Servoregulation:
Lead/Lag & PID

ELEC 3004: Systems: Signals & Controls
Dr. Surya Singh

Lecture 16

elec3004@itee.uq.edu.au
http://robotics.itee.uq.edu.au/~elec3004/

May 4, 2017

Lecture Schedule:

<table>
<thead>
<tr>
<th>Week</th>
<th>Date</th>
<th>Lecture Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>28 Feb</td>
<td>Introduction</td>
</tr>
<tr>
<td>2</td>
<td>7 Mar</td>
<td>Systems as Maps &amp; Signals as Vectors</td>
</tr>
<tr>
<td>3</td>
<td>9 Mar</td>
<td>Systems: Linear Differential Systems</td>
</tr>
<tr>
<td>4</td>
<td>14 Mar</td>
<td>Sampling Theory &amp; Dots Acquisition</td>
</tr>
<tr>
<td>5</td>
<td>16 Mar</td>
<td>Aliasing &amp; Anti-aliasing</td>
</tr>
<tr>
<td>6</td>
<td>21 Mar</td>
<td>Discrete Time Analysis &amp; Z-Transform</td>
</tr>
<tr>
<td>7</td>
<td>23 Mar</td>
<td>Second-Order LTID (Laplace Transform Review)</td>
</tr>
<tr>
<td>8</td>
<td>28 Mar</td>
<td>Frequency Response</td>
</tr>
<tr>
<td>9</td>
<td>30 Mar</td>
<td>Filter Analysis</td>
</tr>
<tr>
<td>10</td>
<td>1 Apr</td>
<td>Digital Filters (IIR) &amp; Filter Analysis</td>
</tr>
<tr>
<td>11</td>
<td>3 Apr</td>
<td>Digital Filter (FIR)</td>
</tr>
<tr>
<td>12</td>
<td>6 Apr</td>
<td>Digital Windows</td>
</tr>
<tr>
<td>13</td>
<td>8 Apr</td>
<td>FFT</td>
</tr>
<tr>
<td>14</td>
<td>11 Apr</td>
<td>Digital Windows</td>
</tr>
<tr>
<td>15</td>
<td>13 Apr</td>
<td>Active Filters &amp; Estimation</td>
</tr>
<tr>
<td>16</td>
<td>18 Apr</td>
<td>Holiday</td>
</tr>
<tr>
<td>17</td>
<td>20 Apr</td>
<td>Digital Filters (IIR) &amp; Filter Analysis</td>
</tr>
<tr>
<td>18</td>
<td>23 Apr</td>
<td>Digital Filter (FIR)</td>
</tr>
<tr>
<td>19</td>
<td>25 Apr</td>
<td>Filter Analysis</td>
</tr>
<tr>
<td>20</td>
<td>27 Apr</td>
<td>Digital Windows</td>
</tr>
</tbody>
</table>

© 2017 School of Information Technology and Electrical Engineering at The University of Queensland
Follow Along Reading:

- **B. P. Lathi**
  - *Signal processing and linear systems*
  - 1998
  - TK5102.9.L38 1998

- **G. Franklin, J. Powell, M. Workman**
  - *Digital Control of Dynamic Systems*
  - 1990
  - TJ216.F72 1990
  - [Available as UQ Ebook]

Today

- **P - I - D**
- **FPW**
  - Chapter 4: Discrete Equivalents to Continuous Transfer Functions: The Digital Filter

- **FPW**
  - Chapter 5: Design of Digital Control Systems Using Transform Techniques

Next Time

Feedback as a Filter
Time Response

![Time Response Diagram]

Frequency Domain Analysis

- Bode (Magnitude + Phase Plots)
- Nyquist Plot (Polar)
In This Way Feedback May Be Seen as a Filter

- Ex: Lightly Damped Robot Arm

![Graph showing amplitude and phase transfer function](image)

-40 dB/decade = double integrator

Resonance

Lead/Lag
**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 20\( x \) \( a \)

**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 10\( x \) \( b \)
BREAK

PID (Intro)
**Proportional Control**

A discrete implementation of proportional control is identical to continuous; that is, where the continuous is

\[ u(t) = K_p e(t) \Rightarrow D(s) = K_p, \]

the discrete is

\[ u(k) = K_p e(k) \Rightarrow D(z) = K_p \]

where \( e(t) \) is the error signal as shown in Fig 5.2.

