Dead-beat controller design

In sampled data control systems the controller is realised by an intelligent device, typically by a PLC (Programmable Logic Controller). The task of the PLC is to realise the controller algorithm and to handle the signals used for operation of the controller (filtering, A/D and D/A converters, signal conditioning interfaces). As the controller algorithm is realised by software, there is a possibility to apply different special, more sophisticated control algorithms. One of these algorithms ensures the accurate settling of the output signal during a finite, small number of the sampling periods. In the literature this algorithm is referred as the dead-beat controller algorithm. To understand the essence of the method - for sake of simplicity - we consider a stable plant without dead time. The reference signal is a unit step. (Let us remark that the dead-beat controller can be designed for unstable plants with dead time considering a reference signal different of unit step as well, in a bit more complex way).

In the sequel a design procedure is derived in three steps. This solution satisfies the practical requirements as well. In the first step the controller is designed for the fastest behaviour when the output signal is settled in one sampling step. It will be seen that with this design with typical sampling times the control signal could be extremely high, moreover in most cases oscillations occur between the sampling points. In the second step the design is modified to avoid intersampling oscillations. It will be shown that cancellation of zeros outside of the unit circle is the reason for oscillations. Zeros of the plant pulse transfer function are separated for cancellable and non-cancellable ones and only the cancellable zeros will appear in the controller algorithm. This modification of the algorithm increases the settling time. If the control signal is still higher than allowed, the solution can be refined using a so-called design polynomial. In this case the settling time is increased further (but still remains finite).

The design is executed in the ‘z’ operator domain. Interesting feature of the design method is the fact that it removes undesirable time domain properties (oscillations, too high values of the control signal) by considerations done in the ‘z’ operator domain. The basic task is the design of a sampled data controller.

\[
\begin{align*}
\text{r}[k] \rightarrow e[k] & \xrightarrow{C(z)} u[k] \xrightarrow{D/A} u(t) \xrightarrow{P(s)} y(t) \\
\text{U}(z) \xrightarrow{A/D} \text{Y}(z) & \xrightarrow{P(z)} \text{Y}(s)
\end{align*}
\]

The hybrid (continuous-discrete) problem is converted first to a discrete problem. Pulse transfer function \( P(z) \) of the plant is determined which considers the D/A converter and hold element together with the plant transfer function \( P(s) \). Sampling time \( T_s \) has also to be given. Then design the controller \( C(z) \) and check the closed loop system performance analysing the continuous signals not only in the sampling points, but also between them.

\[
\begin{align*}
\text{r}[k] \rightarrow e[k] & \xrightarrow{C(z)} u[k] \xrightarrow{P(z)} y[k]
\end{align*}
\]

Speaking about controllers given by their transfer functions let us analyse realisation aspects. Be the pulse transfer function of the controller:

\[
C(z) = \frac{b_3z^3 + b_2z^2 + b_1z + b_0}{z^3 + a_2z^2 + a_1z + a_0}
\]

In order to use the shift operator \( z^{-1} \) let us divide both the numerator and the denominator by third power of \( z \).

\[
C(z) = \frac{b_3 + b_2z^{-1} + b_1z^{-2} + b_0z^{-3}}{1 + a_2z^{-1} + a_1z^{-2} + a_0z^{-3}}
\]
As

\[ C(z) = \frac{Z\{u[k]\}}{Z\{e[k]\}} \]

with cross-multiplication we get

\[ u[k] + a_u u[k-1] + a_v u[k-2] + a_w u[k-3] = b_x e[k] + b_z e[k-1] + b_y e[k-2] + b_w e[k-3] \]

or

\[ u[k] = b_x e[k] + b_z e[k-1] + b_y e[k-2] + b_w e[k-3] - a_u u[k-1] - a_v u[k-2] - a_w u[k-3] \]

It is seen that the pulse transfer function of the controller can be transformed simply to a recursive difference equation. MATLAB function \texttt{dlsim} calculates the output of a discrete element on the basis of its difference equation for a given input signal. The above interpretation of pulse transfer function \( C(z) \) means that the controller in a sampled data control system is implemented by a recursive algorithm.

