

In this digital era, PID controllers have evolved from basic textbook structure to more sophisticated algorithms. Features such as setpoint/derivative weightings and anti-windup scheme are often added to improve the closed-loop response. In our previous article [A Decorated PID Controller](#), we consider a PID structure with modification and additional functions as follows

- A feedback diagram with this advanced PID controller is constructed using Xcos palettes as in Figure 1. In equation form, this controller can be described as

with

$$e_d(s) = W_d r(s) - y(s)$$

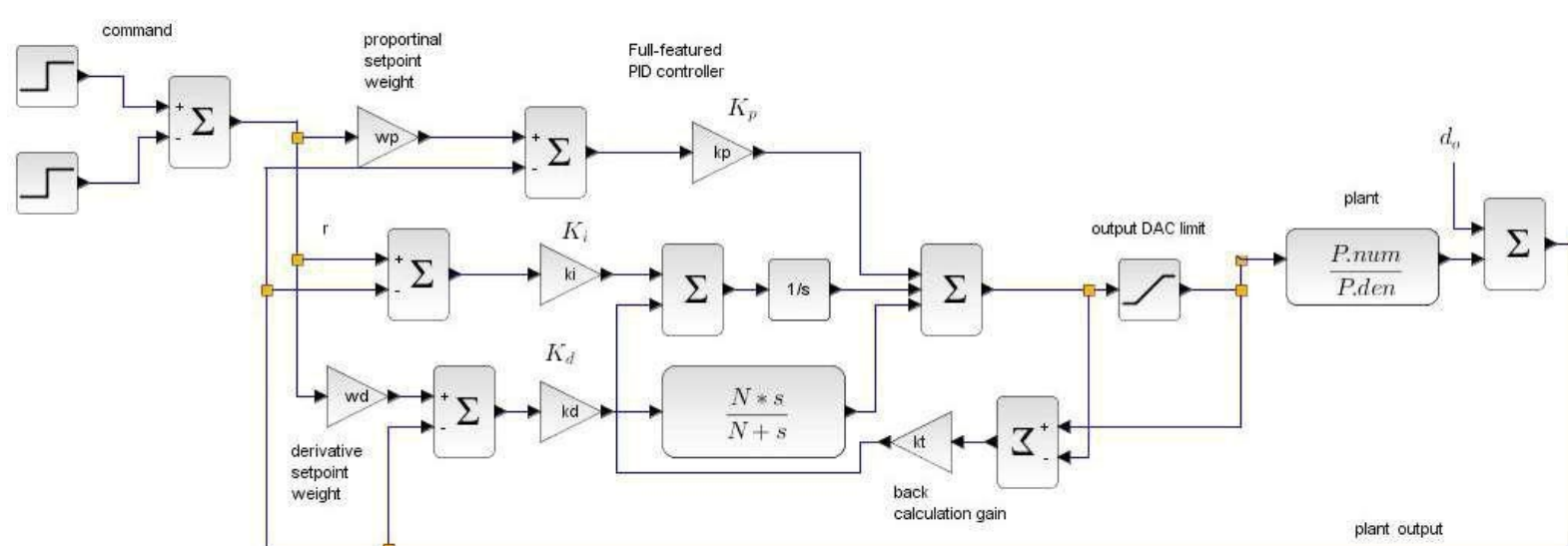


Figure 1 advanced PID feedback diagram

where $r(s)$, $y(s)$, $u(s)$ and $u_{sat}(s)$, are reference command, plant output, controller output, and saturated controller output, respectively. As described in our Discrete-time PID implementation article, using backward difference relationship

Equation (1) can be converted to z-domain as

$$u(z) = K_p e_p(z) + \frac{K_i T_s z}{z-1} e(z) + \frac{K_t T_s z}{z-1} e_{us}(z) + \frac{K_d N(z-1)}{(1+NT_s)z-1} K_d e_d(z) \quad (3)$$

Rewrite (3) in terms of z^{-1}

$$u(z) = K_p e_p(z) + \frac{K_i T_s}{1-z^{-1}} e(z) + \frac{K_t T_s}{1-z^{-1}} e_{us}(z) + \frac{K_d N(1-z^{-1})}{1+NT_s-z^{-1}} K_d e_d(z) \quad (4)$$

