POLITECNICO DI MILANO





Laboratorio di Fondamenti di Automatica

Ingegneria Elettrica

Sessione 1/3

Danilo Caporale [caporale@elet.polimi.it]

Parte 1: comandi base di MATLAB

Prima di tutto…

- Avviare MATLAB:
  - Start → Programmi → Matlab R20xx → Matlab.
  - Quando il sistema sarà pronto: **>>**

- Chi: Mathworks http://www.mathworks.it/.
- Cosa: strumento di calcolo scientifico, standard di riferimento per il calcolo numerico matriciale (MATrix LABoratory). [Alternative open source: Scilab, Octave]
- Quando: creato alla fine degli anni '70, oggi siamo alla release R2012b (Windows/Linux/OS X).
- Come: linguaggio MATLAB, le cui basi impareremo oggi.
- Perché: ci evita di dover reinventare la ruota ogni volta che dobbiamo fare dei conti col calcolatore (l'alternativa è scrivere il codice numerico in C/Fortran/Python/Java/whatever).

  MATLAB ha una funzione più o meno per tutto e sono disponibili numerosi Toolboxes che aggiungono funzionalità nuove e interessanti.

  Dispone di numerose funzioni per la visualizzazione 2D/3D dei dati.

Concetti fondamentali

- Per eseguire un comando bisogna scriverlo e premere INVIO.
- Provare le seguenti semplici operazioni:
  - >> 5+2
  - >> 5-2
  - >> 5*2
  - >> 5/2
  - >> 5^2
  - >> 3+2*(4+3)
  - >> 2.54*8/2.6
  - >> 6.3-2.1045
  - >> 3.6^2
  - >> 1 + 2^2
  - >> sqrt(4)
  - >> cos(pi)

Come usare queste slides:

in blu sono mostrati i comandi da provare in prima persona in MATLAB;

in rosso i nuovi comandi introdotti ad ogni slide;

per prendere confidenza il modo migliore è provare a cambiare i comandi e vedere cosa succede;

per alcuni esercizi è utile avere a portata di mano **carta e penna** su cui controllare i risultati.

MATLAB dispone di un ottimo sistema di aiuto che si può chiamare in due modi:

>> help nomecomando → aiuto mostrato nella stessa finestra in cui si danno i comandi.

>> doc nomecomando → aiuto mostrato in una finestra dedicata.

Provare a chiedere aiuto per la funzione radice quadrata:

```
>> help sqrt
>> doc sqrt
```

Comandi utili

Il punto e virgola sopprime l'output a video.

```
>> 5+2;
```

Notazione esponenziale:

```
>> 1e2
```
1×10^2

```
>> 1.5e-3
```
1.5×10^{-3}

Il risultato dell'ultima operazione viene memorizzato nella variabile ans (answer).

Creare nuove variabili è semplice (non serve specificare il tipo di variabile, al contrario dei linguaggi tradizionali tipo C).

```
>> 3+2
>> 1 + ans
>> 3 – ans
>> a = 1
>> b = 2
>> c = 'ciao'
>> disp(c)
>> disp(c+a)
```
→ cosa è successo?

```
>> sprintf('%s', c+a)
```
→ rispetto a prima si sta forzando MATLAB ad interpretare c+a come una stringa di testo

Finestre principali di MATLAB

Finestra di comando o Command window: dove scriviamo i comandi e ne leggiamo i risultati.

Si pulisce col comando:

>> clc

Finestra di cronologia dei comandi o Command history: possiamo vedere i comandi passati ed eventualmente ripeterli facendo doppio-click su un comando.

Workspace: tiene traccia di tutte le variabili create dall'esecuzione dei comandi e ci dà qualche informazione su di esse.

>> workspace → per richiamare il workspace da riga di comando

Le stesse informazioni del workspace ce le dà il comando:

>> whos

Il comando clear cancella le variabili dal workspace

Se per sbaglio si chiude una di queste finestre la si può riaprire dal menu Desktop nella barra dei menu.

Provare:

```
>> a=5; b=3; c='ciao'; d=1+i;
>> whos          → NB: la tipizzazione delle variabili
                   (double, char, complex ecc) viene effettuata automaticamente da MATLAB
>> whos b
>> clear b       → cancella solo b
>> whos b
>> whos
>> clear all     → cancella tutte le variabili presenti nel workspace
>> whos
```

Matrici

In MATLAB (MATrix LABoratory) tutto è una matrice, anche i numeri (matrici 1x1).

