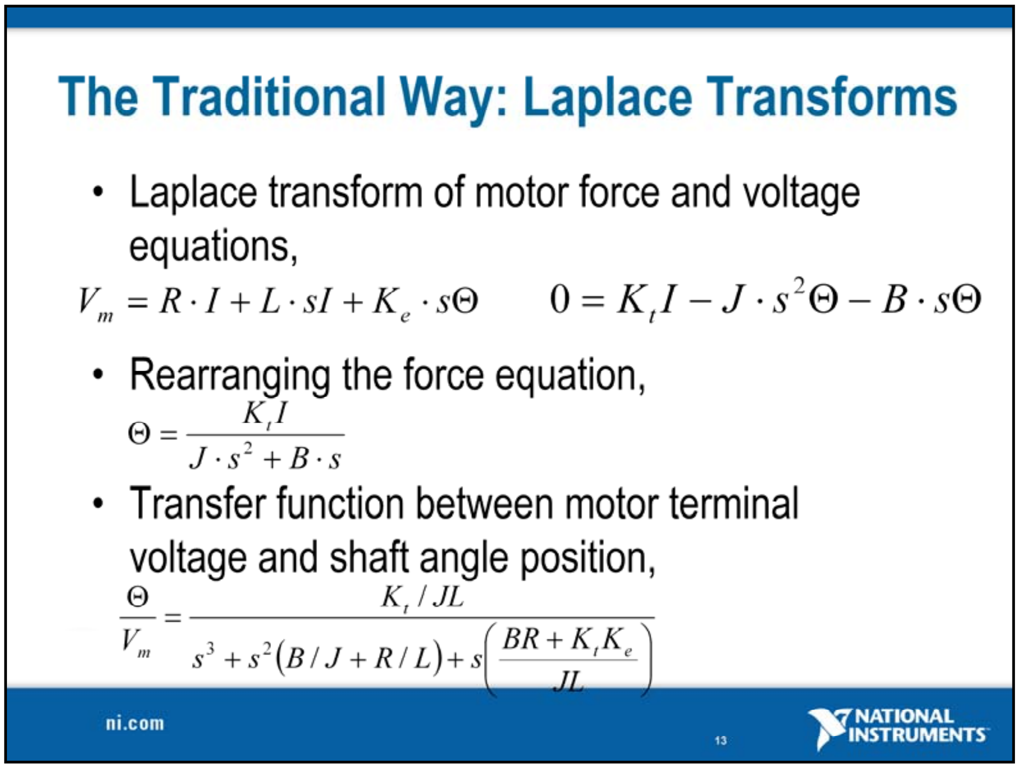
$$\frac{di}{dt} = -\frac{R}{L}i - \frac{K_e}{L}\frac{d\theta}{dt} + \frac{v_m}{L}$$

ni.com

12

NATIONAL INSTRUMENTS

Don’t worry about the math. The only thing to identify in this slide is the mechanical equation and the electrical equation.



The Traditional Way: Laplace Transforms

- Laplace transform of motor force and voltage equations,

$$V_m = R \cdot I + L \cdot sI + K_e \cdot s\Theta \qquad 0 = K_t I - J \cdot s^2\Theta - B \cdot s\Theta$$

- Rearranging the force equation,

$$\Theta = \frac{K_t I}{J \cdot s^2 + B \cdot s}$$

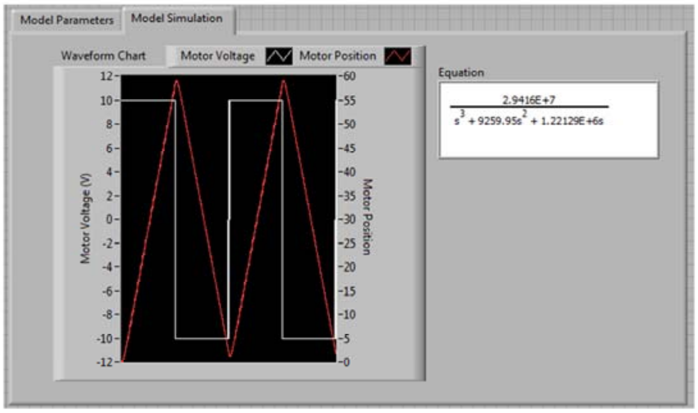
- Transfer function between motor terminal voltage and shaft angle position,

$$\frac{\Theta}{V_m} = \frac{K_t / JL}{s^3 + s^2\left(B/J + R/L\right) + s\left(\frac{BR + K_t K_e}{JL}\right)}$$

ni.com 13 NATIONAL INSTRUMENTS

f we follow the process we have described previously we will get to the final transfer function

Brushed DC Motor Simulation – Old

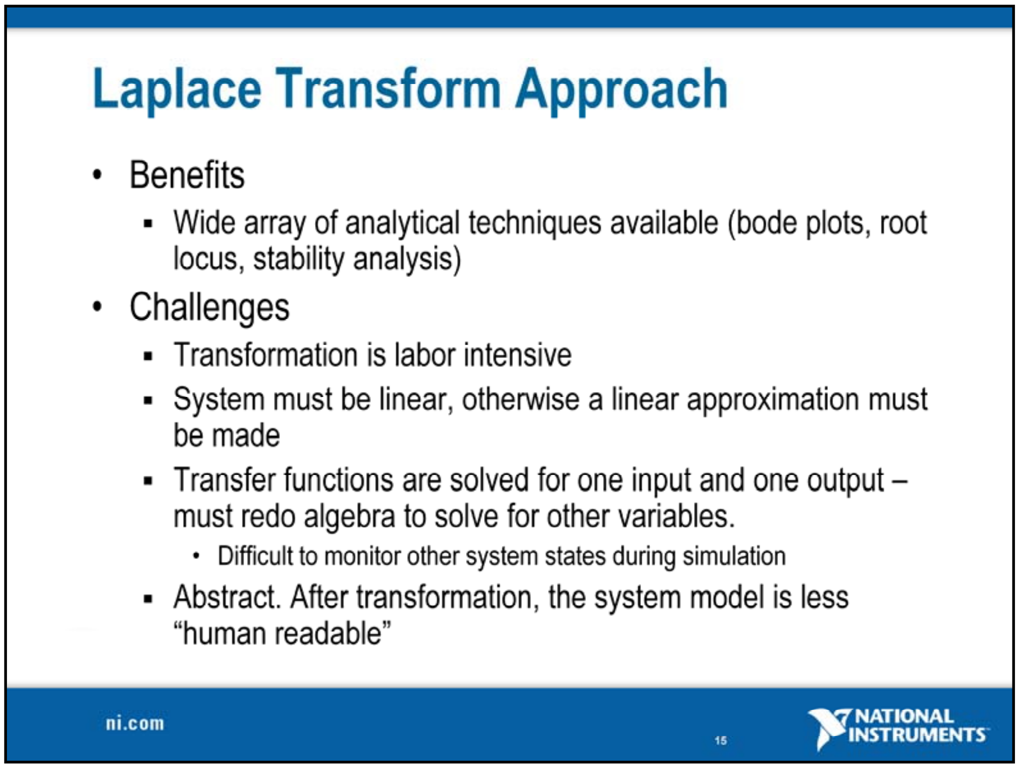


$$\frac{2.9416E+7}{s^3 + 9259.95s^2 + 1.22129E+6s}$$

ni.com

14

NATIONAL INSTRUMENTS



Laplace Transform Approach

- Benefits
  - Wide array of analytical techniques available (bode plots, root locus, stability analysis)
- Challenges
  - Transformation is labor intensive
  - System must be linear, otherwise a linear approximation must be made
  - Transfer functions are solved for one input and one output – must redo algebra to solve for other variables.
    - Difficult to monitor other system states during simulation
  - Abstract. After transformation, the system model is less "human readable"

ni.com 15 NATIONAL INSTRUMENTS

Benefits

Wide array of analytical techniques available (bode plots, root locus, stability analysis, …)

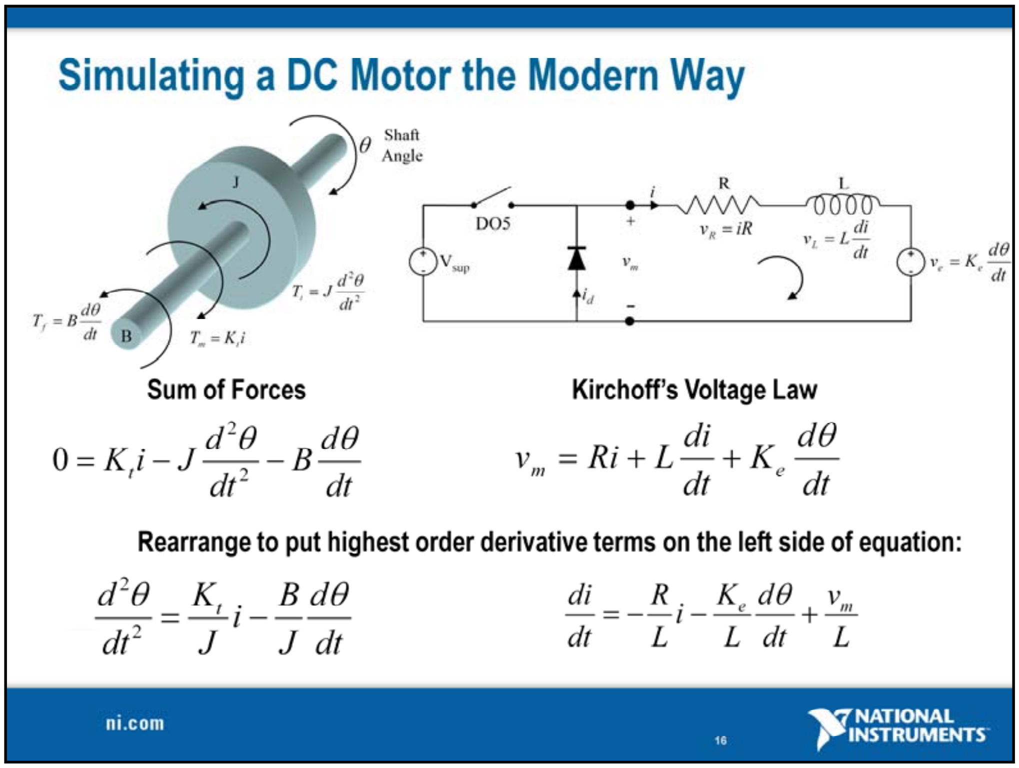
Challenges

Transformation is labor intensive

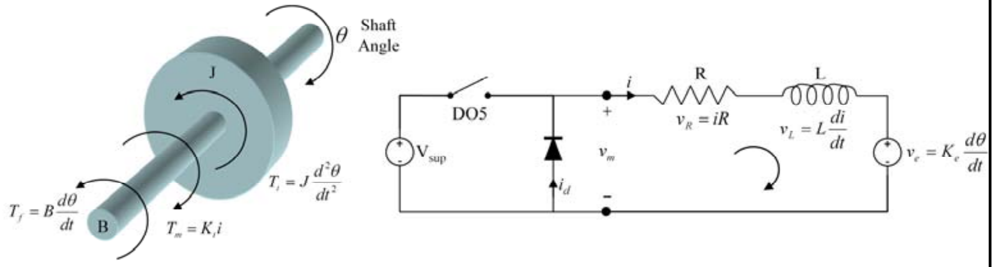
System must be linear otherwise a linear approximation must be made

Transfer functions are solved for one input and one output– must redo algebra to solve for other variables.

Difficult to monitor other system states during simulation



Simulating a DC Motor the Modern Way



Sum of Forces

$$0 = K_t i - J\frac{d^2\theta}{dt^2} - B\frac{d\theta}{dt}$$

Kirchoff's Voltage Law

$$v_m = Ri + L\frac{di}{dt} + K_e\frac{d\theta}{dt}$$

Rearrange to put highest order derivative terms on the left side of equation:

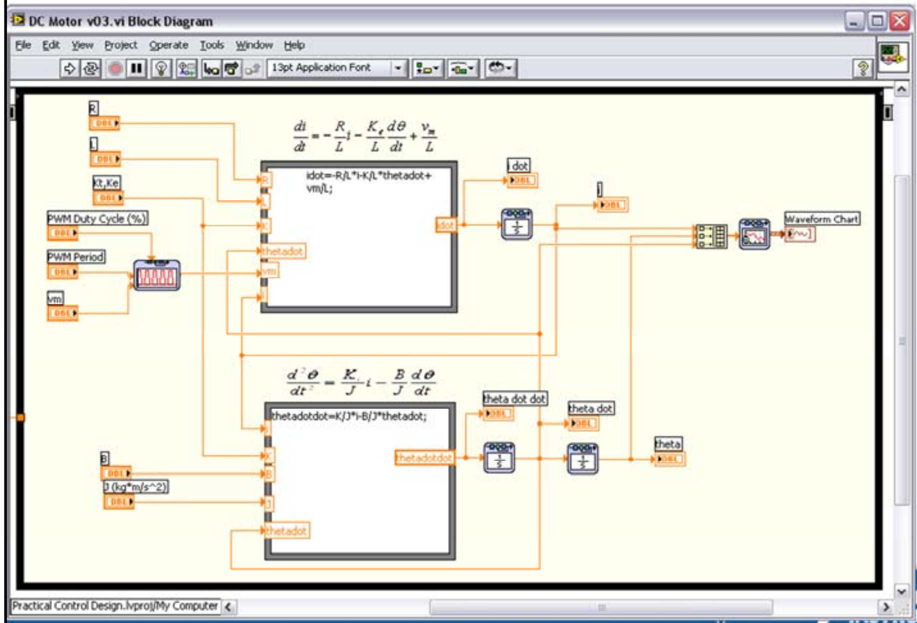
$$\frac{d^2\theta}{dt^2} = \frac{K_t}{J}i - \frac{B}{J}\frac{d\theta}{dt}$$

$$\frac{di}{dt} = -\frac{R}{L}i - \frac{K_e}{L}\frac{d\theta}{dt} + \frac{v_m}{L}$$

ni.com 16 NATIONAL INSTRUMENTS

Now, let’s try the other approach. We still are going to use the same equations, but this time, instead of following the process I’ve followed before I’m going to implement directly the differential equations in LabVIEW using a structure called “Formula Node”

Brushed DC Motor Simulation – Text


DC Motor v03.vi Block Diagram
File Edit View Project Operate Tools Window Help
13pt Application Font
R
L
Kt,Ke
PWM Duty Cycle (%)
PWM Period
vm
B
J (kg*m/s^2)
idot
i
Waveform Chart
theta dot dot
theta dot
theta
Practical Control Design.lvproj/My Computer

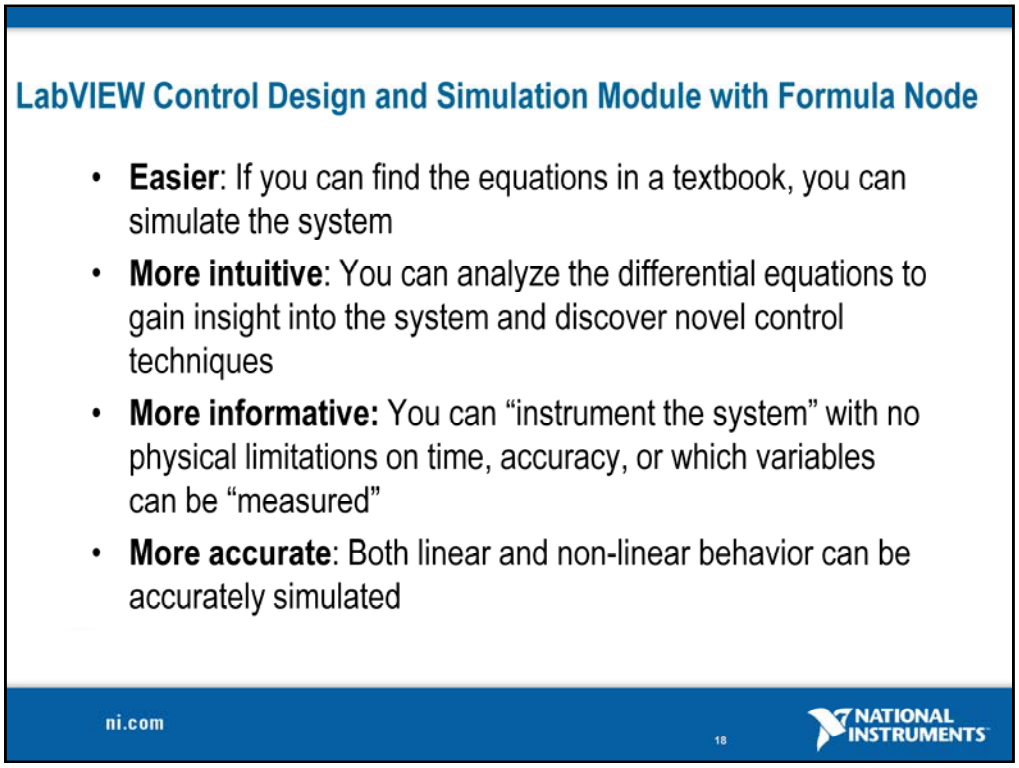
$$\frac{di}{dt} = -\frac{R}{L}i - \frac{K_e}{L}\frac{d\theta}{dt} + \frac{v_m}{L}$$

```
idot=-R/L*i-K/L*thetadot+
vm/L;
```

$$\frac{d^2\theta}{dt^2} = \frac{K_t}{J}i - \frac{B}{J}\frac{d\theta}{dt}$$

```
thetadotdot=K/J*i-B/J*thetadot;
```

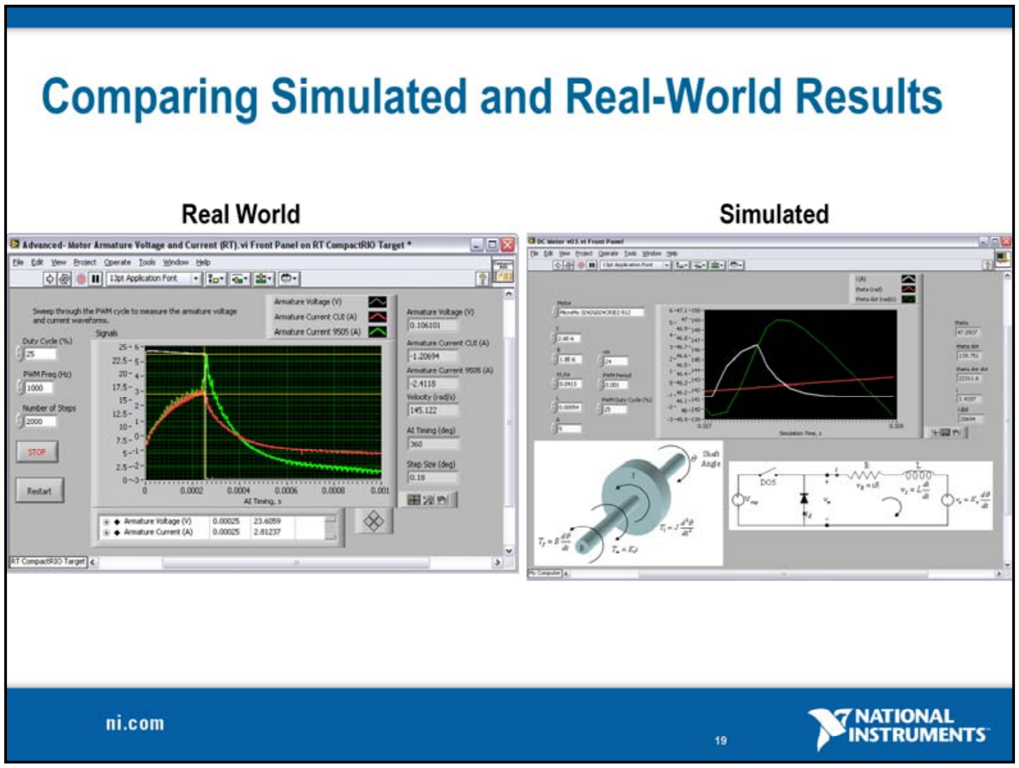
This block diagram represents the brushed DC Motor simulated plant. As you can easily see, we have used textual math (in the form of formula node) to implement the equations we found previouly. The only neccesary “extra” math are the integrator blocks to calculate integrated variables from their derivatives (position and speed from acceleration and current for derivative of current)



LabVIEW Control Design and Simulation Module with Formula Node

- **Easier**: If you can find the equations in a textbook, you can simulate the system
- **More intuitive**: You can analyze the differential equations to gain insight into the system and discover novel control techniques
- **More informative:** You can “instrument the system” with no physical limitations on time, accuracy, or which variables can be “measured”
- **More accurate**: Both linear and non-linear behavior can be accurately simulated

ni.com 18



Comparing Simulated and Real-World Results

Real World

Simulated

Advanced- Motor Armature Voltage and Current (RT).vi Front Panel on RT CompactRIO Target *

Sweep through the PWM cycle to measure the armature voltage and current waveforms.

