Mathematical Models of Dynamic Systems

Methods for modeling dynamic systems

- **Analytical method (physical modeling)**: Apply physical laws (Newton's laws of motion, Kirchoff' law for electric circuits, electromagnetics laws, etc.) to derive the mathematical relationship between the inputs and the outputs of the system

- **Experimental method (system identification)**: Collect input and output values of the system over time to derive expressions describing the input-output relationship

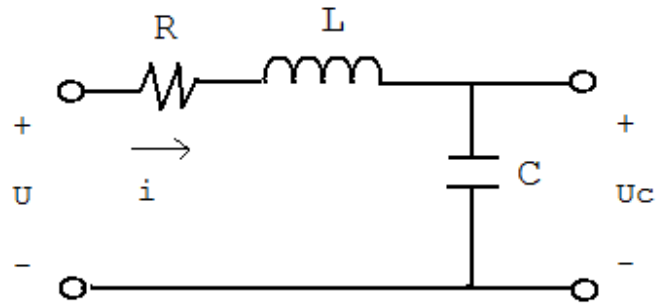
The system model can be used for:

- **System analysis:** To determine system stability, controlability, observability, …
- **System simulation:** To predict system response and understand its behavior
- **Control design:** To modify system response to various commands as desired

Analytical Method of Modeling

In general, the mathematical model of a continuous-time, lumped parameter dynamic system can be represented by an ordinary differential equation (ODE)

Example: For the RLC circuit, describe the behavior of the output u_c versus the input u.



Derive the model:

KVL: $u = u_R + u_L + u_C;$

where $u_R = Ri;\ \ u_L = L\frac{di}{dt};$

$i_C = C\left(\frac{du_c}{dt}\right);$ and $i = i_C$

Substitute the above into the KVL equation:

$$u = LC\left(\frac{d^2u_c}{dt^2}\right) + RC\left(\frac{du_c}{dt}\right) + u_C$$

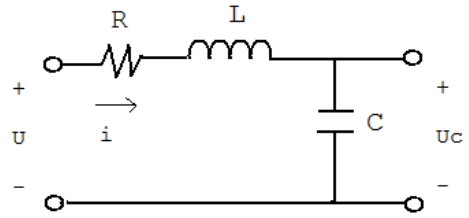
or

$$\boldsymbol{u = LC\,\ddot{u}_C + RC\,\dot{u}_C + u_C} \qquad (2^{nd} \text{ order ODE})$$

Laplace Transform and Transfer Function

Apply Laplace transform (assuming zero initial conditions):

$$U(s) = [LCs^2 + RCs + 1]\, U_C(s)$$



Transfer function:

$$G(s) = \frac{U_C(s)}{U(s)} = \frac{\mathbf{1}}{[\boldsymbol{LCs^2+RCs+1}]}$$

MATLAB: The Matlab function laplace() in Symbolic Toolbox can be used to find the Laplace transform of a given function

Example: Find the Laplace transform of the function $f = e^{bt}\cos(at + c)$

```
>> syms s t a b c;   F=laplace(exp(b*t)*cos(a*t+c)
```

And the result is:

$$F(s) = \frac{\cos(c)(s-b)}{(s-b)^2+a^2} - \frac{\sin(c)a}{(s-b)^2+a^2}$$

Similarly, the inverse-Laplace transform of F(s) can be found using the function ilaplace() in the Symbolic Toolbox

```
>> f=ilaplace(F)
```

Model Representation in Matlab

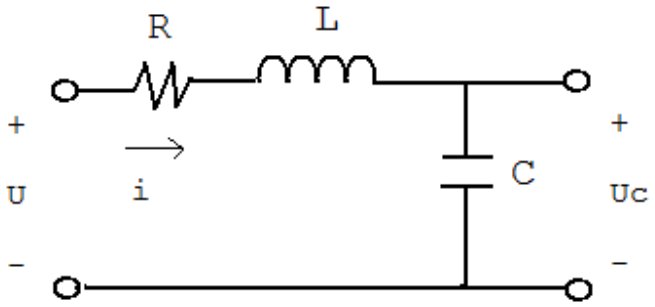
Transfer function representation in Matlab

For the RLC example, using component values (R=100Ω, L=1H, C=1F):

```
>> R=100; L=1; C=1;
>> G=tf([1],[L*C R*C 1])
```

