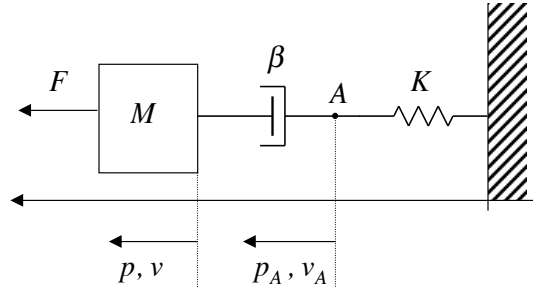


I esercitazione presso il LAIB

Esercizio #1.a: simulazione della risposta di un sistema meccanico

Si consideri il seguente sistema dinamico meccanico SISO LTI a tempo continuo:



che, scegliendo come variabili di ingresso $u = [F]$, stato $x = [x_1, x_2]^T = [v, p_A]^T$ ed uscita $y = [v]$, ha la seguente rappresentazione in variabili di stato:

$$\begin{cases} \dot{x}_1 &= -\frac{K}{M}x_2 + \frac{1}{M}u \\ \dot{x}_2 &= x_1 - \frac{K}{\beta}x_2 \\ y &= x_1 \end{cases}$$

dove M è la massa del corpo puntiforme, K è la costante di elasticità della molla, β è il coefficiente di attrito viscoso dello smorzatore, F è la forza applicata alla massa.

Assumendo $M = 0.2 \text{ kg}$ ed $F(t) = F_0 \cos(\omega_0 t) \text{ N}$, si simuli il comportamento del sistema nei seguenti casi:

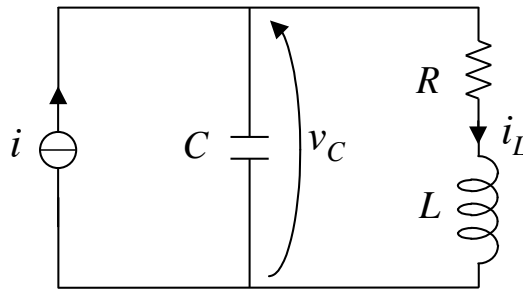
- 1) l'ingresso $F(t)$ è un gradino unitario ($F_0 = 1 \text{ N}$, $\omega_0 = 0 \text{ rad/s}$), mentre i valori numerici degli altri parametri e dello stato iniziale sono:
 - 1.a) $\beta = 0.1 \text{ Ns/m}$, $K = 2 \text{ N/m}$, $x(t=0) = x_0 = [0, 0]^T$;
 - 1.b) $\beta = 0.01 \text{ Ns/m}$, $K = 2 \text{ N/m}$, $x(t=0) = x_0 = [0, 0]^T$;
 - 1.c) $\beta = 10 \text{ Ns/m}$, $K = 20 \text{ N/m}$, $x(t=0) = x_0 = [0, 0]^T$;
 - 1.d) $\beta = 0.1 \text{ Ns/m}$, $K = 2 \text{ N/m}$, $x(t=0) = x_0 = [0, 0.2]^T$;
- 2) l'ingresso $F(t)$ è di tipo sinusoidale con $F_0 = 1 \text{ N}$ ed $\omega_0 = 4 \text{ rad/s}$, mentre i valori numerici degli altri parametri e dello stato iniziale sono:
 - 2.a) $\beta = 0.1 \text{ Ns/m}$, $K = 2 \text{ N/m}$, $x(t=0) = x_0 = [0, 0]^T$;
 - 2.b) $\beta = 0.01 \text{ Ns/m}$, $K = 2 \text{ N/m}$, $x(t=0) = x_0 = [0, 0]^T$;
 - 2.c) $\beta = 10 \text{ Ns/m}$, $K = 20 \text{ N/m}$, $x(t=0) = x_0 = [0, 0]^T$;
 - 2.d) $\beta = 0.1 \text{ Ns/m}$, $K = 2 \text{ N/m}$, $x(t=0) = x_0 = [0, 0.2]^T$.

Tracciare i grafici degli andamenti dell'evoluzione temporale degli stati e dell'uscita.

Comandi MATLAB da prendere in considerazione: `ss`, `lsim`

Esercizio #1.b: simulazione della risposta di un sistema elettrico

Si consideri il seguente sistema dinamico elettrico SISO LTI a tempo continuo:



che, scegliendo come variabili di ingresso $u = [i]$, stato $x = [x_1, x_2]^T = [v_C, i_L]^T$ ed uscita $y = [v_C]$, ha la seguente rappresentazione in variabili di stato:

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

Assumendo $C = 0.2 \text{ F}$ ed $i(t) = i_0 \cos(\omega_0 t) \text{ A}$, si simuli il comportamento del sistema nei seguenti casi:

