EEN1047
CONTROL SYSTEMS ANALYSIS

Dr Brendan Hayes & Dr Mingming Liu

School of Electronic Engineering
Dublin City University

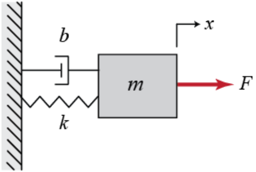
Semester 1 2024-2025

Section 3

Linear Systems Review

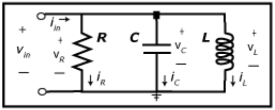
3.1 Physical Systems Behaviour

- Physical systems can be modelled using differential equations
  - Spring–mass–damper system:



$$m.\frac{d^2x(t)}{dt^2} + b.\frac{dx(t)}{dt} + k.x(t) = F(t)$$

  - Electrical circuits:



$$\frac{v_L(t)}{R} + C.\frac{dv_L(t)}{dt} + \frac{1}{L}.\int_0^t v_L(t).dt = i(t)$$

3.2 State Space Models

- State space models can be formed from these original differential equation models.
- **State space** models are a set of input, output and state variables related by first–order differential equations.
- State-space models can be used for models that are not linear, where initial conditions are not zero, and/or there are multiple inputs/outputs.
- For continuous, LTI systems, the **general state space description** is:

$$\begin{aligned} \dot{\mathbf{x}}(t) &= A\mathbf{x}(t) + B\mathbf{u}(t) \\ \mathbf{y}(t) &= C\mathbf{x}(t) + D\mathbf{u}(t) \end{aligned} \tag{1}$$

- The **eigenvalues** of the system matrix A indicate important performance behaviour of the system.
- The characteristic equation can be found using:

$$\lambda(s) = |\lambda\mathcal{I} - A| = 0$$

3.3 Laplace Transform

- The time response can be difficult to obtain from differential equation models.
- The Laplace transform allows for the substitution of algebraic equations for these differential equations.
- It does this by transforming the time-domain model into a frequency-domain model.
- **Definition:**

$$F(s)=\mathcal{L}\{f(t)\}=\int_0^\infty e^{-st}f(t)dt, \quad s=\sigma+j\omega, \quad \sigma,\omega\in\Re$$

- Example for spring–mass–damper:

3.3 Laplace Transform

- The time response can be difficult to obtain from differential equation models.
- The Laplace transform allows for the substitution of algebraic equations for these differential equations.
- It does this by transforming the time-domain model into a frequency-domain model.
- **Definition:**

$$F(s)=\mathcal{L}\{f(t)\}=\int_0^\infty e^{-st}f(t)dt, \quad s=\sigma+j\omega, \quad \sigma,\omega\in\Re$$

- Example for spring–mass–damper:

Laplace Transform Example

$$F(s) = m\left(s^2X(s)-sx(0^-)-\frac{dx(0^-)}{dt}\right)+b\Big(sX(s)-x(0^-)\Big)+kX(s)$$

Assuming zero initial conditions:

$$F(s) = ms^2X(s) + bsX(s) + kX(s)$$

3.4 Transfer Function Models

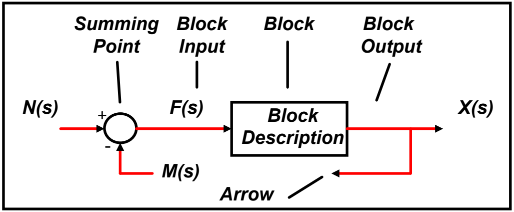
- **Definition:** a **Transfer Function** is defined as the ratio of the Laplace transforms of the output variable to the input variable, assuming *all initial conditions are zero*.
- Can only be defined for linear, time-invariant (**LTI**) systems.