To implement this PID scheme as a computer algorithm, we have to convert (4) to a difference equation. It is straightforward to show that (4) can be rewritten as

$$\begin{aligned} u(z) = & a_1 z^{-1} u(z) + a_2 z^{-2} u(z) + b_1 e_p(z) + b_2 z^{-1} e_p(z) + b_3 z^{-2} e_p(z) + c_1 e(z) + c_2 z^{-1} e(z) \\ & + c_3 e_{us}(z) + c_4 z^{-1} e_{us}(z) + d_1 e_d(z) + d_2 z^{-1} e_d(z) + d_3 z^{-2} e_d(z) \end{aligned} \quad (5)$$

with coefficients

$$\begin{aligned} a_1 &= \frac{2+NT_s}{1+NT_s}, & a_2 &= -\frac{1}{1+NT_s}, & b_1 &= K_p, & b_2 &= -K_p \frac{2+NT_s}{1+NT_s}, \\ b_3 &= \frac{K_p}{1+NT_s}, & c_1 &= K_i T_s, & c_2 &= -\frac{K_i T_s}{1+NT_s}, & c_3 &= K_t T_s, \\ c_4 &= -\frac{K_t T_s}{1+NT_s}, & d_1 &= \frac{K_d N}{1+NT_s}, & d_2 &= -\frac{2K_d N}{1+NT_s}, & d_3 &= \frac{K_d N}{1+NT_s} \end{aligned}$$

So the corresponding difference equation is

$$\begin{aligned} u(n) = & a_1 u(n-1) + a_2 u(n-2) + b_1 e_p(n) + b_2 e_p(n-1) + b_3 e_p(n-2) + c_1 e(n) + c_2 e(n-1) \\ & + c_3 e_{us}(n) + c_4 e_{us}(n-1) + d_1 e_d(n) + d_2 e_d(n-1) + d_3 e_d(n-2) \end{aligned} \quad (6)$$

Response Comparison via Simulation

Equation (6) is ready for implementation on a target processor. Before that phase, we want to make sure that our equation and coefficients are without error. One easy way is to perform simulation on Xcos and compare the response to the original continuous-time PID controller. For this purpose, we construct a model advpid_imp.zcos as shown in Figure 2, consisting of 2 feedback loops. The upper loop is controlled by discrete-time PID in the form (5), and the lower loop contains the continuous-time PID. The simulation results from the two closed-loop systems are then compared to verify how well they match.