![Proportional Control Diagram](image)

**Derivative Control**

For continuous systems, derivative or rate control has the form

\[ u(t) = K_p T_D \hat{e}(t) \Rightarrow D(s) = K_p T_D s \]

where \( T_D \) is called the *derivative time*. Differentiation can be approximated in the discrete domain as the first difference, that is,

\[ u(k) = K_p T_D \frac{(e(k) - e(k - 1))}{T} \Rightarrow \]

\[
D(z) = K_p T_D \frac{1 - z^{-1}}{T} = K_p T_D \frac{z - 1}{T_z}.
\]

In many designs, the compensation is a sum of proportional and derivative control (or PD control). In this case, we have

\[ D(z) = K_p \left(1 + \frac{T_D(z - 1)}{T_z}\right). \]

or, equivalently,

\[
D(z) = K \frac{z - \alpha}{z}
\]
### Derivative Control [2]

- Similar to the lead compensators
  - The difference is that the pole is at $z = 0$

  [Whereas the pole has been placed at various locations along the $z$-plane real axis for the previous designs.]

- In the continuous case:
  - Pure derivative control represents the ideal situation in that there is no destabilizing phase lag from the differentiation
  - The pole is at $s = -\infty$

- In the discrete case:
  - $z=0$
  - However this has phase lag because of the necessity to wait for one cycle in order to compute the first difference

### 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

\[
|C(j\omega)| = \frac{1}{\tau_d} \\
\text{Bode plot of a zero}
\]

PD for 2\textsuperscript{nd} Order Systems

- Consider:
  \[
  \frac{Y(s)}{R(s)} = \frac{(K_P + K_D s)}{Js^2 + (B + K_D)s + K_P}
  \]
- Steady-state error: \( e_{ss} = \frac{B}{K_P} \)
- Characteristic equation: \( Js^2 + (B + K_D)s + K_P = 0 \)
- Damping Ratio: \( \zeta = \frac{B + K_D}{2\sqrt{K_P J}} \)
  \( \Rightarrow \) It is possible to make \( e_{ss} \) and overshoot small (\( \downarrow \)) by making \( B \) small (\( \downarrow \)), \( K_P \) large (\( \uparrow \)), \( K_D \) such that \( \zeta \) between \( [0.4 - 0.7] \)
**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 Control**

For continuous systems, we integrate the error to arrive at the control,

\[ u(t) = \frac{K_p}{T_i} \int_{t_0}^{t} e(t) \, dt \Rightarrow D(s) = \frac{K_p}{T_i s}, \]

where \( T_i \) is called the integral, or reset time. The discrete equivalent is to sum all previous errors, yielding

\[ u(k) = u(k-1) + \frac{K_p T}{T_i} e(k) \Rightarrow D(z) = \frac{K_p T}{T_i (1 - z^{-1})} = \frac{K_p T z}{T_i (z - 1)}. \]  (5.60)

Just as for continuous systems, the primary reason for integral control is to reduce or eliminate steady-state errors, but this typically occurs at the cost of reduced stability.
Integral: P Control only

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

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

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

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

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

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

(never truly goes away)

\[
\frac{1}{s+a}
\]

**Now with added integral action**

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

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

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

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

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

**Must go to zero for constant \( w \)!!**
Proportional-Integral-Derivative control is the control engineer’s hammer*
  – For P,PI,PD, etc. just remove one or more terms

\[ C(s) = k \left( 1 + \frac{1}{\tau Is} + \tau ds \right) \]

*Everything is a nail. That’s why it’s called “Bang-Bang” Control 😊

• Three basic types of control:
  – Proportional
  – Integral, and
  – Derivative

• The next step up from lead compensation
  – Essentially a combination of proportional and derivative control
The user simply has to determine the best values of

- $K_p$
- $T_D$ and
- $T_I$

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

$$D(z) = K_p \left(1 + \frac{T_z}{T_I(z - 1)} + \frac{T_D(z - 1)}{T_z}\right).$$
**PID as Difference Equation**

\[
\frac{U(z)}{E(z)} = D(z) = K_p + K_i \left( \frac{T_z}{z - 1} \right) + K_d \left( \frac{z - 1}{T_z} \right)
\]

\[
u(k) = \left[ K_p + K_i T + \left( \frac{K_d}{T} \right) \right] \cdot e(k) - [K_d T] \cdot e(k - 1) + [K_i] \cdot u(k - 1)
\]