Or, first define the Laplace variable and then write the transfer function

```
>> s=tf('s');
>> G=1/(L*C*s^2+R*Cs+1)
```



Either way results in:

```
Transfer function:
       1
--------------
s^2 + 100 s + 1
```

The complex variable 's' could be replaced, as:

```
>> G.variable='p'

Transfer function:
       1
--------------
p^2 + 100 p + 1
```

Modification of Models in Matlab

To add a time delay to the TF:

```
>> G.iodelay=0.5

Transfer function:
                        1
exp(-0.5*p) * ---------------
              p^2 + 100 p + 1
```

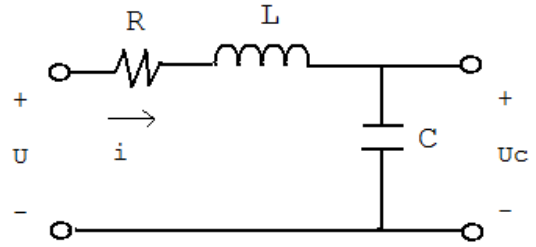
To convert the continuous TF, G(s), to a discrete TF, Gd(z), with sample-time, T=0.01 sec:

```
>> Gdisc=c2d(G,0.01)

Transfer function:
           3.679e-005 z + 2.642e-005
z^(-50) * -------------------------
            z^2 - 1.368 z + 0.3679

Sampling time (seconds): 0.01
```

Other Mathematical Model Forms

- **Various system model representations, and Matlab commands:**
  - Transfer function model: `SYS = tf([num],[den])`
  - Zero-Pole-Gain model: `SYS = zpk(Z,P,K)`
  - State space model: `SYS = ss(A,B,C,D)`
  - Block diagram representation

Example: RLC circuit



- Its state-space model is (given $x_1 = u_c, x_2 = \dot{u}_c$):

$$\begin{cases} \begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & \frac{1}{C} \\ -\frac{1}{L} & -\frac{R}{L} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \frac{1}{L} \end{bmatrix} u \\ y = [1 \quad 0] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + [0]u \end{cases}$$

Matlab code:

```
>> R=100; L=1; C=1;
>> A=[0,1/C;-1/L,-R/L];
>> B=[0;1/L]; C=[1,0]; D=0;
>> sys=ss(A,B,C,D);
```

Result:

```
a =
          x1     x2
   x1   -100     -1
   x2      1      0
b =
         u1
   x1     1
   x2     0
c =
         x1   x2
   y1     0    1
d =
         u1
   y1     0
```

Converting Models to Different Forms

Converting transfer function to zero-pole-gain (zpk) form:

```
>> Gzpk=zpk(G)

Zero/pole/gain:
        1
------------------
(s+99.99) (s+0.01)
```

Converting transfer function to state-space form:

```
>> sys=ss(G)

a =
         x1    x2
   x1  -100    -1
   x2     1     0
b =
        u1
   x1    1
   x2    0
c =
        x1  x2
   y1    0   1
d =
        u1
   y1    0
```

Retrieving state-space model data:

```
>> [A B C D]=ssdata(sys)
```

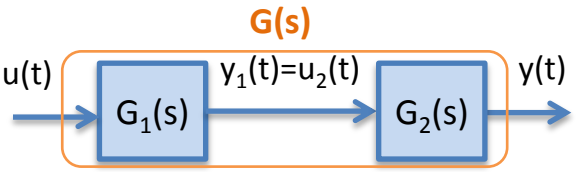
or

```
>> [num,den]=tfdata(G,'v');
>> [A B C D]=tf2ss(num,den)
```

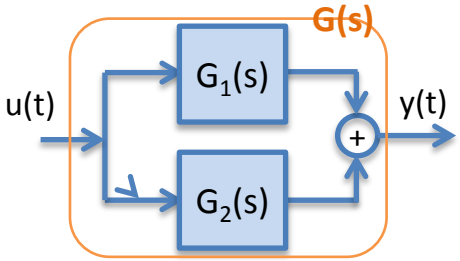
Block Representation

Block algebra:

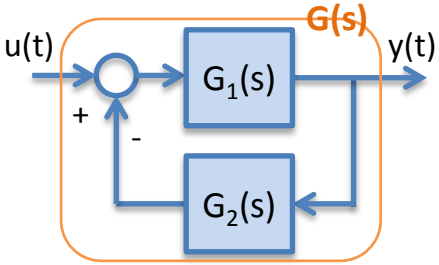
- Series connection: $G(s) = G_1(s)G_2(s)$
  - `G=series(G1,G2)`



- Parallel connection: $G(s) = G_1(s) + G_2(s)$
  - `G=parallel(G1,G2)`

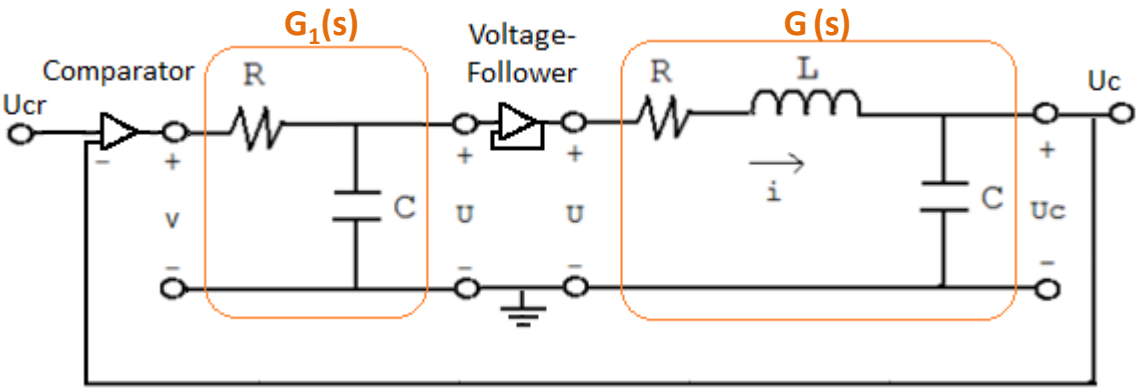


- Feedback connection: $G(s) = \frac{G_1(s)}{1+G_1(s)G_2(s)}$
  (negative feedback)
  - `G=feedback(G1,G2,-1)`



Block Algebra

- **Connect the two circuit blocks in negative feedback structure**
  - Find the closed-loop transfer function (TF)
    - **Note**: There is a buffer (voltage-follower) for one-directional flow



Matlab code:

```
>> G1=1/(R*C*s+1)
>> Tcl=feedback(series(G1,G),-1)
Transfer function:
              1
---------------------------
100 s^3 + 10001 s^2 + 200 s
>> Tcl=minreal(Tcl);          % To perform pole-zero cancellation
```

Function balreal() finds a balanced realization of a stable system
Function minreal() finds the minimal realization of a system

Equivalent State-Space Representation

- **A system can have many state-space (ss) models**
  - These models are all equivalent.
  - Can convert from one to another using an invertible similarity transformation matrix T.
- **Canonical ss models have minimum number of variables**
  - Companion form
  - Modal (block-diagonal) form

- **Example: (converting plant model sys to equivalent modal form sys_m)**

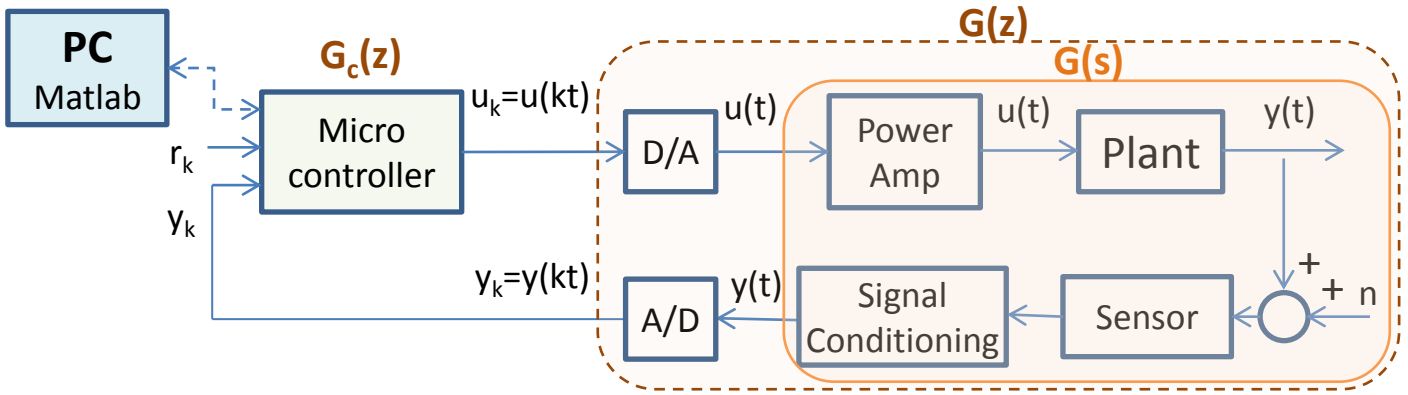
```
>> [syst_m,T]=canon(sys,'modal')

a =
            x1       x2
   x1   -99.99        0
   x2        0    -0.01
b =
            u1
   x1       -1
   x2    -0.01
```

```
c =
             x1        x2
   y1      0.01   -0.9999
d =
         u1
   y1    0
Continuous-time model.
T =
   -1.0002    -0.0100
   -0.0100    -1.0002
```

