01211433 Vision and Control of Industrial Robots

PID Controllers

ตัวควบคุม PID

หัวข้อบรรยาย

- คุณสมบัติ
- รูปแบบของตัวควบคุม PID
- ผลจากพารามิเตอร์
- การอิมพลิเมนต์
- การปรับแต่ง
- Anti-windup
- Auto tuning

ตัวควบคุม PID
(Proportional Integral Derivative Controllers)

- ใช้ในงานควบคุมอุตสาหกรรมมากกว่า 90 %
- ไม่ต้องการการออกแบบระบบควบคุมหรือโมเดลของพลานต์
- อิมพลิเมนต์และปรับแต่งง่าย
- มีหลายรูปแบบ

อัลกอริทึ่ม PID มาตรฐาน

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d \frac{de(t)}{dt}\right)$$

K=proportional gain T_i = integral time T_d=derivative time

เมื่อแปลงลาปลาซ $$u(s) = C(s)e(s)$$

$$C(s) = K\left(1 + \frac{1}{sT_i} + sT_d\right)$$

อัลกอริทึม PID ที่นิยมใช้

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt}$$

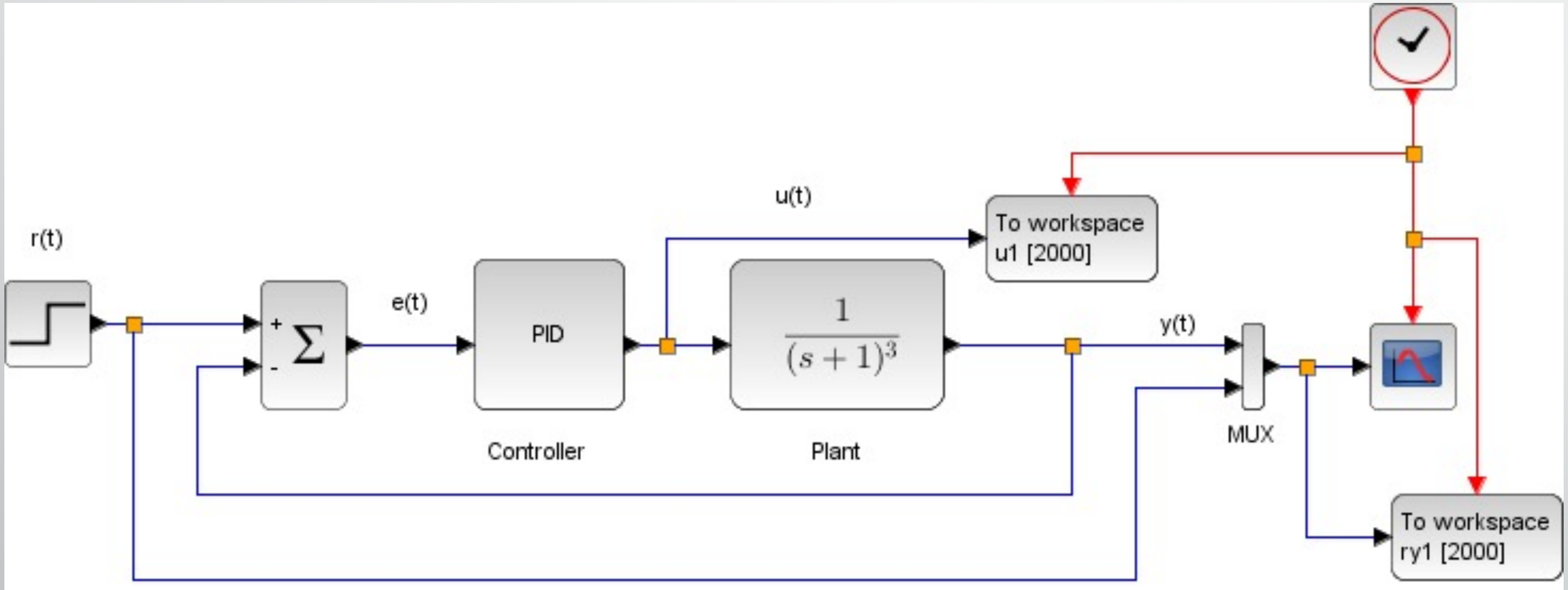
เมื่อแปลงลาปลาซ $u(s) = C(s)e(s)$

$$C(s) = K_p + \frac{K_i}{s} + K_d s$$

การแปลงพารามิเตอร์จากรูปแบบมาตรฐาน

$$K_p = K, \quad K_i = \frac{K}{T_i}, \quad K_d = KT_d$$

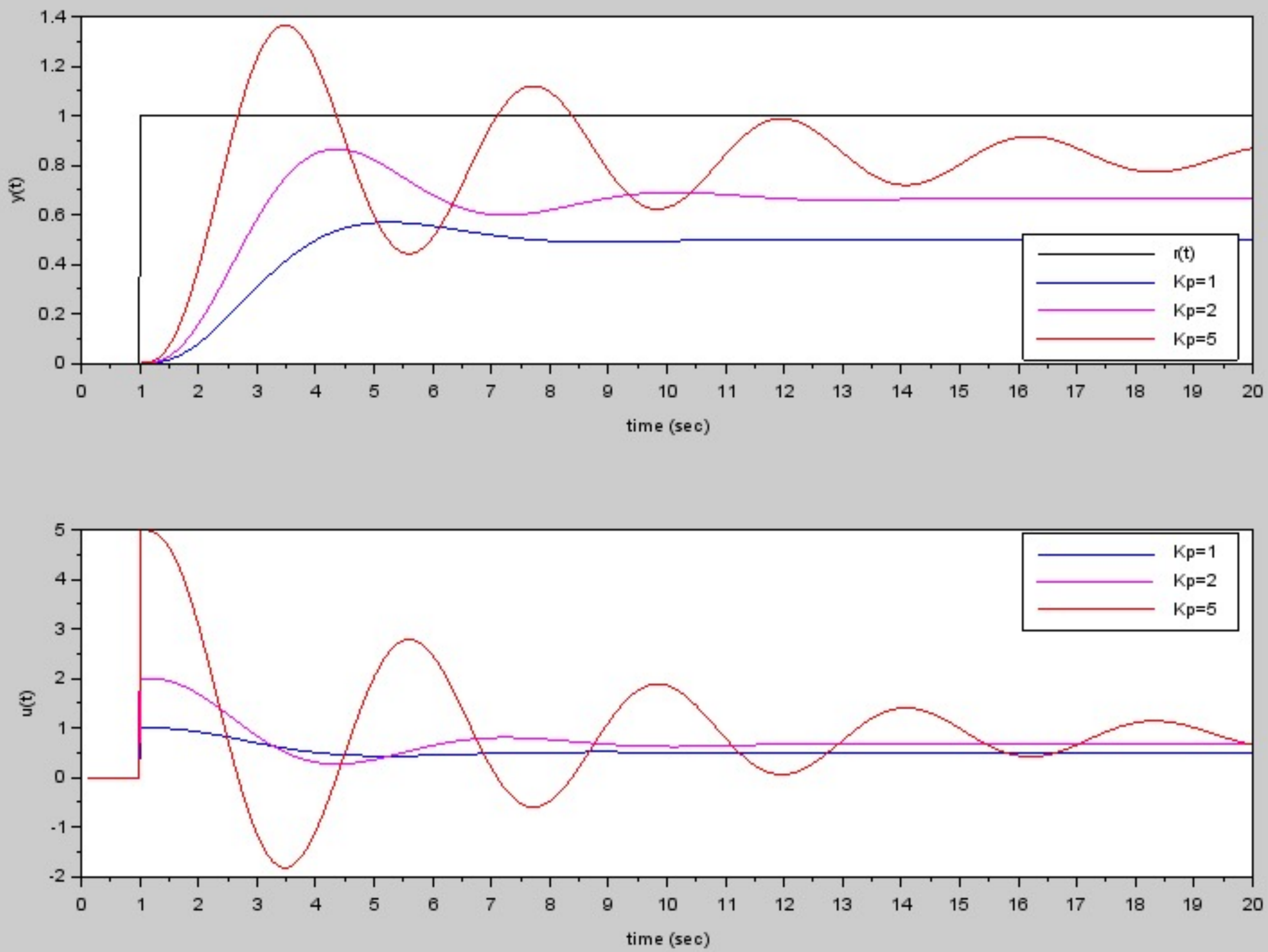
โมเดลการป้อนกลับ PID บน XCos



การควบคุมสัดส่วน (proportional control)

- ใช้การป้อนกลับอย่างง่าย
  - $u_P(t) = Ke(t)$ โดย $e(t) = r(t) - y(t)$
- ไม่สามารถขจัดค่าแตกต่างในสถานะนิ่ง (steady-state error)
- หากต้องการลด SS error ให้น้อยมากจะต้องเพิ่มอัตราขยาย K
  - เกิดผลเสียต่อเสถียรภาพ

