Design of Digital Compensators

or

How I learned to stop worrying and love the z-transform

Paul Pounds

17 May 2012

University of Queensland
Previously on Digital Controls...

• We saw how to approximate continuous dynamic systems with difference equations

• We introduced the z-domain and learned how to convert difference equations to discrete transfer functions

• We saw the relationship between pole positions in the z-plane and characteristic system responses
Feedback control

You guys know how to make a feedback controller, right?

Right…?

... guys?

Anybody?
Two classes of control design

The system...

- Isn’t fast enough
- Isn’t damped enough
- Overshoots too much
- Requires too much control action
  ("Performance")

- Attempts to spontaneously disassemble itself
  ("Stability")
Recall dynamic responses

• Moving pole positions change system response characteristics

More damped

Faster

Pure integrator

More Oscillatory

"More unstable"

More damped

"More unstable"
Recall dynamic responses

• Ditto the z-plane:

More damped

Pure integrator

Faster

“More unstable”
The fundamental control problem

The poles are in the wrong place

How do we get them where we want them to be?
Recall the root locus

- We know that under feedback gain, the poles of the closed-loop system move
  - The root locus tells us where they go!
  - We can solve for this analytically*

\[ \frac{1}{s(s + 1)} \]

Root loci can be plotted for all sorts of parameters, not just gain!
Dynamic compensation

• We can do more than just apply gain!
  – We can add dynamics into the controller that alter the open-loop response

\[
\begin{align*}
\text{compensator} & : -y \quad s + 2 \\
\text{plant} & : \frac{1}{s(s + 1)} \\
\text{combined system} & : \frac{-y}{\frac{s + 2}{s(s + 1)}}, \quad y
\end{align*}
\]
But what dynamics to add?

• Recognise the following:
  – A root locus starts at poles, terminates at zeros
    “Holes eat poles”
  – Closely matched pole and zero dynamics cancel
  – The locus is on the real axis to the left of an odd number of poles (treat zeros as ‘negative’ poles)
Some standard approaches

• Control engineers have developed time-tested strategies for building compensators
• Three classical control structures:
  – Lead
  – Lag
  – Proportional-Integral-Derivative (PID)
    (and its variations: P, I, PI, PD)

How do they work?
Lead/lag compensation

• Serve different purposes, but have a similar dynamic structure:

\[ D(s) = \frac{s + a}{s + b} \]

Note:
Lead-lag compensators come from the days when control engineers cared about constructing controllers from networks of op amps using frequency-phase methods. These days pretty much everybody uses PID, but you should at least know what the heck they are in case someone asks.
Lead compensation: $a < b$

- Acts to decrease rise-time and overshoot
  - Zero draws poles to the left; adds phase-lead
  - Pole decreases noise
- Set $a$ near desired $\omega_n$; set $b$ at ~3 to 20x $a$
Lag compensation: $a > b$

- Improves steady-state tracking
  - Near pole-zero cancellation; adds phase-lag
  - Doesn’t break dynamic response (too much)
- Set $b$ near origin; set $a$ at ~3 to 10x $b$
PID – the Good Stuff

• Proportional-Integral-Derivative control is the control engineer’s hammer*
  – For P, PI, PD, etc. just remove one or more terms

\[ C(s) = k \left( 1 + \frac{1}{\tau_i s} + \tau_d s \right) \]

*Everything is a nail
PID – the Good Stuff

• PID control performance is driven by three parameters:
  – $k$: system gain
  – $\tau_i$: integral time-constant
  – $\tau_d$: derivative time-constant

You’re already familiar with the effect of gain.
What about the other two?
Integral

• Integral applies control action based on accumulated output error
  – Almost always found with P control

• Increase dynamic order of signal tracking
  – Step disturbance steady-state error goes to zero
  – Ramp disturbance steady-state error goes to a constant offset

Let’s try it!
Integral

- Consider a first order system with a constant load disturbance, \( w \); (recall as \( t \to \infty, s \to 0 \))

\[
y = k \frac{1}{s + a} (r - y) + w
\]

\[
(s + a)y = k(r - y) + (s + a)w
\]

\[
(s + k + a)y = kr + (s + a)w
\]

\[
y = \frac{k}{s + k + a} r + \frac{(s + a)}{s + k + a} w
\]

Steady state gain = \( a/(k+a) \)

(never truly goes away)
Now with added integral action

\[
y = k \left(1 + \frac{1}{\tau_i s}\right) \frac{1}{s + a} (r - y) + w
\]

\[
y = k \frac{s + \tau_i^{-1}}{s} \frac{1}{s + a} (r - y) + w
\]

\[
s(s + a)y = k(s + \tau_i^{-1})(r - y) + s(s + a)w
\]

\[
(s^2 + (k + a)s + \tau_i^{-1})y = k(s + \tau_i^{-1})r + s(s + a)w
\]

\[
y = \frac{k(s + \tau_i^{-1})}{(s^2 + (k + a)s + \tau_i^{-1})} r + \frac{s(s + a)}{k(s + \tau_i^{-1})} w
\]