---

**PID Implementation**

- **Non-Interacting**

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

- **Interacting Form**

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

- Note: Different \(K, T_i\) and \(T_d\)
Operational Amplifier Circuits for Compensators

- PD
  \[ G_p(z) = \frac{R_2}{R_1(R_2C_p + 1)} \]

- PI
  \[ G_i(z) = \frac{R_2}{R_1(R_2C_i + 1)} \]

- Lead or lag
  \[ G_l(z) = \frac{R_2}{R_1(R_2C_l + 1)} \]
  Lead if \( RC_l > RC_i \)
  Lag if \( RC_l < RC_i \)

(Yet Another Way to See PID)

---

PID Algorithm (in Z-Domain):

\[ D(z) = K_p \left( 1 + \frac{T_z}{T_i(z-1)} + \frac{T_D(z-1)}{T_z} \right) \]

- As Difference equation:
  \[ u(t_k) = u(t_{k-1}) + K_p \left[ \left( 1 + \frac{T_i}{T_i} + \frac{T_D}{T_i} \right) e(t_k) + \left( -1 - \frac{2T_D}{T_i} \right) e(t_{k-1}) + \frac{T_D}{T_i} e(t_{k-2}) \right] \]

- Pseudocode [Source: Wikipedia]:
  ```plaintext```
  previous_error = 0, integral = 0
  start:
  error = setpoint - measured_value
  integral = integral + error*dt
  derivative = (error - previous_error)/dt
  output = Kp*error + Ki*integral + Kd*derivative
  previous_error = error
  wait [dt]
  goto start```

---
Another way to see P I|D

- Derivative
  D provides:
  - High sensitivity
  - Responds to change
  - Adds “damping” & ∴ permits larger $K_p$
  - Noise sensitive
  - Not used alone
    (∴ its on rate change of error – by itself it wouldn’t get there)
  → “Diet Coke of control”

- Integral
  - Eliminates offsets
    (makes regulation 😊)
  - Leads to Oscillatory behaviour
  - Adds an “order” but instability
    (Makes a 2\textsuperscript{nd} order system 3\textsuperscript{rd} order)

→ “Interesting cake of control”

Seeing PID – No Free Lunch

- The energy (and sensitivity) moves around
  (in this case in “frequency”)

- Sensitivity reduction at low frequency unavoidably leads to sensitivity increase at higher frequencies.

PID Intuition & Tuning

• Tuning – How to get the “magic” values:
  – Dominant Pole Design
  – Ziegler Nichols Methods
  – Pole Placement
  – Auto Tuning

• Although PID is common it is often poorly tuned
  – The derivative action is frequently switched off!
    (Why ∴ it’s sensitive to noise)
  – Also lots of “I” will make the system more transitory &
    leads to integrator wind-up.

PID Intuition

\[ u(t) = K \left[ e(t) + \frac{1}{T_i} \int e(s) \, ds + T_d \frac{de(t)}{dt} \right] \]

• P:
  – Control action is proportional to control error
  – It is necessary to have an error to have a non-zero control signal

• I:
  – The main function of the integral action is to make sure that the
    process output agrees with the set point in steady state
The purpose of the derivative action is to improve the closed loop stability.

The instability “mechanism” “controlled” here is that because of the process dynamics it will take some time before a change in the control variable is noticeable in the process output.

The action of a controller with proportional and derivative action may be interpreted as if the control is made proportional to the predicted process output, where the prediction is made by extrapolating the error by the tangent to the error curve.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Rise time</th>
<th>Overshoot</th>
<th>Settling time</th>
<th>Steady-state error</th>