ผลตอบสนองขั้นบันไดต่อการปรับค่า Kp

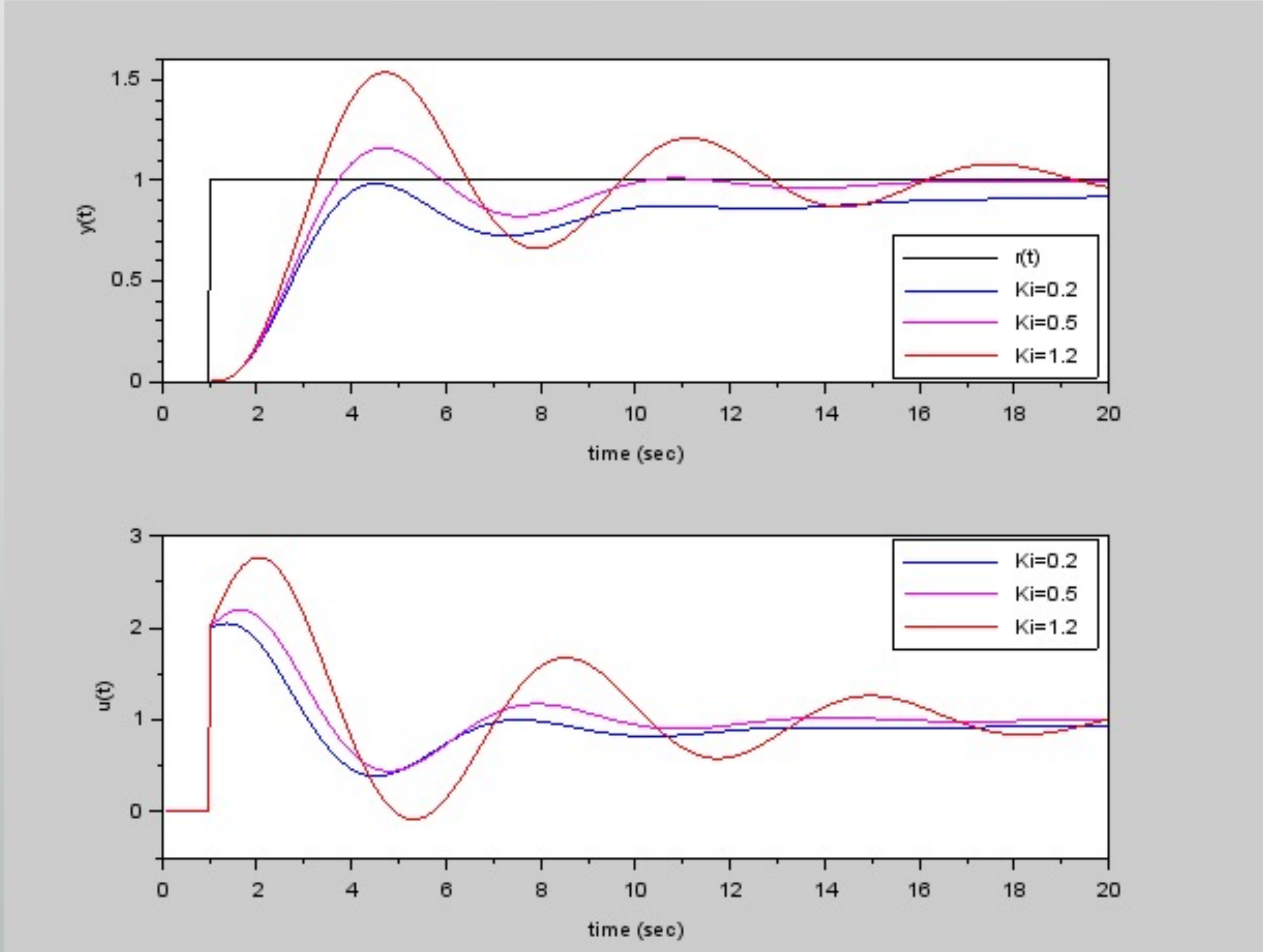


พจน์ปริพันธ์ (Integral term)

$$\frac{K_i}{s}$$

- วัตถุประสงค์เพื่อขจัดค่าแตกต่างในสถานะนิ่ง
- การเพิ่มค่า Ki อาจทำให้ระบบแกว่งและมีการพุ่งเกินสูง
- หากเพิ่มค่า Ki มากเกินไประบบป้อนกลับอาจเสียเสถียรภาพ

ผลตอบสนองขั้นบันไดต่อการปรับค่า Ki



Kp ตั้งไว้คงที่เท่ากับ 2

พจน์อนุพันธ์ (derivative term)

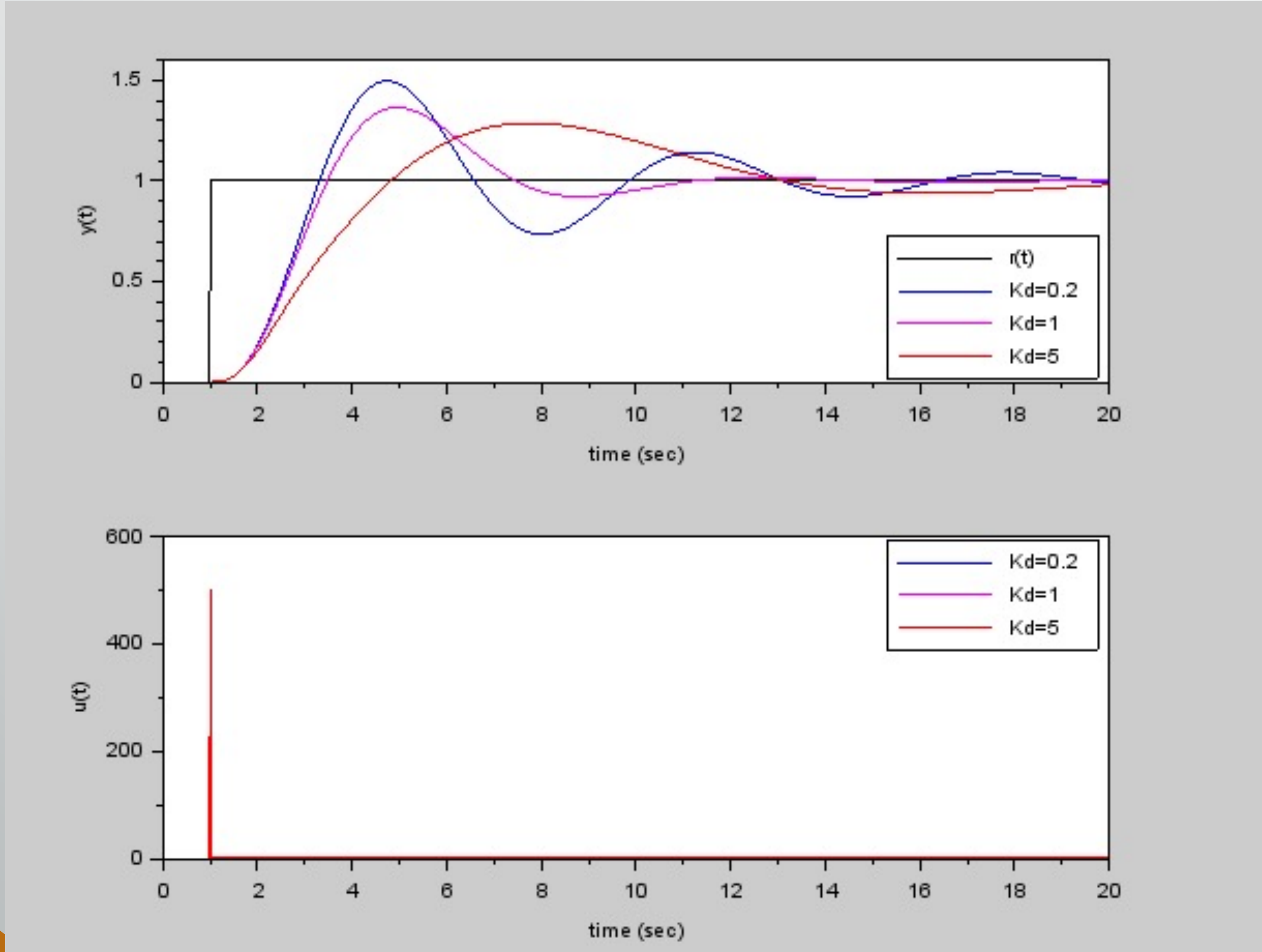
$$K_d s$$

- สามารถช่วยให้ผลตอบสนองเข้าสู่สถานะนิ่งได้เร็วขึ้น
- ถ้าปรับมากเกินไปเกิดผลเสียกับผลตอบสนอง
- ขยายสัญญาณรบกวน
- ในทางปฏิบัติอิมพลิเมนต์เป็นวงจรกรอง

$$\frac{NK_d}{1+N/s}$$

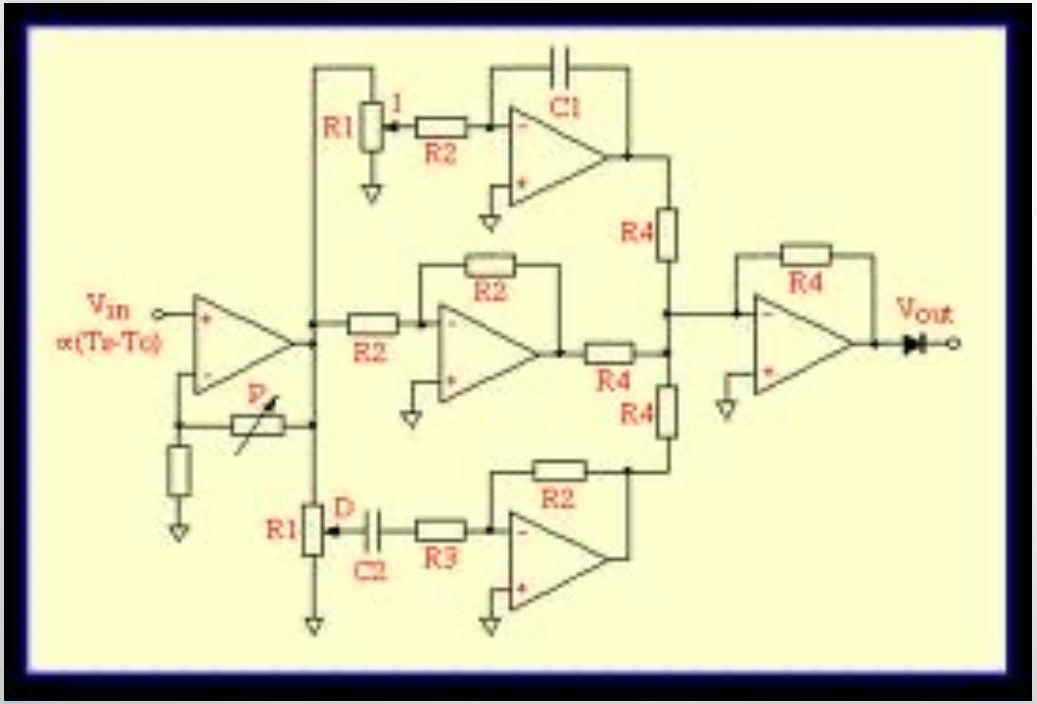
N มีค่าประมาณ 2-20

ผลตอบสนองขั้นบันไดต่อการปรับค่า Kd



Kp =2, Ki = 1.2

Analog PID Implementation



Digital PID Implementation

```
double e, e1, e2, u, delta_u;

k1= kp + ki + kd;

k2=-kp - 2*kd;

k3= kd;

void pid( )

{

        e2 = e1;                                 // update error variables

        e1 = e;

        y = readADC( );                              // read variable from sensor

        e = setpoint - y;                            // compute new error

        delta_u = k1*e + k2*e1 + k3*e2;              // PID algorithm (3.17)

        u = u + delta_u;

        if (u > UMAX) u = UMAX;                  // limit to DAC range

        if(u < umin) u = UMIN;

        writeDA(u);                              // send to DAC hardware

}
```