Duty Cycle (%) 25

PWM Freq (Hz) 1000

Number of Steps 2000

Armature Voltage (V) 0.106101

Armature Current CUI (A) -1.20694

Armature Current 9505 (A) -2.4118

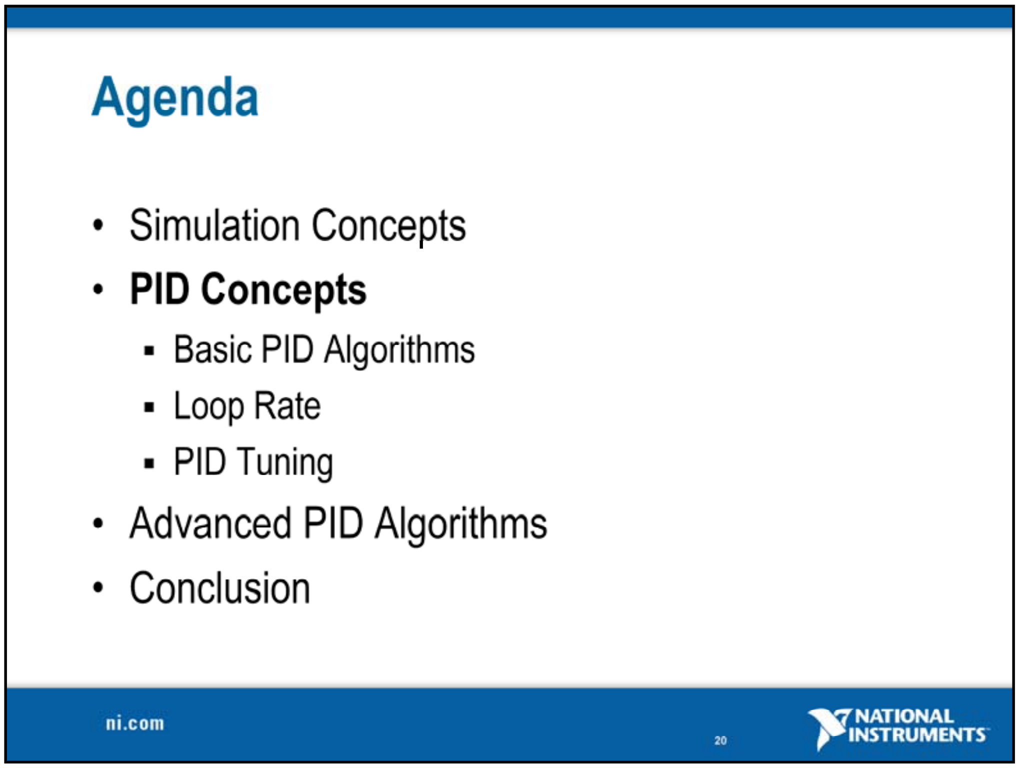
Velocity (rad/s) 145.122

AI Timing (deg) 360

Step Size (deg) 0.18

Armature Voltage (V)	0.00025	23.6059
Armature Current (A)	0.00025	2.81237

Simulation, albeit being a very useful tool, can’t totally replace the real world system. Slide you see the response to an step input both on the real Brushless DC motor and the corresponding software simulation. Notice that, although the behavior generally matches, they are not identical.

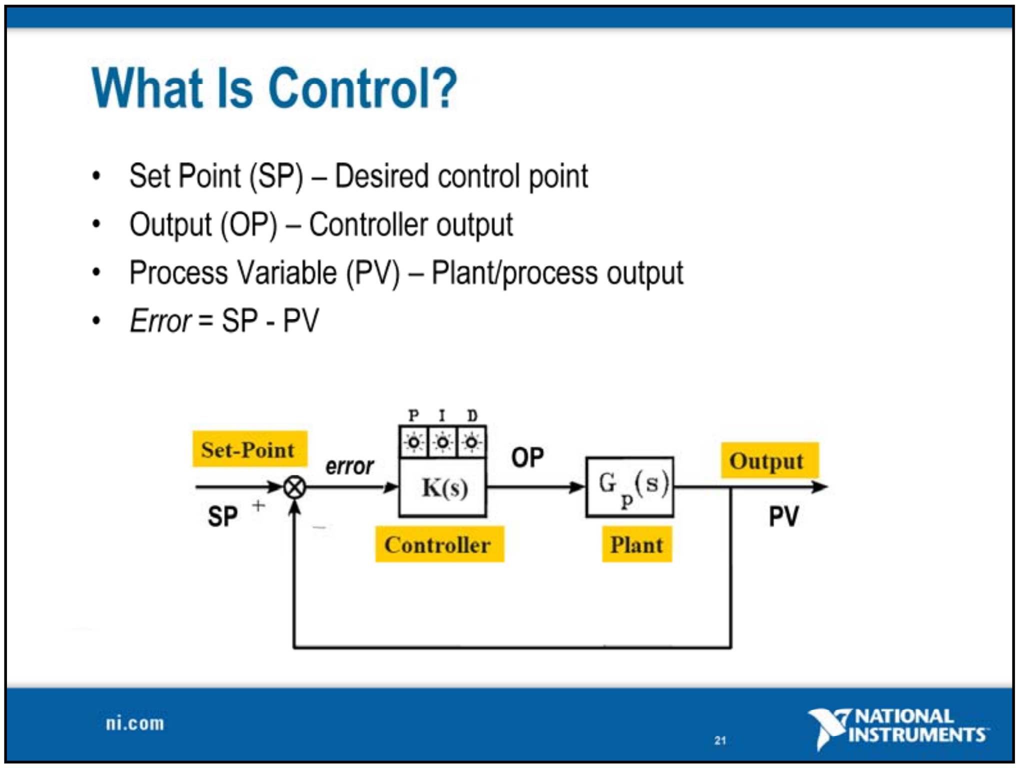


Agenda

- Simulation Concepts
- **PID Concepts**
  - Basic PID Algorithms
  - Loop Rate
  - PID Tuning
- Advanced PID Algorithms
- Conclusion

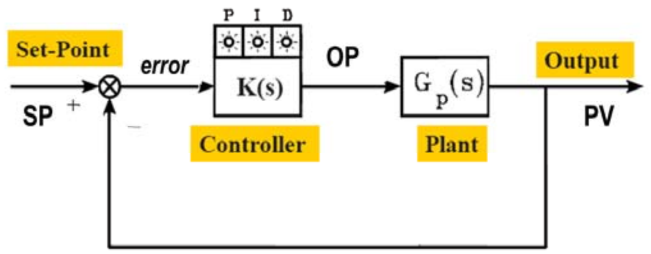
ni.com 20 NATIONAL INSTRUMENTS

Let's talk now about how to implement PID controllers. There is a lot of literature on how to tune PID controllers, but there isn't that much information on how fast the control loop should be. This is what we are going to tacking if the first section. Then we will cover how what different PID algorithms are present in the LabVIEW software platform and when to use them and, of course, we will finish with a section on tuning



What Is Control?

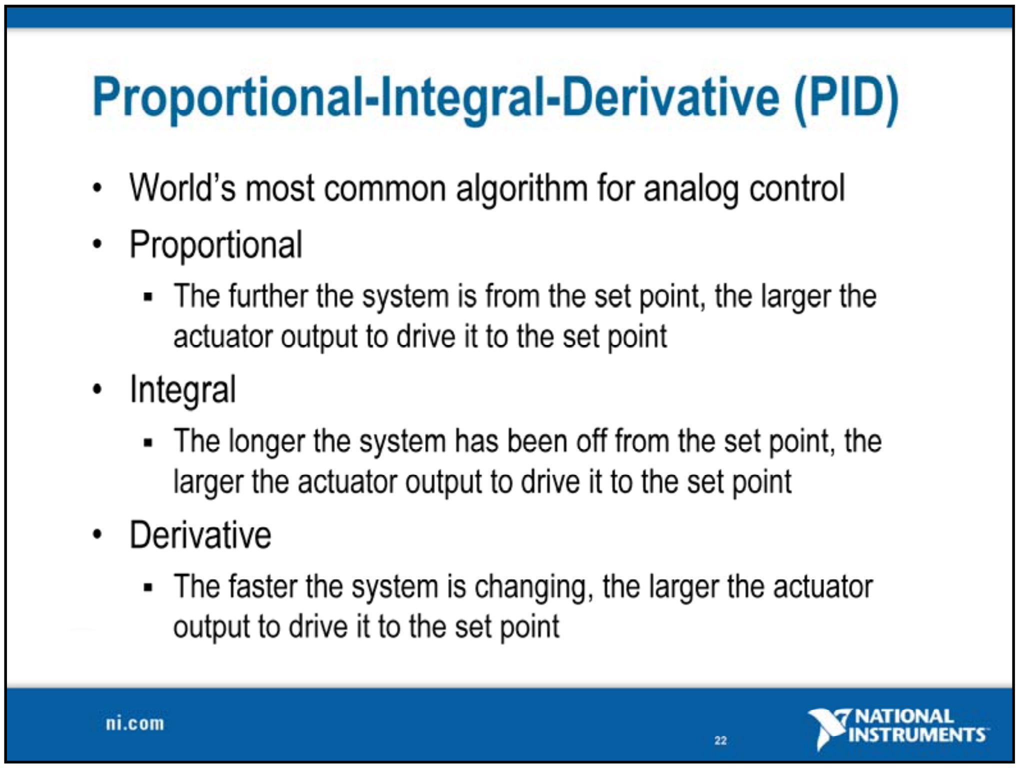
- Set Point (SP) – Desired control point
- Output (OP) – Controller output
- Process Variable (PV) – Plant/process output
- *Error* = SP - PV



ni.com 21 NATIONAL INSTRUMENTS

After talking about the loop rate, let’s move to the next sub-section

PID control is the cornerstone of control algorithms and is used in a wide range of industries for control of many different types of process, machines, and systems. In a typical control system there is a setpoint which is the desired control output, then there is the controller output which is the response of the PID control algorithm, the process variable is the result on the physical system, an this is then compared to the setpoint and the error is used as an input to the controller.



Proportional-Integral-Derivative (PID)

- World's most common algorithm for analog control
- Proportional
  - The further the system is from the set point, the larger the actuator output to drive it to the set point
- Integral
  - The longer the system has been off from the set point, the larger the actuator output to drive it to the set point
- Derivative
  - The faster the system is changing, the larger the actuator output to drive it to the set point

ni.com 22 NATIONAL INSTRUMENTS

World's most common algorithm for analog control

Proportional

The further the system is from the set point, the larger the actuator output to drive it to the set point

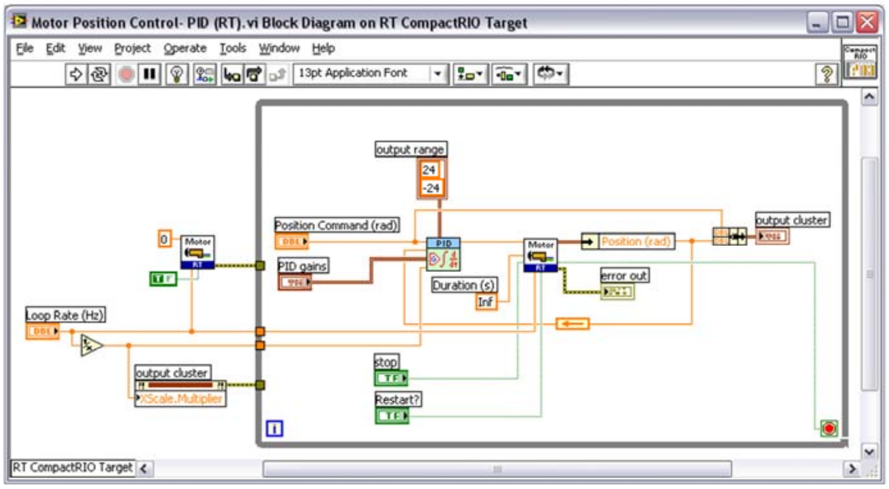
Integral

The longer the system has been off from the set point, the larger the actuator output to drive it to the set point

Derivative

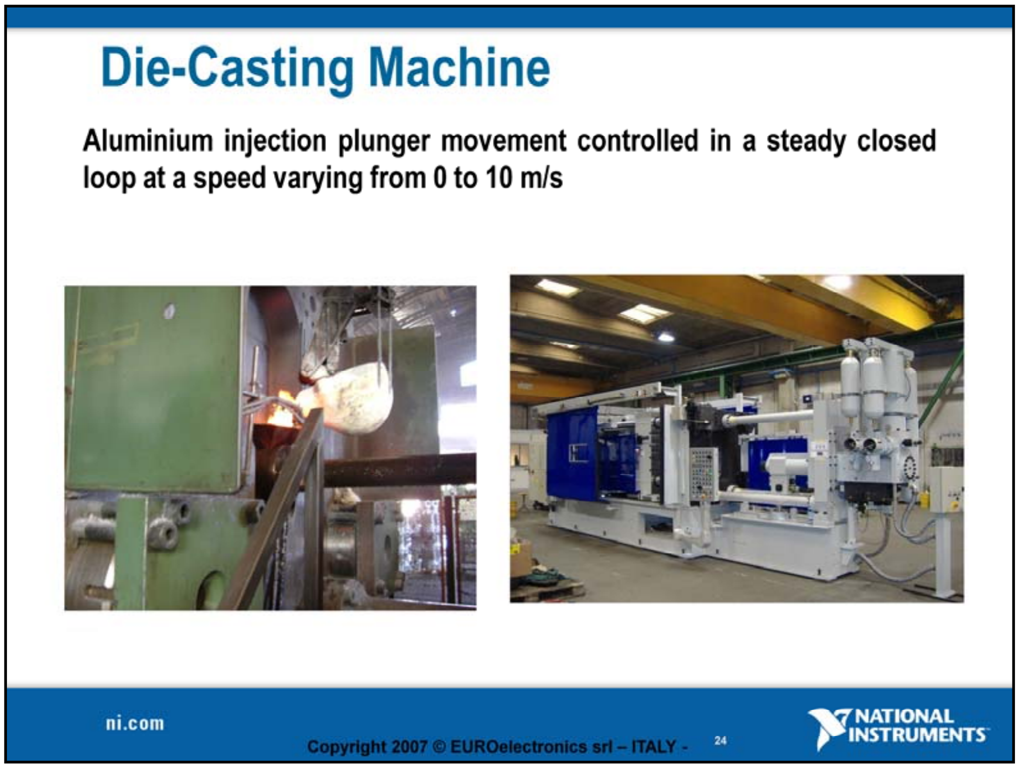
The faster the system is changing, the larger the actuator output to drive it to the set point

Demo: LabVIEW PID Control



ni.com

23



Die-Casting Machine

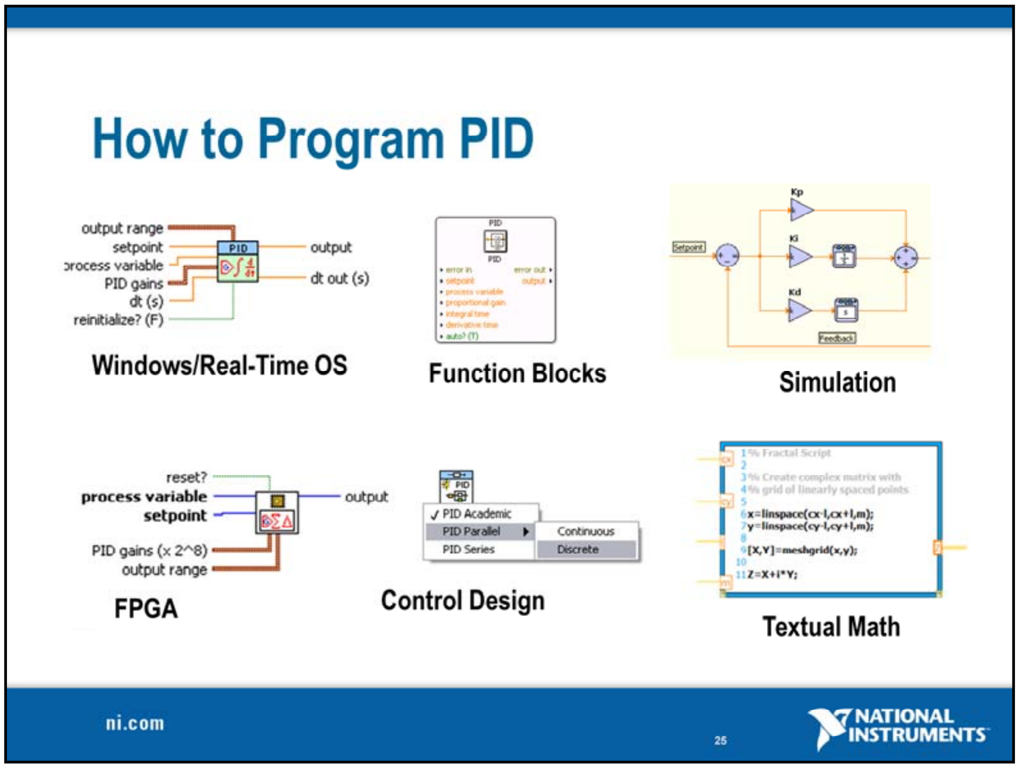
Aluminium injection plunger movement controlled in a steady closed loop at a speed varying from 0 to 10 m/s

ni.com

Copyright 2007 © EUROelectronics srl – ITALY - 24

NATIONAL INSTRUMENTS

EUROelectronics is a machine builder that was asked to design a closed-loop hydraulic cylinder control system for a die-casting press machine. The high-speed press moves anywhere from 0 to 10 m/s and therefore requires a high-speed control system. To meet this challenge, we relied on the NI LabVIEW FPGA Module and CompactRIO hardware. With the integrated FPGA on the CompactRIO controller, we could develop a system capable of low-level customization using commercially available tools. To meet the unique requirements of the application, we implemented a highly optimized encoder interface in the FPGA to measure the cylinder position while programming the system entirely in LabVIEW.