<th>Stability</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_p$</td>
<td>↓</td>
<td>✴</td>
<td>Minimal change</td>
<td>↓</td>
<td>↓</td>
</tr>
<tr>
<td>$K_I$</td>
<td>↓</td>
<td>✴</td>
<td>✴</td>
<td>Eliminate</td>
<td>↓</td>
</tr>
<tr>
<td>$K_D$</td>
<td>Minor change</td>
<td>↓</td>
<td>No effect / minimal change</td>
<td>Improve (if $K_D$ small)</td>
<td></td>
</tr>
</tbody>
</table>
PID Intuition: P and PI

Responses of P, PI, and PID control to

(a) step disturbance input  
(b) step reference input
**PID Example**

- A 3rd order plant: b=10, \( \zeta=0.707 \), \( \omega_n=4 \)

\[
G(s) = \frac{1}{s(s + b)(s + 2\zeta \omega_n)}
\]

- PID:

  - Kp=855:
    
    \( \cdot 40\% \) Kp = 370

---

**Ziegler-Nichols Tuning – Reaction Rate**

FPW § 5.8.5 [p.224]

**Table 5.2** Ziegler-Nichols tuning parameters using transient response.

<table>
<thead>
<tr>
<th></th>
<th>( K_p )</th>
<th>( T_i )</th>
<th>( T_D )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P )</td>
<td>1/RL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( PI )</td>
<td>0.9/RL</td>
<td>3L</td>
<td></td>
</tr>
<tr>
<td>( PID )</td>
<td>1.2/RL</td>
<td>2L</td>
<td>0.5L</td>
</tr>
</tbody>
</table>
Quarter decay ratio

Ziegler-Nichols Tuning – Stability Limit Method
FPW § 5.8.5 [p.226]
- Increase $K_P$ until the system has continuous oscillations
  $\equiv K_U$ : Oscillation Gain for “Ultimate stability”
  $\equiv P_U$ : Oscillation Period for “Ultimate stability”

<table>
<thead>
<tr>
<th></th>
<th>$K_p$</th>
<th>$T_I$</th>
<th>$T_D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P$</td>
<td>$0.5K_u$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$PI$</td>
<td>$0.45K_u$</td>
<td>$P_u/1.2$</td>
<td></td>
</tr>
<tr>
<td>$PID$</td>
<td>$0.6K_u$</td>
<td>$P_u/2$</td>
<td>$P_u/8$</td>
</tr>
</tbody>
</table>
Ziegler-Nichols Tuning

\[ C(i\omega_u) = K \left( 1 + i\left(\omega_u T_d - \frac{1}{\omega_u T_i}\right) \right) \approx 0.6K_u(1 + 0.467i) \]

Break!: Fun Application: Linear Algebra & KVL!

We can write this as:

\[
\begin{pmatrix}
1 & 1 & 1 \\
-2 & 3 & 0 \\
0 & -3 & 6
\end{pmatrix}
\begin{pmatrix}
I_1 \\
I_2 \\
I_3
\end{pmatrix}
= 
\begin{pmatrix}
0 \\
24 \\
0
\end{pmatrix}
\]

So we have:

\[
\begin{pmatrix}
I_1 \\
I_2 \\
I_3
\end{pmatrix}
= 
\begin{pmatrix}
1 & 1 & 1 \\
-2 & 3 & 0 \\
0 & -3 & 6
\end{pmatrix}^{-1}
\begin{pmatrix}
0 \\
24 \\
0
\end{pmatrix}
\]

Using a computer algebra system to perform the inverse and multiply by the constant matrix, we get:

\[ I_1 = -6 \text{ A} \]
\[ I_2 = 4 \text{ A} \]
\[ I_3 = 2 \text{ A} \]

We observe that \( I_1 \) is negative, as expected from the circuit diagram.

Break!: Fun Application: Linear Algebra & KCL!

We solve this using a computer as follows. We just write the coefficient matrix on the left, find the inverse (raise the matrix to the power -1) and multiply the result by the constant matrix.

You can use Matlab, Mathcad or similar math software to do this. WolframAlpha is a free alternative.

\[
X = \begin{bmatrix}
72 & 0 & -17 & -35 & 0 & 0 & 0 \\
0 & 122 & -35 & 0 & 0 & -87 & 34 \\
-17 & -35 & 149 & 0 & -28 & -35 & -34 & 0 \\
0 & 0 & -35 & 0 & 105 & -34 & 0 & -27 \\
-35 & 0 & 0 & 0 & 105 & -43 & 0 & 5 \\
\end{bmatrix}^{-1} \begin{bmatrix}
26 \\
34 \\
-13 \\
27 \\
5 \\
\end{bmatrix}
\]

\[
= \begin{bmatrix}
-0.46801 \\
0.42902 \\