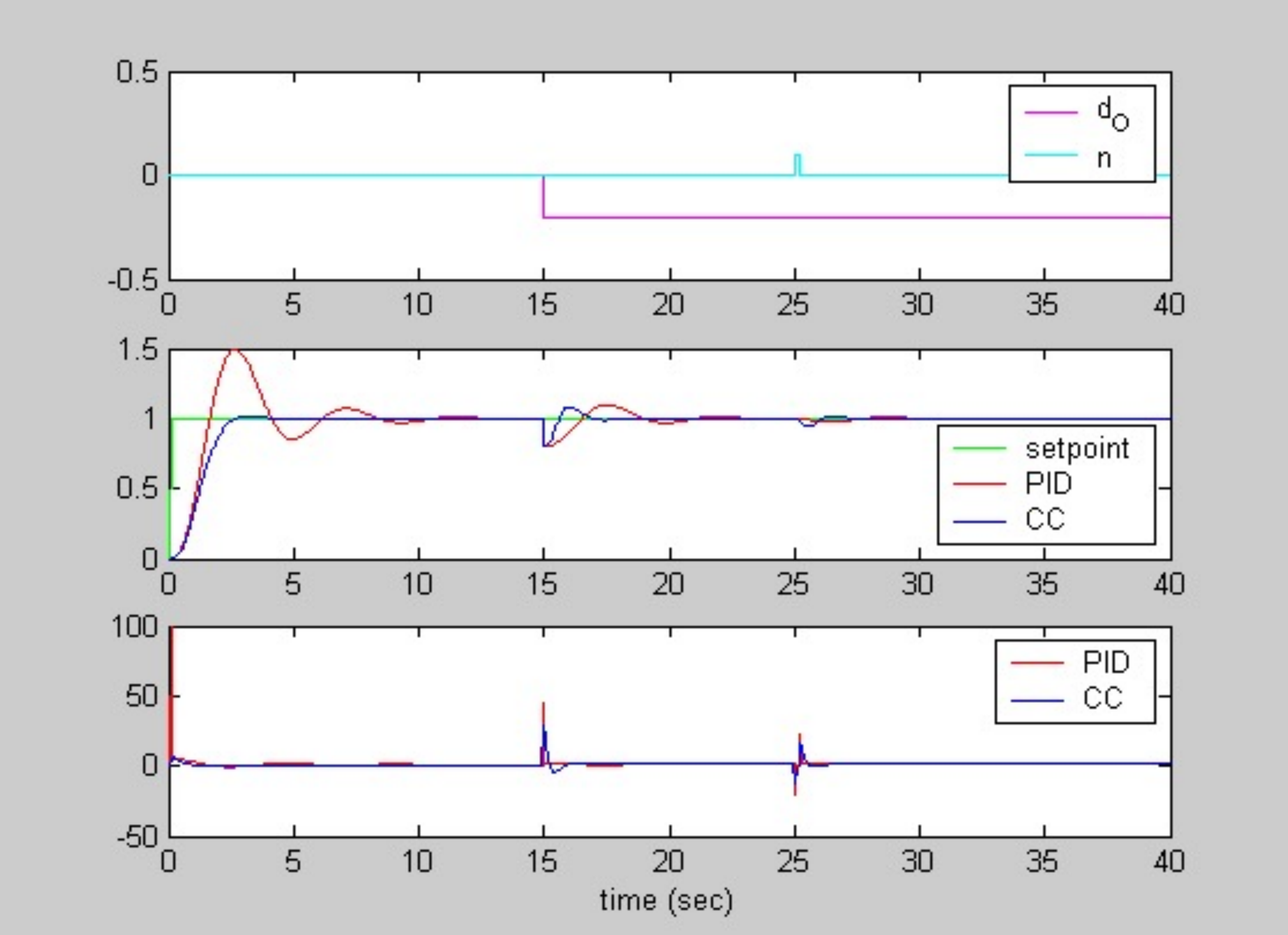
When PID Performs Well

- Essentially first order processes
  - For 1st order P control is state feedback
  - PI control is actually sufficient; D not needed
- Essentially second order processes
  - For 2nd order PD control is state feedback
- Derivative action is beneficial when
  - Time constants differ in magnitude
  - Tight control of higher order system is required (higher order dynamics prevent the use of high proportional gain; D provides damping and speeds up transcient response.)

When More Sophisticated Control is Needed

- Tight control of higher order processes
- Systems with long delay times
- System with lightly-damped oscillatory modes
- Systems with large parameter variations/high uncertainty
- When the disturbance characteristics are important
- Highly-coupled MIMO systems

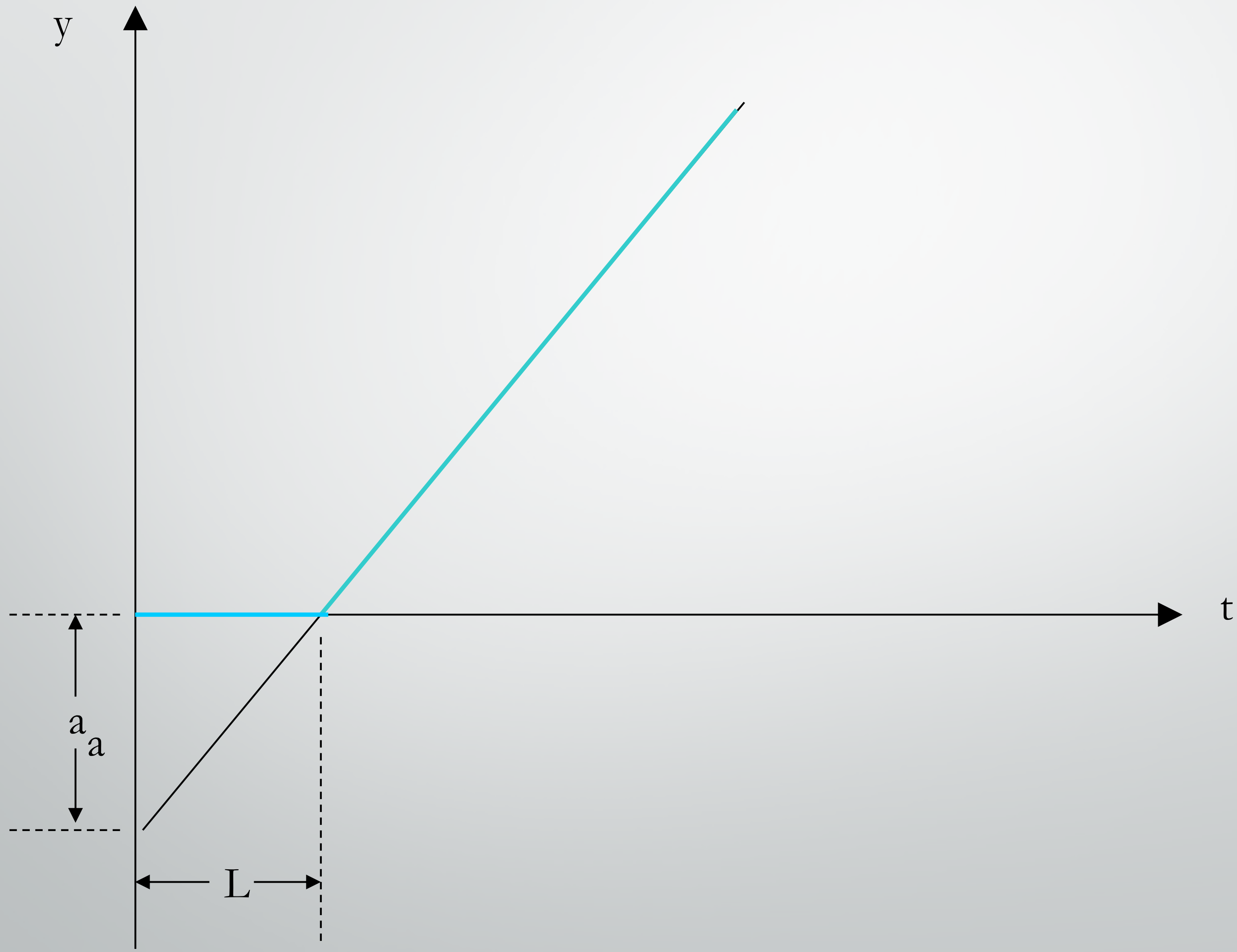
Comparison between PID & more complex controller



PID Tuning

- Ziegler-Nichols
- Relay feedback
- Chein-Hrones-Reswick
- Cohen-Coon
- Kappa-Tau
- Analytical approaches (Haalman's method)
  - Specify T(s) (or L(s) = C(s)P(s)) and solve for C(s)
- Optimization-Based methods

Ziegler-Nichols step response method



PID parameters from Z-N step response method (standard form)

Controller	K	T_i	T_d
P	1/a	-	-
PI	0.9/a	3L	-
PID	1.2/a	2L	L/2

Ziegler-Nichols Frequency Response Method

- Find "ultimate gain" K_u and "ultimate period" T_u
  - Turn off I and D term by setting T_i =inf and T_d=0
  - Increase gain K until output oscillates
  - Record the gain K_u and period T_u
  - Calculate P, I, and D from table

PID parameters from Z-N frequency response method (standard form)

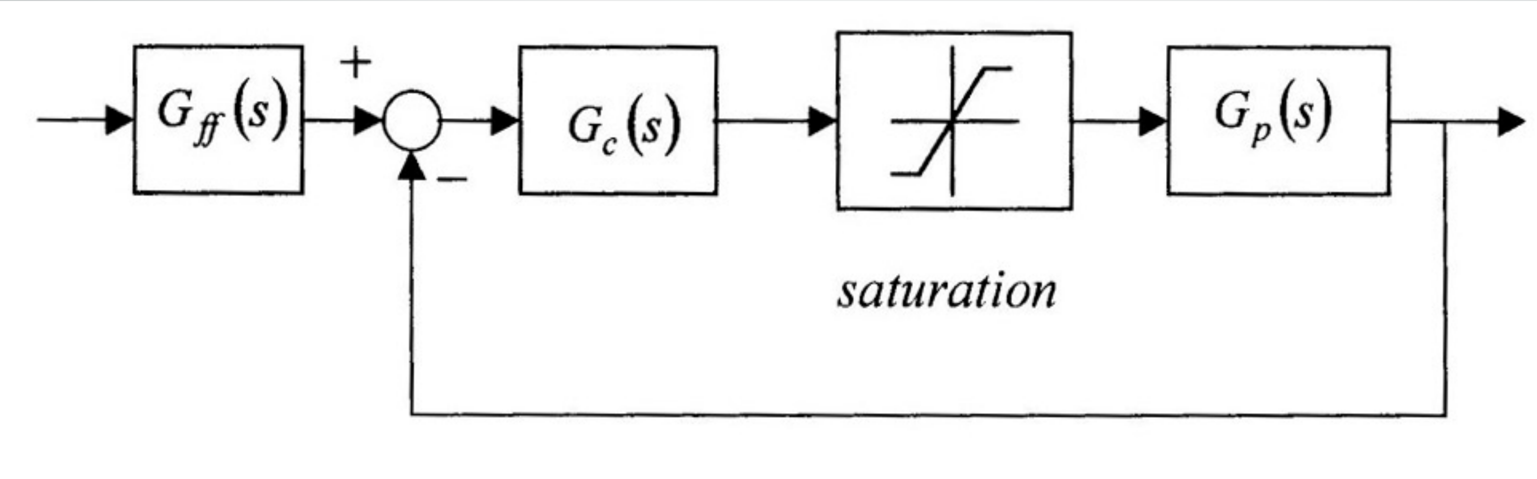
Controller	K	T_i	T_d
P	$0.5K_u$	-	-
PI	$0.4K_u$	$0.8T_u$	-
PID	$0.6K_u$	$0.5T_u$	$0.125T_u$

PID parameters from Z-N frequency response method (parallel form)