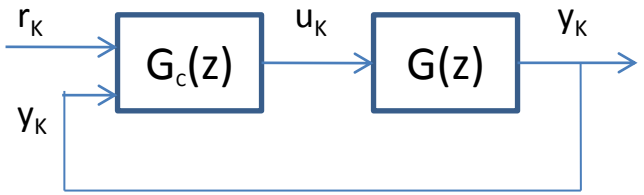
Similarity Transformation matrix, T

Embedded Control Systems

- **Basic components of an embedded control system**:
  - Analog systems (plant, sensors, power-amps, …)
  - Digital microcontroller
  - PC (programming and monitoring environment, such as Matlab)



- **Viewed from the perspective of the controller:**
  - Discrete system and control models
- **Goal:**
  - Model G(z)
  - Design $G_c(z)$



Data-Based Modeling

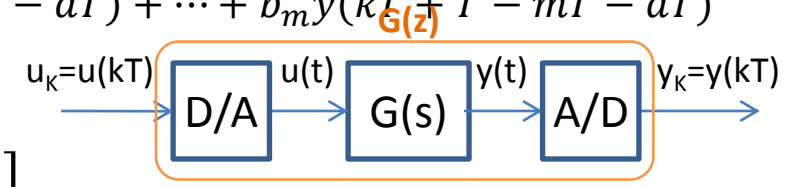
Z-transform of discrete signals: $z = e^{Ts}, \quad z^{-1} = e^{-Ts}$

$$\mathfrak{Z}\{y(kT)\} = Y(z); \quad \mathfrak{Z}\{y(kT-T)\} = z^{-1}Y(z); \quad \mathfrak{Z}\{y(kT+T)\} = zY(z) - zy(0)$$

Given a discrete TF: $\frac{Y(z)}{U(z)} = G(z) = \frac{b_1 + b_2 z^{-1} + \cdots + b_m z^{-m+1}}{1 + a_1 z^{-1} + a_2 z^{-2} + \cdots + a_n z^{-n}} z^{-d}$

The corresponding difference equation:

$$y(kT) + a_1 y(kT-T) + \cdots + a_n y(kT-nT) = b_1 u(kT-dT) + \cdots + b_m y(kT+T-mT-dT)$$



Collected input-output data: $U = \begin{bmatrix} u_1 \\ \vdots \\ u_M \end{bmatrix}; \; Y = \begin{bmatrix} y_1 \\ \vdots \\ y_M \end{bmatrix};$

Arrange difference equation at these points: $Y = \phi\theta + \varepsilon$

$$\theta = [-a_1 \quad \dots \quad -a_n \quad b_1 \quad \dots \quad b_m]^T;$$

$$\phi = \begin{bmatrix} y(0) & \dots & y(1-n) & u(1-d) & \dots & u(2-m-d) \\ & \vdots & & & \vdots & \\ y(M-1) & \dots & y(M-n) & u(M-d) & \dots & u(M+1-m-d) \end{bmatrix};$$

Residual error: $\varepsilon = [\varepsilon(1) \quad \dots \quad \varepsilon(M)]^T;$

Least-square estimation of θ:

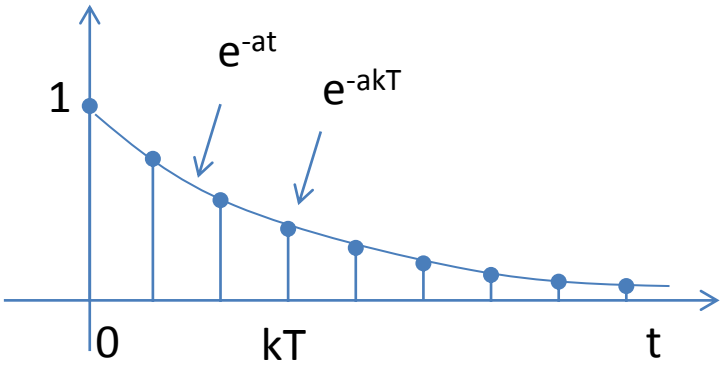
- If the matrix $(\phi^T\phi)$ has full rank (it is invertible), then estimate of θ is:

$$\hat{\theta} = (\phi^T\phi)^{-1}\phi^T y$$

Conversion Between Continuous and Discrete-Time Signals

- Z-transform converts continuous signals into discrete signals

Example:



Continuous signal:

$$e(t) = e^{-at}\,\mathbf{1}(t)$$

Discretized signal, at sample-time t=kT:

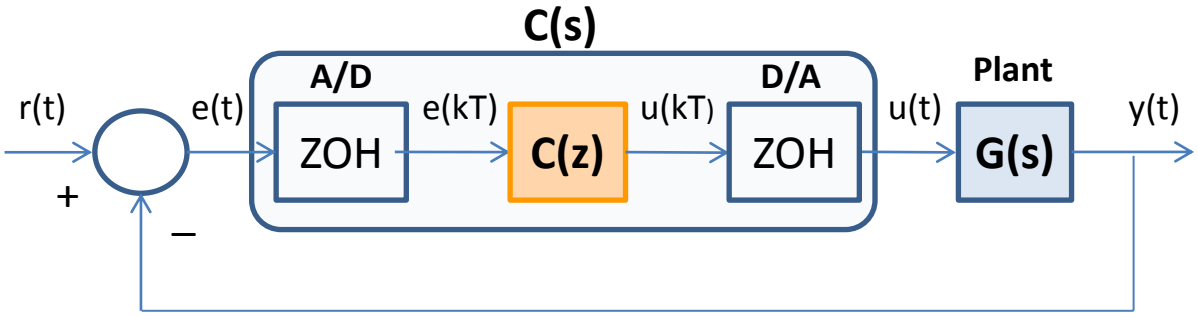
$$e(kT) = e^{-akT}\,\mathbf{1}(kT)$$

Z-transform of the discrete signal:

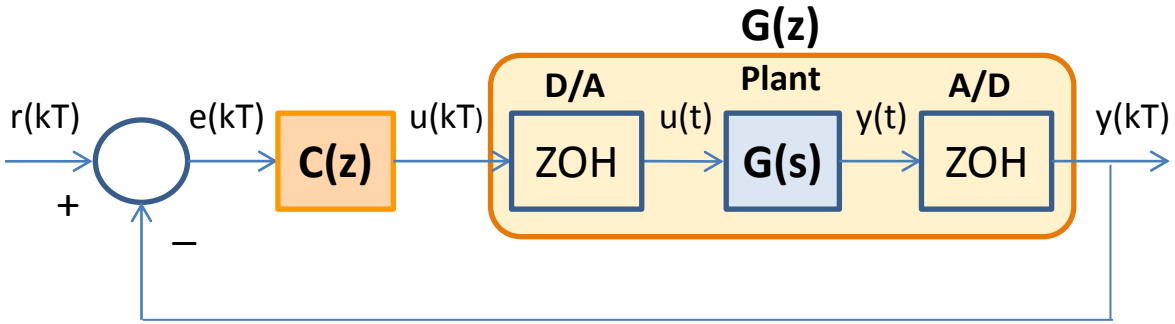
$$\begin{aligned} E(z) &= \mathfrak{Z}\{e(kT)\} \\ &= \sum_{k=-\infty}^{\infty} e(kT)z^{-k} \\ &= \sum_{k=0}^{\infty} e^{-akT}z^{-k} \\ &= \sum_{k=0}^{\infty} (e^{-aT}z^{-1})^{k} \\ &= \frac{1}{1-e^{-aT}z^{-1}} \\ &= \frac{z}{z-e^{-aT}} \qquad , \quad \text{for } |z| > e^{-aT} \end{aligned}$$