Returning to the design of the dead-beat controller let us consider the following plant given by its transfer function

\[ P(s) = \frac{1}{(1+5s)(1+10s)} \]

The sampling time is \( T_s=1 \) sec.

\[ \text{Ts}=1; \]
\[ s=zpk('s'); \]
\[ z=zpk('z',Ts); \]
\[ Ps=1/((1+5*s)*(1+10*s)) \]

In order to get an impression about the system let us calculate its unit step response.

\[ \text{step(Ps)} \]

The pulse transfer function of the plant together with the zero order hold is obtained

\[ Pz=c2d(Ps,Ts) \]

\[
\begin{align*}
Bz & = 0.9048 \\
Az & = 0.0090559 \\
Bz & = -0.8187 \\
Az & = -0.9048 \\
\end{align*}
\]

First design the dead-beat controller ensuring settling process during one sampling period. The condition for this is that the resulting transfer function of the closed loop between the output signal and the input signal (supposed to be a sampled unit step) be a one step shift, namely the shift operator \( z^{-1} \).

\[ T(z) = \frac{C(z)P(z)}{1+C(z)P(z)} = z^{-1} \]

Hence the controller pulse transfer function is expressed as

\[ C(z) = \frac{T(z)}{P(z)(1-T(z))} = \frac{1}{P(z)(z-1)} \]

Express \( P(z) \) as a ratio of two polynomials:

\[ P(z) = \frac{B(z)}{A(z)} \]

Then

\[ C(z) = \frac{A(z)}{B(z)(z-1)} \]

The discrete transfer function of the controller is calculated in Matlab by

\[ \text{Tz}=1/z \]
\[ \text{Cz}=\text{Tz}/(\text{Pz}*(1-\text{Tz})) \]
\[ \text{Cz}=\text{minreal(Cz)} \]

\[ C(z) = 110.425 \frac{(z-0.8187)(z-0.9048)}{(z+0.9048)(z-1)} \]

The closed loop system behaviour can be visualised by the \texttt{step} command.
The control signal is displayed:

\[ \text{step}(\frac{C_z P_z}{1 + C_z P_z}) \]

The system shows one time step delay. More accurate behaviour can be investigated by a Simulink model.

The reason of the oscillations is the fact that the controller contains the zeros of the plant as its poles, and some of these poles result oscillations in control signal \( u[k] \). (see the appendix at the end).

The zeros of the plant (roots of \( B(z) \)) appear in the controller as poles,

\[ C(z) = \frac{A(z)}{B(z)(z-1)} \]

\( P(z) \) has only one zero, \( z_1 = -0.9048 \). Examine this in more detail.

\[ C_1z = \frac{1}{z+0.9048} \]

\[ \text{step}(C_1z) \]

This component causes the oscillation. Convert back this pole to the continuous domain. Since, \( s = -\ln(z)/T_s \)

\[ p_1 = \log(-0.9048) \]

\[ p_1 = -0.1000 + 3.1416i \]

Here we just emphasise that typically poles of negative real value cause the oscillations. Let us separate the cancellable and non-cancellable zeros of the process pulse transfer function according to

\[ B(z) = B^+(z)B^-(z) \]

where \( B^+(z) \) contains the compensable and \( B^-(z) \) the non compensable roots. If a zero is not compensated it will appear in the closed loop transfer function.

Design a controller that does not compensate the non compensable roots.

\[ T(z) = \frac{B^-(z)}{z^k} \]
where

\[ k = 1 + \text{deg}(B^r), \quad B^r_n(z) = \frac{B^r(z)}{B^r(1)} \]

The \( z^{-1} \) component is necessary to get a realisable system (the degree of the denominator is higher than the degree of the numerator). Another requirement is that the static gain is 1, that is \( B^r(z) \) is normalized.

\[ B(z) = B^r(z)B^r(1) \]

\[ B^r(1) = 1 + 0.9048, \quad B^r_n(z) = \frac{z + 0.9048}{1.9048} = 0.525z + 0.475 \]

and the MATLAB program

\[
\begin{align*}
Bm & = (z + 0.9048) \\
Bpn & = Pz.k*dcgain(Bm) \\
Bmn & = Bm/dcgain(Bm) \\
Tz & = Bmn/(z^2) \\
Cz & = Tz/(Pz*(1-Tz)) \\
(Cz & = \text{minreal}(Cz, 0.001))
\end{align*}
\]

It is seen that there are no oscillations and the overexcitation in the control signal is also less than before. The system became slower, now the output signal reaches the steady state during two sampling steps. As the maximum value of the control signal is still too high, its value of about 50 would exceed the possibilities of a usual actuator. So we have to find a modification of the design which would decrease the value of the overexcitation keeping the property of a finite settling time.