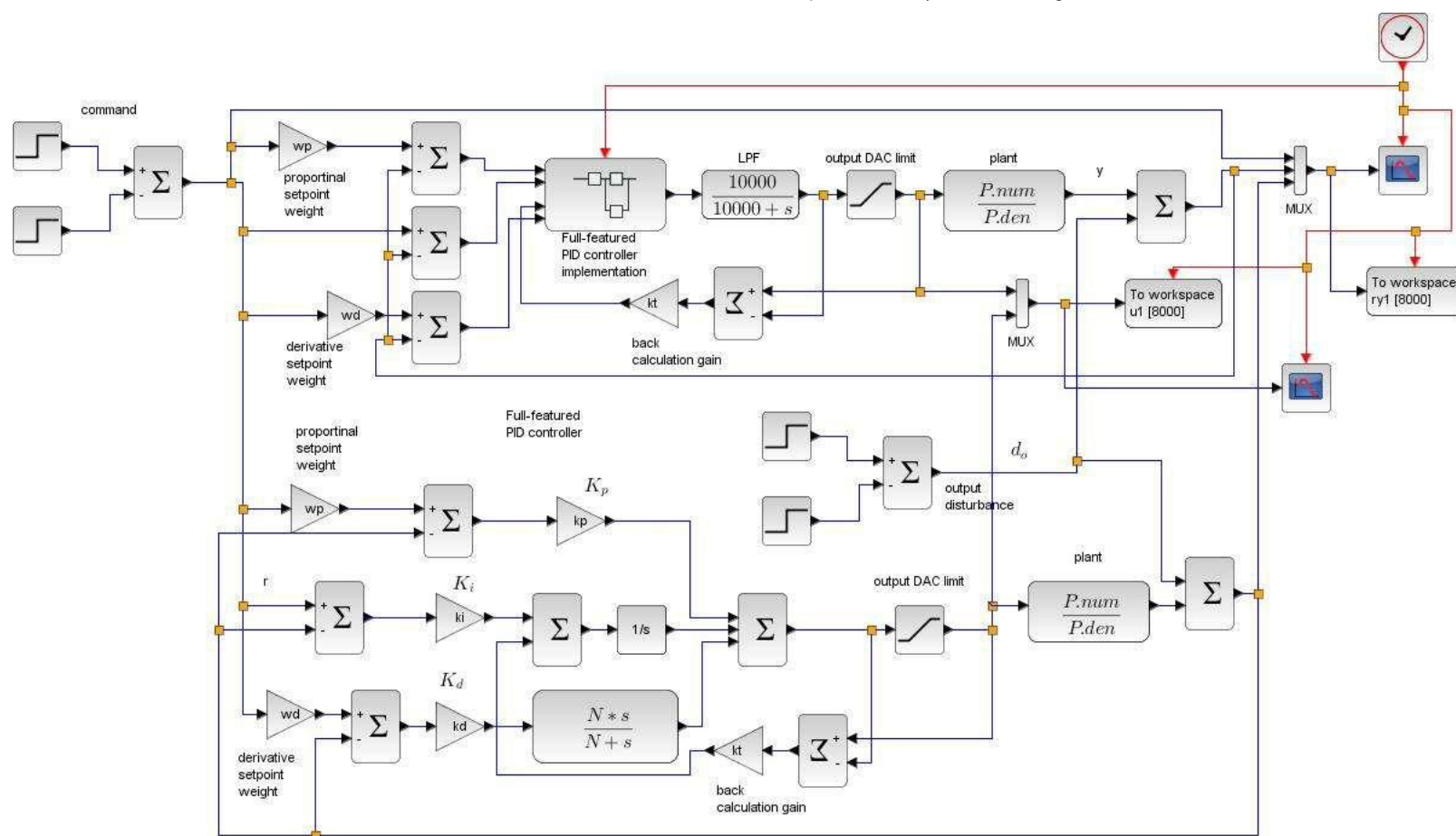


Figure 2 model advpid_imp.zcos for discrete and continuous PID comparison

Note that the discrete-time PID in the upper loop is contained in a superblock. The internal details are shown in Figure 3, which corresponds to the discrete-time controller (5).