5.193 \times 10^{-3} \\
-0.22243 \\
-0.27848 \\
0.21115 \\
0.20904 \\
\end{bmatrix}
\]


Next Time...

- Digital Feedback Control
- Review:  
  - Chapter 2 of FPW
- More Pondering??
Two cases for 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”)
Dynamic compensation

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

\[
\begin{align*}
\text{compensator} & : \quad u \quad \rightarrow \quad y \\
\text{plant} & : \quad \frac{1}{s(s+1)} \\
\text{combined system} & : \quad \frac{s+2}{s(s+1)}
\end{align*}
\]

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)
The Root Locus (Quickly)

- The transfer function for a closed-loop system can be easily calculated:

\[ y = CH(r - y) \]
\[ y + CHy = CHr \]
\[ \therefore \frac{y}{r} = \frac{CH}{1 + CH} \]

- We often care about the effect of increasing gain of a control compensator design:

\[ \frac{y}{r} = \frac{kCH}{1 + kCH} \]

Multiplying by denominator:

\[ \frac{y}{r} = \frac{kC_nH_n}{C_dH_d + kC_nH_n} \]

characteristic polynomial
The Root Locus (Quickly)

- Pole positions change with increasing gain
  - The trajectory of poles on the pole-zero plot with changing $k$ is called the "root locus"
  - This is sometimes quite complex

(In practice you’d plot these with computers)

Designing in the Purely Discrete...

Analyse/design a discrete controller $D(z)$:

by considering the purely discrete time system:

Closed loop system transfer function: $\frac{Y(z)}{R(z)} = \frac{G(z)D(z)}{1 + G(z)D(z)}$

How do the closed loop poles relate to
  - stability?
  - performance?
Now in discrete

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

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

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

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

---

Sampling a continuous-time system

suppose \( \dot{x} = Ax \)

sample \( x \) at times \( t_1 \leq t_2 \leq \cdots \): define \( z(k) = x(t_k) \)

then \( z(k+1) = e^{(t_{k+1}-t_k)A}z(k) \)

for uniform sampling \( t_{k+1} - t_k = h \), so

\[
z(k+1) = e^{hA}z(k),
\]

a discrete-time LDS (called discretized version of continuous-time system)

Source: Boyd, Lecture Notes for EE263, 10-22
Piecewise constant system

consider time-varying LDS \( \dot{x} = A(t)x, \) with

\[
A(t) = \begin{cases} 
A_0 & 0 \leq t < t_1 \\
A_1 & t_1 \leq t < t_2 \\
\vdots & 
\end{cases}
\]

where \( 0 < t_1 < t_2 < \cdots \) (sometimes called jump linear system)

for \( t \in [t_i, t_{i+1}] \) we have

\[
x(t) = e^{(t-t_i)A_1} \cdots e^{(t_3-t_2)A_2} \cdots e^{(t_2-t_1)A_1} e^{t_1A_0} x(0)
\]

(matrix on righthand side is called state transition matrix for system, and denoted \( \Phi(t) \))

---

Qualitative behaviour of \( x(t) \)

suppose \( \dot{x} = Ax, \ x(t) \in \mathbb{R}^n \)

then \( x(t) = e^{tA}x(0); \ X(s) = (sI - A)^{-1}x(0) \)

\( i \)th component \( X_i(s) \) has form

\[
X_i(s) = \frac{a_i(s)}{X(s)}
\]

where \( a_i \) is a polynomial of degree < \( n \)

thus the poles of \( X_i \) are all eigenvalues of \( A \) (but not necessarily the other way around)

---

Source: Boyd, Lecture Notes for EE263, 10-23

Source: Boyd, Lecture Notes for EE263, 10-24
Qualitative behaviour of $x(t)$ [2]

first assume eigenvalues $\lambda_i$ are distinct, so $X_i(s)$ cannot have repeated poles

then $x_i(t)$ has form

$$x_i(t) = \sum_{j=1}^{n} \beta_{ij} e^{\lambda_j t}$$