Let us complete the control algorithm with a design polynomial, which will “lead” the finite time settling process. For example choosing design polynomial

\[
F(z) = \frac{1}{3}z^{-1} + \frac{1}{3}z^{-2} + \frac{1}{3}z^2 + \frac{1}{3}z + 1
\]

its smoothing effect is shown by its unit step response.

\[
Fz = (z^2 + z + 1)/(3*z^2)
\]

The control equation with the design polynomial:

\[ T(z) = F(z) \frac{B^r_n(z)}{z^k} \]

and the controller algorithm:

\[ C(z) = \frac{A(z)F(z)}{B^r_n(z)[z^k - B^r_n(z)F(z)]} \]

Let us observe that \( F(1) = 1 \), so the design polynomial does not affect the zero static error.

\[
Tz = Fz*Bmn/(z^2) \\
Cz = Tz/(Pz*(1-Tz)) \\
Cz = \text{minreal}(Cz)
\]

The dead-beat controller can be designed also in cases when the plant contains dead time. Let us suppose that the continuous dead time \( T_d \) is a multiple integer of the sampling time \( T_s \). Be this ratio \( d = T_d/T_s \).

\[ P(z) = \frac{B(z)}{A(z)}z^{-d} \]
Supposing that the continuous plant is stable and the reference input is a unit step, the closed loop transfer function is written as:

$$T(z) = \frac{B_{\text{c}}(z)}{z^k}$$

but now the value of $k$ has to be increased by $d$ steps.

$$k = 1 + \text{deg}(B^\text{c}) + d$$

Consequently the open loop pulse transfer function is

$$C(z)P(z) = \frac{z^{-s}B_{\text{c}}^\text{c}(z)}{1 - z^{-k}B_{\text{c}}^\text{c}(z)},$$

$$C(z) = \frac{z^dA(z)}{B_{\text{c}}^\text{c}(z)[z^s - B_{\text{c}}^\text{c}(z)]}.$$ 

In the previous example a $T_d=1$ sec dead time is added to the continuous process ($d=1$). Calculate the controller and simulate the behaviour of the control system.

The controller:

```matlab
» Td=1
» d=Td/Ts
» Tz=Bmn/(z^(2+d))
» Cz=Tz/(Pz*(1-Tz))
» Cz=minreal(Cz)
```

The delay can be simulated in Simulink with an added delay block (Simulink–>Continuous–>Transport Delay). Similarly to the previous discussion the design can be modified with a design polynomial.

**Appendix:**

Let us analyse the contour of a conjugate complex pair with a given damping factor in the $z$ domain. In the $s$ domain the constant $\zeta$ lines are straight lines $s = \sigma + j\omega$ going through the origo where for a given $\sigma$ value $\omega = \frac{\sigma}{\zeta} \sqrt{1 - \zeta^2}$. These complex $s$ values are transformed to the $z$ domain by relationship $z = e^{sT\text{r}}$, to curves of heart shape. As a demonstration be

```matlab
» szigma=0:0.01:1.6;
» zeta=0.4;
» Ts=1;
» z=exp(Ts*(-szigma+j*sqrt(1-zeta*zeta)*szigma/zeta));
» plot(real(z),imag(z), real(z),-imag(z)),grid;
```

Those roots of polynomial $B(z)$ which belong to polynomial $B_{\text{c}}(z)$ are inside of the closed curve (where the damping factor is higher than on the contour). Those roots of polynomial $B(z)$ which belong to polynomial $B_{\text{c}}(z)$ are on the contour or outside of it.