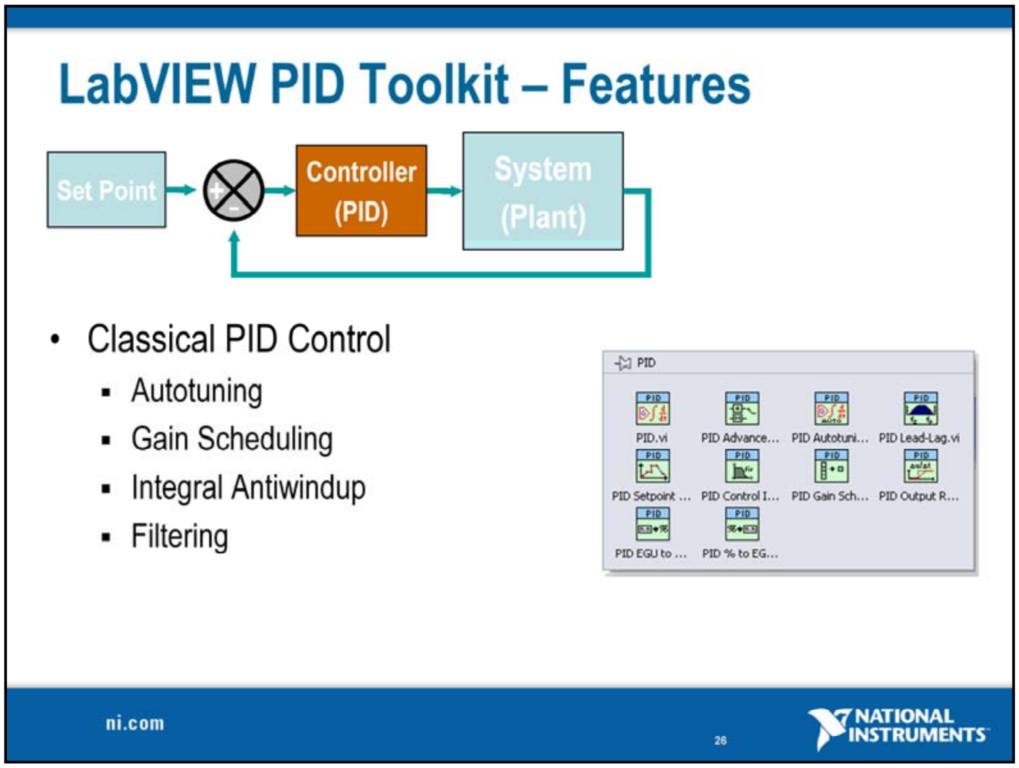
How to Program PID

```
% Fractal Script

% Create complex matrix with
% grid of linearly spaced points

x=linspace(cx-l,cx+l,m);
y=linspace(cy-l,cy+l,m);

[X,Y]=meshgrid(x,y);

Z=X+i*Y;
```

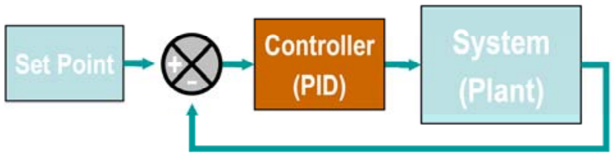
As LabVIEW offers a number of models of computation, it also offer different approaches to implement PID algorithms. This options include

-industrial grade PID algorithms

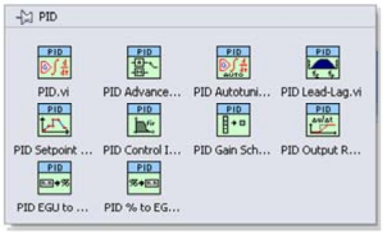
- PID with 6011 format
- ODE based format for optimal design
- Hardware specif PID like the one for FPGA
- both graphical and textual options



LabVIEW PID Toolkit – Features



- Classical PID Control
  - Autotuning
  - Gain Scheduling
  - Integral Antiwindup
  - Filtering



ni.com 26 NATIONAL INSTRUMENTS

On the screen we can see a typical close loop control implementation. In this case the controller shown in the orange box could be the PID we are design. This PID could include a number of features to improve its performance such as

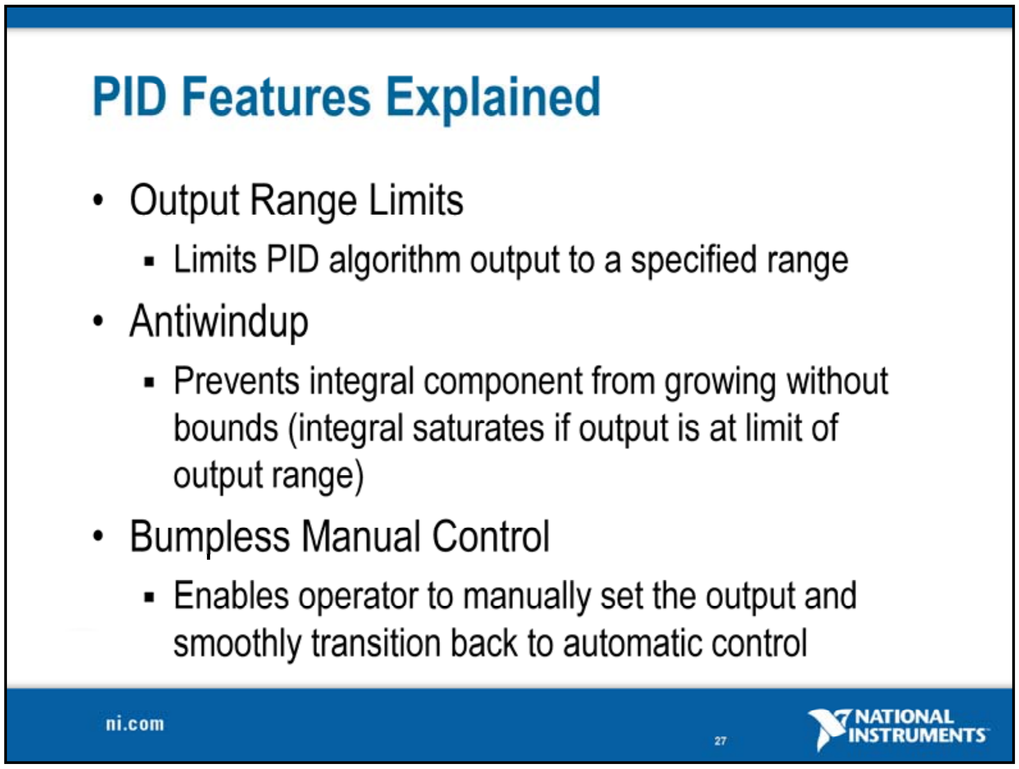
- Antiwind up: where the integral component stops integrating if saturation is reached

- Limit saturation: to protects the outputs

- Gain Scheduling; allow different sets of PID gains that will be applied based on the Process variable

- Bumpless gain transfer: to allow smooth transition when changing gains either because we are using Gain Scheduling or are in the process of manually tuning

- and more

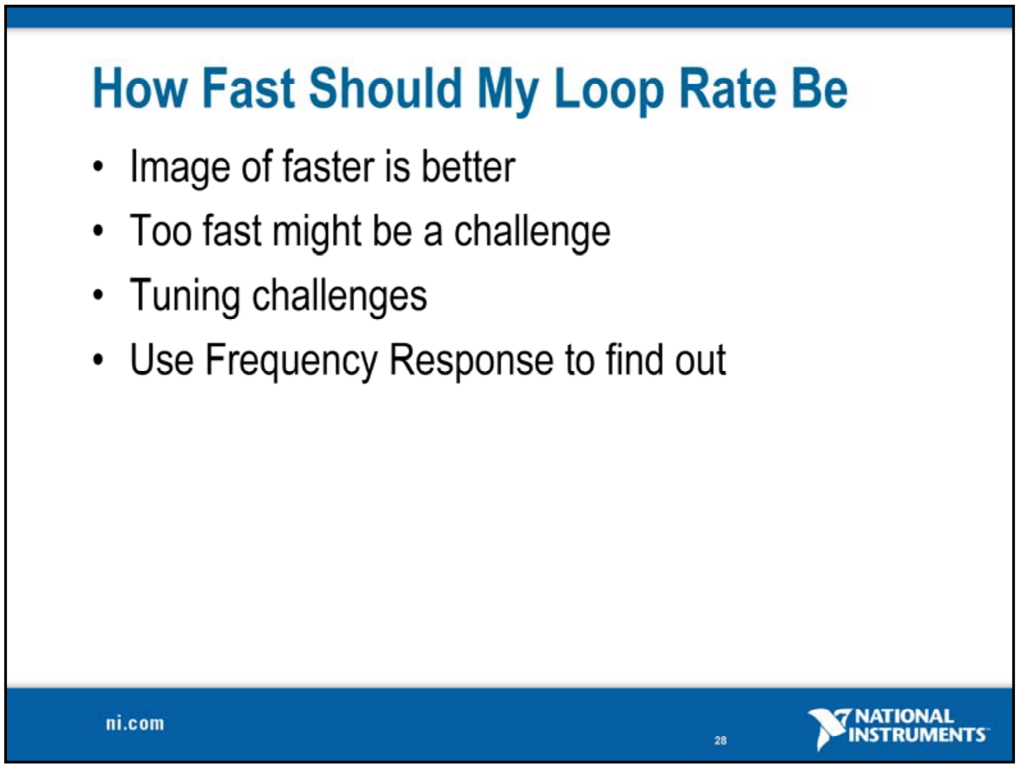


PID Features Explained

- Output Range Limits
  - Limits PID algorithm output to a specified range
- Antiwindup
  - Prevents integral component from growing without bounds (integral saturates if output is at limit of output range)
- Bumpless Manual Control
  - Enables operator to manually set the output and smoothly transition back to automatic control

ni.com

27



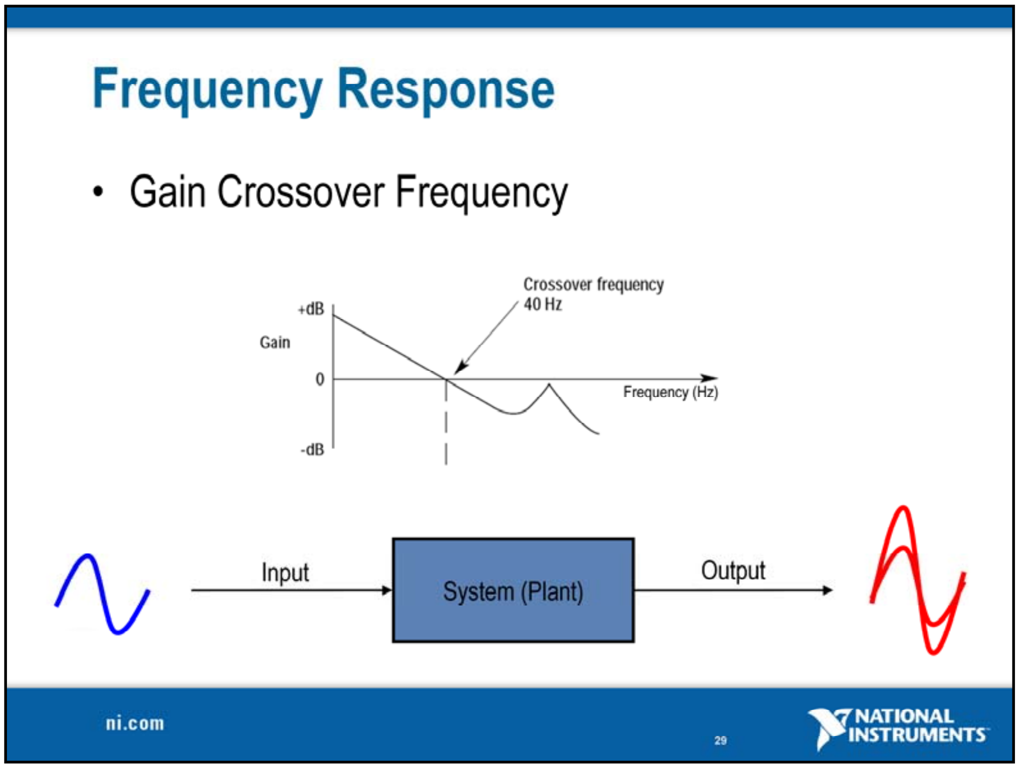
How Fast Should My Loop Rate Be

- Image of faster is better
- Too fast might be a challenge
- Tuning challenges
- Use Frequency Response to find out

ni.com 28 NATIONAL INSTRUMENTS

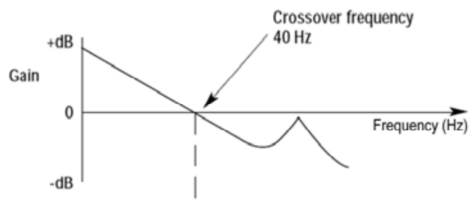
When deciding which loop rate, there is the general idea that faster loop rates are better. While this is often the truth, sometimes high loop rates might uncover undesirable efect on the systems to be controlled. Also, higher loop rates might require extra programming in the form of microprocessor or FPGA and might unnecessary increment the complexity of the application. Last but not least, faster control loops are typically more difficult to tune that slow ones.

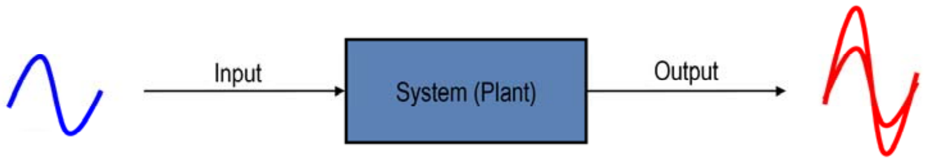
We are going to cover how to estimate the necessary loop rate of the system by using frequency response.



Frequency Response

- Gain Crossover Frequency



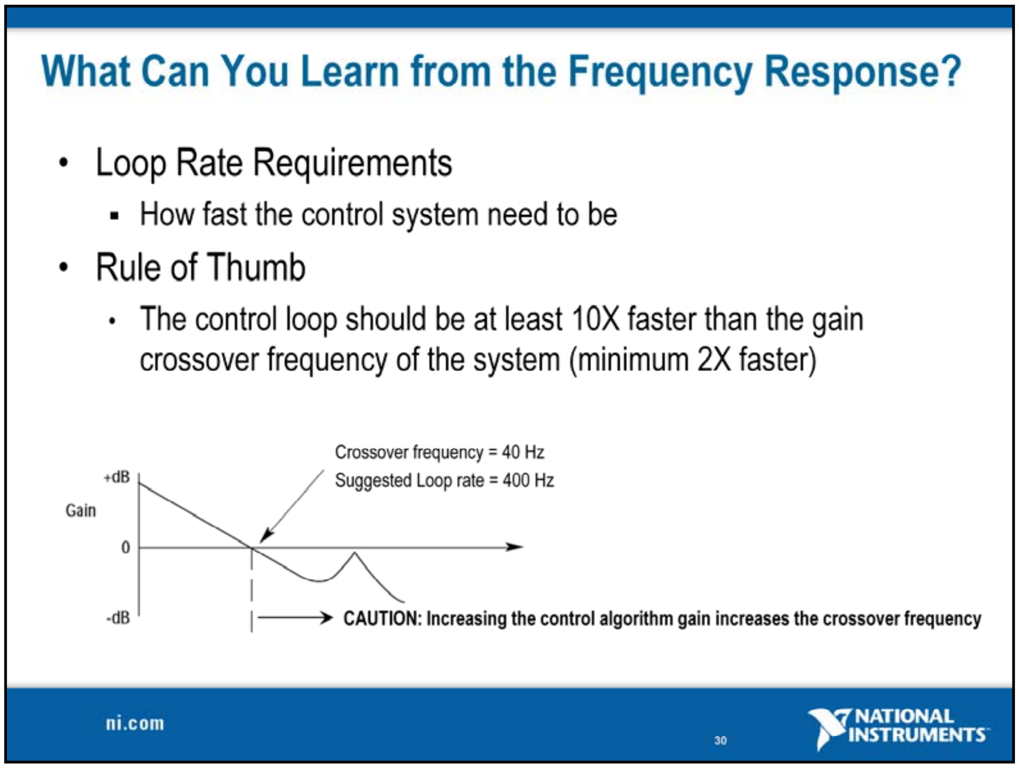


ni.com

29

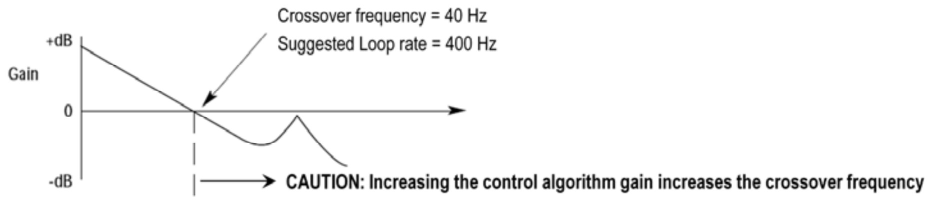
Gain Crossover Frequency is The frequency at which the amplitude response of the system has a gain of 1 (0 dB)

Plot shows the frequency response for a DC motor. “The first thing we notice is the pronounced spike in
the gain plot at a frequency of around 2kHz. This is caused by shaft resonance, torsional oscillation in the shaft between the motor and the tach. Observe that the phase plot drive dramatically through the critical 180° line at this point. This means that the loop gain at this frequency must be less than unity (0dB), otherwise the system will oscillate.”