where $\beta_{ij}$ depend on $x(0)$ (linearly)

eigenvalues determine (possible) qualitative behavior of $x$:

- eigenvalues give exponents that can occur in exponentials
- real eigenvalue $\lambda$ corresponds to an exponentially decaying or growing term $e^{\lambda t}$ in solution
- complex eigenvalue $\lambda = \sigma + j\omega$ corresponds to decaying or growing sinusoidal term $e^{\sigma t} \cos(\omega t + \phi)$ in solution

Source: Boyd, Lecture Notes for EE263, 10-25

---

Qualitative behaviour of $x(t)$ [3]

first assume eigenvalues $\lambda_i$ are distinct, so $X_i(s)$ cannot have repeated poles

then $x_i(t)$ has form

$$x_i(t) = \sum_{j=1}^{n} \beta_{ij} e^{\lambda_j t}$$

where $\beta_{ij}$ depend on $x(0)$ (linearly)

eigenvalues determine (possible) qualitative behavior of $x$:

- eigenvalues give exponents that can occur in exponentials
- real eigenvalue $\lambda$ corresponds to an exponentially decaying or growing term $e^{\lambda t}$ in solution
- complex eigenvalue $\lambda = \sigma + j\omega$ corresponds to decaying or growing sinusoidal term $e^{\sigma t} \cos(\omega t + \phi)$ in solution

Source: Boyd, Lecture Notes for EE263, 10-26
Qualitative behaviour of $x(t)$ [4]

- $\Re \lambda_j$ gives exponential growth rate (if $> 0$), or exponential decay rate (if $< 0$) of term
- $\Im \lambda_j$ gives frequency of oscillatory term (if $\neq 0$)

![Eigenvectors Diagram]

Source: Boyd, Lecture Notes for EE263, 10-27

Qualitative behaviour of $x(t)$ [5]

Now suppose $A$ has repeated eigenvalues, so $X_t$ can have repeated poles.

Express eigenvalues as $\lambda_1, \ldots, \lambda_r$ (distinct) with multiplicities $n_1, \ldots, n_r$, respectively ($n_1 + \cdots + n_r = n$)

Then $x(t)$ has form

$$x(t) = \sum_{j=1}^{r} P_{ij}(t)e^{\lambda_j t}$$

where $P_{ij}(t)$ is a polynomial of degree $< n_j$ (that depends linearly on $x(0)$)

Source: Boyd, Lecture Notes for EE263, 10-28
**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

Euler’s method*

- Dynamic systems can be approximated† by recognising that:
  $$\dot{x} \approx \frac{x(k + 1) - x(k)}{T}$$

- As $T \to 0$, approximation error approaches 0

*Also known as the forward rectangle rule
†Just an approximation – more on this later $T$
An example!

Convert the system \( \frac{Y(s)}{X(s)} = \frac{s+2}{s+1} \) into a difference equation with period \( T \), using Euler’s method.

1. Rewrite the function as a dynamic system:

\[
sY(s) + Y(s) = sX(s) + 2X(s)
\]

Apply inverse Laplace transform:

\[
\dot{y}(t) + y(t) = \dot{x}(t) + 2x(t)
\]

2. Replace continuous signals with their sampled domain equivalents, and differentials with the approximating function

\[
\frac{y(k+1) - y(k)}{T} + y(k) = \frac{x(k+1) - x(k)}{T} + 2x(k)
\]

Simplify:

\[
y(k+1) - y(k) + Ty(k) = x(k+1) - x(k) + 2Tx(k)
y(k+1) + (T-1)y(k) = x(k+1) + (2T-1)x(k)
\]

\[
y(k+1) = x(k+1) + (2T-1)x(k) - (T-1)y(k)
\]

We can implement this in a computer.

Cool, let’s try it!
Back to the future

A quick note on causality:

• Calculating the “(k+1)th” value of a signal using

\[ y(k + 1) = x(k + 1) + Ax(k) - By(k) \]

relies on also knowing the next (future) value of \( x(t) \).

(this requires very advanced technology!)

• Real systems always run with a delay:

\[ y(k) = x(k) + Ax(k - 1) - By(k - 1) \]

Back to the example!

