PID Controller with Second Order Filtering of Mesured Signal

K. J. Åström

1. Introduction

There are good reasons to filter the measured signal by a second order filter instead of the first order filter that is traditionally used in PID control. Measurement noise will generate less control activity and the robustness is performed because of the improved high-frequency roll-off. In this note we give an impementation that is also very efficient, the computations increase only marginally.

2. Filtering

Let the measured signal be y and let y_f denote the filtered signal. For a second order filter with time constant T_f and relative damping $\zeta = 0.707$ we have the following relation between the Laplace transforms of the signals.

$$Y_f = \frac{1}{1 + sT_f + (sT_f)^2/2} Y.$$

Hence,

$$(1 + sT_f + (sT_f)^2/2) Y_f = Y \tag{1}$$

Introducing the state variables

$$\begin{aligned} X_1 &= Y_f \\ X_2 &= sT_f Y_f, \end{aligned} \tag{2}$$

we find

$$\begin{aligned} sT_f X_1 &= X_2 \\ sT_f X_2 &= (sT_f)^2 Y_f = 2(Y - Y_f - sT_f Yf) \end{aligned}$$

where the last equality follows from (1). Transforming to the time domain we get

$$\begin{aligned} T_f \frac{dx_1}{dt} &= x_2 \\ T_f \frac{dx_2}{dt} &= -2x_1 - 2x_2 + 2y \end{aligned} \tag{3}$$

Approximating the derivative with a backward difference we find

$$\begin{aligned}(T_f/h)(x_1 - x_1^-) &= x_2\\ (T_f/h)(x_2 - x_2^-) &= -2x_1 - 2x_2 + 2y\end{aligned}$$

where x_k denotes the current value and x_k^- the value at the previous sampling instant. Solving for x_1 and x_2 gives

$$\begin{aligned}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} &= \begin{pmatrix} 1 & -h/T_f \\ 2h/T_f & 1+2h/T_f \end{pmatrix}^{-1} \left(\begin{pmatrix} x_1^- \\ x_2^- \end{pmatrix} + \begin{pmatrix} 0 \\ 2(h/T_f)y \end{pmatrix} \right)\\ &= \frac{1}{det} \begin{pmatrix} 1+2h/T_f & h/T_f \\ -2h/T_f & 1 \end{pmatrix} \left(\begin{pmatrix} x_1^- \\ x_2^- \end{pmatrix} + \begin{pmatrix} 0 \\ 2(h/T_f)y \end{pmatrix} \right)\\ &= \frac{1}{det} \begin{pmatrix} (1+\frac{2h}{T_t})x_1^- + \frac{h}{T_f}x_2^- + 2\left(\frac{h}{T_f}\right)^2 y \\ -\frac{2h}{T_f}x_1^- + x_2^- + 2\frac{h}{T_f}y \end{pmatrix}\end{aligned}$$

where

$$det = 1 + \frac{2h}{T_f} + 2\left(\frac{h}{T_f}\right)^2$$

Simplification of the expressions give

$$\begin{aligned}x_1 &= \frac{T_f^2 + 2hT_f}{T_f^2 + 2hT_f + 2h^2}x_1^- + \frac{hT_f}{T_f^2 + 2hT_f + 2h^2}x_2^- + \frac{2h^2}{T_f^2 + 2hT_f + 2h^2}y\\ &= x_1^- + \frac{hT_f}{T_f^2 + 2hT_f + 2h^2}x_2^- + \frac{2h^2}{T_f^2 + 2hT_f + 2h^2}(y - x_1^-)\\ x_2 &= -\frac{2hT_f}{T_f^2 + 2hT_f + 2h^2}x_1^- + \frac{T_f^2}{T_f^2 + 2hT_f + 2h^2}x_2^- + \frac{2hT_f}{T_f^2 + 2hT_f + 2h^2}y\\ &= \frac{T_f^2}{T_f^2 + 2hT_f + 2h^2}x_2^- + \frac{2hT_f}{T_f^2 + 2hT_f + 2h^2}(y - x_1^-)\end{aligned} \tag{4}$$

The estimated rate of change of process output is

$$\frac{dy_f}{dt} \approx \frac{x_2}{T_f} \tag{5}$$

3. Rescaling the State Variables

It is hihgly desirable to have an algorithme that permits $T_f = 0$. To achieve this we introduce new state variables

$$\begin{aligned}y_1 &= x_1\\ y_2 &= \frac{h}{T_f}x_2 \approx h\frac{dy_f}{dt}\end{aligned}$$

Equation (4) can then be written as

$$\begin{aligned} y_1 &= y_1^- + \frac{T_f^2}{T_f^2+2hT_f+2h^2} y_2^- + \frac{2h^2}{T_f^2+2hT_f+2h^2}(y-y_1^-) = y_1^- + y_2 \\ y_2 &= \frac{T_f^2}{T_f^2+2hT_f+2h^2} y_2^- + \frac{2h^2}{T_f^2+2hT_f+2h^2}(y-y_1^-) \end{aligned} \tag{6}$$

When $T_f = 0$ the equation becomes

$$\begin{aligned} y_1 &= y \\ y_2 &= y - y_1^- \end{aligned}$$

and the estimated velocity is

$$\frac{dy_f}{dt} \approx \frac{y - y_1^-}{h}$$

Stability

An advantage of using backward difference approximations is that they often give difference equations that are stable for a wide range of parameters. Equation 6 is a discrete time dynamical system with the dynamics matrix

$$A = \begin{pmatrix} \dfrac{T_f^2+2hT_f}{T_f^2+2hT_f+2h^2} & \dfrac{T_f^2}{T_f^2+2hT_f+2h^2} \\ -\dfrac{2h^2}{T_f^2+2hT_f+2h^2} & \dfrac{T_f^2}{T_f^2+2hT_f+2h^2} \end{pmatrix}$$

