436-433 Mechanical Systems  
Part B: Digital Control  
Lecture BL6

- Transformations between state-space realisations  
- Modal form  
- Discrete analysis using MATLAB  
- Direct digital design: root locus
Transformations between state-space realizations

• As in the continuous case, different realizations of a discrete transfer function may be related by a non-singular transformation matrix:

\[ x(k) = Tw(k) \]

\[ x(k+1) = \Phi x(k) + \Gamma u(k) \]
\[ y(k) = Cx(k) \]

may be transformed to one with state variables \( w \):

\[ Tw(k+1) = \Phi Tw(k) + \Gamma u(k) \]
\[ y(k) = CTw(k) \]

i.e.,
\[ w(k+1) = T^{-1} \Phi Tw(k) + T^{-1} \Gamma u(k) \]
\[ y(k) = CTw(k) \]
**Modal form**

- The natural (unforced) motions of a discrete system

\[ x(k + 1) = \Phi x(k) \]

are of the form

\[ x(k) = pz^k \]

- Substituting this assumed form of solution:

\[ pz^{k+1} = zpz^k = \Phi pz^k \]

i.e., \[zp = \Phi p\]

- Hence

\[ [zI - \Phi]p = 0 \]

- The condition for this equation to have a non-trivial solution for the eigenvector (mode shape) \(p\) is

\[ \det[zI - \Phi] = 0 \]

- The solutions \(z = \lambda_i (i = 1, 2, \ldots, n)\) of this *characteristic equation* are the *eigenvalues* of the system
Modal form ...

- Given the eigenvalues $\lambda_i$, we can solve $[\lambda_i I - \Phi]p^{(i)} = 0$ for the mode shapes $p^{(i)}$.

- Provided the eigenvalues are all distinct, the modal matrix $P = [p^{(1)} p^{(2)} p^{(3)} \ldots p^{(n)}]$ is the transformation matrix to a modal (diagonal) realization:

  \[
  \begin{align*}
  w(k+1) &= \Lambda w(k) + P^{-1} \Gamma u(k) \\
  y &= CPw(k)
  \end{align*}
  \]

  where \[ \Lambda = P^{-1} \Phi P = 
  \begin{bmatrix}
  \lambda_1 & 0 & \cdots & 0 \\
  0 & \lambda_2 & \cdots & 0 \\
  \vdots & \vdots & \ddots & \vdots \\
  0 & 0 & \cdots & \lambda_n
  \end{bmatrix} \]
Complex eigenvalues

• Note that if some eigenvalues occur in complex conjugate pairs, it is more convenient to work with coupled 'second-order modes'

• Thus, for example, if we have eigenvalues

$$\lambda_1 = \alpha + j\beta \text{ and } \lambda_2 = \lambda_1^* = \alpha - j\beta$$

with corresponding complex eigenvectors

$$p^{(1)} = u + jv \text{ and } p^{(2)} = p^{(1)*} = u - jv$$

we use $u$ and $v$ as columns in the modal matrix, in place of $p^{(1)}$ and $p^{(2)}$

• Then, the 'diagonalized' $\Phi$ matrix will be of the form

$$\Lambda = P^{-1}\Phi P = \begin{bmatrix} \alpha & \beta & \cdots & 0 \\ -\beta & \alpha & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}$$

corresponding to coupled state equations

$$w_1(k+1) = \alpha w_1(k) + \beta w_2(k)$$
$$w_2(k+1) = -\beta w_1(k) + \alpha w_2(k)$$
MATLAB commands
Example: amp/motor considered previously

\[ G(s) = \frac{5}{s(s^2 + 2s + 5)} \]

- Continuous LTI object
  ```matlab
  num = 5; den = [1 2 5 0];
  G = tf(num, den)
  Transfer function:
  5
  ------------------
  s^3 + 2 s^2 + 5 s
  ```

- Display poles and zeros
  ```matlab
  G = zpk(G)
  Zero/pole/gain:
  5
  ------------------
  s (s^2 + 2s + 5)
  ```
Discretization with \texttt{c2d}

- The function \texttt{c2d} provides several discretization options ('\texttt{zoh}', '\texttt{foh}', '\texttt{tustin}', ...)
- The default '\texttt{zoh}' is appropriate for getting a discrete model of a continuous plant (from DAC to ADC)

```matlab
>> T = 0.1; D = c2d(G, T); % gives discrete ZPK object
Zero/pole/gain:
0.00079133 (z+3.543) (z+0.2554)
----------------------------------------
(z-1) (z^2 - 1.774z + 0.8187)
Sampling time: 0.1
```
• **Discretization of state-space model**
  - the continuous model provided to `c2d` should be the appropriate state-space realization
  - e.g., suppose the amp/motor model is derived thus:

  ![Block Diagram](image)

  - Appropriate state variables are \( x = [i_a \, \omega \, \theta]^T \), yielding state equations:

  \[
  \begin{bmatrix}
  i_a \\
  \dot{\omega} \\
  \dot{\theta}
  \end{bmatrix} =
  \begin{bmatrix}
  -2 & -1.25 & 0 \\
  4 & 0 & 0 \\
  0 & 1 & 0
  \end{bmatrix}
  \begin{bmatrix}
  i_a \\
  \omega \\
  \theta
  \end{bmatrix}
  +
  \begin{bmatrix}
  1.25 \\
  0 \\
  0
  \end{bmatrix}u
  \]

  \[
  y = \begin{bmatrix}
  0 & 0 & 1
  \end{bmatrix}x
  \]
MATLAB commands ...

• Then, create MATLAB continuous \texttt{ss} object:
  » \texttt{a = [-2 -1.25 0; 4 0 0; 0 1 0];}
  » \texttt{b = [1.25 0 0]'; c = [0 0 1]; d = 0;}
  » \texttt{Gss = ss(a, b, c, d);}
  » \texttt{set(Gss, 'InputName','U', 'OutputName','Q')}
  » \texttt{set(Gss, 'StateName',{'Ia','W', 'Q'})}

• Check the transfer function:
  » \texttt{Gzpk = zpk(Gss)}
    
    \texttt{Zero/pole/gain from input "U" to output "Q":}
    
    \texttt{5}
    
    \texttt{------------------}
    
    \texttt{s (s^2 + 2s + 5)}

• Discretize:
  » \texttt{T = 0.1; Dss = c2d(Gss, T)}
- Discretize:

```matlab
» T = 0.1; Dss = c2d(Gss, T)

a =

\begin{bmatrix}
Ia & W & Q \\
Ia & 0.79692 & -0.11235 & 0 \\
W & 0.35953 & 0.97668 & 0 \\
Q & 0.018654 & 0.099209 & 1 \\
\end{bmatrix}

b =

\begin{bmatrix}
U \\
Ia & 0.11235 \\
W & 0.023317 \\
Q & 0.00079133 \\
\end{bmatrix}

c =

\begin{bmatrix}
Ia & W & Q \\
Q & 0 & 0 & 1 \\
\end{bmatrix}

d =

\begin{bmatrix}
U \\
Q & 0 \\
\end{bmatrix}

Sampling time: 0.1
Discrete-time system.
```
Matlab commands ...