```plaintext
T = 0.02; //period of 50 Hz, a number pulled from thin air
A = 2*T-1; //pre-calculated control constants
B = T-1;

...

while(1)
{
    if(interrupt_flag) //this triggers every 20 ms
    {
        x0 = x; //save previous values
        y0 = y;
        x = update_input(); //get latest x value
        y = x + A*x0 - B*y0; //do the difference equations
        update_output(y); //write out current value
    }
}
```

(The actual calculation)
**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:
\[ s = \frac{2z - 1}{Tz + 1} \]

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 \((z + 1)\) 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. Platypus Is Divine!

---

**Discrete design process**

- Handy rules of thumb:
  - Sample rates can be as low as twice the system bandwidth
    - but 5 to 10× for “stability”
    - 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
Review: Direct Design: 
Second Order Digital Systems

Consider the z-transform of a decaying exponential signal:

\[ y(t) = e^{-at} \cos(bt) U(t) \]  \( (U(t) = \text{unit step}) \)

- sample: \[ y(kT) = r^k \cos(k\theta) U(kT) \] with \( r = e^{-aT} \) & \( \theta = bT \)

- transform: \[ Y(z) = \frac{1}{2} \left( \frac{z}{z - re^{j\theta}} + \frac{1}{z - re^{-j\theta}} \right) \]

\[ \frac{z(z - r \cos \theta)}{(z - re^{j\theta})(z - re^{-j\theta})} \]

- e.g. \( y_k \) is the pulse response of \( G(z) \):

\[ G(z) = \frac{z(z - r \cos \theta)}{(z - re^{j\theta})(z - re^{-j\theta})} \]

poles:
\[ \{ z = re^{j\theta}, z = re^{-j\theta} \} \]

zeros:
\[ \{ z = 0, z = r \cos \theta \} \]
Response of 2nd order system [1/3]

Responses for varying $r$:

- $r < 1$
  - exponentially decaying envelope

- $r = 1$
  - sinusoidal response with $2\pi/\theta$ samples per period

- $r > 1$
  - exponentially increasing envelope

Response of 2nd order system [2/3]

Responses for varying $\theta$:

- $\theta = 0$
  - decaying exponential

- $\theta = \pi/2$
  - $2\pi/\theta = 4$ samples per period

- $\theta = \pi$
  - 2 samples per period
Response of 2nd order system [3/3]

Some special cases:

- for $\theta = 0$, $Y(z)$ simplifies to:
  \[ Y(z) = \frac{z}{z - r} \]
  exponentially decaying response

- when $\theta = 0$ and $r = 1$:
  \[ Y(z) = \frac{z}{z - 1} \]
  unit step

- when $r = 0$:
  \[ Y(z) = 1 \]
  unit pulse

- when $\theta = 0$ and $-1 < r < 0$:
  samples of alternating signs

2nd Order System Response

- Response of a 2nd order system to increasing levels of damping:
Damping and natural frequency

\[ z = e^{sT} \text{ where } s = -\zeta \omega_n \pm j \omega_n \sqrt{1 - \zeta^2} \]

Pole positions in the z-plane

- Poles inside the unit circle are stable
- Poles outside the unit circle are unstable
- Poles on the unit circle are oscillatory
- Real poles at \( 0 < z < 1 \) give exponential response
- Higher frequency of oscillation for larger
- Lower apparent damping for larger and r
2\textsuperscript{nd} Order System Specifications

Characterizing the step response:

- Rise time (10% \(\rightarrow\) 90%): \(t_r \approx \frac{1.6}{\omega_0}\)
- Overshoot: \(M_p \approx \frac{e^{-\pi\zeta}}{\sqrt{1 - \zeta^2}}\)