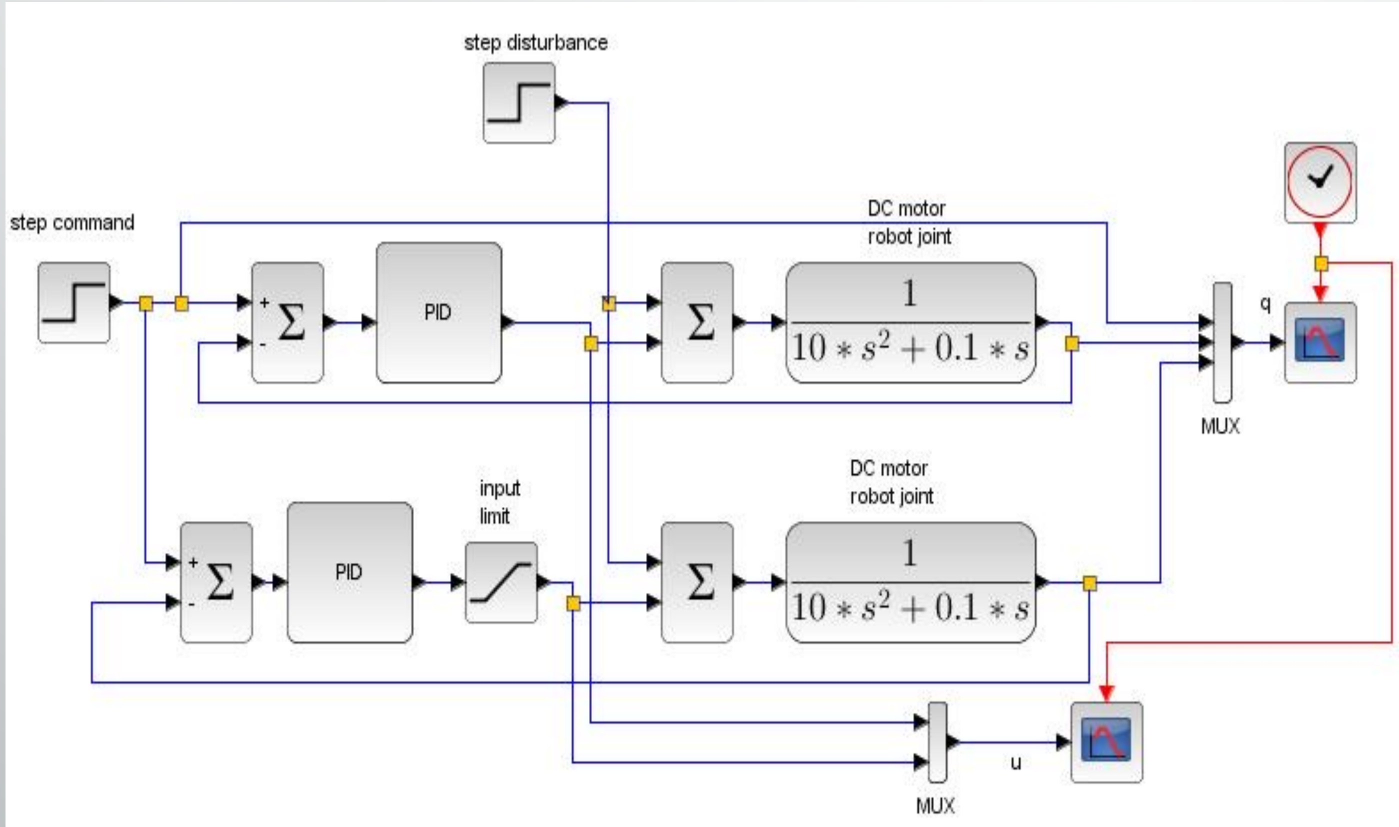
Controller	K_p	K_i	K_d
P	$0.5K_u$	-	-
PI	$0.4K_u$	$0.5K_u/T_u$	-
PID	$0.6K_u$	$1.2K_u/T_u$	$0.075K_uT_u$

Integrator Windup

- When the actuator saturates, the loop is effectively open
- When the controller contains an integrator within, the integrators output will drift higher since the integrator is open-loop unstable. This is referred to as integrator windup
- The error signal needs to change signs before the integrator output will start to return towards zero.
- Thus, the actuator remains saturated even after error changes sign as the controller's output stays high while the integrator output drifts back

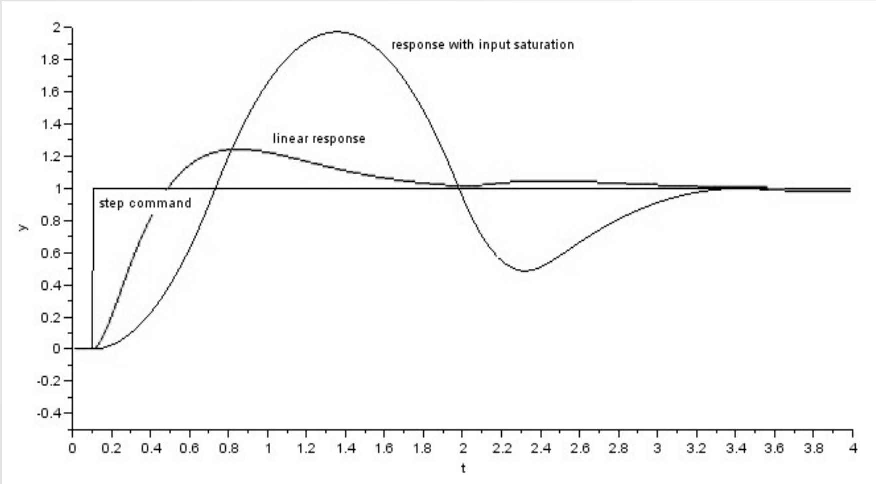


Integrator Windup Example

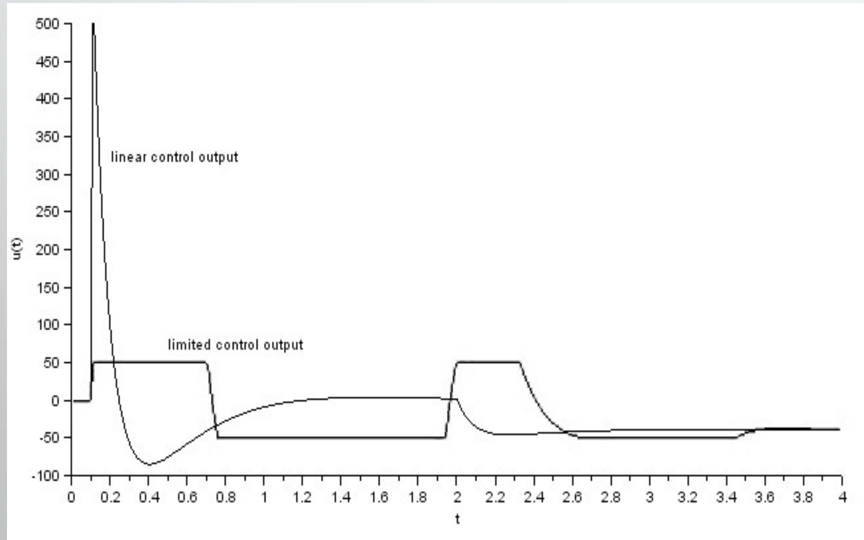


pid_ilim.zcos

Integrator Windup -Simulation Result



Plant output



Controller output

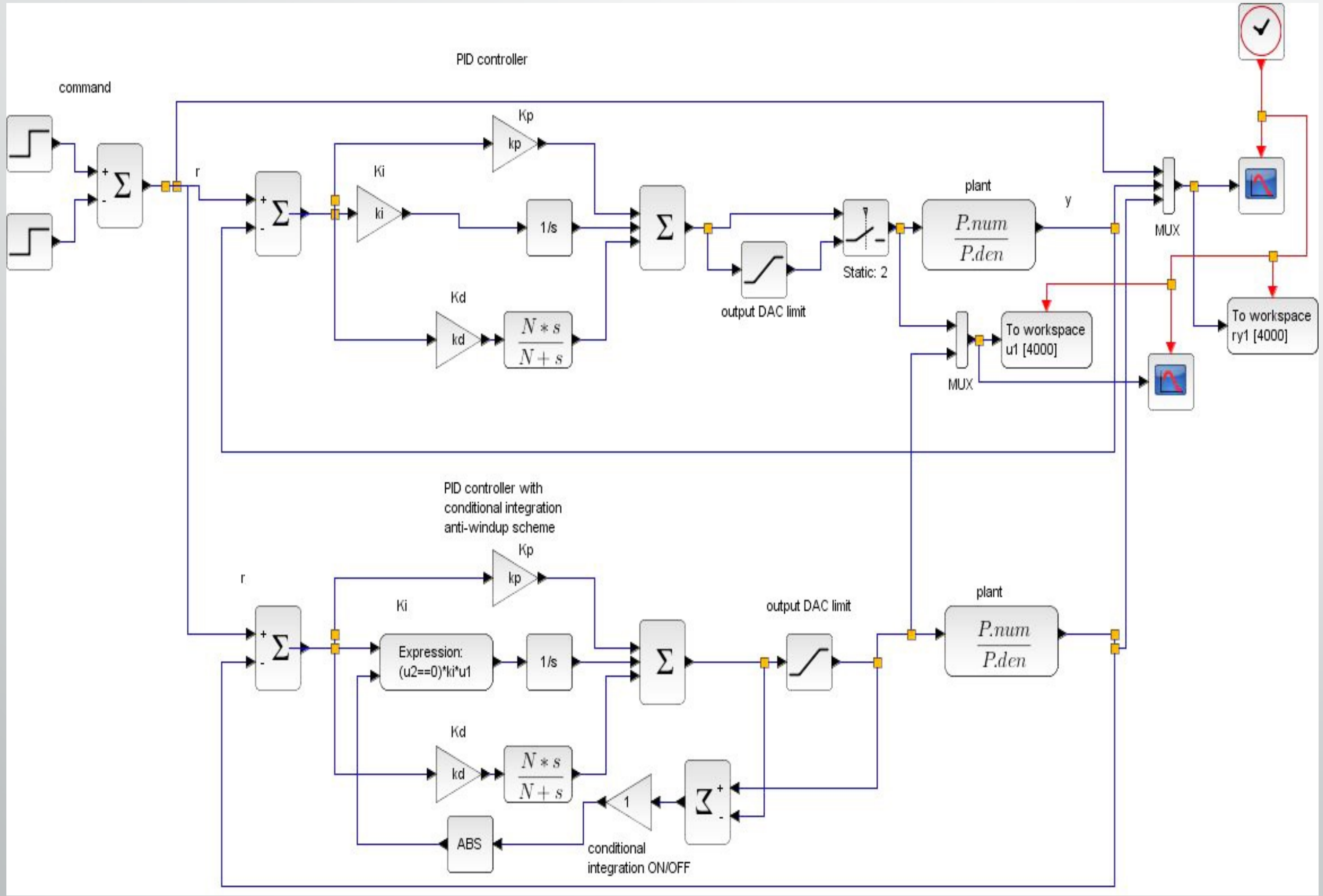
Anti-Windup Methods

- Conditional Integration
  - Integration is switched off
  - May be done in several different manners; e.g.,
    - Switch off when control error is large
    - Switch off when actuator is saturated
- Back Calculation
  - When actuator output saturates, the integral is recomputed such that its output keeps the control at the saturation limit.
  - Actually done through a filter so that anti-windup is not initiated by short periods of saturation; e.g., those induced by noise