- 1) l'ingresso $i(t)$ è un gradino unitario ($i_0 = 1 \text{ A}$, $\omega_0 = 0 \text{ rad/s}$), mentre i valori numerici degli altri parametri e dello stato iniziale sono:
 - 1.a) $R = 10 \Omega$, $L = 0.5 \text{ H}$, $x(t=0) = x_0 = [0, 0]^T$;
 - 1.b) $R = 100 \Omega$, $L = 0.5 \text{ H}$, $x(t=0) = x_0 = [0, 0]^T$;
 - 1.c) $R = 0.1 \Omega$, $L = 0.05 \text{ H}$, $x(t=0) = x_0 = [0, 0]^T$;
 - 1.d) $R = 10 \Omega$, $L = 0.5 \text{ H}$, $x(t=0) = x_0 = [0, 0.2]^T$;
- 2) l'ingresso $i(t)$ è di tipo sinusoidale con $i_0 = 1 \text{ A}$ ed $\omega_0 = 4 \text{ rad/s}$, mentre i valori numerici degli altri parametri e dello stato iniziale sono:
 - 2.a) $R = 10 \Omega$, $L = 0.5 \text{ H}$, $x(t=0) = x_0 = [0, 0]^T$;
 - 2.b) $R = 100 \Omega$, $L = 0.5 \text{ H}$, $x(t=0) = x_0 = [0, 0]^T$;
 - 2.c) $R = 0.1 \Omega$, $L = 0.05 \text{ H}$, $x(t=0) = x_0 = [0, 0]^T$;
 - 2.d) $R = 10 \Omega$, $L = 0.5 \text{ H}$, $x(t=0) = x_0 = [0, 0.2]^T$.

Tracciare i grafici degli andamenti dell'evoluzione temporale degli stati e dell'uscita.

Comandi MATLAB da prendere in considerazione: ss, lsim

- **SS** Create state-space models or convert LTI model to state space.

You can create a state-space model by:

`SYS = SS(A,B,C,D)` Continuous-time model

`SYS = SS(A,B,C,D,T)` Discrete-time model with sampling time T (Set `T=-1` if undetermined)

`SYS = SS` Default empty state-space model

`SYS = SS(D)` Static gain matrix

`SYS = SS(A,B,C,D,LTISYS)` State-space model with LTI properties inherited from the LTI model `LTISYS`.

All the above syntaxes may be followed by Property/Value pairs. (Type `help ltiprops` for details on assignable properties). Setting `D=0` is interpreted as the zero matrix of adequate dimensions. The output `SYS` is an **SS** object.

`SYS = SS(SYS)` converts an arbitrary LTI model `SYS` to state space, i.e., computes a state-space realization of `SYS`.

- **LSIM** Simulation of the time response of LTI systems to arbitrary inputs.

LSIM(SYS,U,T) plots the time response of the LTI model **SYS** to the input signal described by **U** and **T**. The time vector **T** consists of regularly spaced time samples and **U** is a matrix with as many columns as inputs and whose *i*-th row specifies the input value at time **T(i)**. For instance,

```
t = 0:0.01:5; u = sin(t); lsim(sys,u,t)
```

simulates the response of **SYS** to $u(t) = \sin(t)$ during 5 seconds.

In discrete time, **U** should be sampled at the same rate as the system (**T** is then redundant and can be omitted or set to the empty matrix).

In continuous time, the sampling period **T(2)-T(1)** should be chosen small enough to capture the details of the input signal. The time vector **T** is resampled when intersample oscillations may occur.

LSIM(SYS,U,T,X0) specifies an additional nonzero initial state **X0** (for state-space systems only).

LSIM(SYS1,SYS2,...,U,T,X0) simulates the response of multiple LTI systems **SYS1**, **SYS2**,... on a single plot. The initial condition **X0** is optional. You can also specify a color, line style, and marker for each system, as in

```
lsim(sys1,'r',sys2,'y--',sys3,'gx',u,t).
```

When invoked with left hand arguments,

```
[Y,T] = LSIM(SYS,U,...)
```

returns the output history **Y** and time vector **T** used for simulation. No plot is drawn on the screen. The matrix **Y** has **LENGTH(T)** rows and as many columns as outputs in **SYS**.

For state-space systems,

```
[Y,T,X] = LSIM(SYS,U,...)
```

also returns the state trajectory **X**, a matrix with **LENGTH(T)** rows and as many columns as states.

Esercizio #2: calcolo di funzioni di trasferimento

Si calcolino le funzioni di trasferimento dei sistemi dinamici precedentemente considerati.

Comandi MATLAB da prendere in considerazione: ss2tf

- **SS2TF** State-space to transfer function conversion.

[NUM,DEN] = SS2TF(A,B,C,D,iu) calculates the transfer function:

$$H(s) = \frac{\text{NUM}(s)}{\text{DEN}(s)} = C(sI-A)^{-1}B+D$$

of the system: $\dot{x} = Ax + Bu$, $y = Cx + Du$ from the *iu*'th input. Vector **DEN** contains the coefficients of the denominator in descending powers of *s*. The numerator coefficients are returned in matrix **NUM** with as many rows as there are outputs *y*.