What Can You Learn from the Frequency Response?

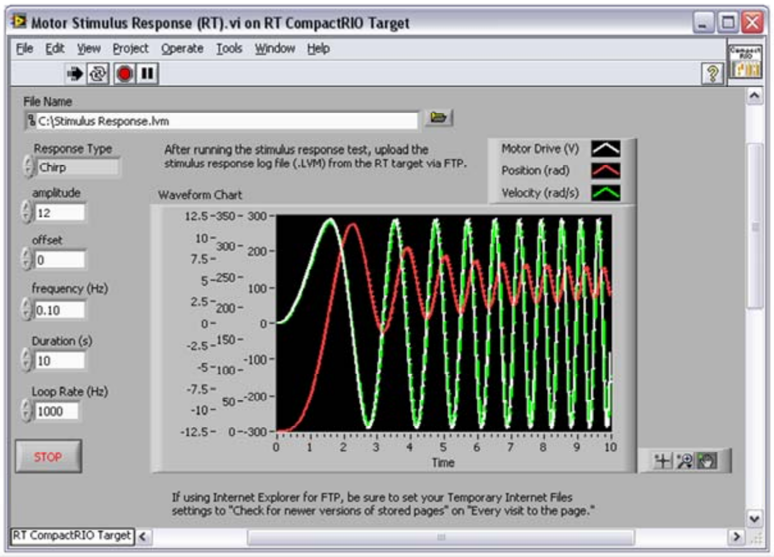
- Loop Rate Requirements
  - How fast the control system need to be
- Rule of Thumb
  - The control loop should be at least 10X faster than the gain crossover frequency of the system (minimum 2X faster)



ni.com

30

Demo: Frequency Response Demo


Motor Stimulus Response (RT).vi on RT CompactRIO Target
File Edit View Project Operate Tools Window Help
Waveform Chart
Motor Drive (V)
Position (rad)
Velocity (rad/s)
Time
RT CompactRIO Target

File Name: C:\Stimulus Response.lvm

Response Type: Chirp

amplitude: 12

offset: 0

frequency (Hz): 0.10

Duration (s): 10

Loop Rate (Hz): 1000

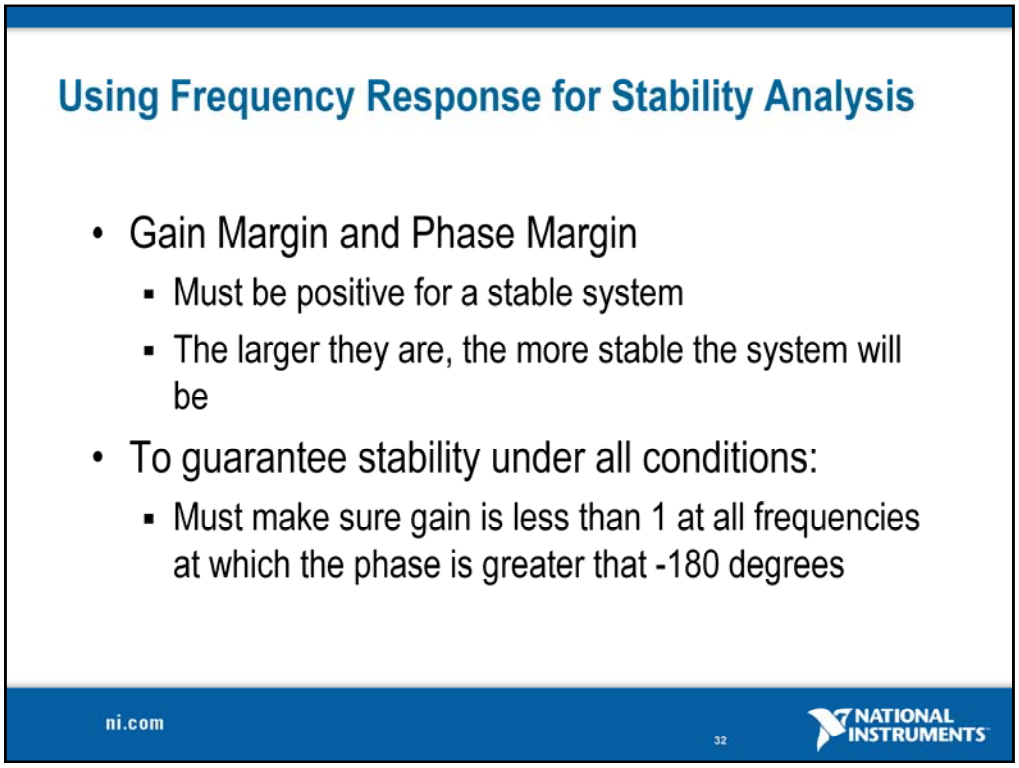
STOP

After running the stimulus response test, upload the stimulus response log file (.LVM) from the RT target via FTP.

If using Internet Explorer for FTP, be sure to set your Temporary Internet Files settings to "Check for newer versions of stored pages" on "Every visit to the page."

ni.com

31



Using Frequency Response for Stability Analysis

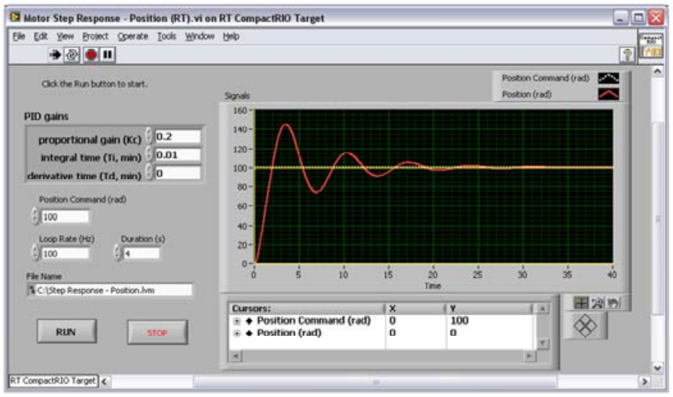
- Gain Margin and Phase Margin
  - Must be positive for a stable system
  - The larger they are, the more stable the system will be
- To guarantee stability under all conditions:
  - Must make sure gain is less than 1 at all frequencies at which the phase is greater that -180 degrees

ni.com 32 NATIONAL INSTRUMENTS

Once found the frequency of the system, it can be used not only to look for the loop rate, but also the stabilty of the system

PID Tuning and Loop Rates

- Increasing the P gain makes your system faster.
  - Result: You may need a faster control loop rate.


Motor Step Response - Position (RT).vi on RT CompactRIO Target
File Edit View Project Operate Tools Window Help
Signals
Position Command (rad)
Position (rad)
Time
RUN
STOP
RT CompactRIO Target

Click the Run button to start.

PID gains
proportional gain (Kc) 0.2
integral time (Ti, min) 0.01
derivative time (Td, min) 0

Position Command (rad) 100

Loop Rate (Hz) 100

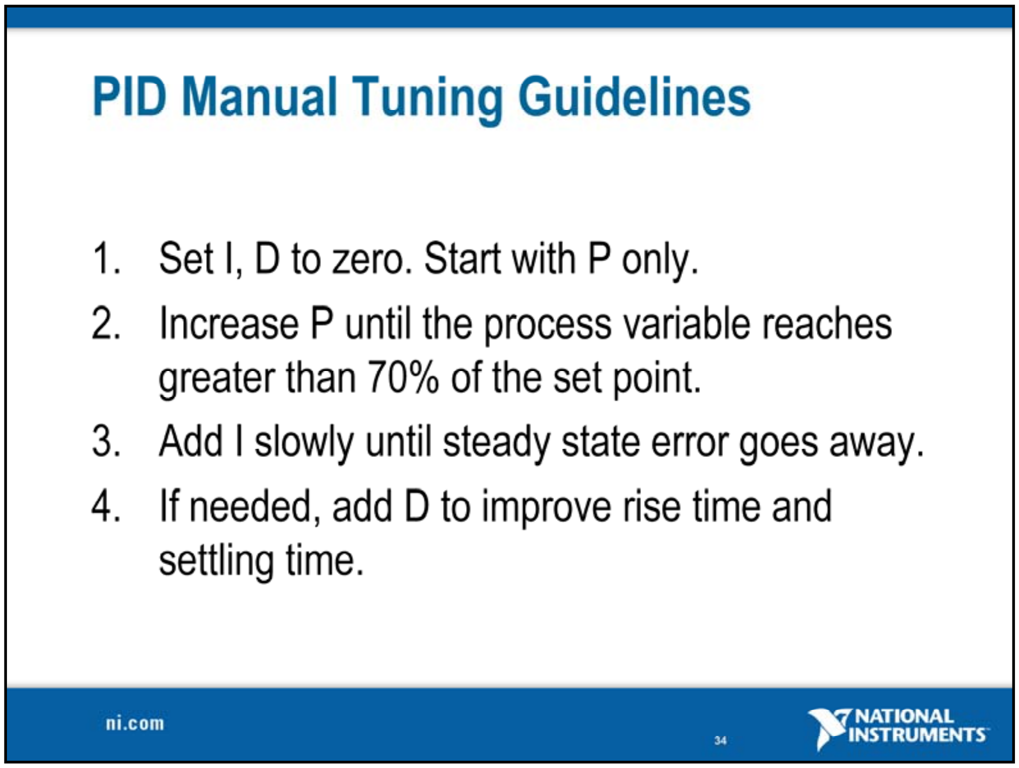
Duration (s) 4

File Name C:\Step Response - Position.lvm

Cursors:	X	Y
Position Command (rad)	0	100
Position (rad)	0	0

ni.com 33 NATIONAL INSTRUMENTS

Tuning has an effect in the loop rate. So even when we have mentioned before how to calculate the loop rate, it might be needed to increase that number after tuning.

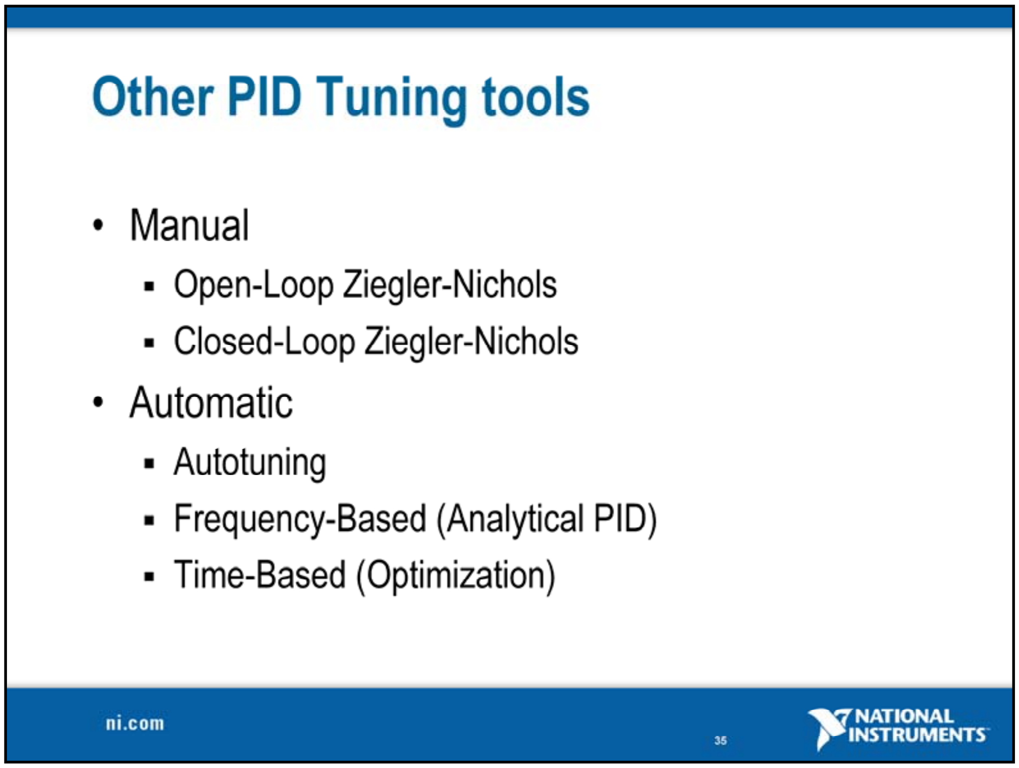


PID Manual Tuning Guidelines

1. Set I, D to zero. Start with P only.
2. Increase P until the process variable reaches greater than 70% of the set point.
3. Add I slowly until steady state error goes away.
4. If needed, add D to improve rise time and settling time.

ni.com 34 NATIONAL INSTRUMENTS

Now that we have explain what is PID and how to implement it, let’s talk about how to tune it. What you have in the screen is a set of easy rules. Beware that, although this rules work often, we can not ensure the optimal performance nor stability on the closed loop system



Other PID Tuning tools

- Manual
  - Open-Loop Ziegler-Nichols
  - Closed-Loop Ziegler-Nichols
- Automatic
  - Autotuning
  - Frequency-Based (Analytical PID)
  - Time-Based (Optimization)

ni.com 35 NATIONAL INSTRUMENTS

There is a lot of literature about how to tune PID controller in different application areas like motion, process control etc. We can classify these techniques in two areas

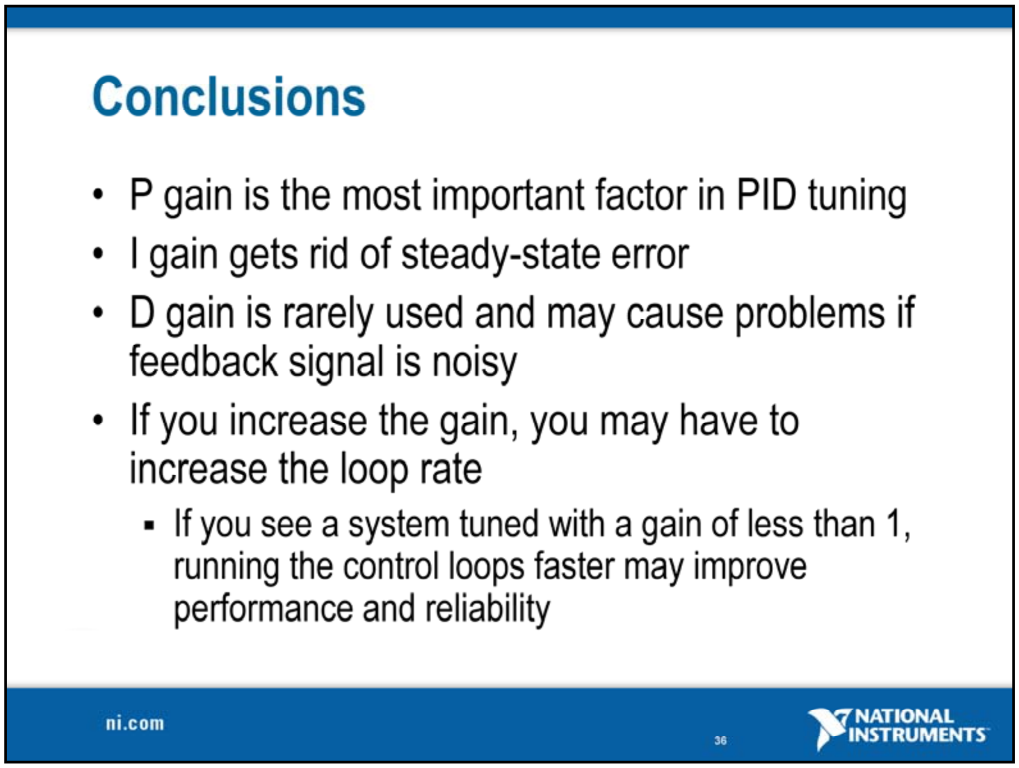
Manual: Where the operator has to control the plant and, based in inputs and outputs will tune the PID algorithm. The most common techniques or set of rules are Ziegler-Nichols. In open loop, we measure the step response of the system and based on it we can tune our controller. In the close loop case, we will zero both the integral and derivative gain and will increase the P gain until the response of the system oscillates. Based on this oscillation we can find a set of PID gains

Automatic. These are software tools that will help the control engineer to find the right set of gains without the need of manual operations or to make the system unstable. Some of these tools are

Autotuning: Included with the PID toolkit: This tool automatically excite the control system and find a suitable set of PID gains

Frequency based: Included with the control design andn simulation module there are examples on how to use the optimizer to find PID gains based on frequency response of the system. This technique, called "Analytical PID" finds gains that makes the sytem stable, but can't ensure optimal performance.