Le matrici e i vettori si definiscono dandone gli elementi tra parentesi quadre:

- Gli elementi per colonna sono separati da virgole o da spazi
- Le righe sono separate dal punto e virgola

```
>> A = [1, 2, 3, 4] → virgole sono opzionali, bastano gli spazi
>> A = [1  2  3  4]
>> whos A

>> B = [1 2 3 4; 5 6 7 8] → il punto e virgola indica di andare a capo su una nuova riga
>> C = [1 2 3; 4 5 6]' → l'apostrofo indica il trasposto
>> whos
>> clear B → cancella le variabili presenti nel workspace.
>> whos
>> clear all
>> whos
```

Provare a definire le seguenti matrici:

1	2	3
4	5	6
7	8	9

log(exp(1))	0	3	-15	21
43	5	-1	78	103
7	8	9	1e4	sqrt(9)

Operazioni tra matrici

>> A = [1 2; 3 4]

>> B = eye(2) → matrice identità di dimensione 2x2

>> A – B

>> A + B

>> A * B → Prodotto righe per colonne (algebra)

>> Z = zeros(2,5) → matrice di 0, dimensione 2x5

>> U = ones(2,5) → matrice di 1, dimensione 2x5

>> d = [1; 0]

>> A*d → questa operazione viene eseguita

>> Z*d → errore! Provare ad eseguire questa operazione con carta e penna. Si può fare? Ovviamente nel calcolo matriciale bisogna stare attenti alle dimensioni dei vettori che stiamo usando. Il comando whos può essere utile per debug.

>> inv(A) → matrice inversa

>> A^(-1) → matrice inversa

>> inv(A)*A → matrice identità

Operazioni con i vettori

```
>> x = [1 2 3 4]
>> y = [1 2 3 4]'
>> A = [x; 2 3 4 5; 3 4 5 6]
```

Nota: si può andare a capo quando si definisce una matrice.

```
>> A = [1 2 3 4;
2 3 4 5;
3 4 5 6]
```

Intervalli di numeri:

```
>> b = 1:5                  → 1 2 3 4 5 (passo 1)
>> c = 1:2:5                → 1 3 5 (passo 2)
>> d = linspace(1,10,3)     → vettore riga di 3 elementi, equidistanziati tra 1
                              e 10
>> e = logspace(1,3,3)      → vettore riga di 3 elementi, distanziati
                              logaritmicamente tra 10^1 e 10^3
```

Operazioni con i vettori

```
>> a = 1:3
>> b = a + 5
```

>> a * b → si sta chiedendo a MATLAB di moltiplicare 2 matrici, cioè fare il prodotto riga per colonna: in MATLAB i vettori riga (colonna) sono matrici con una sola riga (colonna)

a * b = [1 2 3] * [6 7 8] = 1*6 + 2*???

Quello che si può fare invece è moltiplicare elemento per elemento i due vettori. Operatore punto:

```
>> a.*b
```

a .* b = [1 2 3] * [6 7 8] = [1*6 2*7 3*8]

```
>> a./b
>> a.^2
```

Operazioni con i vettori

NB: moltiplicare per uno scalare si può sempre fare, sia con i vettori che con le matrici.

```
>> 2*a
>> 2*eye(5)
```

Una funzione MATLAB in generale può restituire più di un argomento.

Esempio:

```
>> a = [1, 2, -3, -4, 30, 1]
>> max(a)          → restituisce 30
>> [m,k] = max(a)  → restituisce in m il massimo, 30; restituisce in k
                     l'indice dell'elemento massimo, cioè 5
```

Estrarre sottomatrici/sottovettori

>> A = [1 2 3 4; 2 3 4 5; 3 4 5 6]

>> A(1,:) → prima riga. I due punti indicano di prendere tutti gli elementi.

>> A(:,1) → prima colonna.

>> A(end,1) → ultima riga, end indica l'ultimo elemento del vettore.

>> A(end,end)

>> A(:,2:3) → seconda e terza colonna, tutte le righe.

>> A([1 3],[2 4]) → prima e terza riga, seconda e quarta colonna.

Fare qualche prova per prendere confidenza con questi comandi.

Creare matrici diagonali o estrarre la diagonale di una matrice

Comando diag:

- Quando usato su una matrice, estrae i valori sulla diagonale principale:

```
>> A = [1 2; 3 4]
>> diag(A)
```

- Quando usato su un vettore, crea una matrice i cui elementi sulla diagonale sono dati dal vettore argomento:

```
>> v = 1:5
>> diag(v)
```

Cosa fanno i comandi seguenti?

```
>> diag(diag(A))
>> diag(diag(v))
```

Numeri e funzioni notevoli

Provare a giocare con i comandi seguenti (5 minuti):

>> imm = sqrt(-1) → è già definito come i

>> pi → pi greco

>> exp(1) → numero di Nepero

>> log(exp(1)) → logaritmo naturale

>> log10(10) → logaritmo base 10

>> cos(0) → coseno, argomento in radianti

>> sin(0) → seno, argomento in radianti

>> acos(0) → arcocoseno, risultato in radianti

>> asin(0) → arcoseno, risultato in radianti

>> abs(-2) → valore assoluto

>> abs(1+i) → modulo vettore complesso 1+i

>> angle(1+i) → fase o anomalia del vettore complesso 1+i, in radianti

>> angle(1+i)*180/pi → in gradi. Il comando rad2deg fa la stessa cosa.