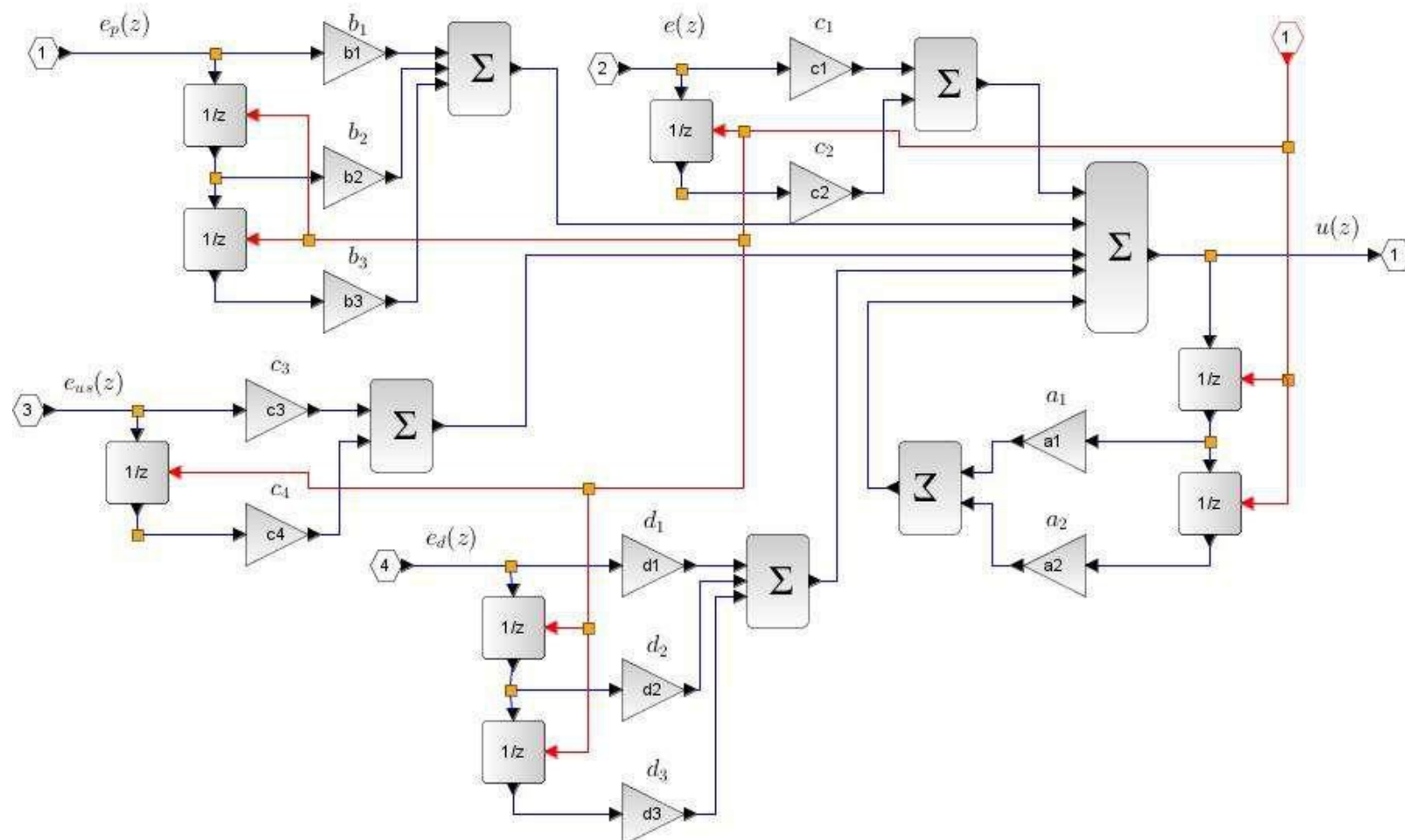


Figure 3 the internal details of discrete-time PID superblock

Also, at the output of discrete-time PID controller, a LPF transfer function is inserted to prevent an algebraic loop error, normally occurred with hybrid simulation. The LPF pole is chosen well above the closed-loop bandwidth so the filter does not have noticeable effect on the responses.

For easy editing, all the parameters in advpid_imp.zcos are initialized using a script file advpid_imp.sce. The plant is chosen as a third-order lag transfer function

$$P(s) = \frac{1}{(s+1)^3} \quad (7)$$

which can be created in Scilab by

```
s = poly(0, 's');
P = syslin('c', 1/(s+1)^3);
```



Then, controller parameters are assigned values. These can be chosen as you like since the purpose of this simulation is to compare the responses. Here the PID gains are obtained from Zieglers Nichols frequency domain tuning method, and others are assigned some practical values.

<code>kp = 4.8;</code>	<code>// PID gains</code>
<code>ki = 2.7;</code>	
<code>kd = 2.1;</code>	
<code>N = 10;</code>	<code>// filter coefficient</code>
<code>kt = 1.2;</code>	<code>// back calculation gain for anti-windup</code>
<code>wp = 0.7;</code>	<code>// setpoint weight for proportional term</code>
<code>wd = 0.1;</code>	<code>// setpoint weight for derivative term</code>
<code>Ts = 0.01;</code>	<code>// sampling peroid</code>

For sampling period Ts, the value should match the simulation sampling period in the clock.

The parameters left to be assigned are the limits in saturation block. Put in some reasonable values such that some saturation effect happens during transient, since we prefer response comparison with the back calculation term activated. Too small the limit range would cause severe performance degradation. By some trial and error, we are finally satisfied with these values for saturation block

```
ulim = 2000;

llim = -2000;
```

Finally, the coefficients in (5) need to be computed. We introduce additional variables x1 and x2 for terms that appear in several places.



```
x1      = (1+N*Ts);  
x2      = (2+N*Ts);  
a1      = x2/x1;  
a2      = -1/x1;  
b1      = kp;  
b2      = -kp*x2/x1;  
b3      = kp/x1;  
c1      = ki*Ts;  
c2      = -ki*Ts/x1;  
c3      = kt*Ts;  
c4      = -kt*Ts/x1;  
d1      = kd*N/x1;  
d2      = -2*kd*N/x1;  
d3      = kd*N/x1;
```

After all parameters are assigned, interactively or by executing advpid_imp.sce, we proceed by clicking on the simulation start button. The simulation results in Figure 4 show that the plant and controller outputs from continuous and discrete PID are almost identical. This makes us confident that the discrete-time PID and its coefficients are derived correctly.