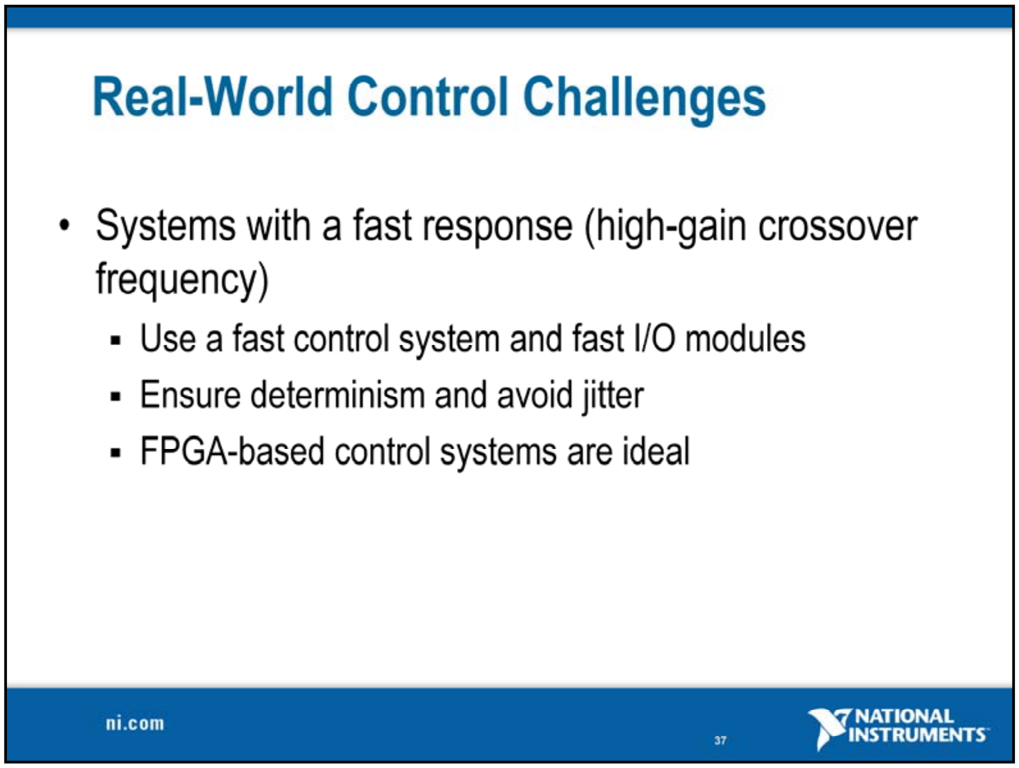
Time Based: where use of the Optimal Design functionality to determine the parameters of a PID controller. The continuous-time system, G1, is represented by a second order transfer function. The controller parameters, P,I,D and optimized within a bounded range of parameters and subject to time domain constraints on outputs and control actions.



Conclusions

- P gain is the most important factor in PID tuning
- I gain gets rid of steady-state error
- D gain is rarely used and may cause problems if feedback signal is noisy
- If you increase the gain, you may have to increase the loop rate
  - If you see a system tuned with a gain of less than 1, running the control loops faster may improve performance and reliability

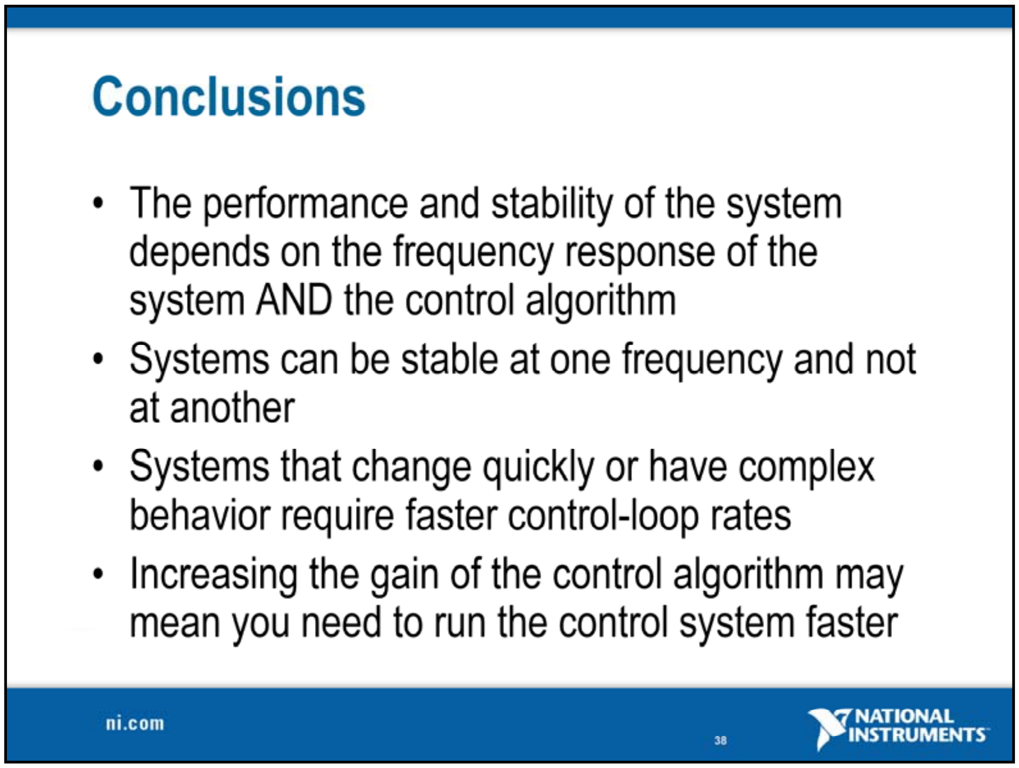
ni.com 36 NATIONAL INSTRUMENTS



Real-World Control Challenges

- Systems with a fast response (high-gain crossover frequency)
  - Use a fast control system and fast I/O modules
  - Ensure determinism and avoid jitter
  - FPGA-based control systems are ideal

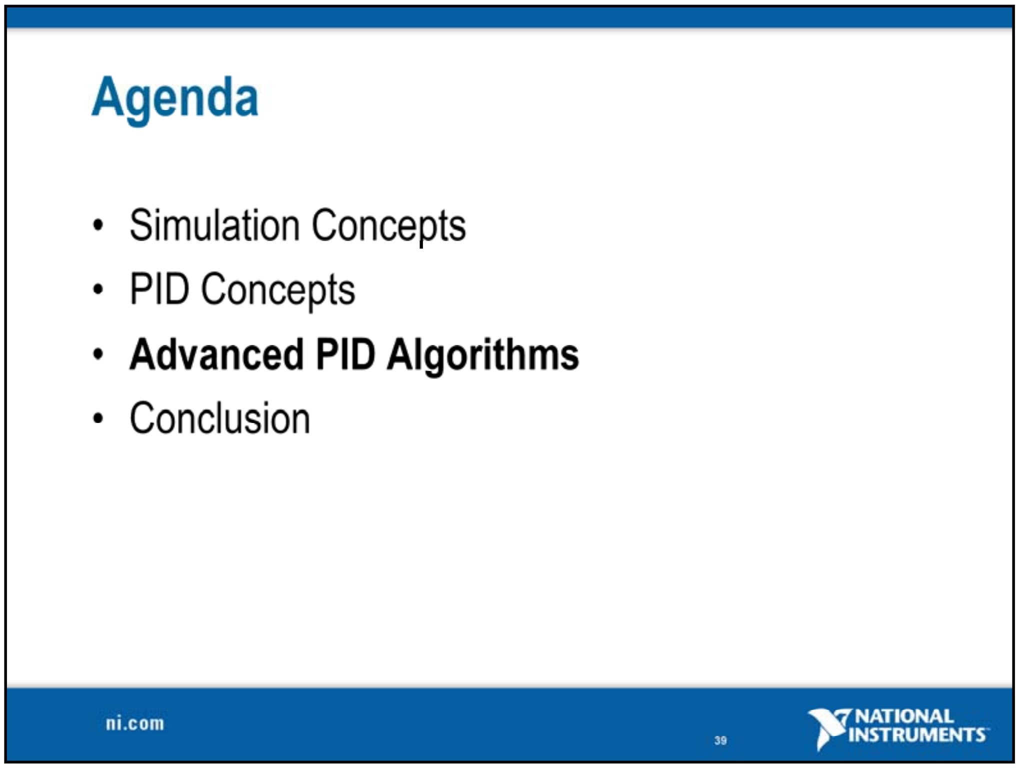
ni.com 37 NATIONAL INSTRUMENTS



Conclusions

- The performance and stability of the system depends on the frequency response of the system AND the control algorithm
- Systems can be stable at one frequency and not at another
- Systems that change quickly or have complex behavior require faster control-loop rates
- Increasing the gain of the control algorithm may mean you need to run the control system faster

ni.com 38 NATIONAL INSTRUMENTS



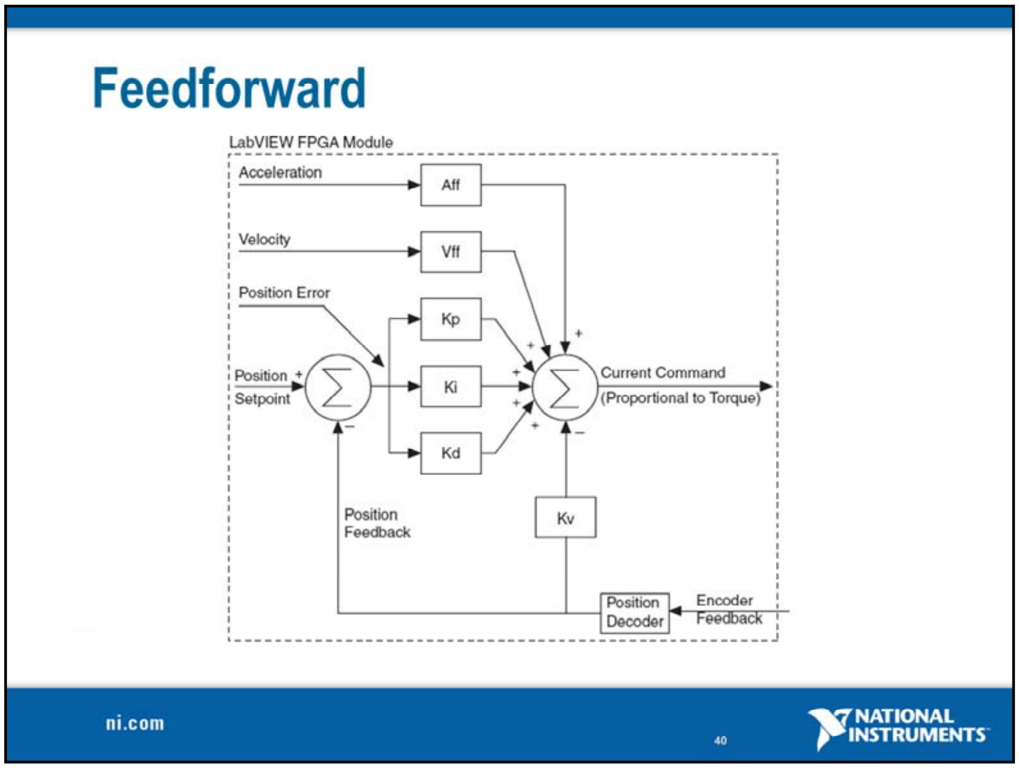
Agenda

- Simulation Concepts
- PID Concepts
- **Advanced PID Algorithms**
- Conclusion

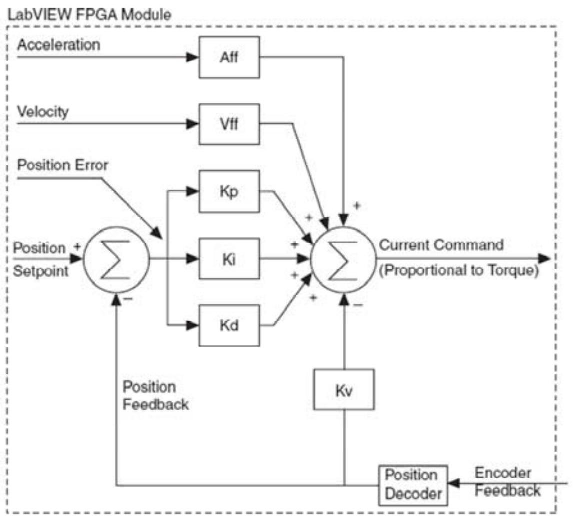
ni.com

39

What we are going to cover now is some techniques to improve the performance of standard PID controllers.



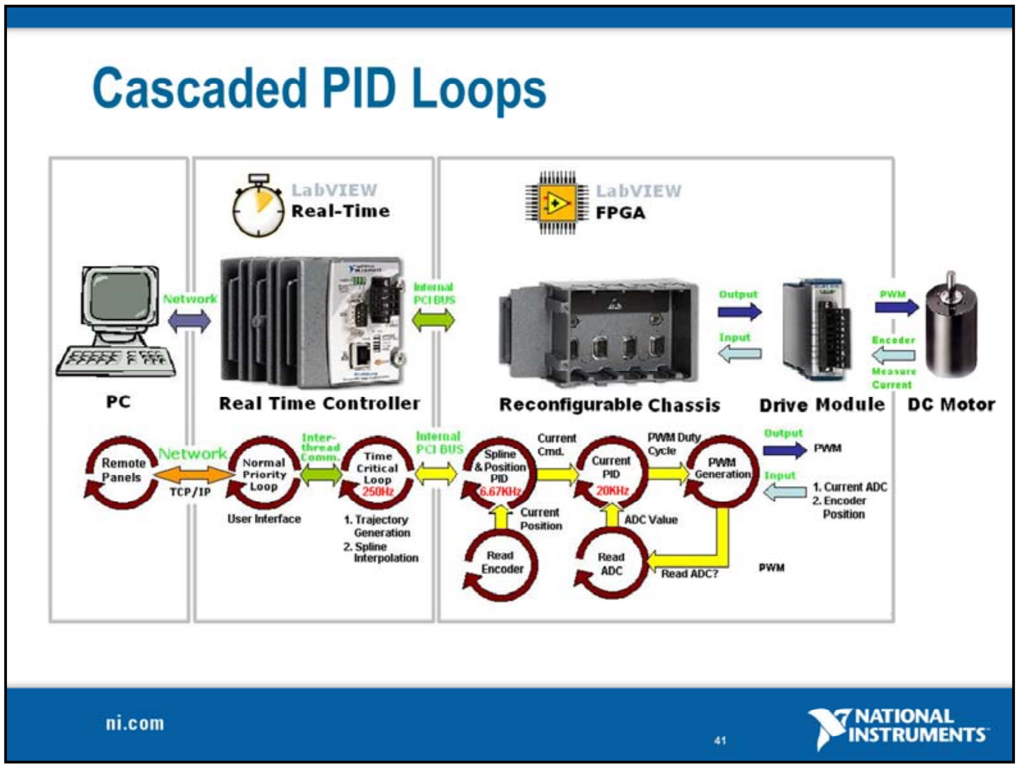
Feedforward



ni.com 40 NATIONAL INSTRUMENTS

Feed Forward consist on adding other values to the controller based on external information. Typical examples of Feed Forward are present in motion control of mechanical systems. In these cases the output of the controller is not only affected by the PID gains, but also by other magnitudes (desired speed and acceleration) which also will be scaled.

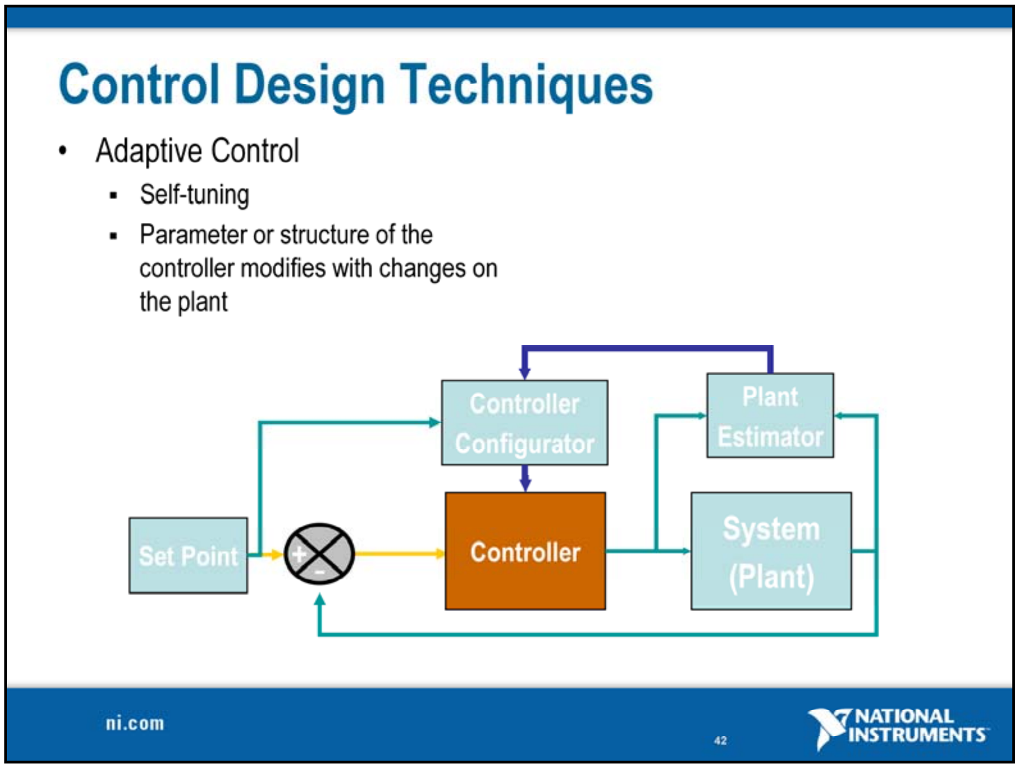
What you see on the screen is one example present if the FPGA fabric of our motion controllers



Cascaded PID Loops

ni.com 41

The aforementioned techniques are not mutually exclusive. You can combine current PID control with high-level optimal controllers like MPC or linear-quadratic regulator (LQR). These controllers, called hierarchical controllers, benefit from the use of high-level algorithms when working with nonlinearities and multiple inputs and outputs. This offers a set of smaller, easier-to-handle control problems that PID algorithms can address.