Conditional Integration

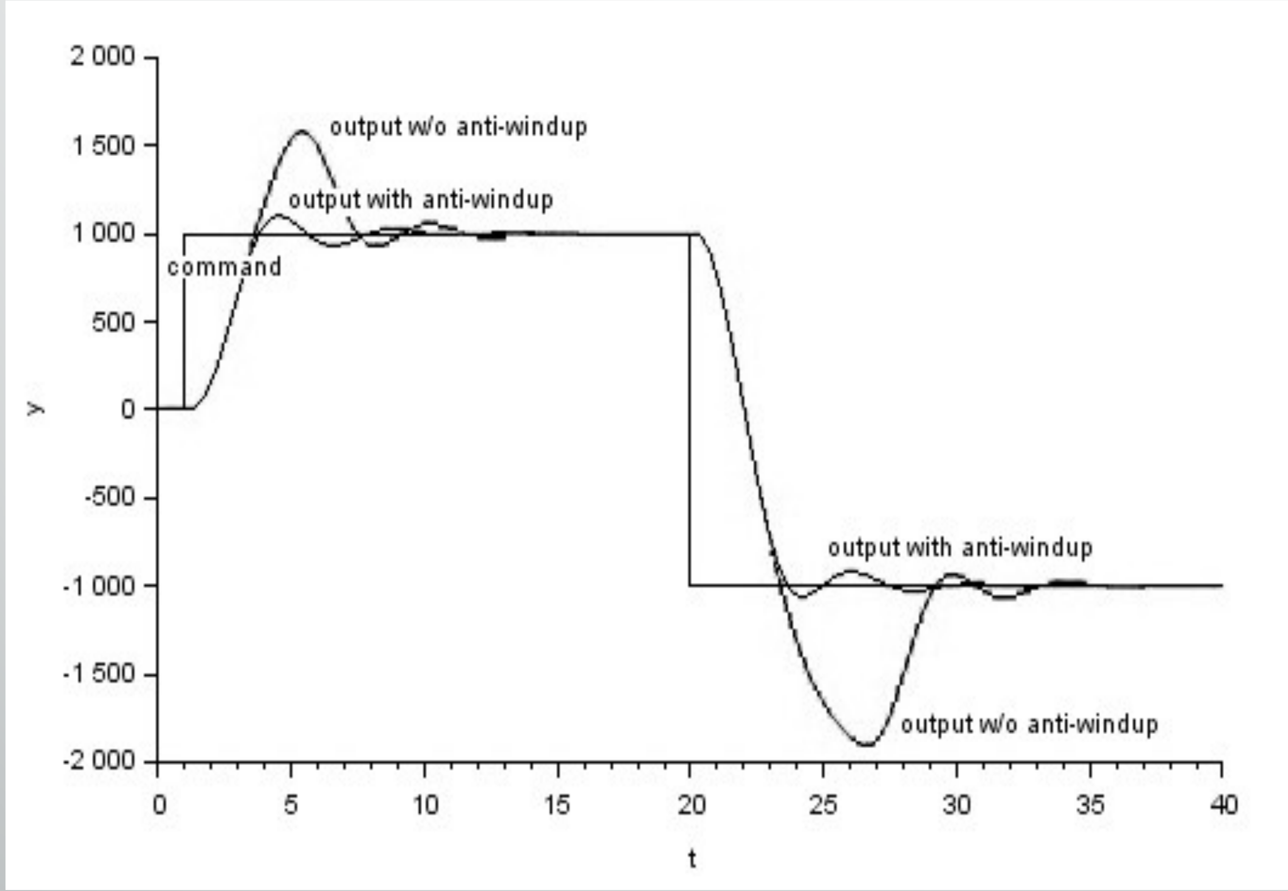
- Integration is switched off
- May be done in several different manners; e.g.,
  - Switch off when control error is large
  - Switch off when actuator is saturated
- Note: both of these techniques can get stuck with non-zero error if the integral term has a large value at the time of switch-off. To avoid this:
  - Switch off when actuator is saturated and the integrator input is such that it causes the control to become more saturated.