3.4 Transfer Function Models

- **Definition:** a **Transfer Function** is defined as the ratio of the Laplace transforms of the output variable to the input variable, assuming *all initial conditions are zero*.
- Can only be defined for linear, time-invariant (**LTI**) systems.
- Given:
$$G(s) = \frac{Y(s)}{U(s)} = \frac{Num(s)}{Den(s)},$$
  the denominator of the TF is called the **characteristic polynomial**, $Den(s)$.
- The **characteristic equation** is defined by letting the characteristic polynomial equal to zero, $Den(s) = 0$.
- The roots of this characteristic equation are called **poles**
  - These poles occur as real numbers or complex conjugate pairs, they are values for s.
- The roots of $Num(s) = 0$ are called the **zeros** of the TF.

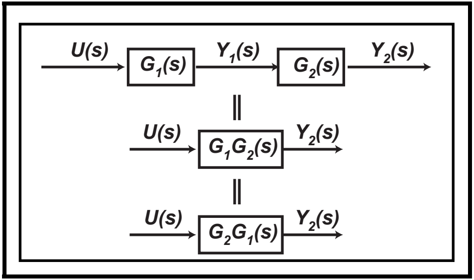
3.5 Block Diagrams

- Large systems may consist of many components, each of which may be represented by a tranfer function.
- **Block diagrams** are used as diagrammatical means of representing the relationships between the inputs and outputs of different systems.
- They consist of uni-directional blocks of interest.



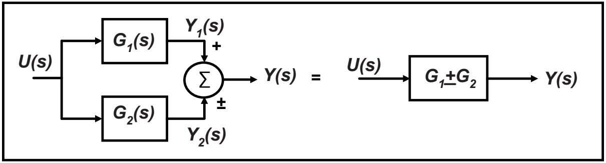
3.6 Blocks in Series

- Complicated block diagrams can be simplified or reduced by combining blocks together using block diagram algebra.
- There are 3 basic operations, depending on the connection between the blocks.
- Blocks that are in **series (cascade connection)** are multiplied together - if individual systems are linear, then multiplication is commutative.

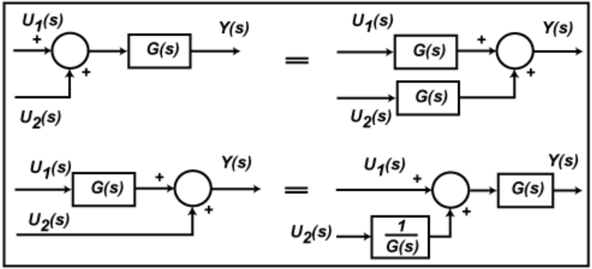


3.7 Blocks and Summers

- Blocks that are in **parallel** are added together - if individual systems are linear, then addition is commutative and distributive

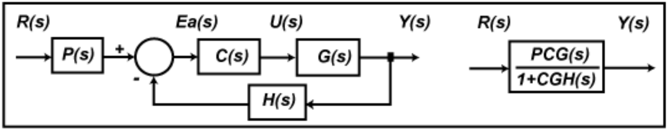


- Summation points can be moved in a diagram



3.8 Block Diagram Feedback

- A **feedback connection** occurs when the output of one block is fed back to the input of an earlier block.
- Loops in block diagrams are common and can be eliminated/simplified.
- Example: **the canonical block diagram**



  - This is the most common block diagram for systems and control.
  - The simplified representation represents the complete feedback system; it is the **Closed Loop Transfer Function, CLTF**.
  - The path from the summer and the output is called the **Forward Path**
  - The path from the output to the summer is called the **Feedback Path**

3.9 Block Diagrams CLTF

- From the diagram above:

Derivation

$$Y(s) = C(s)G(s).E_a(s)$$

$$E_a(s) = P(s).R(s) - H(s).Y(s)$$

$$Y(s) = CG(P.R(s) - H.Y(s))$$

$$(1+CGH).Y(s) = PCG.R(s)$$

$$\text{CLTF: } T(s) = \frac{Y(s)}{R(s)} = P.\frac{CG}{1+CGH}$$

- $H(s)$ is the feedback transfer function; if $H(s) = 1$ the system is a **unity feedback system**.
- $P(s)$ is a pre-filter and is usually assumed to be unity.
- A unity feedback systems has CLTF:

$$T(s) = \frac{Y(s)}{R(s)} = \frac{CG}{1+CG}$$

3.10 MATLAB for Polynomials

- Polynomials (e.g. denominator of TF) are entered as matrices in MATLAB.
  - Enter polynomial coefficients into a matrix; coefficients are placed in the matrix starting with the coefficient of the highest power of s & ending with the coefficient of s^0 term.
  - **N.B.** - remember to put the coefficient of the highest power of s (typically 1) into the matrix for the polynomial in MATLAB.
  - There is always 1 term more than the order of the polynomial in the MATLAB matrix. MATLAB assumes that the last entry in the polynomial matrix is the coefficient of s^0 and intreprets the other coefficients accordingly.

3.10 MATLAB for Polynomials

- Polynomials (e.g. denominator of TF) are entered as matrices in MATLAB.
  - Enter polynomial coefficients into a matrix; coefficients are placed in the matrix starting with the coefficient of the highest power of s & ending with the coefficient of s^0 term.
  - **N.B.** - remember to put the coefficient of the highest power of s (typically 1) into the matrix for the polynomial in MATLAB.
  - There is always 1 term more than the order of the polynomial in the MATLAB matrix. MATLAB assumes that the last entry in the polynomial matrix is the coefficient of s^0 and intreprets the other coefficients accordingly.

Example

polyCoeffs = [6 7 −8 9]	$6x^3 + 7x^2 - 8x - 9$
deng = [3 2 5]	$3s^2 + 2s + 5$
numg = [1 0 5]	$s^2 + 5$
denh = [1 0 −1 0 0]	$s^4 - s^2$

3.11 MATLAB for Manipulating Polynomials

- The **conv** function multiplies two polynomials.
- The **roots** function finds the roots of a polynomial.
- The **poly** function finds a polynomial from the roots.

Example

```
polynom1 = [1 2 3];
polynom2 = [3 4 5];
polyTotal = conv(polynom1,polynom2); % [3 10 22 22 15]

rootsPolyTotal = roots(polyTotal);
% -1 +/- j1.41, -0.667 +/- j1.11

polyNew = poly([-1-0.5j -1+0.5j -0.58 -100]);
% [1 102.58 260.41 241.725 72.5]
polyNew2 = poly([-1-j*0.5; -1+j*0.5; -0.58; -100]); % same result
```

3.12 MATLAB for LTI Models

- The **tf** function creates a Transfer Function model.
- The **zpk** function creates a Zero-Pole model.
- The **ss** function creates a State-Space model.

Example

```
numPoly = [4 5];
denPoly = conv([1 3],[1 2]);
sysGtf = tf(numPoly,denPoly)
%      4 s + 5
%   -------------
%   s^2 + 5 s + 6
sysGzpk = zpk([-1.25],[-3 -2],4);
%   4 (s+1.25)
%   -----------
%   (s+3) (s+2)
aG = [-2 0.866;0 -3];bG = [0;2];cG = [-1.732 2];dG=0;
sysGss = ss(aG,bG,cG,dG);
sysGcheck = tf(sysGss);      % can convert between models
```

3.13 MATLAB for Model Data

- The attributes of the LTI objects can be accessed directly.
- The **tfdata, ssdata, zpkdata** functions also extract this data.
- The poles and zeros the LTI objects can be accessed directly with **pole, zero**.