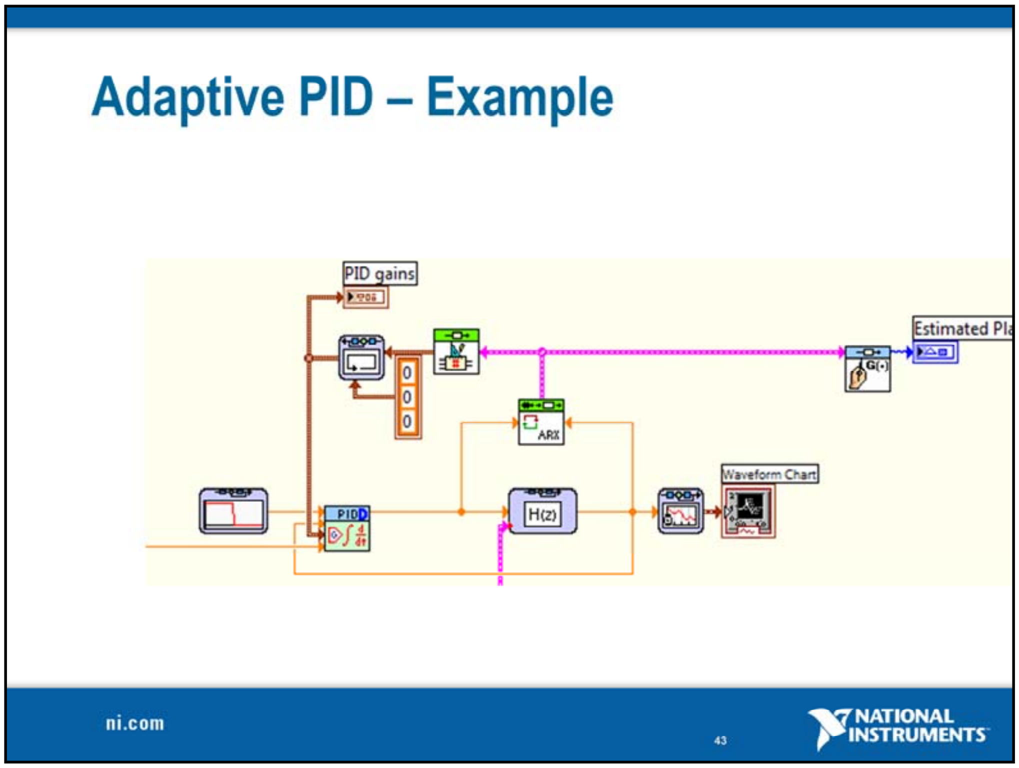
Control Design Techniques

- Adaptive Control
  - Self-tuning
  - Parameter or structure of the controller modifies with changes on the plant

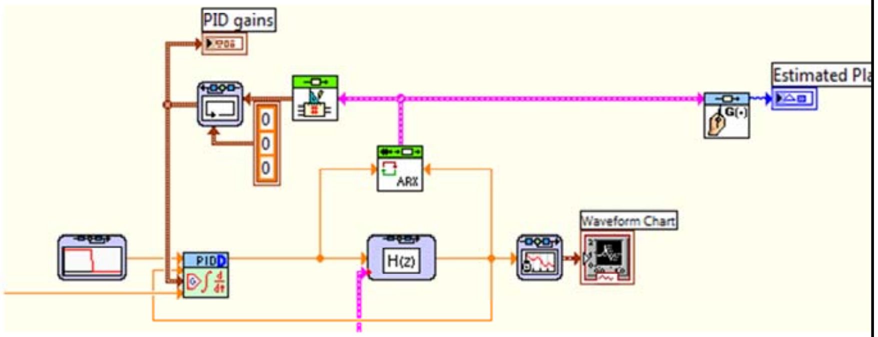
Controller Configurator
Plant Estimator
Set Point
+
-
Controller
System (Plant)

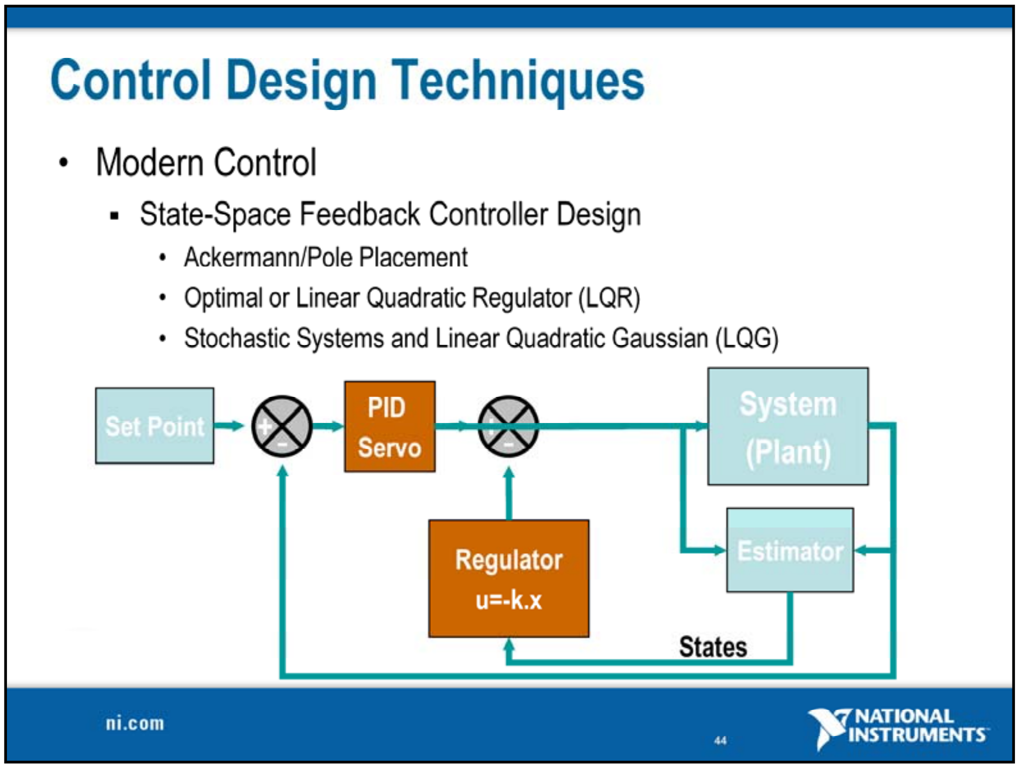
ni.com 42 NATIONAL INSTRUMENTS

To further extend PID performance on systems that change over time, you can use another advanced variant of PID controllers that involves gains change, depending on the dynamics of your system. While gain scheduling works only with the plant output to define the operating range, adaptive PID considers both inputs and outputs to find the gains. Figure 1 shows an example of an adaptive PID implementation running in LabVIEW, where the LabVIEW PID Control and LabVIEW System Identification toolkits are combined to generate the adaptive algorithm.



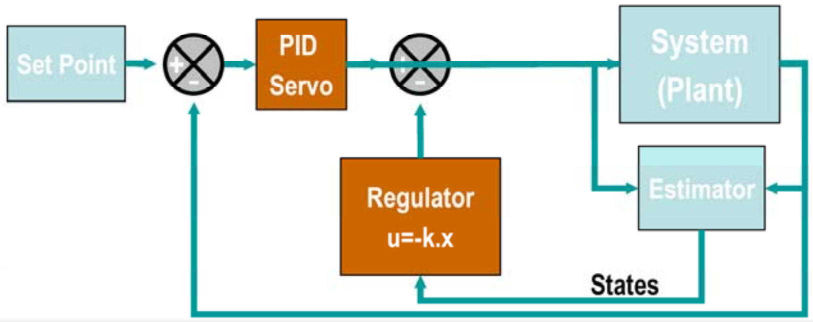
Adaptive PID – Example





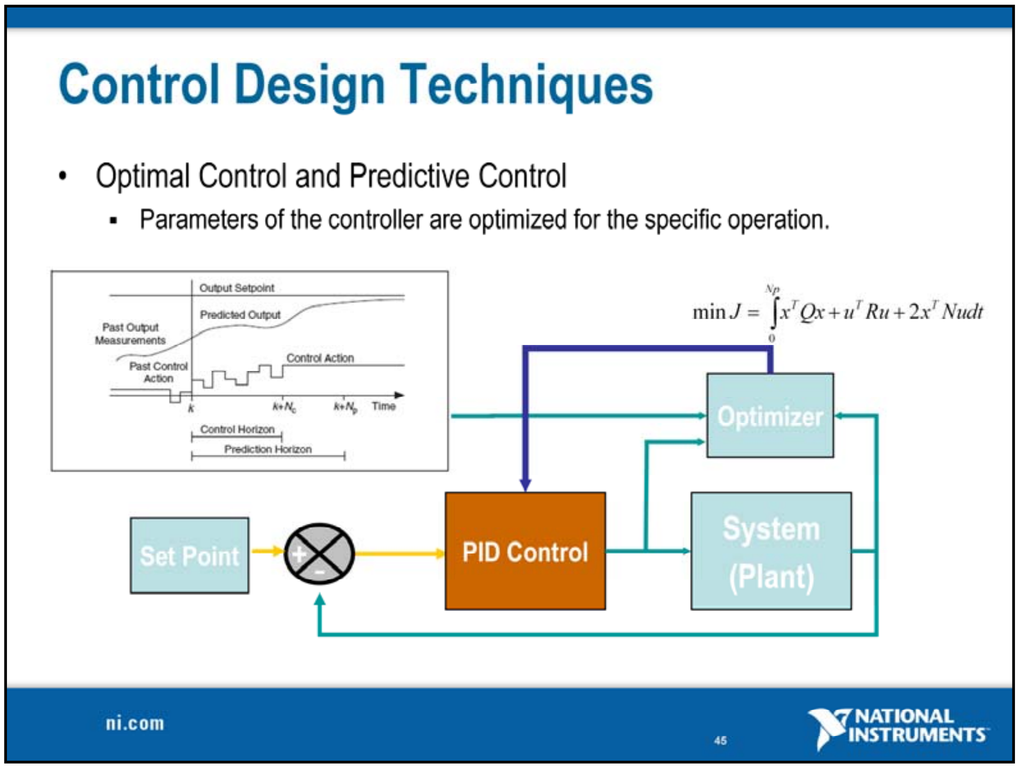
Control Design Techniques

- Modern Control
  - State-Space Feedback Controller Design
    - Ackermann/Pole Placement
    - Optimal or Linear Quadratic Regulator (LQR)
    - Stochastic Systems and Linear Quadratic Gaussian (LQG)



ni.com 44 NATIONAL INSTRUMENTS

PID benefits can be combined with other controllers like clasic controllers. Here we are showing where to implement an estimator so that a variable to be controller can be observed. A typical case would be DC motor control, were not always current can be sensed. An estimator could be used to find information on what the system is performance and act based on such information



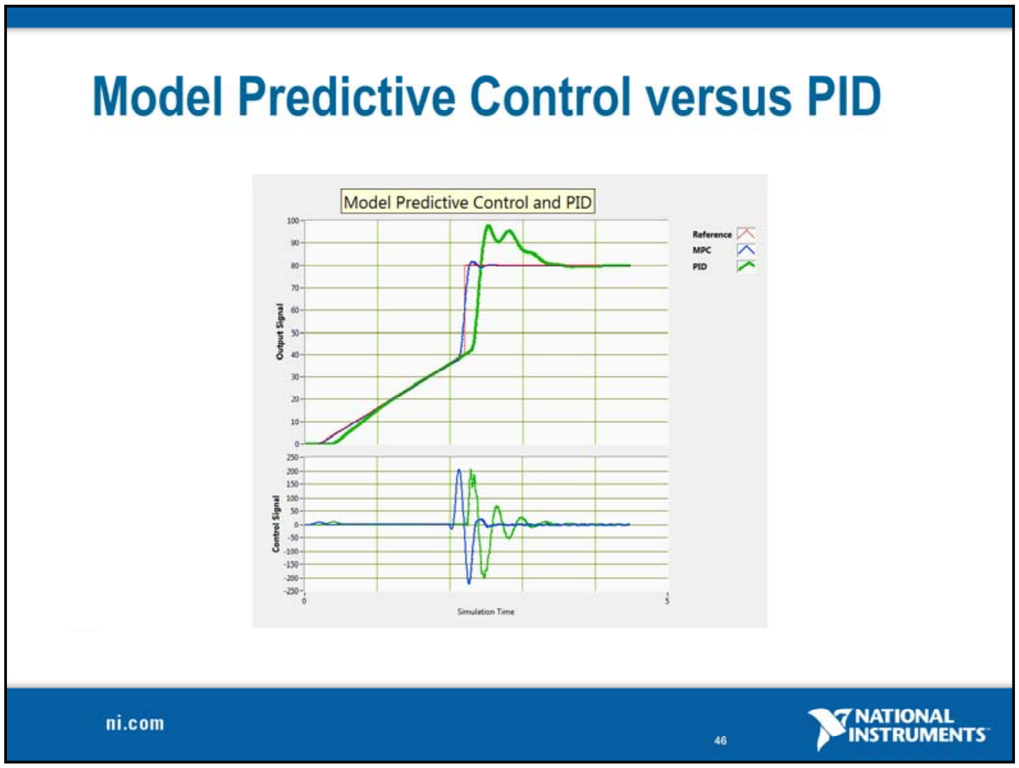
Control Design Techniques

- Optimal Control and Predictive Control
  - Parameters of the controller are optimized for the specific operation.

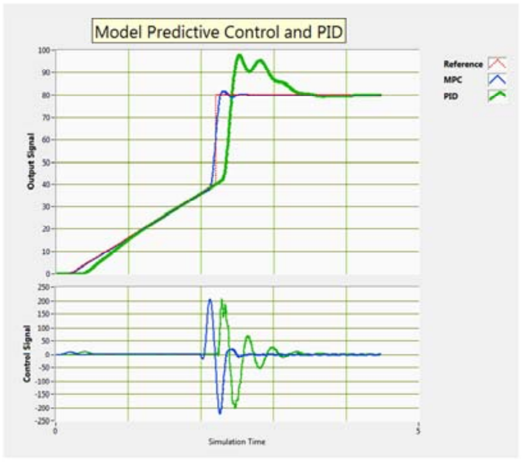
$$\min J = \int_0^{Np} x^T Qx + u^T Ru + 2x^T Nudt$$

Traditional feedback controllers adjust control action in response to a change in the output setpoint of the plant. Model predictive control (MPC), also included in the LabVIEW Control Design and Simulation Module, is a technique that engineers use to construct controllers that can adjust the control action before a change in the output setpoint actually occurs. This predictive ability, when combined with traditional feedback operation, enables a controller to make adjustments that are smoother and closer to the optimal control action values.

Figure 4 shows a comparison of an MPC controller with a PID controller running on two temperature chambers. As you can see, the more advanced control system can better track changes on the setpoint because it has knowledge on how the setpoint is going to change and how the system reacts to a given change in the control variable.

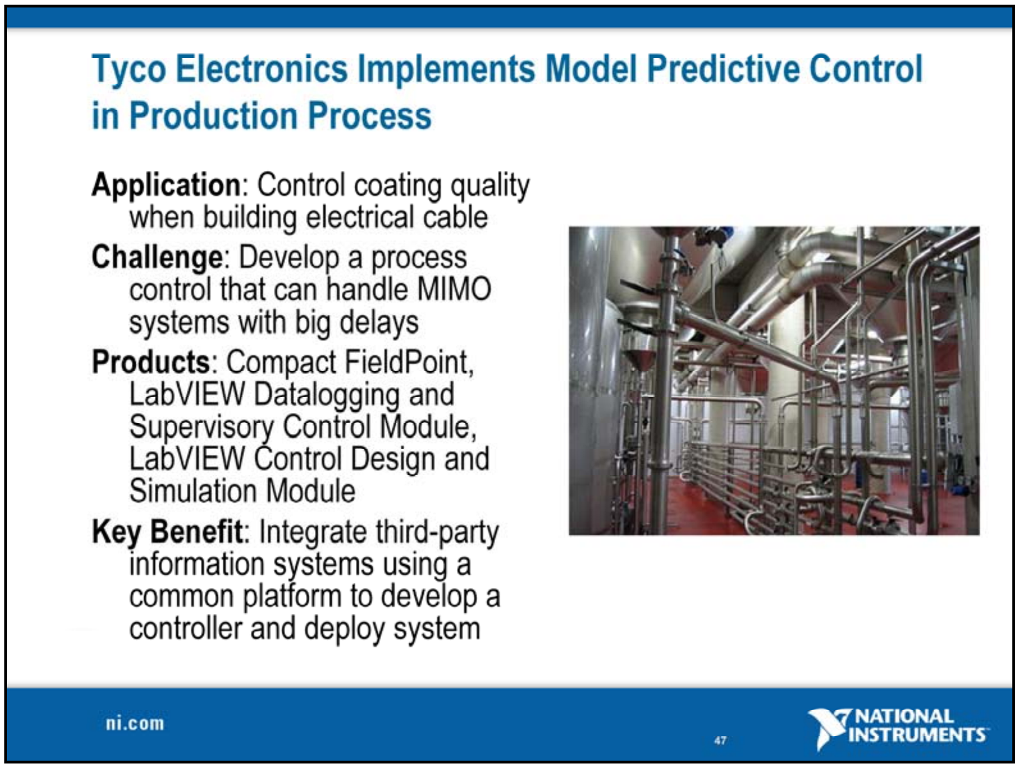


Model Predictive Control versus PID



ni.com

46



Tyco Electronics Implements Model Predictive Control in Production Process

Application: Control coating quality when building electrical cable

Challenge: Develop a process control that can handle MIMO systems with big delays

Products: Compact FieldPoint, LabVIEW Datalogging and Supervisory Control Module, LabVIEW Control Design and Simulation Module

Key Benefit: Integrate third-party information systems using a common platform to develop a controller and deploy system

ni.com 47 NATIONAL INSTRUMENTS

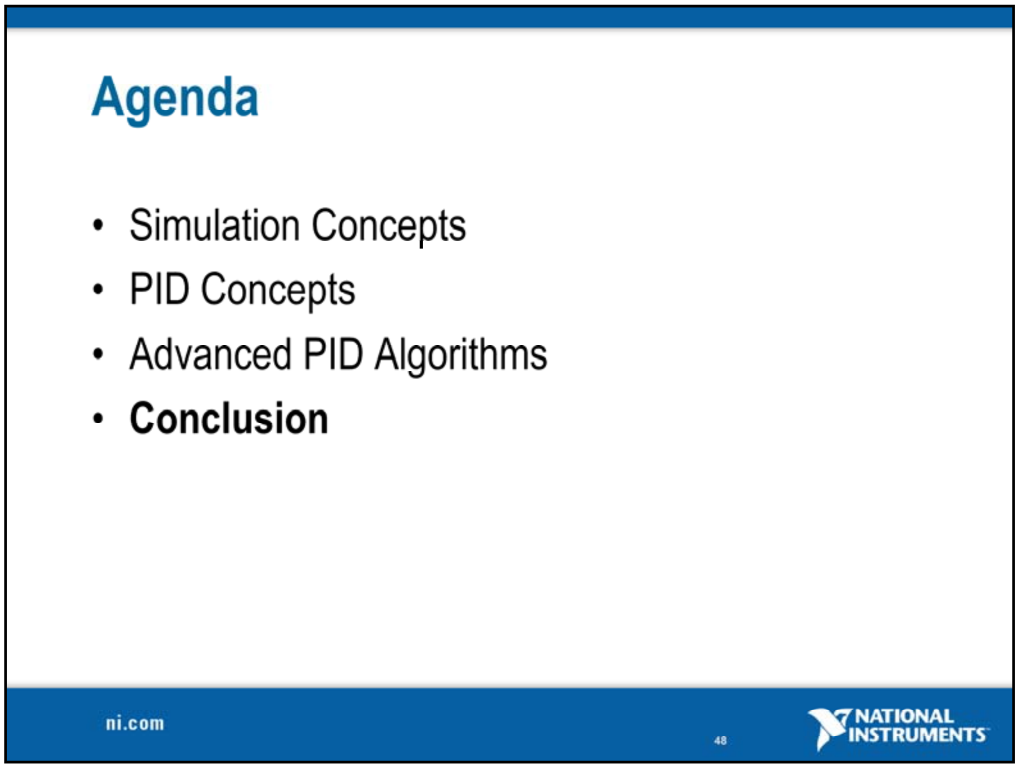
http://sine.ni.com/cs/app/doc/p/id/cs-11491

Our LabVIEW application consists of user interface process setup, monitoring, alarm, and control. It also includes features such as recipe loading, gauge normalization, system configuration, material utilization tracking, product quality tracking, and quality summary printing. The LabVIEW libraries include around 400 VIs and example code that made our complicated application development easier. In the application, we used a tab control to switch among different user screens. We also used trending and histogram charts for real-time and statistical data display, sequence structure for software logic flow, and DataSocket OPC configuration for easy OPC connections. We implemented the LabVIEW Advanced Signal Processing Toolkit for frequency analysis and filter design and used event structures to handle online controller tuning and operator interrupts.