Conditional Integration Xcos model

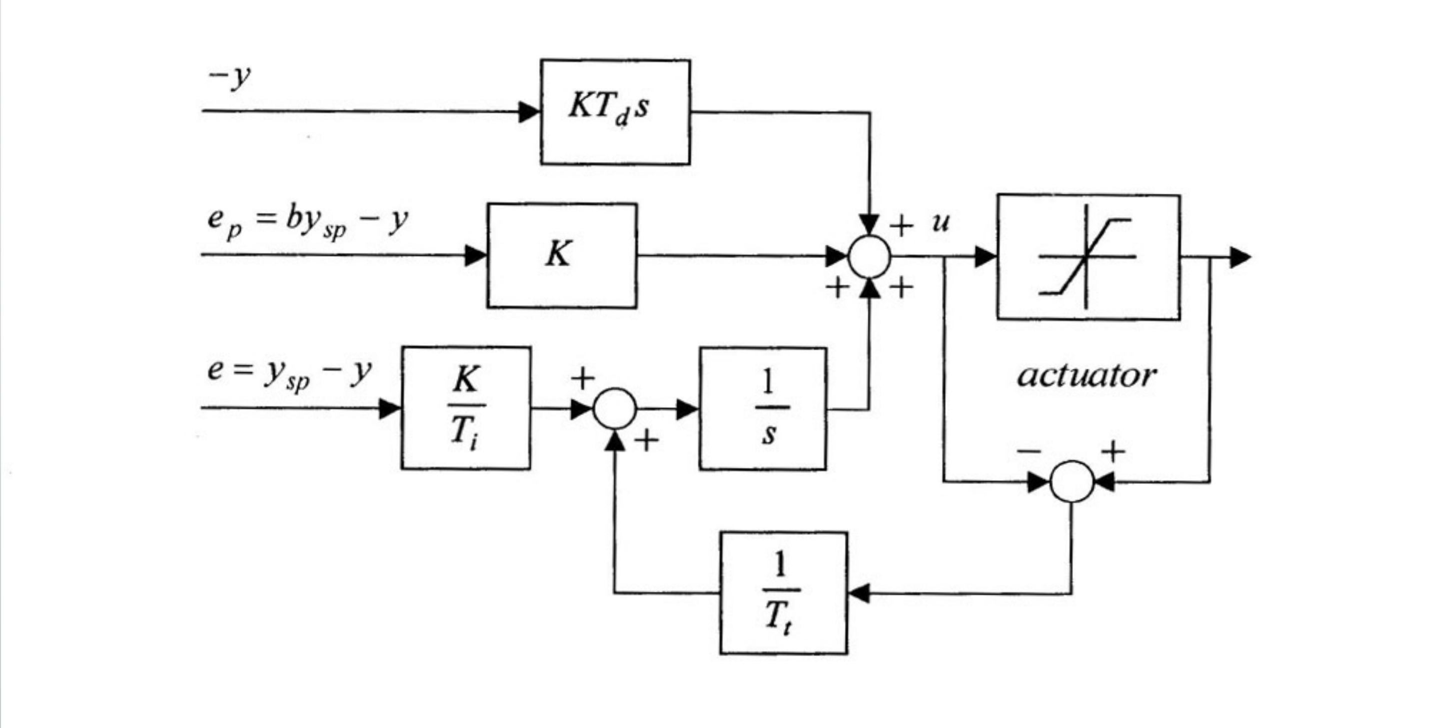


awupid_m1.zcos

Conditional Integration – Simulation Result

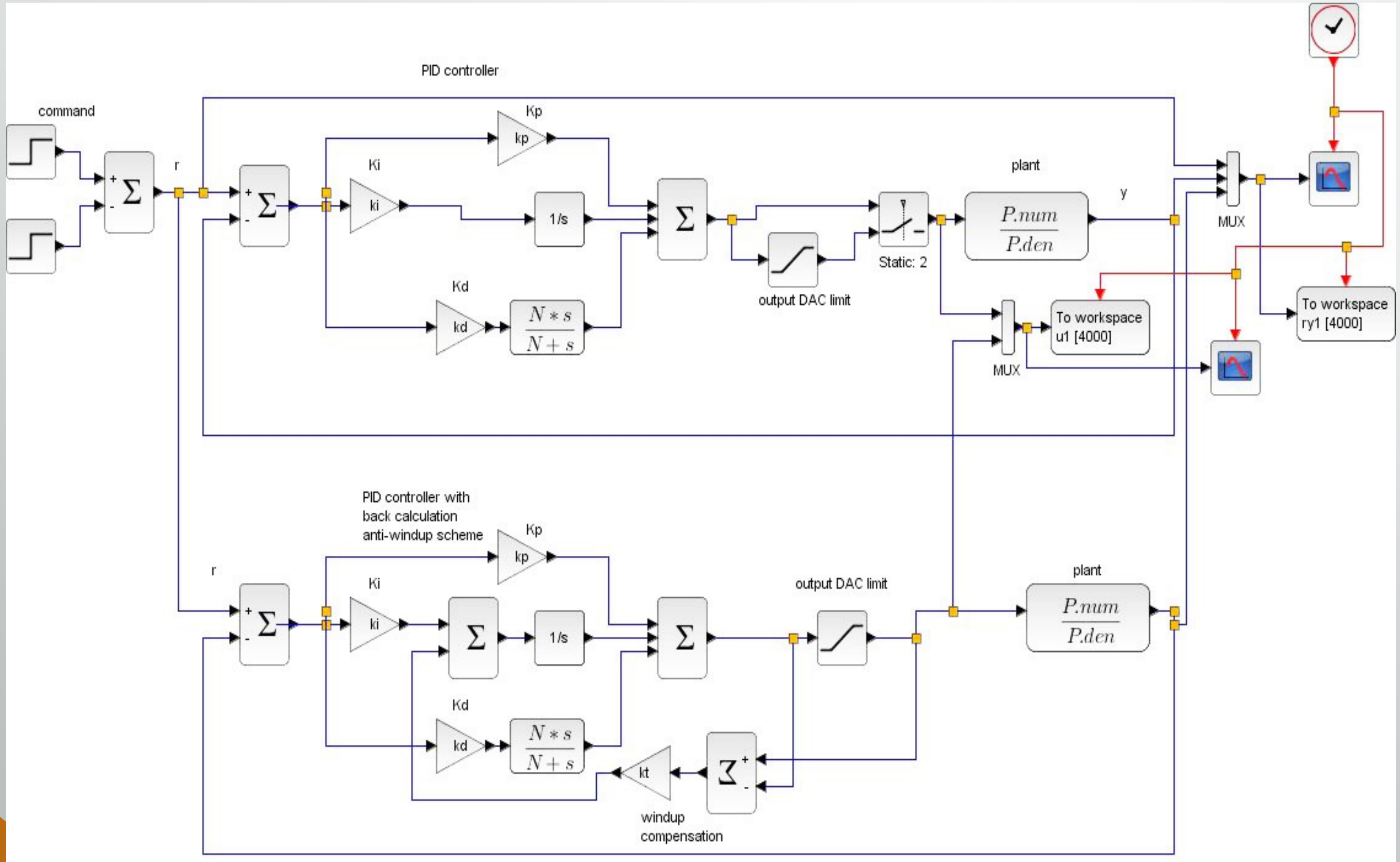


Back Calculation



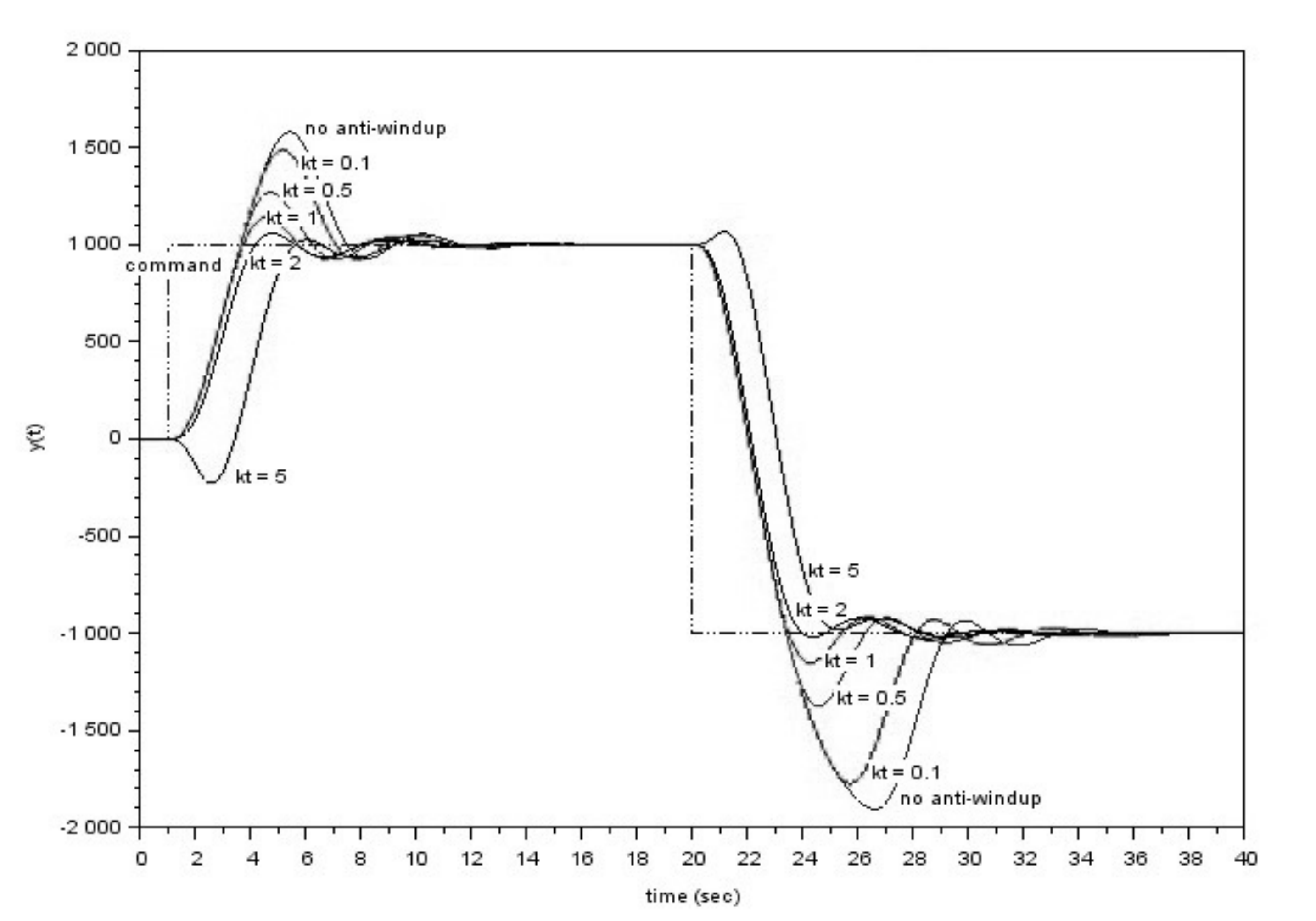
- Can be viewed as supplying a supplementary feedback path around integrator that only becomes active during saturation. This stabilizes the integrator when the main feedback loop is open due to saturation.
- Tracking time constant Ti determines how quickly after saturation the integrator is reset.