Example

```
numSysG = sysGtf.num{1};
numSysG = cell2mat(sysGtf.num)
% [0 4 5]
gainG = sysGzpk.k;   % 4
sysGcMatrix = sysGss.c   % [-1.7320    2.0000]

[numGout,denGout] = tfdata(sysGtf,'v');
[aGout,bGout,cGout,dGout] = ssdata(sysGss);
[zGout,pGout,kGout] = zpkdata(sysGzpk,'v');
% or [zGout,pGout,kGout] = zpkdata(sysGtf,'v');

zeroGtf = zero(sysGtf);
poleGtf = pole(sysGtf);
```

3.14 MATLAB for Polynomials

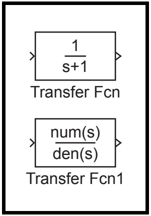
- It is very important **NOT** to use matrix operations with TFs.
- The **series** function multiplies (cascades) TFs, do not use *.
- The **feedback** functions calculates the overall closed-loop feedback TF in MATLAB; do not use *, \.

Example

```
sysCtf  = tf([1],[1 2]); \\
sysGtf  = tf([4],[1 3]); sysHtf = [4]; \\
sysCGtf = series(sysCtf, sysGtf); \\
%         4
%   ---------------
%    s^2 + 5s + 6
sysCLtf = feedback(sysCGtf, sysHtf);
%         4
%   ---------------
%    s^2 + 5s + 22
sysWrong = (sysGtf*sysCtf)/(1+sysGtf*sysCtf*sysHtf)
%             4s^2 + 20s + 24
%   --------------------------------
%    s^4 + 10s^3 + 53^2 +140s +132
```

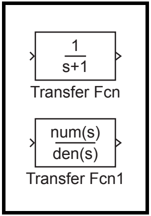
3.15 SIMULINK for LTI Models 1

- **Transfer Fcn block** (in SIMULINK continuous) - takes the numerator and denominator polynomials.

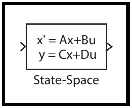


3.15 SIMULINK for LTI Models 1

- **Transfer Fcn block** (in SIMULINK continuous) - takes the numerator and denominator polynomials.



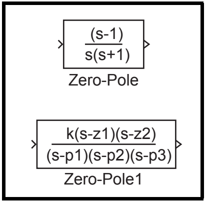
- **State–Space block** (in SIMULINK continuous) - takes the state space model matrices.



- Avoid typing model parameters directly into the SIMULINK blocks.

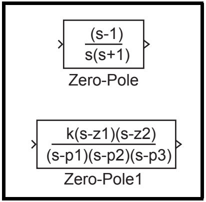
3.16 SIMULINK for LTI Models 2

- **Zero–Pole block** (in SIMULINK continuous) - takes the zeros, poles and gain of the model.

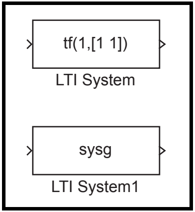


3.16 SIMULINK for LTI Models 2

- **Zero–Pole block** (in SIMULINK continuous) - takes the zeros, poles and gain of the model.

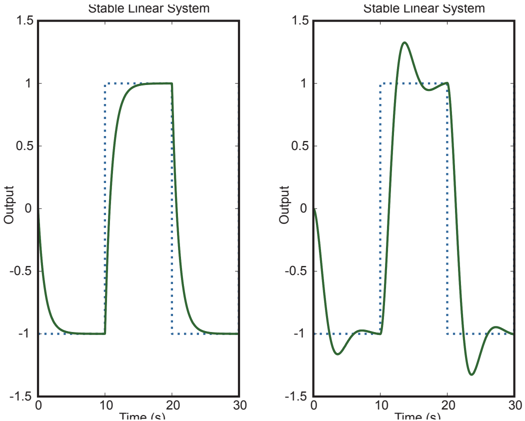


- **LTI block** (in SIMULINK Control Systems Toolbox) - takes any of the system models described above (tf, ss, zpk).



3.17 LTI Stability Responses

- An LTI system is **asymptotically stable** if all states decay to zero. An LTI system is **stable** if the response of the system is bounded for *every* bounded input (**BIBO**).



3.18 LTI Stability Examples

- A **stable system** has poles (roots of the characteristic equation) exclusively in the left–half s–plane.