- Settling time (to 1\%): \(t_s = \frac{4.6}{\zeta \omega_0}\)

\[\text{Why 4.6? It's -ln(1%) }\]
\[\rightarrow e^{-\pi \zeta} = 0.01 \rightarrow \zeta \omega_0 = 4.6 \rightarrow t_s = \frac{4.6}{\zeta \omega_0}\]

- Steady state error to unit step: \(e_s\)
- Phase margin: \(\phi_{PM} \approx 100\zeta\)

\[\text{Characterizing the step response:}\]

- Rise time (10% \(\rightarrow\) 90%) & Overshoot:
  \(t_r, M_p \rightarrow \zeta, \omega_0\) : Locations of dominant poles
- Settling time (to 1%):
  \(t_s \rightarrow\) radius of poles: \(|z| < 0.01\zeta\)
- Steady state error to unit step:
  \(e_{ss} \rightarrow\) final value theorem \(e_{ss} = \lim_{z \rightarrow 1} \{z - 1\} F(z)\)
Ex: System Specifications $\rightarrow$ Control Design [1/4]

Design a controller for a system with:

- A continuous transfer function: $G(s) = \frac{0.1}{s(s + 0.1)}$
- A discrete ZOH sampler
- Sampling time ($T_s$): $T_s = 1s$
- Controller:
  
  $$u_k = -0.5u_{k-1} + 13(e_k - 0.88e_{k-1})$$

The closed loop system is required to have:

- $M_p < 16\%$
- $t_s < 10s$
- $e_{ss} < 1$

Ex: System Specifications $\rightarrow$ Control Design [2/4]

1. **(a)** Find the pulse transfer function of $G(s)$ plus the ZOH

   $$G(z) = (1 - z^{-1})Z\left\{\frac{G(s)}{s}\right\} = \frac{(z - 1)}{z} \cdot 2\left\{\frac{0.1}{s^2(s + 0.1)}\right\}$$

   e.g. look up $Z\{a/s^2(s + a)\}$ in tables:

   $$G(z) = \frac{(z - 1)}{z} \cdot \frac{(0.1 - 1 + e^{-0.1})z + (1 - e^{-0.1} - 0.1e^{-0.1})}{0.1(z - 1)^2(z - e^{-0.1})}$$

   $$= \frac{0.0484(z + 0.9672)}{(z - 1)(z - 0.9048)}$$

   **(b)** Find the controller transfer function (using $z =$ shift operator):

   $$\frac{U(z)}{E(z)} = D(z) = 13\frac{(1 - 0.88z^{-1})}{(1 + 0.5z^{-1})} = 13\frac{(z - 0.88)}{(z + 0.5)}$$
Ex: System Specifications $\rightarrow$ Control Design [3/4]

2. Check the steady state error $e_{ss}$ when $r_k = \text{unit ramp}$

$$e_{ss} = \lim_{k \to \infty} e_k = \lim_{z \to 1} (z - 1)E(z)$$

$$E(z) = \frac{1}{1 + D(z)G(z)}$$

$$R(z) = \frac{T_z}{(z - 1)^2}$$

$$\text{so} \quad e_{ss} = \lim_{z \to 1} \left\{ (z - 1) \frac{T_z}{(z - 1)^2} \frac{1}{1 + D(z)G(z)} \right\} = \lim_{z \to 1} \frac{T}{(z - 1)^2} \frac{0.0484(z + 0.9672)}{(z - 1)(z - 0.9048)D(1)}$$

$$= \frac{1 - 0.9048}{0.0484(1 + 0.9672)D(1)} = 0.96$$

$$\Rightarrow e_{ss} < 1 \quad \text{(as required)}$$

Ex: System Specifications $\rightarrow$ Control Design [4/4]

3. Step response: overshoot $M_p < 16\% \quad \Rightarrow \quad \zeta > 0.5$

settling time $t_s < 10 \quad \Rightarrow \quad |z| < 0.01^{1/10} = 0.63$

The closed loop poles are the roots of $1 + D(z)G(z) = 0$, i.e.

$$1 + 13\left(\frac{z - 0.88}{z + 0.9672}\right)\left(\frac{z + 0.5}{z - 1}\right)(z - 0.9048) = 0$$

$$\Rightarrow \quad z = 0.88, \quad -0.050 \pm j0.304$$

But the pole at $z = 0.88$ is cancelled by controller zero at $z = 0.88$, and

$$z = -0.050 \pm j0.304 = re^{\pm j\theta} \quad \Rightarrow \quad \begin{cases} r = 0.31, \quad \theta = 1.73 \\ \zeta = 0.56 \end{cases}$$

all specs satisfied!
LTID Stability

Characteristic roots location and the corresponding characteristic modes [1/2]
Characteristic roots location
and the corresponding characteristic modes [2/2]