With an abundance of powerful software features, we completed process modeling using logged data and LabVIEW modeling tools. We then used these LabVIEW simulation results in our real application with further controller tuning. We implemented the multivariable model predictive control systems on more than 10 manufacturing lines with excellent Six Sigma process control levels.

The control systems not only control the product quality to meet specifications but also stop the lines automatically if the product quality is out of specification. We achieved unmanned overnight production thereby increasing productivity and reducing machine downtime. In addition, we reduced coaxial final quality test time because the control system tracks the product quality and prints a label to identify whether the product passed or failed. We anticipated that control system modification and population to other manufacturing processes would help us reap further Six Sigma and lean manufacturing benefits.

Coaxial manufacturing involves complicated processes with many variables affecting its final product quality. Due to the nature of multiple-input and multiple-output processes with large variable time delays between the inputs and the outputs, we chose multivariable model predictive control to control these processes. We use LabVIEW to implement the model predictive control via the LabVIEW Control Design and Simulation Module. We also use it, along with OPC and Microsoft SQL technologies, for user interface, data sharing, and data logging. We successfully implemented the control systems based on LabVIEW on many similar coaxial manufacturing lines to automatically control coaxial product quality and maintain unmanned overnight production.

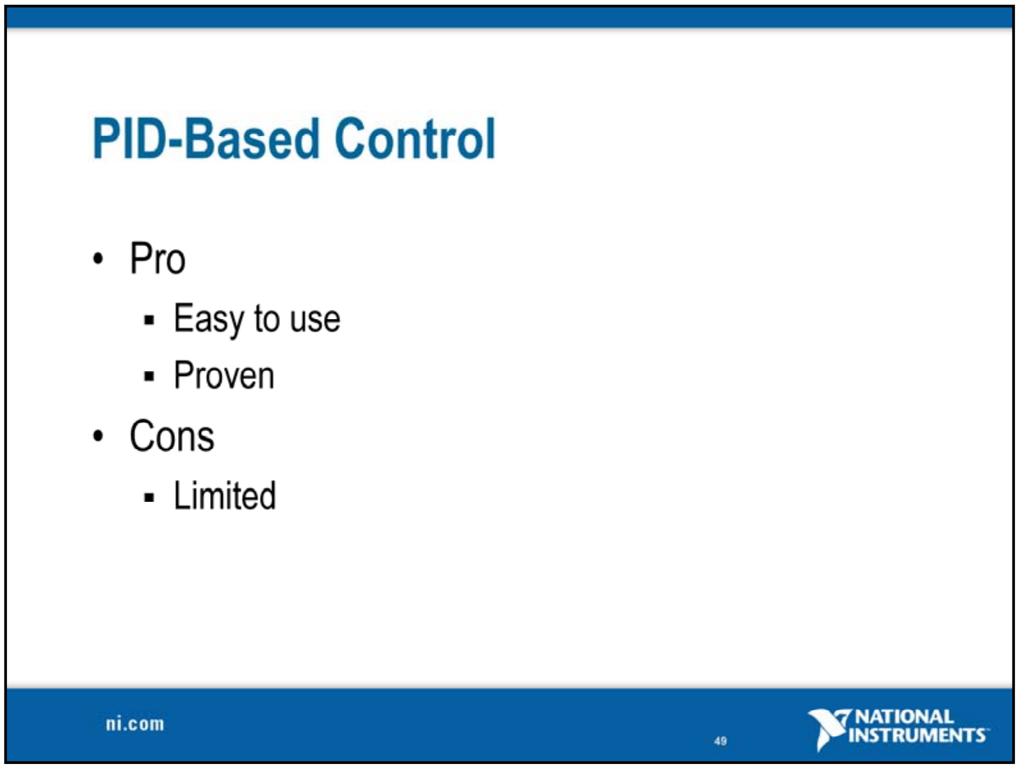


Agenda

- Simulation Concepts
- PID Concepts
- Advanced PID Algorithms
- **Conclusion**

ni.com

48



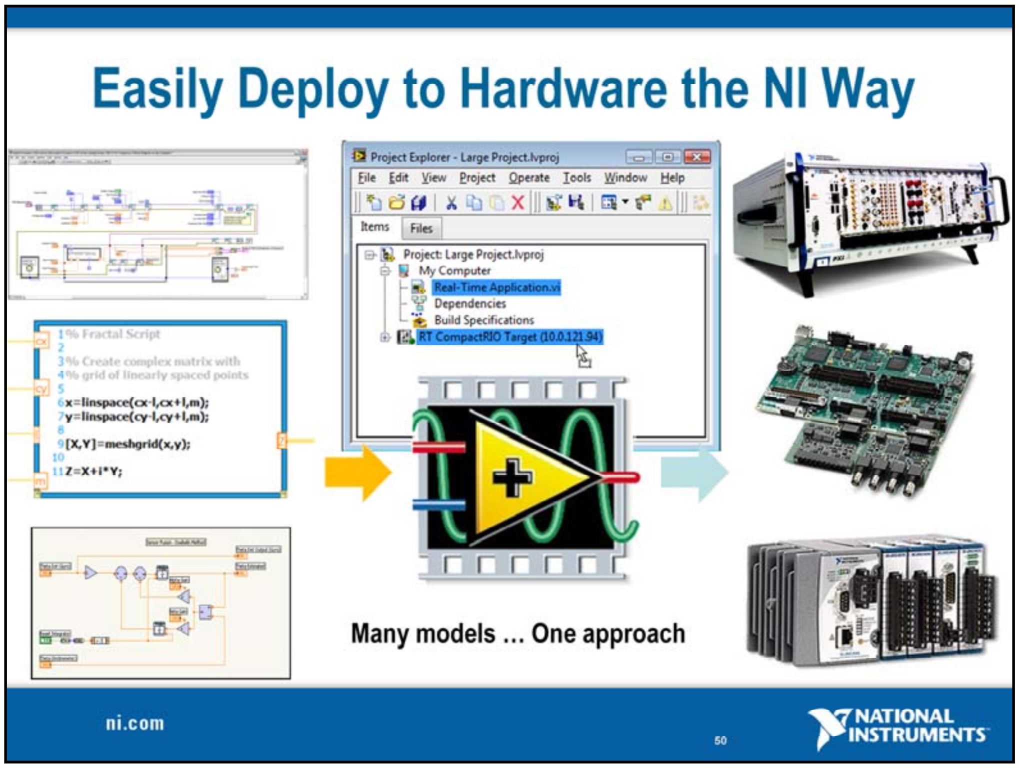
PID-Based Control

- Pro
  - Easy to use
  - Proven
- Cons
  - Limited

ni.com

49

NATIONAL INSTRUMENTS



Easily Deploy to Hardware the NI Way

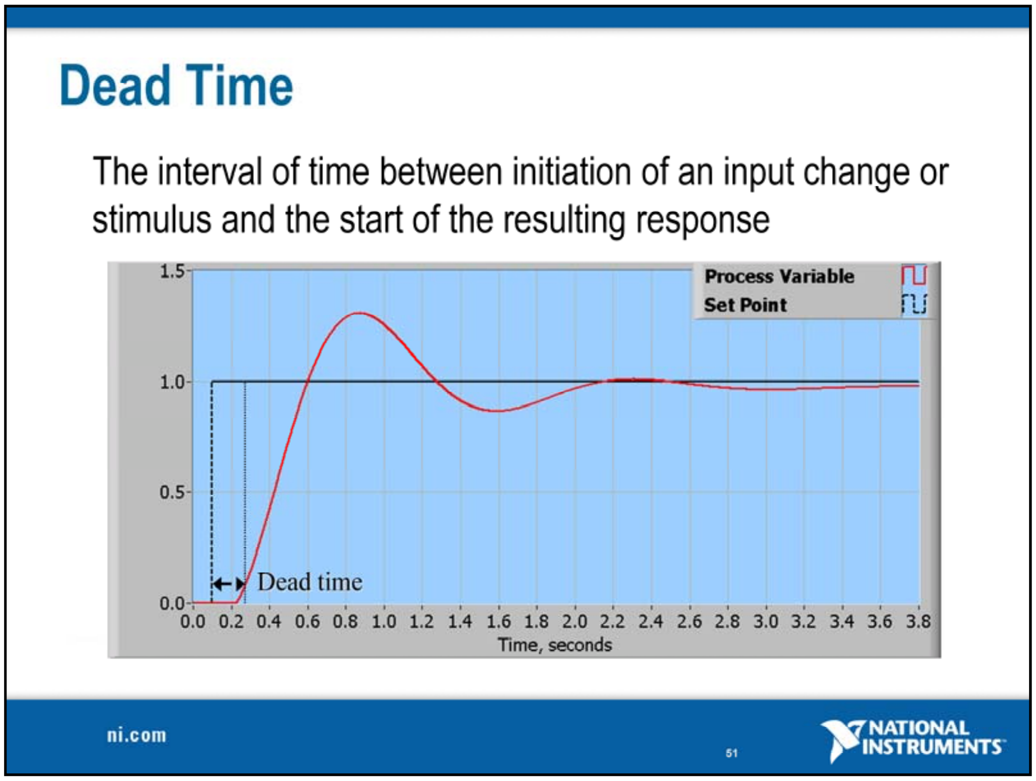
Project Explorer - Large Project.lvproj
File Edit View Project Operate Tools Window Help
Items Files
Project: Large Project.lvproj
My Computer
Real-Time Application.vi
Dependencies
Build Specifications
RT CompactRIO Target (10.0.121.94)
1 % Fractal Script
3 % Create complex matrix with
4 % grid of linearly spaced points
x=linspace(cx-l,cx+l,m);
y=linspace(cy-l,cy+l,m);
[X,Y]=meshgrid(x,y);
Z=X+i*Y;
NATIONAL INSTRUMENTS

Many models … One approach

ni.com 50

Once we have seen what PID is, what can it be programed and some options to improve it, the next step is how to deploy to a Real Time target, so it can be executed robustly thus ensuring the reliability and performance of the control systems.

This process could be long and tedious and require specific knowledge of embedded systems. Luckly, National Instrument control platform is design so all the low level embedded programming will be taked care of so that the algorithms can be deployed directly into high performance NI PAC platforms such as PXI or compactRIO



Dead Time

The interval of time between initiation of an input change or stimulus and the start of the resulting response

ni.com

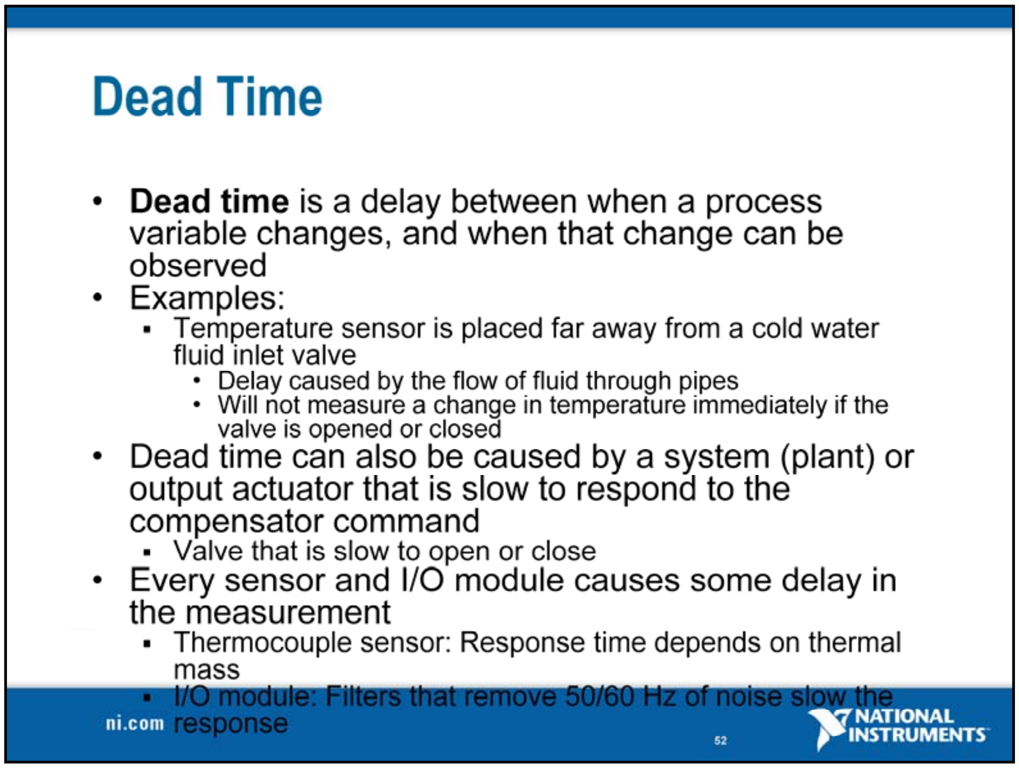
51

Some systems exhibit an undesirable behavior called **Dead time**. **Dead time** is a delay between when a process variable changes, and when that change can be observed. For instance, if a temperature sensor is placed far away from a cold water fluid inlet valve, it will not measure a change in temperature immediately if the valve is opened or closed. **Dead time** can also be caused by a **System (Plant)** or **Output Actuator** that is slow to respond to the **Compensator** command, for instance a valve that is slow to open or close. A common source of deadtime in chemical plants is the delay caused by the flow of fluid through pipes.

Once the performance requirements have been specified, it is time to examine the **System (Plant)** and select an appropriate control scheme. In the vast majority of applications, a **Proportional-Integral-Derivative (PID) Compensator** will provide satisfactory results. However, it is important to understand some of the alternatives to basic PID control and when they are appropriate. It should be noted that, beginning with LabVIEW 6.1, the PID VIs support array inputs for multi-loop PID control applications.

Definitions:
Autotuning: Performing an automatic testing process to determine the controller gains that will provide the best controller performance.

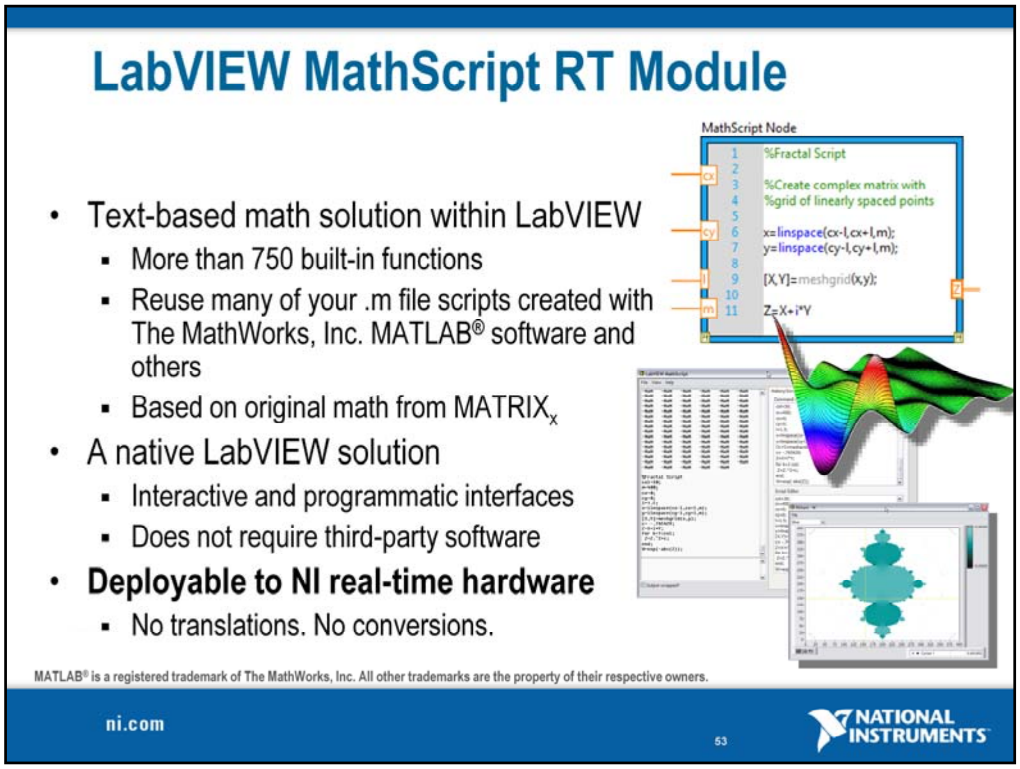
Nonlinear System: A system in which the control parameters that produce a desired response at one operating point might not produce a satisfactory response at another operating point. For instance, a chamber partially filled with fluid will exhibit a much faster response to heater output when nearly empty than it will when nearly full of fluid.