Example

$$G_1(s) = \frac{10}{(s+3)(s+5)}$$

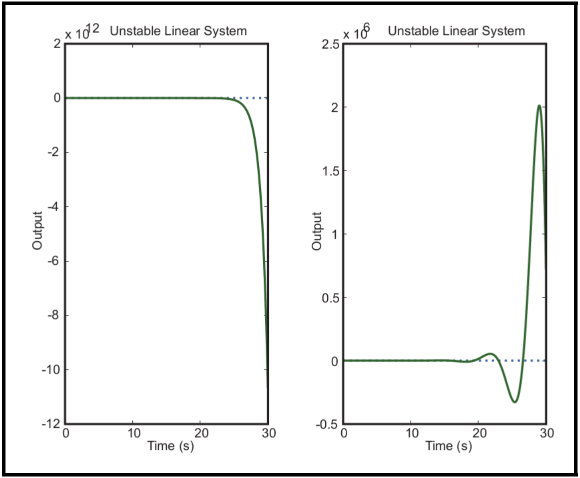
$$G_2(s) = \frac{s+4}{s^3+4s^2+4s+3}$$

$$G_3(s) = \frac{s-4}{s+4}$$

- Only **poles** dictate system stability, **zeros** in the right–half s–plane do *NOT* cause instability.

3.19 LTI Unstable Response

- An LTI system is **unstable** if the response of the system is unbounded for *any* bounded input.



3.20 LTI Unstable Examples

- An **unstable system** has at least one pole in the right–half s–plane and/or imaginary axis poles of multiplicity greater than one.

Example

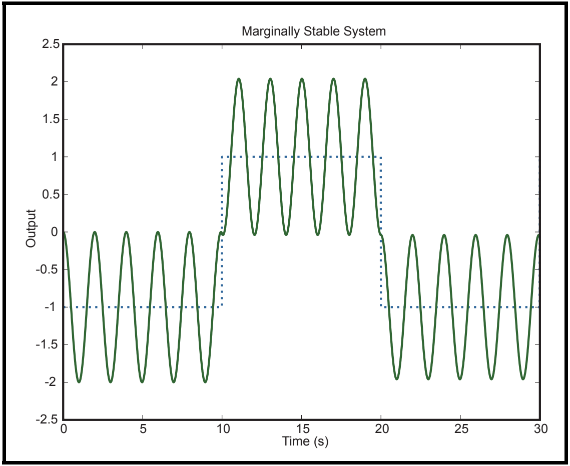
$$G_1(s) = \frac{s+3}{(s+2)(s-3)}$$

$$G_2(s) = \frac{10}{s^2 - 2s + 10}$$

$$G_3(s) = \frac{10}{(s^2+4)^2}$$

3.21 LTI Marginally Stable Response

- A LTI system is **marginally/critically stable** if the natural response to a bounded input is offset or oscillating.



3.22 LTI Marginally Stable Examples

- A **marginally stable** system has distinct complex conjugate pole pairs on the imaginary axis and all other poles in the left–half s–plane

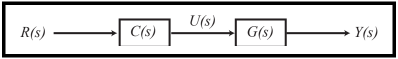
Example

$$G_1(s) = \frac{4}{\left(s^2+4\right)\left(s^2+16\right)}$$

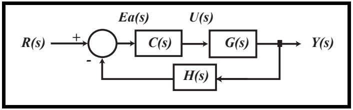
$$G_2(s) = \frac{s-4}{\left(s+2\right)\left(s^2+1\right)}$$

3.23 Linear Feedback Control Systems

- A **Control System** is a set of interconnected systems designed to provide a desired system response.
- **Open Loop System**



  - Has no feedback.
  - The output is generated directly by the input.
- **Closed Loop System**



  - Has a feedback loop; is error based.
  - Compares the output with the desired response.
  - Often $H(s) = 1$, called a **unity feedback** system.
  - Feedback reduces the **sensitivity** of the control system to model inaccuracies and disturbances.