Back Calculation Xcos model

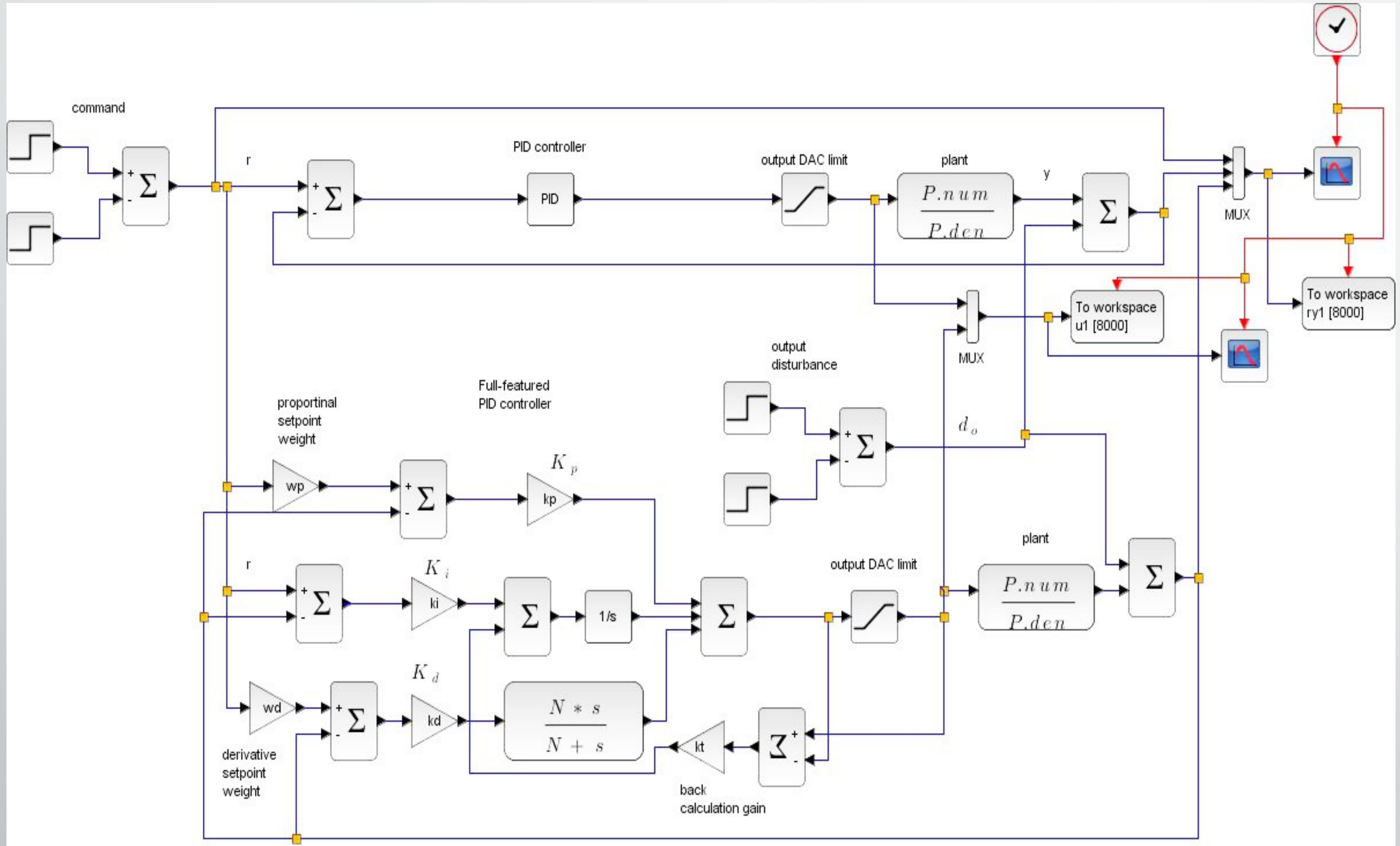


awupid_m2.zcos

Back Calculation – Simulation Results



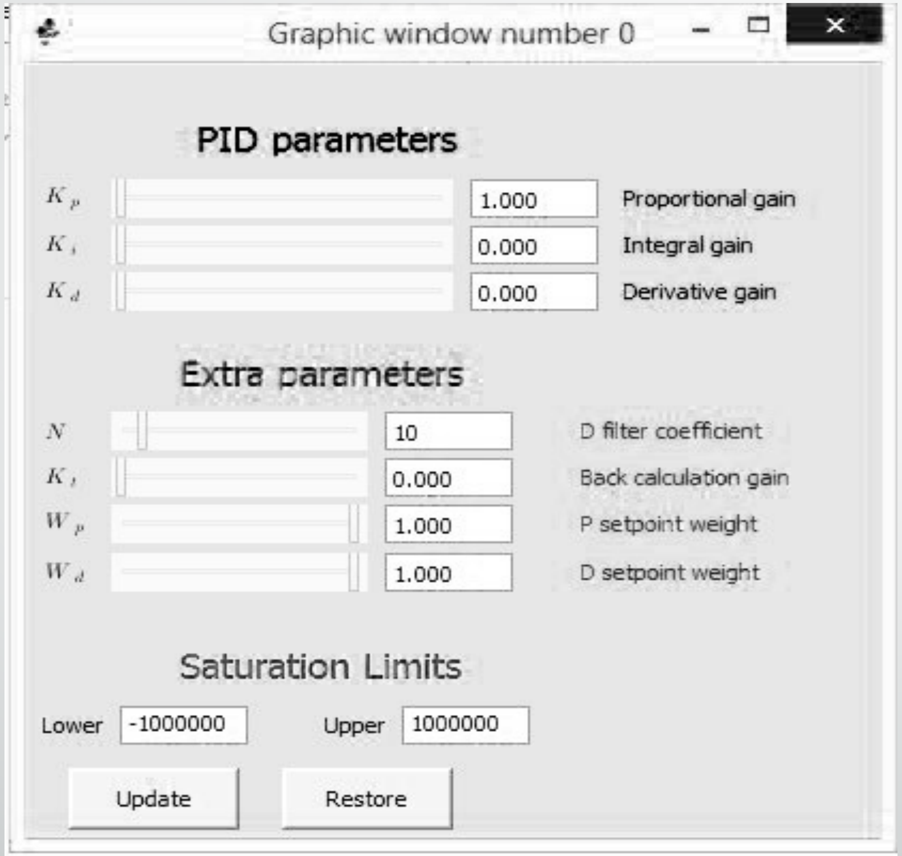
A Full-featured PID Controller



advpid.zcos

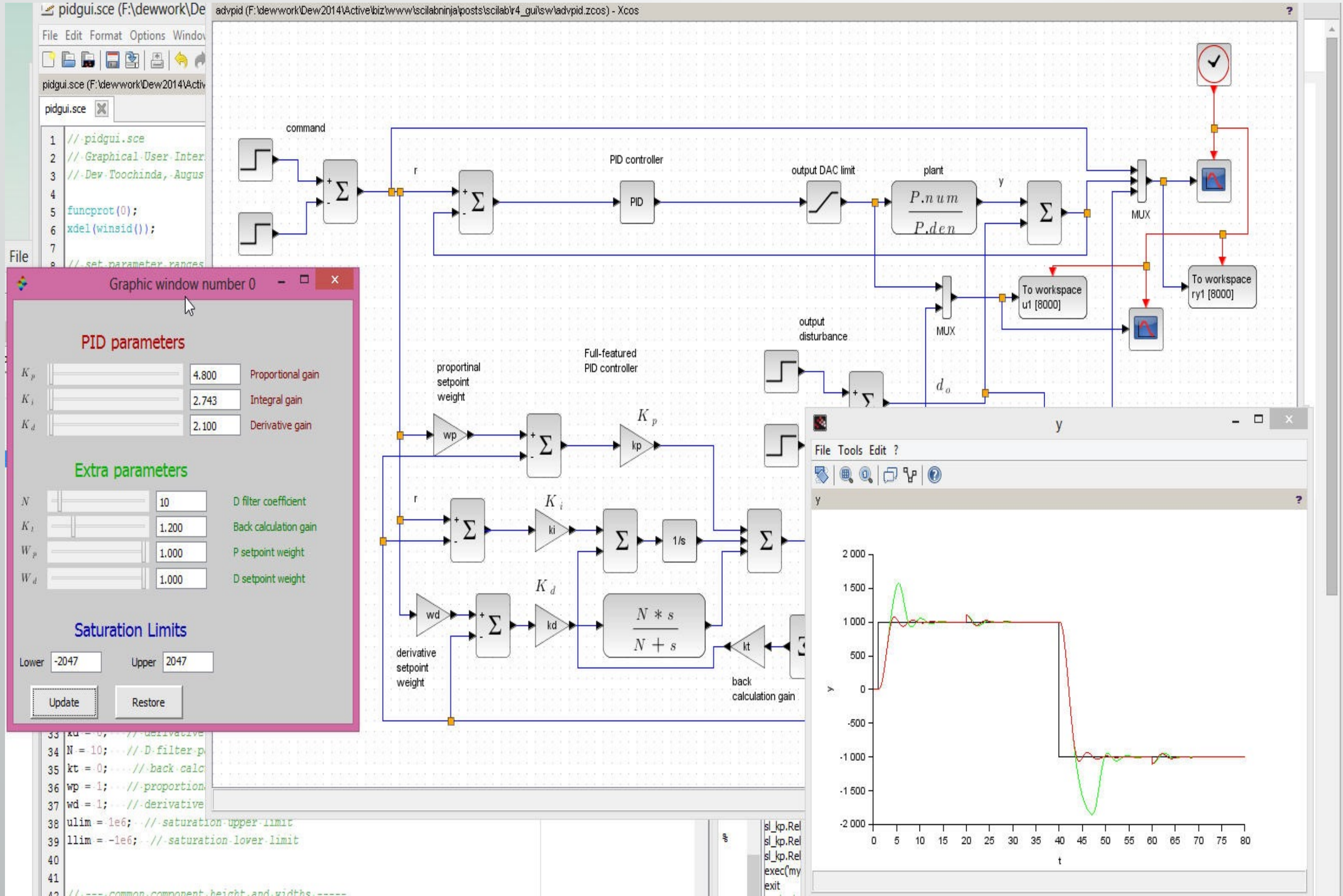
Run setup script advpid.sce

Setup GUI for full-featured PID

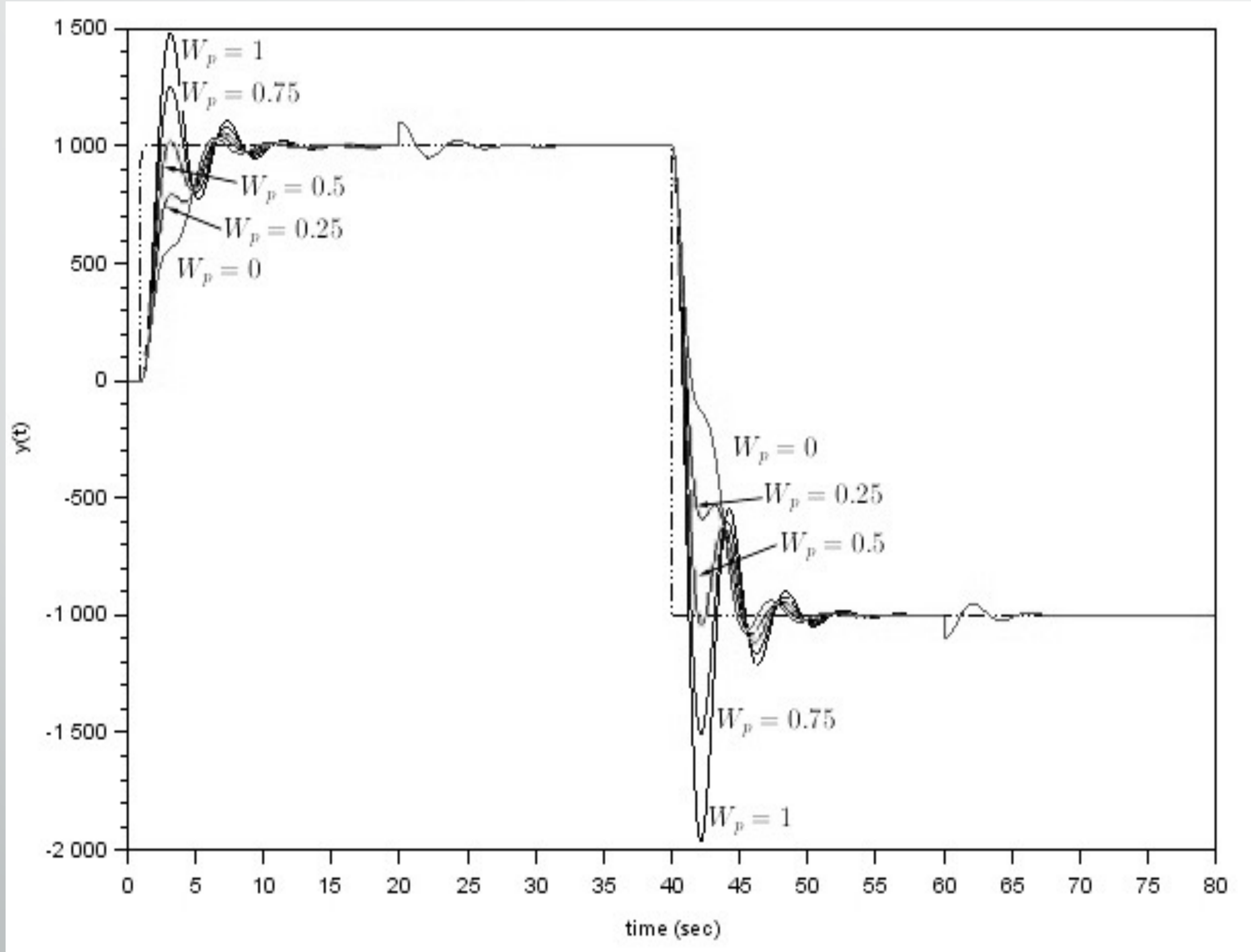


pidgui.sce

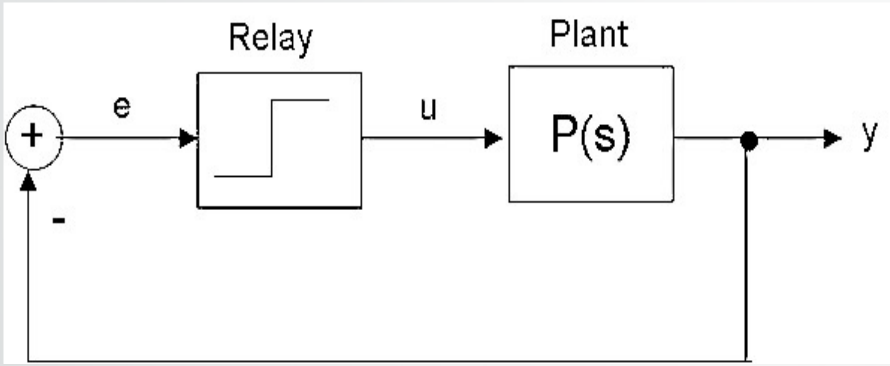
Use `pidgui` to aid in simulation



Step response results



PID Autotuning (relay method)



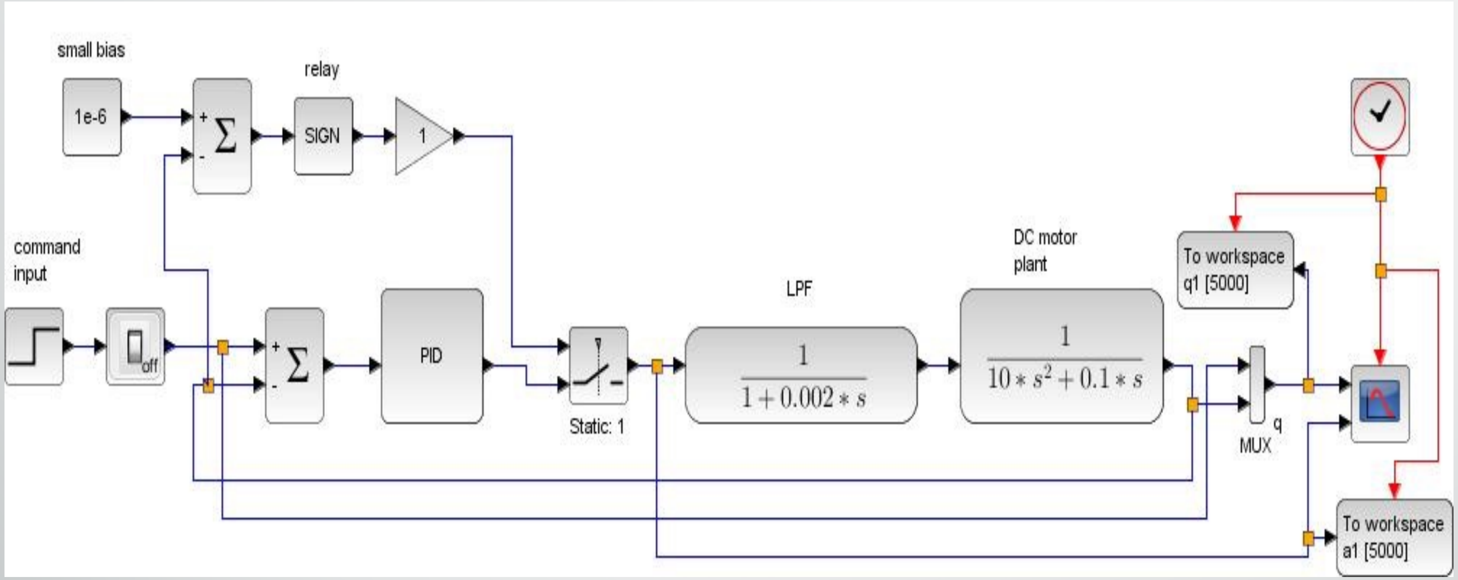
- Based on the manual ZNFD method
- Oscillation is forced by putting a relay in place of controller
- A nonlinear analysis tool called "describing function" is used