Modeling of Sampled-Data Systems

- Closed-loop sampled-data system
  - Equivalent continuous structure



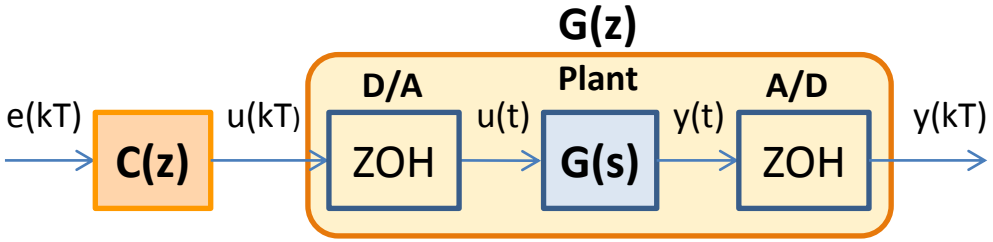
  - Equivalent discrete structure



ZOH = zero-order hold

Conversion Between Continuous and Discrete-Time Models

Z-transform of a continuous system preceded by a D/A (ZOH) and followed by an A/D (sampler)



$$\Rightarrow G(z) = \mathfrak{Z}\left\{\left(\frac{1-e^{-Ts}}{s}\right) G(s)\right\} = \mathfrak{Z}\left\{(1-e^{-Ts})\,\frac{G(s)}{s}\right\} = (1-z^{-1})\,\mathfrak{Z}\left\{\frac{G(s)}{s}\right\}$$

Example:

Continuous transfer function: $G(s) = \frac{a}{s+a}$

$$\Rightarrow \quad G(z) = (1-z^{-1})\,\mathfrak{Z}\left\{\frac{a}{s\,(s+a)}\right\}$$

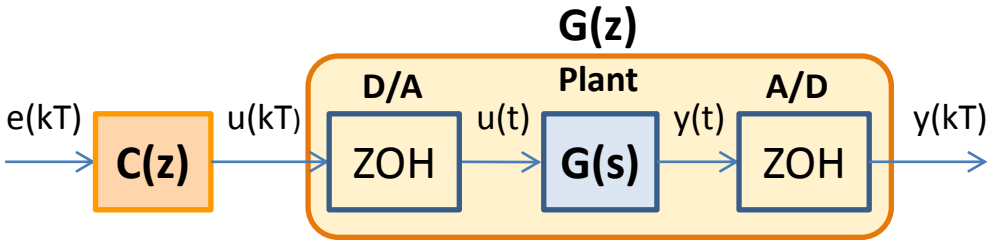
$$= (1-z^{-1})\,\mathfrak{Z}\left\{\frac{1}{s} - \frac{1}{s+a}\right\}$$

$$= (1-z^{-1})\left(\frac{z}{z-1} - \frac{z}{z-e^{-aT}}\right)$$

$$= \frac{1-e^{-aT}}{z-e^{-aT}}$$

- For a discrete system to be stable its poles should lie inside the unit circle.
- If any pole lies outside the unit circle the system will be unstable.

Parameter Identification of Discrete-Time Systems



Example:

$$G(z) = \frac{Y(z)}{U(z)} = \frac{z+1}{z^2 - 0.2\,z + 1} = \frac{z^{-1}+z^{-2}}{1-0.2z^{-1}+z^{-2}}$$

G(s)= Y(s)/U(s) | I.C=0

G(z)=Y(z)/U(z) | I.C=0

$$\Rightarrow (z^2 - 0.2\,z + 1)\,Y(z) = (z+1)\,U(z)$$
$$\Rightarrow (1 - 0.2\,z^{-1} + z^{-2})\,Y(z) = (z^{-1} + z^{-2})U(z)$$

In time-domain:

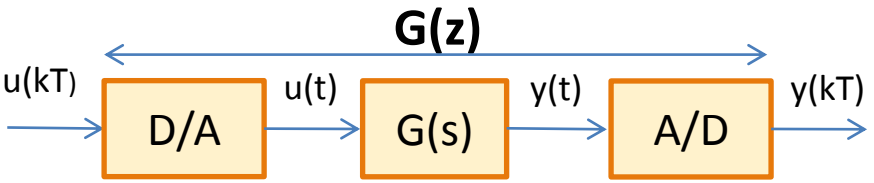
$$\Rightarrow y(k) - 0.2\,y(k-1) + y(k-2) = u(k-1) + u(k-2)$$

or:

$$\Rightarrow y(k) = [y(k-1) \quad y(k-2) \quad u(k-1) \quad u(k-2)] \begin{bmatrix} 0.2 \\ -1 \\ 1 \\ 1 \end{bmatrix}$$

Parameter Identification of Discrete-Time Systems …

More generally:



Discrete transfer function (z-domain):

$$G(z) = \frac{Y(z)}{U(z)} = \frac{b_1 z + b_0}{z^2 + a_1 z + a_0} = \frac{b_1 z^{-1} + b_0 z^{-2}}{1 + a_1 z^{-1} + + a_0 z^{-2}}$$

$$\Rightarrow (z^2 + a_1 z \ + a_0)\, Y(z) = (b_1 z + b_0)\, U(z)$$
$$\Rightarrow (1 + a_1 z^{-1} + + a_0 z^{-2})\, Y(z) = (b_1 z^{-1} + b_0 z^{-2}) U(z)$$

Difference equation (time-domain):

$$\Rightarrow y(k) + a_1 y(k-1) + a_0 y(k-2) = b_1 u(k-1) + b_0 u(k-2)$$

or:

$$\Rightarrow y(k) = [y(k-1) \quad y(k-2) \quad u(k-1) \quad u(k-2)] \begin{bmatrix} -a_1 \\ -a_0 \\ b_1 \\ b_0 \end{bmatrix}$$

Parameter Identification of Discrete-Time Systems …

Algebraic equation:

Writing the difference equation for various k, we get:

k=0 $\Rightarrow$ $$y(1) = [y(0) \quad y(-1) \quad u(0) \quad u(-1)] \begin{bmatrix} -a_1 \\ -a_0 \\ b_1 \\ b_0 \end{bmatrix}$$

k=1 $\Rightarrow$ $$y(2) = [y(1) \quad y(0) \quad u(1) \quad u(0)] \begin{bmatrix} -a_1 \\ -a_0 \\ b_1 \\ b_0 \end{bmatrix}$$

…

Which can be written as:

$$\begin{bmatrix} y(1) \\ y(2) \\ \vdots \end{bmatrix} = \begin{bmatrix} y(0) & y(-1) & u(0) & u(-1) \\ y(1) & y(0) & u(1) & u(0) \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} -a_1 \\ -a_0 \\ b_1 \\ b_0 \end{bmatrix}$$

Equivalently:

$$Y = \Phi\, \theta$$

Parameter Identification of Discrete-Time Systems …

Given: $Y = \Phi\,\theta + \varepsilon$

Minimize: $J = \min_{\theta} \sum_{i}^{M} {\varepsilon_i}^2 = \min_{\theta} \varepsilon^T \varepsilon = \min_{\theta} (Y - \Phi\theta)^T (Y - \Phi\theta)$

Estimate of parameter vector θ (using **Least Mean Squared** algorithm):

$$\theta = (\Phi^T \Phi)^{-1} \Phi^T\, Y$$

Recursive estimate of parameter vector θ (using **gradient** technique):

- **Continuous:** $\dot{\theta} = -\gamma \frac{\partial J}{\partial \theta} = \gamma\, \Phi^T \varepsilon \;\;,\quad \gamma > 0$
- **Discrete:** $\theta(k+1) = \theta(k) - \gamma_d \frac{\partial J}{\partial \theta} = \theta(k) + \gamma_d \Phi^T(k)\varepsilon(k) \;\;,\quad \gamma_d > 0$

MATLAB Commands:

- Converting $G(s)$ *to* $G(z)$: `c2d(sys)`
- Generating and collecting I/O samples, U and Y: `lsim()`
- Identification of the system TF, $G(z) = \frac{b(z^{-1})}{a(z^{-1})} z^{-d}$:
  - Using System Identification toolbox GUI: `ident()`
  - Using System Identification command: `arx()`
    - `H=arx([Y U],[n m d])`
    - n=# of terms in a(z), m=#of terms in b(z), d=delay