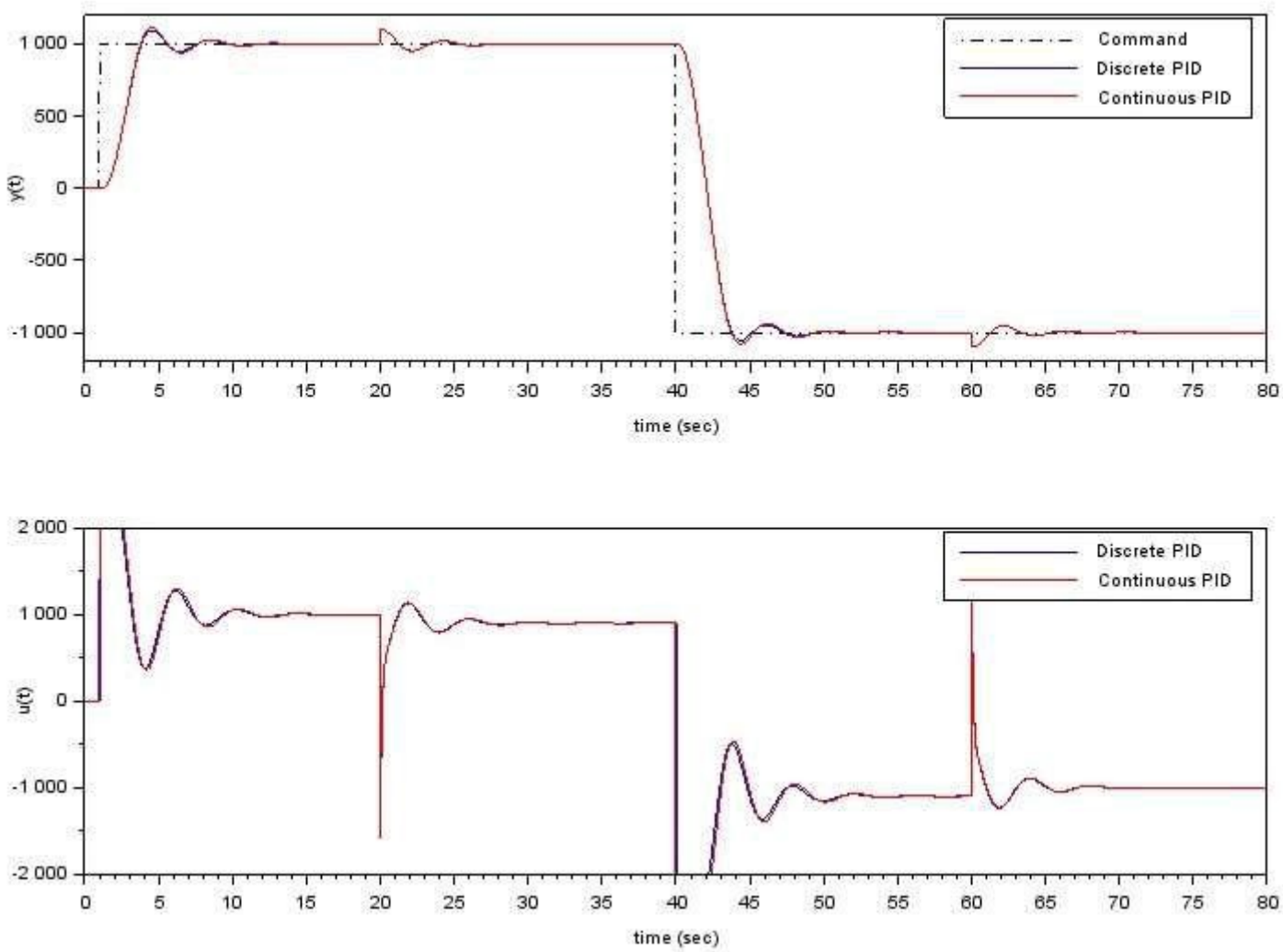


Figure 4 response comparison between continuous and discrete PID



[Download](#)

[Tutorials](#)

[Industries](#)


[Technology](#)

[Services](#)

[Software](#)

[Cloud](#)

[About](#)



WORK WITH US

Email: team@scilab.io

Web: <http://scilab.io/company/careers/>

3 bis rue Saarinen

94528 Rungis - France

ESI Group - 2020

[Cookie settings](#) | [Privacy & Terms Of Use](#)

[Legal notice](#) | [Donate](#)


Tweets by [@Scilab](#)



Scilab

@Scilab

From statistical analysis of the cases by country to a basic model of the spread of the coronavirus. A few elements to understand the dynamic of the situation from an epidemiological point of view:scilab.org/coronavirus-sp...





Apr 28, 2020



Scilab

@Scilab

Statistics - Some Theory
youtu.be/UaLGxwjS50A via [@YouTube](#)

 YouTube

@YouTube



Embed

View on Twitter

https://www.scilab.org/advanced-pid-controller-implementation

6/6