Control System Training

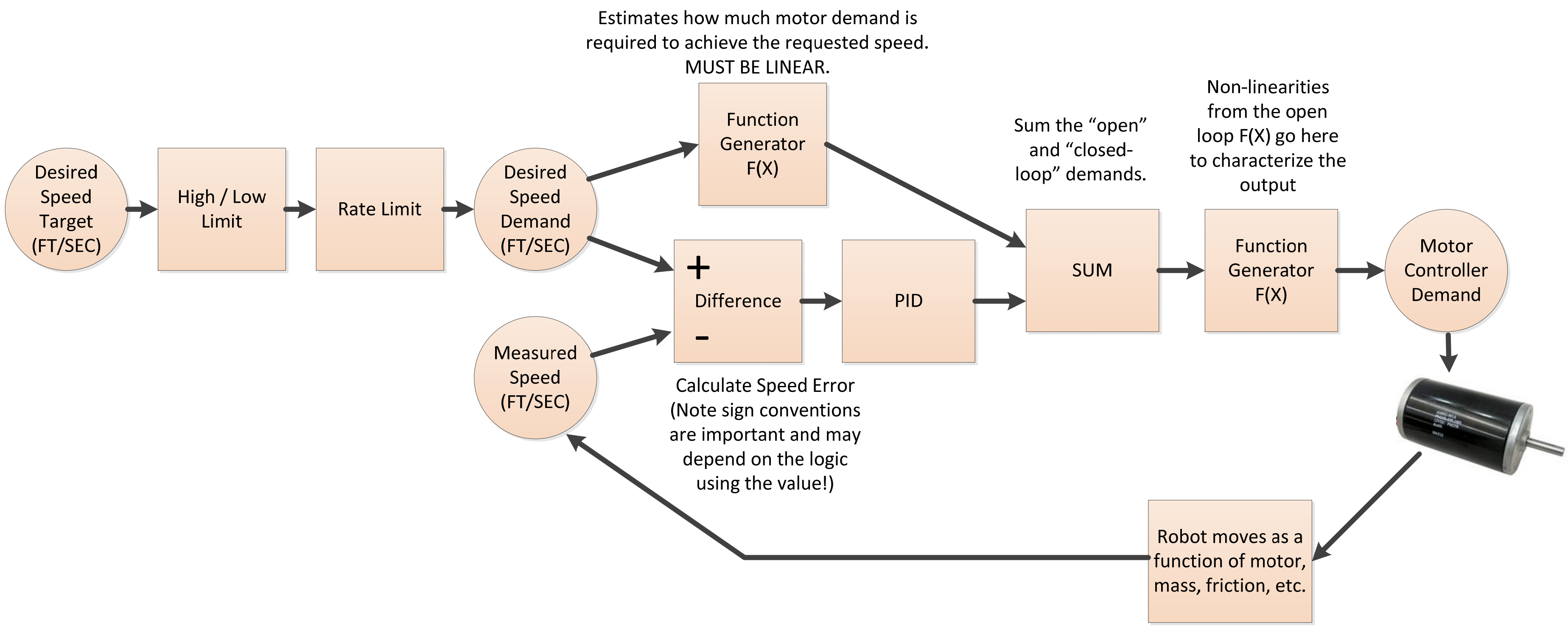
Module 12 – Speed Control

Copyright Notice

Speed Control Concepts

- **The demand to the motor controllers does NOT control speed. It controls how much electricity gets to the motors.**
- **Good speed control should help improve:**
  - Not drifting when moving straight
  - Smoother slow speed control
- **Note:**
  - There is a way to wire and communicate with CAN based motor controllers so that the motor controllers perform speed control. The concepts in this module apply to both configurations. The difference is where some parts of the logic is executed.

Speed Control Block Diagram



- **Feedforward F(x), Output characterization F(x), or PID can be disabled as needed or for tuning.**
- **Left and right motors would have individual circuits.**
  - Other drive types would also have separate circuits.

Tuning - Determine Feedforward

- **Disable PID**
- **Test robot to determine feedforward**
- **If testing is not possible, determine maximum speed and use a linear function.**
- **(Insert trend here…)**

PID

- **Mathematically adjusts output to correct for error**
- **ERR = (SP – PV)**
  - SP = Setpoint (speed demand – FT/SEC)
  - PV = Process variable (actual speed – FT/SEC)
  - In some cases the equation will be PV - SP
- **$ERR_{sc} = K_s$ * ERR**
  - Scale the error to be in the same units as the output
  - K_s = Max Output / Max PV
- **Mathematically:**
  - $Out = K_p \, x \, ERR_{sc} + K_i \, x \int ERR_{sc} + K_d \, x \, d\frac{ERR_{sc}}{dt}$

PID – Numerical Implementation

- **Numerical (programmatic) Implementation:**
  - $I_{err} = I_{err} + 0.5 * (ERR_{sc} + last\_ERR_{sc}) * Time\_Diff$
  - $D_{err} = (ERR_{sc} - last\_ERR_{sc}) / Time\_Diff$
  - $OUT = K_p * ERR_{sc} + K_i * I_{err} + K_d * D_{err}$
- **Ensure the output is within the allowed limits**
  - The individual terms could sum to a value larger than the allowed output!
- **This is a “non-interacting” implementation of a PID.**
  - “Traditional” PID multiplies the integral term by Kp
  - Non-interacting PIDs are more flexible and can work better with a feedforward term.

PID – Things to Consider

- **Integral wind up**
  - Symptom example: PID becomes unresponsive when changing directions!
  - Occurs when integral grows too large or too small
  - Remedy: Various ways to limit integral term (average last proportional + feedforward term over last several scans used to limit integral term)
- **Differentiation multiplies noise**
  - Often the Kd is set to zero
  - Various ways to smooth noise by calculating Derr over last several scans
- **Note:**
  - The C++ and JAVA PID implementation doesn't do either of these. The standard Labview PID does have these built in.

PID - Tuning

- **Goals**
  - Quickly match setpoint and process variable
  - Don't overshoot too much
  - Don't oscilate
- **Need to trend setpoint and process variable**
- **Repeated make step changes (instantaneous movements) of setpoint and adjust Kp, Ki, Kd to obtain desired results.**
- **Leave Kd as zero.**
- **Start with Kp between 0.5 and 1.0. Adjust Ki and Kp to get desired results.**
- **Many other more involved methods to tune PID.**

PID Demo

- **See demo application.**

PID – Additional Things to Consider

- **Consider a manual mode that bypasses the PID**
  - Instrumentation breaks.
  - Different field conditions or robot conditions may invalidate the PID tuning
  - Stuff happens…
  - In manual, the PID output is set to zero. Only the feedforward is used. The PID internally "tracks" the output to all "bumpless" return to "auto" operation.
- **Note:**
  - The C++ and JAVA PID implementation appear to have a manual or auto mode, or manual tracking. The standard Labview "advanced" PID does have these built in.
- **Consider writing your own PID…**

PID – Additional Information

- **This training only covers limited practical information. There is a large amount of information about PID's online.**
- **Including information on:**
  - Traditional and non-interacting forms of the PID equation
  - Response information (over damped, under damped, critically damped.)
  - Information on numerical integration (integration by trapezoids)
  - PID integral windup
  - Derivative smoothing to help reduce noise
  - Tuning methods