- Of course, the same discrete transfer function as before can be obtained from the `ss` model:

```matlab
» Dzpk = zpk(Dss)
Zero/pole/gain from input "U" to output "Q":
0.00079133 (z+3.543) (z+0.2554)
----------------------------------------------
(z-1) (z^2 - 1.774z + 0.8187)
Sampling time: 0.1
```
Modal form

```matlab
» [Dd, T1] = canon(Dss, 'modal')

a =

\[
\begin{bmatrix}
1 & 0 & 0 \\
0 & 0.8868 & 0.17976 \\
0 & -0.17976 & 0.8868
\end{bmatrix}
\]

b =

\[
\begin{bmatrix}
0.1 \\
0.15565 \\
0.24717
\end{bmatrix}
\]

c =

\[
\begin{bmatrix}
0 & 0 & 1 & -0.29091 & -0.21818 \\
0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

d =

\[
\begin{bmatrix}
0 \\
0
\end{bmatrix}
\]

Sampling time: 0.1

Discrete-time system.

\[\lambda_1 = 1\]

\[
\begin{bmatrix}
\alpha & \beta \\
-\beta & \alpha
\end{bmatrix}
\]

from \[\lambda_{2,3} = \alpha \pm j\beta\]
Modal matrix

» [Dd, T1] = canon(Dss, 'modal')
returns the matrix $T_1$ in the transformation $w = T_1x$. Hence, to get the modal matrix we have been using ($x = Pw$):

» $P = \text{inv}(T1)$

$$P =
\begin{bmatrix}
0 & 0 & 0.4545 \\
0 & 0.7273 & -0.3636 \\
1.0000 & -0.2909 & -0.2182
\end{bmatrix}$$

• Look at the eigenvectors and eigenvalues:

» [eVec, Lambda] = eig(Dss.a)

$$eVec = e^{(1)}$$

$u$

$$v$$

$$\begin{bmatrix}
0 & 0 + 0.4545i & 0 - 0.4545i \\
0.7273 & -0.3636i & 0.7273 + 0.3636i \\
-0.2909 & -0.2182i & -0.2909 + 0.2182i
\end{bmatrix}$$

$$\begin{bmatrix}
1.0000 & 0 & 0 \\
0 & 0.8868 + 0.1798i & 0 \\
0 & 0 & 0.8868 - 0.1798i
\end{bmatrix}$$
Direct digital design

- We can use classical design techniques
  - root locus method: just as for continuous systems
  - frequency response methods: requires some modification of continuous method

- We can also use state-space methods
  - pole placement by full state feedback
  - estimation of states
  - regulator and tracking systems

essentially as for continuous systems
Discrete root locus

- Closed-loop transfer function

\[
\frac{Y(z)}{R(z)} = \frac{G_c(z)G_p(z)}{1 + G_c(z)G_p(z)}
\]

- Characteristic equation

\[1 + G_c(z)G_p(z) = 0\]

is in exactly same form as for continuous analysis
Discrete root locus ...

- Let the open-loop transfer function be expressed in Evans' form:

\[
G_c(z)G_p(z) = \frac{K \prod_{i=1}^{m} (z - z_i)}{\prod_{j=1}^{n} (z - p_j)} = \frac{KB(z)}{A(z)}
\]

- Then the characteristic equation of the closed-loop system is

\[
1 + K \frac{B(z)}{A(z)} = 0
\]

or \( K \frac{B(z)}{A(z)} = -1 \)

- Thus, values of \( z \) which are solutions of the characteristic equation satisfy:
  - the magnitude condition \( \left| \frac{KB(z)}{A(z)} \right| = 1 \)
    - Used to associate a root with a gain
  - the phase condition \( \text{ph} \left( \frac{KB(z)}{A(z)} \right) = 180^\circ + 360^\circ l \) (\( l \) is any integer)

Sufficient to define the locus of roots as \( K \) varies from 0 to \( \infty \).
Discrete root locus ...

- Thus, the rules for drawing a discrete root locus are precisely as for the continuous case
  - it is just the association of pole location with transient response which differs

\[ z = e^{sT} = e^{\sigma T} e^{j\omega T} = re^{j\theta} \]

- \( \omega_n = 0.9\omega_N \)
- \( \omega_n = \omega_N \)
- \( \omega_n = 0.1\omega_N = 0.1\pi/T \)
- \( \omega_N = \omega_s/2 = \pi/T \) Nyquist freq.
- \( \zeta = 0.1 \)
- \( \zeta = 0.9 \)
- \( \omega_d = 0.3\omega_N \)
- \( \zeta\omega_n = \text{const} \)
- \( \zeta = 0 \)
Design procedure

- Specify acceptable regions for dominant closed-loop poles based on required transient responses
  - $\zeta, \omega_n$ from continuous time response specs
  - settling time $t_s \approx 4 / \zeta \omega_n \Rightarrow \sigma = -\zeta \omega_n \Rightarrow r = e^{-\zeta \omega_n T}$
  - peak overshoot $M_p = e^{-\pi \zeta / \sqrt{1-\zeta^2}} \Rightarrow \zeta$
  - rise time $t_r \approx 1.8 / \omega_n \Rightarrow \omega_n$
- Steady-state error specs
Steady-state errors

- z-transform of error:
  \[ E(z) = \frac{R(z)}{1 + G_c(z)G_p(z)} \]

- Unit step input:
  \[ R(z) = \frac{z}{z-1} \]

- Final value theorem:
  \[ e(\infty) = \lim_{z \to 1} (z - 1)E(z) \]

  \[ = \lim_{z \to 1} \frac{z}{z-1} \frac{1}{1 + G_c(z)G_p(z)} \]

  \[ = \frac{1}{1 + G_c(1)G_p(1)} \]

Type 1 system

s.s. error for unit ramp input \( r(k) = kT \cdot 1(k) \)

is \( 1/K_v \)

\[ K_v = \lim_{z \to 1} \frac{(z-1)G_c(z)G_p(z)}{Tz} \]

Discrete compensator

ZOH discretization of plant

- 1/(1+\(K_p\)) for Type 0 system
- 0 for Type 1 system
  (one pole at \( z = 1 \))