Dead Time

- **Dead time** is a delay between when a process variable changes, and when that change can be observed
- Examples:
  - Temperature sensor is placed far away from a cold water fluid inlet valve
    - Delay caused by the flow of fluid through pipes
    - Will not measure a change in temperature immediately if the valve is opened or closed
- Dead time can also be caused by a system (plant) or output actuator that is slow to respond to the compensator command
  - Valve that is slow to open or close
- Every sensor and I/O module causes some delay in the measurement
  - Thermocouple sensor: Response time depends on thermal mass
  - I/O module: Filters that remove 50/60 Hz of noise slow the response

ni.com 52 NATIONAL INSTRUMENTS



LabVIEW MathScript RT Module

- Text-based math solution within LabVIEW
  - More than 750 built-in functions
  - Reuse many of your .m file scripts created with The MathWorks, Inc. MATLAB® software and others
  - Based on original math from MATRIX$_x$
- A native LabVIEW solution
  - Interactive and programmatic interfaces
  - Does not require third-party software
- **Deployable to NI real-time hardware**
  - No translations. No conversions.

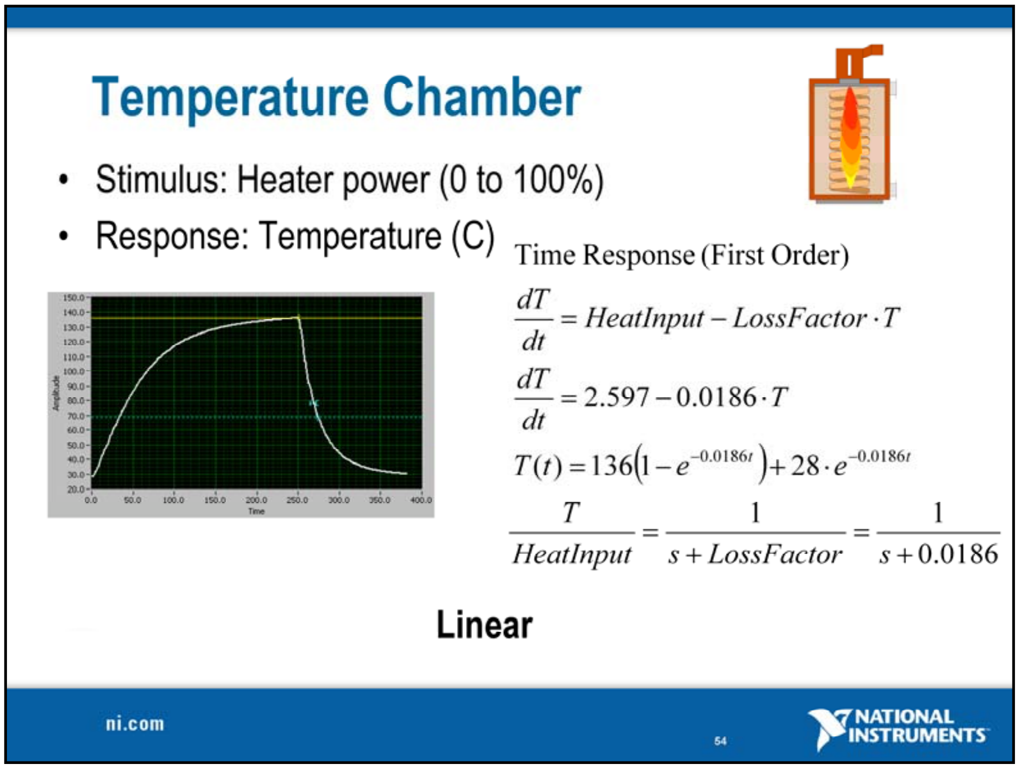
```
%Fractal Script

%Create complex matrix with
%grid of linearly spaced points

x=linspace(cx-l,cx+l,m);
y=linspace(cy-l,cy+l,m);

[X,Y]=meshgrid(x,y);

Z=X+i*Y
```

MATLAB® is a registered trademark of The MathWorks, Inc. All other trademarks are the property of their respective owners.

ni.com 53 NATIONAL INSTRUMENTS

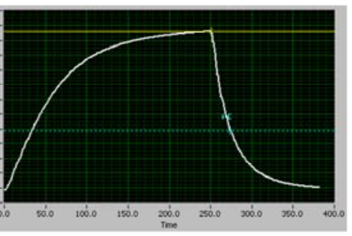
MathScript extends LabVIEW with a math-oriented text-based programming language that is generally compatible with the .m file script syntax used by alternative technical computing software such as the MATLAB software development environment and others. Such compatibility means that you can “instrument your algorithms” by running your .m file scripts with LabVIEW to access productivity enhancing LabVIEW features such as simplified instrument control / data acquisition, file / database access, user interface development and more.

Compatibility also means that students working with MathScript gain industry relevant experience with the widely used .m file script language.



Temperature Chamber

- Stimulus: Heater power (0 to 100%)
- Response: Temperature (C)



Time Response (First Order)

$$\frac{dT}{dt} = HeatInput - LossFactor \cdot T$$

$$\frac{dT}{dt} = 2.597 - 0.0186 \cdot T$$

$$T(t) = 136\left(1 - e^{-0.0186t}\right) + 28 \cdot e^{-0.0186t}$$

$$\frac{T}{HeatInput} = \frac{1}{s + LossFactor} = \frac{1}{s + 0.0186}$$

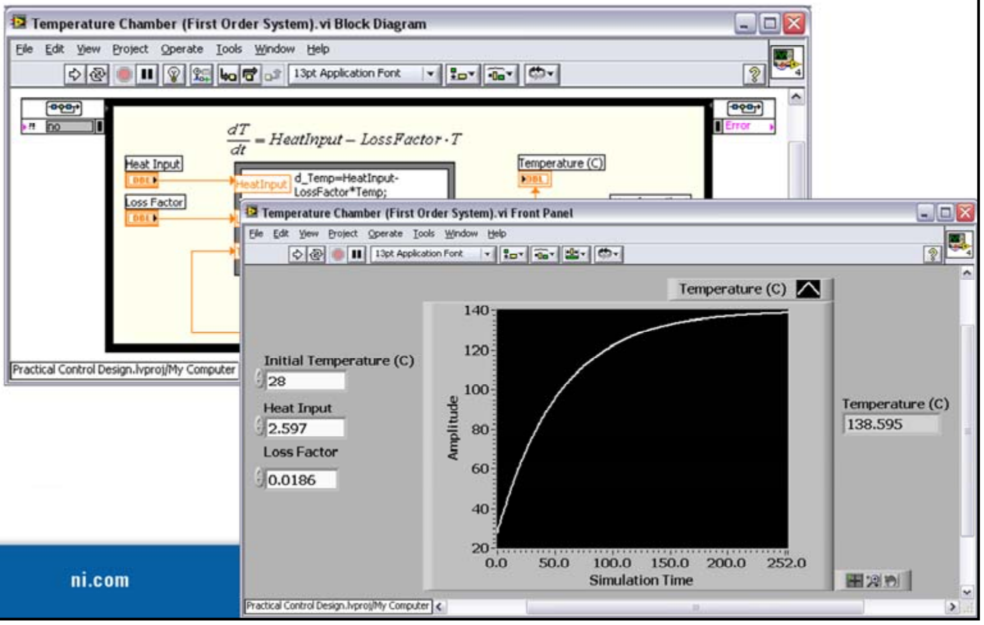
Linear

ni.com

54

NATIONAL INSTRUMENTS

Temperature Chamber Simulation


Temperature Chamber (First Order System).vi Block Diagram
File Edit View Project Operate Tools Window Help
13pt Application Font
Heat Input
Loss Factor
Temperature (C)
Error
Practical Control Design.lvproj/My Computer
Temperature Chamber (First Order System).vi Front Panel
Temperature (C)
Amplitude
Simulation Time

$$\frac{dT}{dt} = HeatInput - LossFactor \cdot T$$

```
d_Temp=HeatInput-
LossFactor*Temp;
```

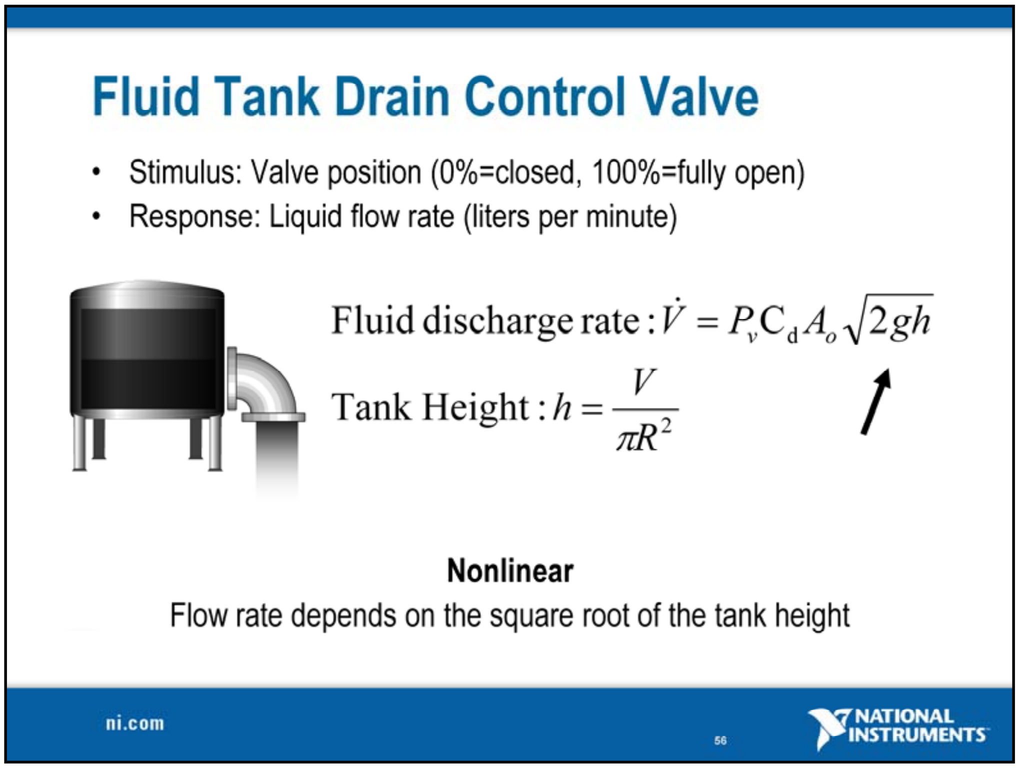
Initial Temperature (C) 28

Heat Input 2.597

Loss Factor 0.0186

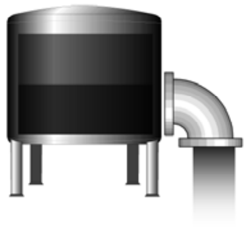
Temperature (C) 138.595

ni.com



Fluid Tank Drain Control Valve

- Stimulus: Valve position (0%=closed, 100%=fully open)
- Response: Liquid flow rate (liters per minute)



Fluid discharge rate : $\dot{V} = P_v C_d A_o \sqrt{2gh}$

Tank Height : $h = \dfrac{V}{\pi R^2}$

$\uparrow$

Nonlinear

Flow rate depends on the square root of the tank height

ni.com 56 NATIONAL INSTRUMENTS

Source: Engineer-in-Training Reference Manual, 8th Edition, Michael R. Lindeburg, 1998

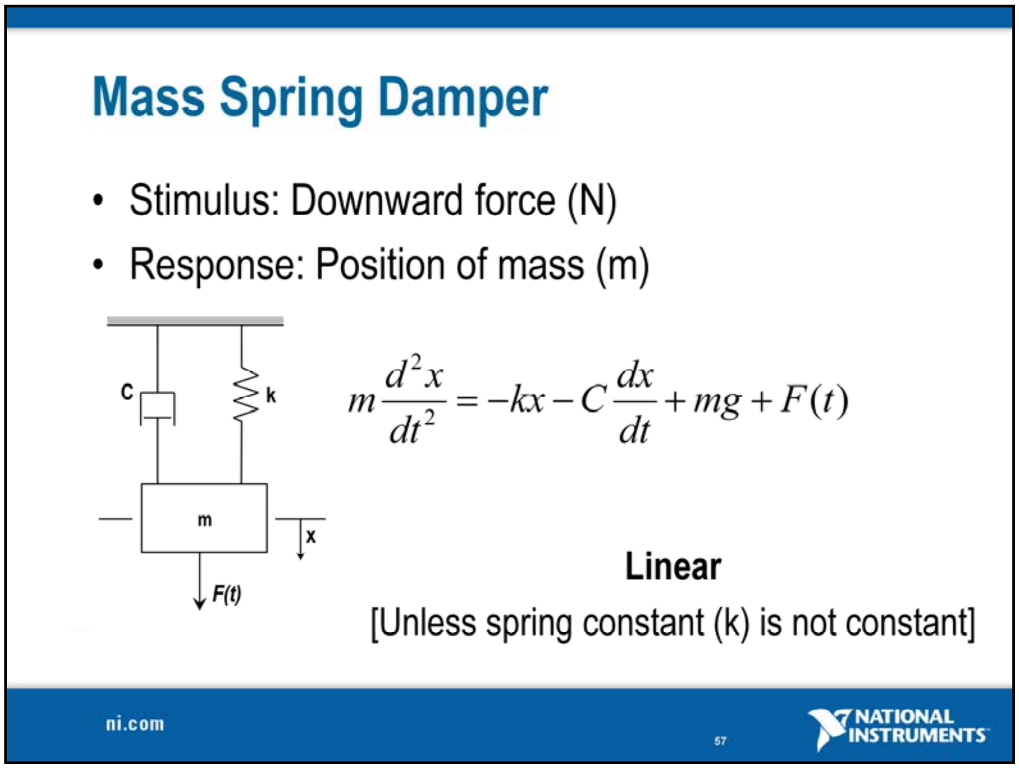
Pv = Valve position (1 = fully open)

Cd = Coefficient of discharge (accounts for turbulence and shape of orifice)

Ao = Area of the orifice

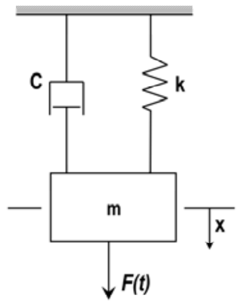
G = Acceleration of gravity

h = Fluid height in tank



Mass Spring Damper

- Stimulus: Downward force (N)
- Response: Position of mass (m)

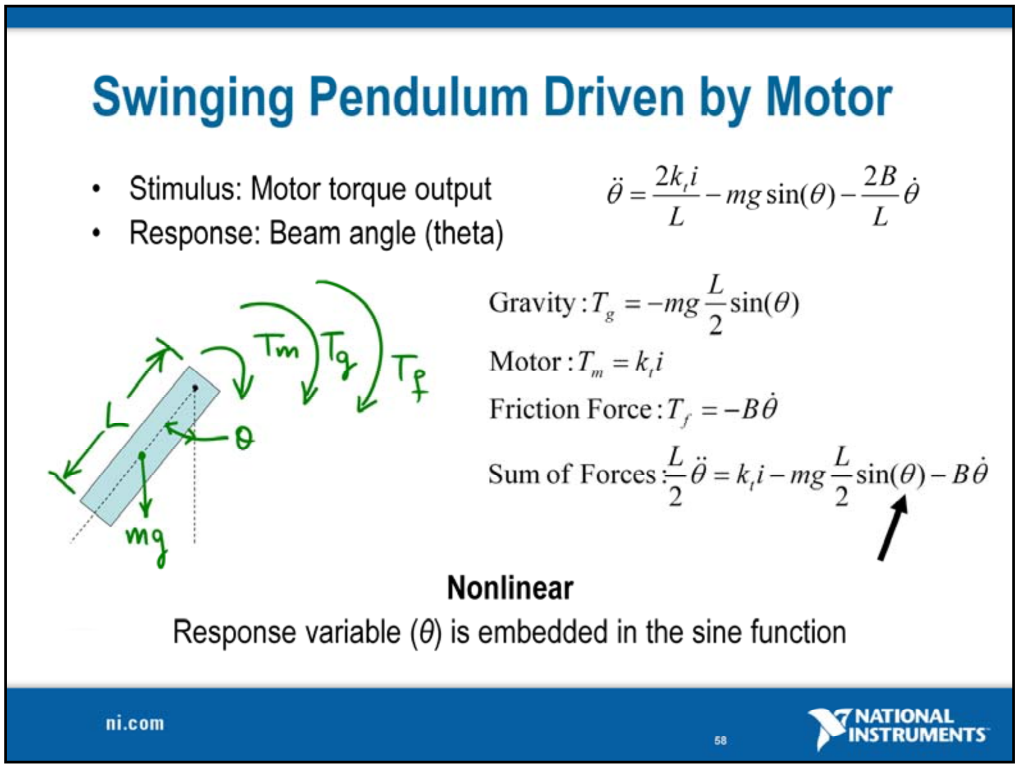


$$m\frac{d^2x}{dt^2} = -kx - C\frac{dx}{dt} + mg + F(t)$$

Linear

[Unless spring constant (k) is not constant]

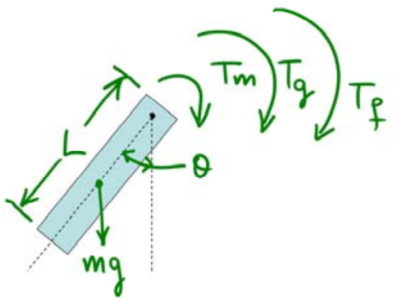
ni.com 57



Swinging Pendulum Driven by Motor

- Stimulus: Motor torque output
- Response: Beam angle (theta)

$$\ddot{\theta} = \frac{2k_t i}{L} - mg\sin(\theta) - \frac{2B}{L}\dot{\theta}$$



$$\text{Gravity}: T_g = -mg\frac{L}{2}\sin(\theta)$$

$$\text{Motor}: T_m = k_t i$$

$$\text{Friction Force}: T_f = -B\dot{\theta}$$

$$\text{Sum of Forces}: \frac{L}{2}\ddot{\theta} = k_t i - mg\frac{L}{2}\sin(\theta) - B\dot{\theta}$$

Nonlinear

Response variable (θ) is embedded in the sine function

ni.com

58

NATIONAL INSTRUMENTS