Esercizio #3: calcolo analitico di risposte nel tempo di sistemi dinamici mediante antitrasformata di Laplace

Si calcolino analiticamente, per condizioni iniziali nulle, le risposte dei sistemi dinamici precedentemente considerati ai seguenti ingressi:

- gradino di ampiezza u_0 : $u(t) = u_0 \cdot \varepsilon(t)$
- rampa unitaria $u(t) = t \cdot \varepsilon(t)$
- coseno di ampiezza u_0 e di pulsazione 4 rad/s: $u(t) = u_0 \cos(4t) \cdot \varepsilon(t)$

Comandi MATLAB da prendere in considerazione: tf, tfdata, residue

- **TF** Creation of transfer functions or conversion to transfer function.

You can create SISO or MIMO transfer functions by

```
SYS = TF(NUM,DEN) Continuous-time model NUM(s)./DEN(s)
```

```
SYS = TF(NUM,DEN,T) Discrete-time model NUM(z)./DEN(z) with sampling time T (Set T=-1 if undetermined)
```

```
SYS = TF Default empty transfer function
```

```
SYS = TF(M) Static gain matrix
```

```
SYS = TF(NUM,DEN,LTISYS) Transfer function with LTI properties inherited from the LTI model LTISYS.
```

All the above syntaxes may be followed by Property/Value pairs. (Type **help ltiprops** for details on assignable properties). The output **SYS** is a TF object.

By default, transfer functions are displayed as functions of '*s*' or '*z*'. Alternatively, you can set the variable name to '*p*' (continuous time) and '*z*⁻¹' or '*q*' (discrete time) by modifying the 'Variable' property.

For SISO models, **NUM** and **DEN** are row vectors listing the numerator and denominator coefficients in

- descending powers of s or z by default
- ascending powers of $q = z^{-1}$ if 'Variable' is set to ' z^{-1} ' or ' q ' (DSP convention).

For MIMO models with **NU** inputs and **NY** outputs, **NUM** and **DEN** are **NY**-by-**NU** cell arrays of row vectors where **NUM**{*i,j*} and **DEN**{*i,j*} specify the transfer function from input *j* to output *i*. For example,

`tf(-5 ; [1 -5 6] , [1 -1] ; [1 1 0])`

specifies the two-output/one-input system

$$\begin{bmatrix} -5/(s-1) \\ (s^2-5s+6)/(s^2+s) \end{bmatrix}$$

SYS = **TF**(**SYS**) converts an arbitrary LTI model **SYS** to transfer function format. The output **SYS** is a **TF** object.

SYS = **TF**(**SYS**, 'inv') uses a fast algorithm for state-space **SYS**, but is typically less accurate for high-order systems.

• **TFDATA** Quick access to transfer function data.

[NUM,DEN] = **TFDATA**(**SYS**) returns the numerator(**s**) and denominator(**s**) of the transfer function **SYS**. **NUM** and **DEN** are cell arrays with as many rows as outputs and as many columns as inputs, and their (**I,J**) entries specify the transfer function from input *J* to output *I*. **SYS** is first converted to transfer function if necessary.

[NUM,DEN,TS,TD] = **TFDATA**(**SYS**) also returns the sample time **TS** and input delays **TD**. For continuous systems, **TD** is a vector with one entry per input channel. For discrete systems, **TD** is the empty matrix [].

For SISO systems, the convenience syntax **[NUM,DEN]** = **TFDATA**(**SYS**, 'v') returns the numerator and denominator as row vectors rather than cell arrays.

• **RESIDUE** Partial-fraction expansion (residues).

[R,P,K] = **RESIDUE**(**B,A**) finds the residues, poles and direct term of a partial fraction expansion of the ratio of two polynomials **B**(**s**)/**A**(**s**). If there are no multiple roots,

$$\frac{B(s)}{A(s)} = \frac{R(1)}{s - P(1)} + \frac{R(2)}{s - P(2)} + \frac{R(n)}{s - P(n)} + K(s)$$

Vectors **B** and **A** specify the coefficients of the numerator and denominator polynomials in descending powers of *s*. The residues are returned in the column vector **R**, the pole locations in column vector **P**, and the direct terms in row vector **K**. The number of poles is **n** = **length**(**A**)-1 = **length**(**R**) = **length**(**P**). The direct term coefficient vector is empty if **length**(**B**) < **length**(**A**), otherwise **length**(**K**) = **length**(**B**)-**length**(**A**)+1.

If **P**(*j*) = ... = **P**(*j+m*-1) is a pole of multiplicity *m*, then the expansion includes terms of the form

$$\frac{R(j)}{s - P(j)} + \frac{R(j+1)}{(s - P(j))^2} + \frac{R(j+m-1)}{(s - P(j))^m}$$

[B,A] = **RESIDUE**(**R,P,K**), with 3 input arguments and 2 output arguments, converts the partial fraction expansion back to the polynomials with coefficients in **B** and **A**.

Warning: Numerically, the partial fraction expansion of a ratio of polynomials represents an ill-posed problem. If the denominator polynomial, **A**(**s**), is near a polynomial with multiple roots, then small changes in the data, including roundoff errors, can make arbitrarily large changes in the resulting poles and residues. Problem formulations making use of state-space or zero-pole representations are preferable.