The characteristic polynomial is

$$z^2 + a_1 z + a_2 = z^2 - \frac{2T_f^2+2hT_f}{T_f^2+2hT_f+2h^2} z + \frac{T_f^2}{T_f^2+2hT_f+2h^2}$$

For all $h > 0$ and $T_f \geq 0$ we have

$$\begin{aligned} 0 \leq a_2 &= \frac{T_f^2}{T_f^2+2hT_f+2h^2} < 1 \\ a_2 + a_1 + 1 &= \frac{2h^2}{T_f^2+2hT_f+2h^2} > 0 \\ a_2 - a_1 + 1 &= \frac{4T_f^2+4hT_f+2h^2}{T_f^2+2hT_f+2h^2} > 0 \end{aligned}$$

The difference equation is thus stable for all values of T_f.

4. Computer Code

A PID controller with zero setpoint weight on the derivative has the following

Table 1 Parameters for a PID Controller.

Controller	Symbol
Gain	K
Integration time	T_i
Derivative time	T_d
Filter time constant	T_f
Reset time constant	T_r
Sampling perion	h
Set point weight on proportional	b

terms.

$$\begin{aligned}
P &= k(by_{sp} - y_f) = k(by_{sp} - y_1)\\
I &= k_i \int_0^t (y_{sp}(\tau) - y_f(\tau))d\tau = \frac{k}{T_i}\int_0^t (y_{sp}(\tau) - y_1(\tau))d\tau\\
D &= -k_d \frac{dy_f}{dt} = -\frac{kT_d}{h} y_2
\end{aligned}$$

To avoid bumps in the control signal when parameters are changed the integral term $I = y_3$ is introduced as the third state of the controller. We have

$$\frac{dy_3}{dt} = \frac{k}{T_i}(y_{sp} - x_1).$$

Approximating the derivative by a forward difference we find

$$y_3 = y_3^- + \frac{kh}{T_i}(y_{sp} - y_1^-).$$

The controller parameters are given in Table 1. It is useful to precompute some parameters to reduce the number of calculations that have to be done at each sampling interval. Introduce

$$\begin{aligned}
p_1 &= \frac{T_f^2}{T_f^2 + 2hT_f + 2h^2}\\
p_2 &= \frac{2h^2}{T_f^2 + 2hT_f + 2h^2}\\
p_d &= k\frac{T_d}{h}\\
p_i &= k\frac{h}{T_i}\\
p_r &= \frac{h}{T_r}
\end{aligned}$$

These calculations have to be repeated when the controller parameters are changed.

The algorithm for the the main loop of the PID controller can then be written as

$$\begin{aligned}
y_1 &= y_1^- + p_1 y_2^- + p_2(y - y_1^-) = y_1^- + y_2\\
y_2 &= p_1 y_2^- + p_2(y - y_1^-)\\
v &= k(b y_{sp} - y_1) + y_3^- - p_d y_2\\
u &= \text{sat}\,(v)\\
y_3 &= y_3^- + p_i(y_{sp} - y_1) + p_r(u - v)
\end{aligned}$$

A skelton program for computer implementation is given in Appendix A. page. The code in the main loop requires 7 multiplications and 10 additions, only 5 multiplications and 6 additions are required in the critical part of the loop. This compares favorably with the conventional PID controller with first order filtering that requires 6 multiplications and 9 additions. Second order filtering thus only requires one addition and one multipication extra

Appencdix A - PID Controller with Second Order Filtering

```
%PID Controller with second order filter

%Bumpless parameter changes
   InitiateParChanges
   kold=k
   bold=b
   GetNewPars
   y3=y3+kold*(bold*ysp-y)-knew*(bnew*ysp-y)
   k=knew
   b=bnew

%Compute parameters
  den=Tf^2+2*h*Tf+2*h^2
  p1=Tf*Tf/den
  p2=2*h*h/den
  pd=K*Td/h
  pi=K*h/Ti
  pr=h/Tr

%Main loop
  adin(ysp)                     ‘‘Read setpoint from analog input
  adin(y)                       ‘‘Read process output from analog input
  y2=p1*y2+p2*(y-y1)            ‘‘Update filter state y1
  y1=y1+y2                      ‘‘Update filter state y2
  v=K*(b*ysp-y1)-bd*y2+y3       ‘‘Compute nominal control
  u=sat(v,ulow,uhih)            ‘‘Saturate output to avoid windup
  daout(u)                      ‘‘Set analog output
  y3=y3+bi*(ysp-y1)+br*(u-v)    ‘‘Update integrator state
```

Appencdix B - PID Controller with First Order Filtering

```
%PID controller with first order filter of derivative

%Compute controller coefficients
   bi=K*h/Ti                     "integral gain
   ad=(2*Td-N*h)/(2*Td+N*h)
   bd=2*K*N*Td/(2*Td+N*h)        "derivative gain
   a0=h/Tt

%Bumpless parameter changes
   I=I+Kold*(bold*ysp-y)-Knew*(bnew*ysp-y)

%Main Loop
   r=adin(ch1)                   "read setpoint from ch1
   y=adin(ch2)                   "read process variable from ch2
   P=K*(b*ysp-y)                 "compute proportional part
   D=ad*D-bd*(y-yold)            "update derivative part
   v=P+I+D                       "compute temporary output
   u=sat(v,ulow,uhigh)           "simulate actuator saturation
   daout(ch1)                    "set analog output ch1
   I=I+bi*(ysp-y)+ao*(u-v)       "update integral
   yold=y                        "update old process output
```