$K_u = \quad N(a) = \frac{4d}{\pi a}$

d = magnitude of relay signal
a = magnitude of plant oscillation

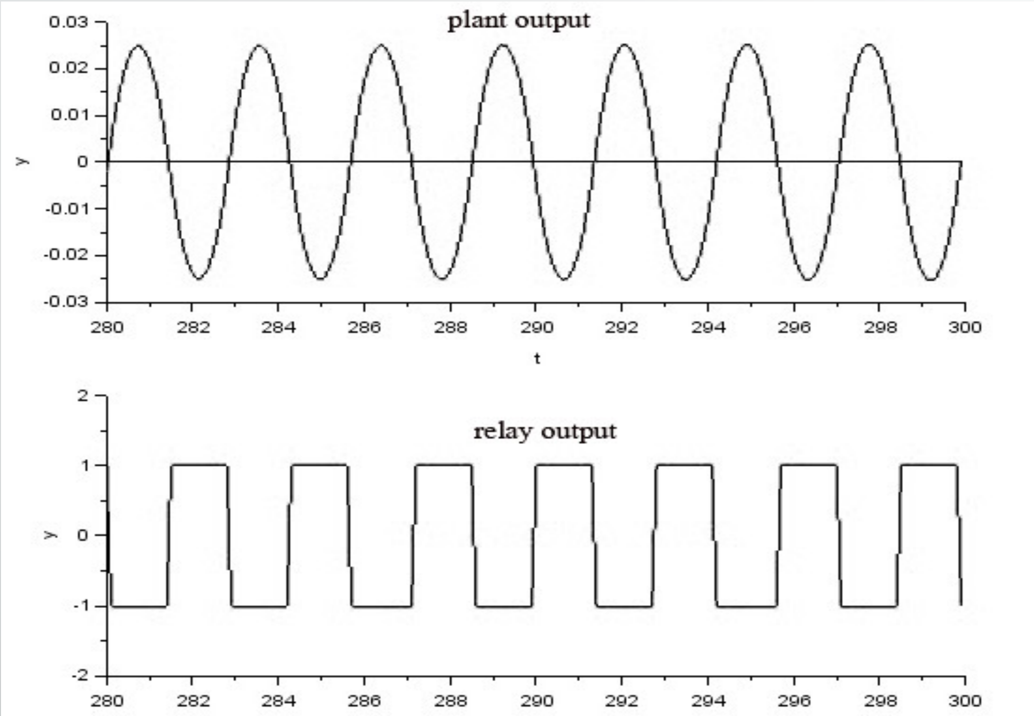
T_u can be determined by software, then use ZNFD tuning rule

PID Autotuning Xcos model



`pid_autotuning.zcos`

Simulation result shows oscillation

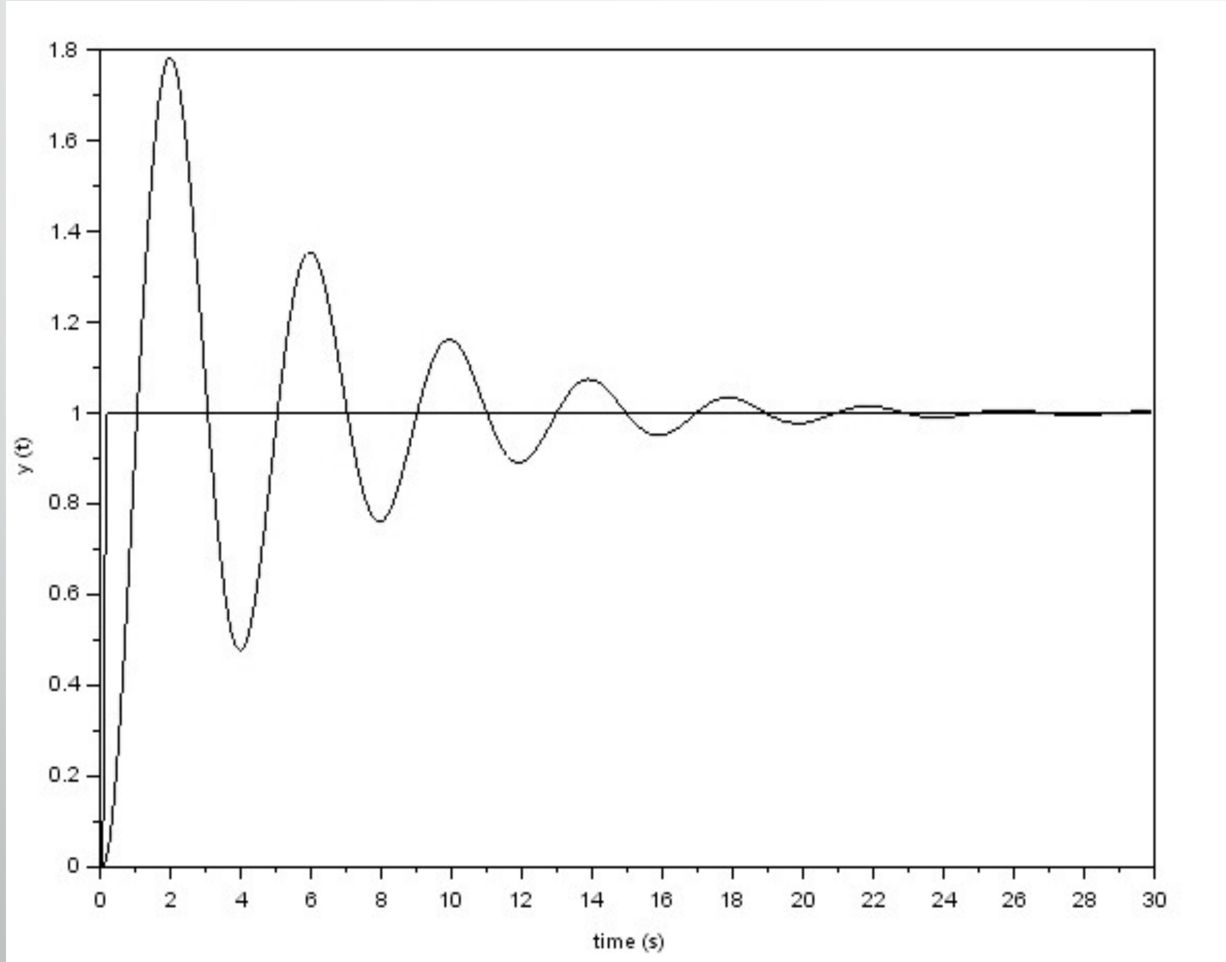


a = 0.025
d = 1

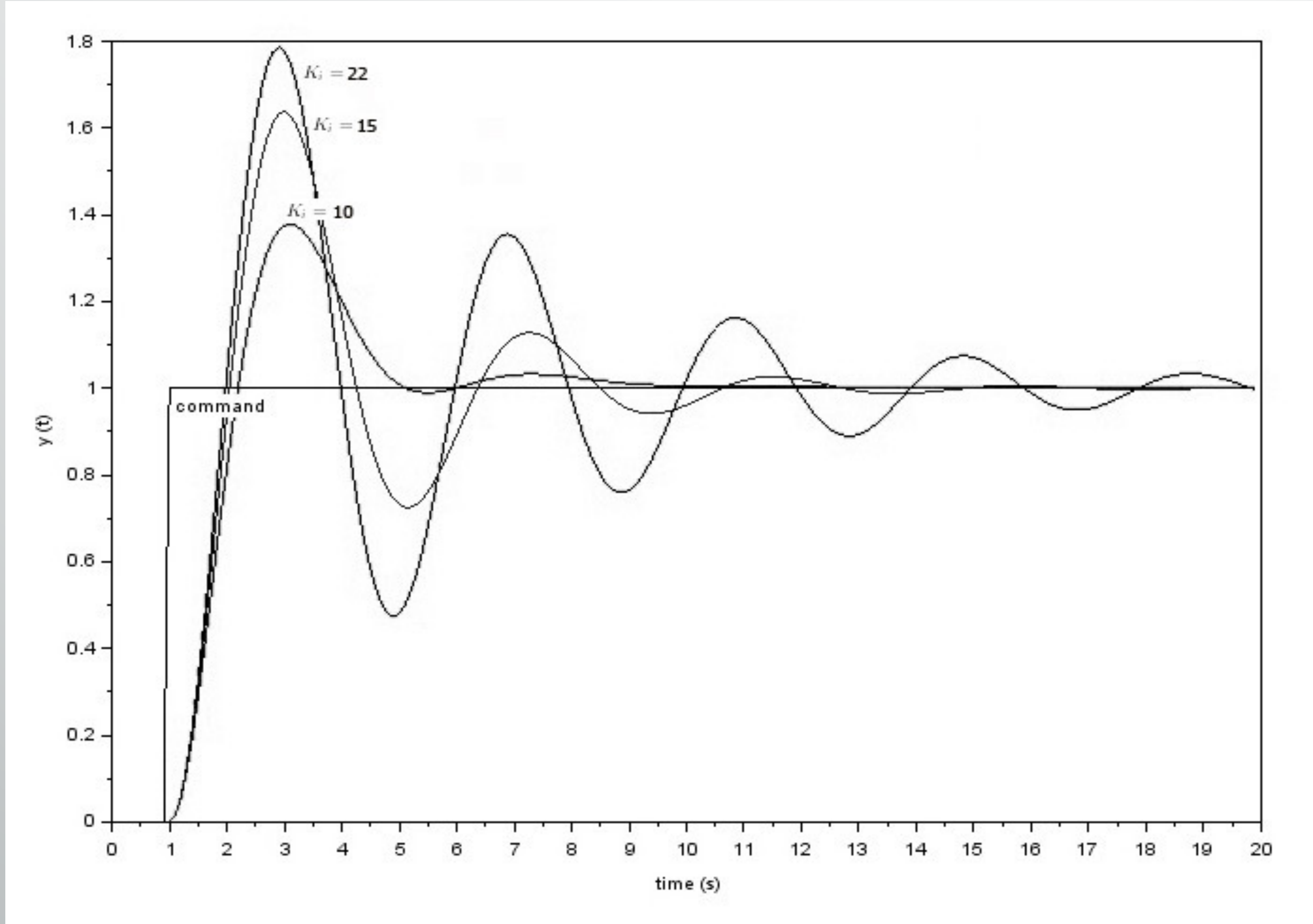
$$K_u = N(a) = 4/\pi a = 51$$

$$T_u = 2.8$$

PID gains from ZNFD table



Reducing Ki for less oscillation



Reference on PID

- Digital PID Controllers, www.controlsystemslab.com
- K. Astrom and T. Hagglund, PID Controllers: Theory, Design, and Tuning, 2nd Edition, Instrument Society of America, 1995.