Must go to zero for constant \(w\)!
Derivative

- Derivative uses the rate of change of the error signal to anticipate control action
  - Increases system damping (when done right)
  - Can be thought of as ‘leading’ the output error, applying correction predictively
  - Almost always found with P control

*What kind of system do you have if you use D, but don’t care about position? Is it the same as P control in velocity space?
Derivative

- It is easy to see that PD control simply adds a zero at \( s = -\frac{1}{\tau_d} \) with expected results
  - Decreases dynamic order of the system by 1
  - Absorbs a pole as \( k \to \infty \)

- Not all roses, though: derivative operators are sensitive to high-frequency noise

![Bode plot of a zero](image-url)
PID

- Collectively, PID provides two zeros plus a pole at the origin
  - Zeros provide phase lead
  - Pole provides steady-state tracking
  - Easy to implement in microprocessors

- Many tools exist for optimally tuning PID
  - Zeigler-Nichols
  - Cohen-Coon
  - Automatic software processes
Be alert

• If gains and time-constants are chosen poorly, all of these compensators can induce oscillation or instability.

• However, when used properly, PID can stabilise even very complex unstable third-order systems
Now in discrete

- Naturally, there are discrete analogs for each of these controller types:

  **Lead/lag:** \[ \frac{1 - \alpha z^{-1}}{1 - \beta z^{-1}} \]

  **PID:** \[ k \left( 1 + \frac{1}{\tau_i (1 - z^{-1})} + \tau_d (1 - z^{-1}) \right) \]

  But, where do we get the control design parameters from? The s-domain?
Emulation vs Discrete Design

• Remember: polynomial algebra is the same, whatever symbol you are manipulating:

  \[ s^2 + 2s + 1 = (s + 1)^2 \]
  \[ z^2 + 2z + 1 = (z + 1)^2 \]

  Root loci behave the same on both planes!

• Therefore, we have two choices:
  – Design in the s-domain and digitise (emulation)
  – Design only in the z-domain (discrete design)
Emulation design process

1. Derive the dynamic system model ODE
2. Convert it to a continuous transfer function
3. Design a continuous controller
4. Convert the controller to the z-domain
5. Implement difference equations in software
Emulation design process

• Handy rules of thumb:
  – Use a sampling period of 20 to 30 times faster than the closed-loop system bandwidth
  – Remember that the sampling ZOH induces an effective $T/2$ delay
  – There are several approximation techniques:
    • Euler’s method
    • Tustin’s method
    • Matched pole-zero
    • Modified matched pole-zero
Tustin’s method

- Tustin uses a trapezoidal integration approximation (compare Euler’s rectangles)
- Integral between two samples treated as a straight line:
  \[ u(kT) = \frac{T}{2} [x(k - 1) + x(k)] \]

Taking the derivative, then z-transform yields:

\[
S = \frac{2 z - 1}{T z + 1}
\]

which can be substituted into continuous models
Matched pole-zero

• If \( z = e^{sT} \), why can’t we just make a direct substitution and go home?

\[
\frac{Y(s)}{X(s)} = \frac{s+a}{s+b} \quad \Rightarrow \quad \frac{Y(z)}{X(z)} = \frac{z-e^{-aT}}{z-e^{-bT}}
\]

• Kind of!
  – Still an approximation
  – Produces quasi-causal system (hard to compute)
  – Fortunately, also very easy to calculate.
Matched pole-zero

The process:

1. Replace continuous poles and zeros with discrete equivalents:
   \[ (s + a) \rightarrow (z - e^{-aT}) \]

2. Scale the discrete system DC gain to match the continuous system DC gain

3. If the order of the denominator is higher than the enumerator, multiply the numerator by \((1 + z^{-1})\) until they are of equal order*

* This introduces an averaging effect like Tustin’s method
Modified matched pole-zero

- We’re prefer it if we didn’t require instant calculations to produce timely outputs
- Modify step 2 to leave the dynamic order of the numerator one less than the denominator
  - Can work with slower sample times, and at higher frequencies
Discrete design process

1. Derive the dynamic system model ODE
2. Convert it to a discrete transfer function
3. Design a digital compensator
4. Implement difference equations in software
5. Pub
Discrete design process

• Handy rules of thumb:
  – Sample rates can be as low as twice the system bandwidth (but 20 to 30 for better performance)
  – A zero at $z = -1$ makes the discrete root locus pole behaviour more closely match the s-plane
  – Beware “dirty derivatives”
    • $dy/dt$ terms derived from sequential digital values are called ‘dirty derivatives’ – these are especially sensitive to noise!
    • Employ actual velocity measurements when possible
Tune-in next time for...

Digital Design Practice

or

“Why doesn’t my compensator work?”

Fun fact: The first UAV was the Hewitt-Sperry Automatic Aeroplane in 1917, 14 years after the Wright bros’ first flight. It could travel 50 km and drop a sandbag within 3.2 km of a target.