Definizioni intuitive, basta cercare.

Funzioni utili per trattare i polinomi: roots

Calcolare le radici del polinomio x^2+2x+1

>> coeff=[1 2 1] → si definisce il vettore dei coefficienti del polinomio, dal coefficiente di grado più alto a quello di grado più basso

>> r=roots(coeff) → radici del polinomio i cui coefficienti sono coeff

>> r=roots([1 2 1]) → stessa cosa, senza passare per la variabile coeff

Esercizio:

Calcolare le radici del polinomio x^2+x+1 e salvarle nel vettore r.

Mostrarle a video coi comandi seguenti:

>> [abs(r) angle(r)*180/pi real(r) imag(r)]

Funzioni utili per trattare i polinomi: poly

Problema: trovare i coefficienti del polinomio che ha radici [-1 -1].

Si scrivono in un vettore le radici del polinomio (l'ordine ovviamente non conta quando scriviamo le radici):

```
>> r = [-1 -1]
```

I coefficienti del polinomio che ha radici r si calcolano col comando poly:

```
>> coeff = poly(r)
```
→ è l'inverso del comando roots

```
>> coeff = poly([-1 -1])
```
→ stessa cosa

Il comando poly restituisce i coefficienti dal grado più alto al più basso.

Sintassi del comando poly:

- Usato su un vettore dà i coefficienti del polinomio che ha come radici gli elementi del vettore, ad es:

```
>> c = poly(r)
```
→ esempio in alto.

- Usato su una matrice dà i coefficienti del polinomio caratteristico della stessa:

```
>> A = [1 2; 0 1]; poly(A)
```
→ confrontare con conti fatti a mano

Funzioni utili per trattare i polinomi: conv

Problema: moltiplicare due polinomi: (x+1)*(x+2)

Si definiscono i coefficienti dei due polinomi nei vettori a e b.

```
>> a = [1 1]; b = [1 2];
```

Si calcolano i coefficienti del polinomio risultante: x^2+2x+2.

Si usa il comando conv:

>> conv(a,b) → confrontare il risultato con conti fatti a mano per convincersi di cosa si è fatto

>> conv([1 2 3],[4 5 6]) → inventare altri esempi con questo comando

NB: l'ordine con cui MATLAB tratta i polinomi è dal coefficiente di grado più alto al più basso. Consultare l'help in caso di dubbi.

Autovalori e polinomio caratteristico

>>A=[1 2;2 1];

Calcoliamo gli autovalori come radici del polinomio caratteristico e creiamo la relativa matrice diagonale:

>>v=roots(poly(A)) → v è il vettore degli autovalori di A: radici (roots) del polinomio caratteristico di A (poly(A)).

>> eig(A) → comando nativo di MATLAB per il calcolo degli autovalori di A. Confrontare con riga precedente.

>> [M,D] = eig(A) → salva in M la matrice degli autovettori di A, in D la matrice diagonale degli autovalori di A.

Calcoliamo ora una matrice M che diagonalizza A:

>>[M,dv]=eig(A); → dv è la matrice diagonale degli autovalori, M è la matrice degli autovettori di A

Verifica:

>>inv(M)*A*M → è diagonale

Operazioni su matrici

Altro esempio di errori dovuti ad approssimazione numerica:

>> A = [.001 1; 1 1];

>> B = eye(2)

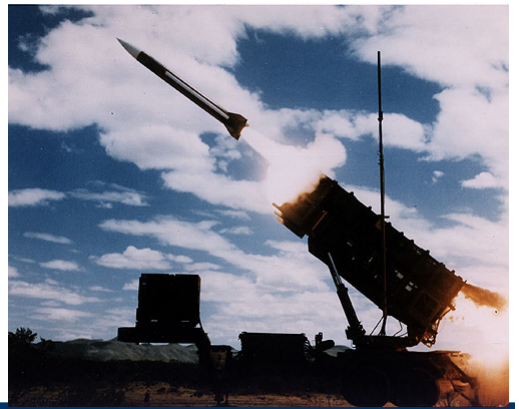
>> inv(A) → matrice inversa

>> inv(A)*A

>> inv(A)*A – B → c'è un residuo dovuto ad errori numerici (*spiegazione intuitiva*: i calcolatori hanno memoria finita e perciò devono arrotondare i numeri). A noi questo non darà fastidio. Chi è interessato potrà approfondire questi concetti nei corsi di analisi numerica.

Per un esempio clamoroso (missili Patriot):

http://sydney.edu.au/engineering/it/~alum/patriot_bug.html



Un po' di grafica

Vogliamo mostrare a video un grafico di sin(2πt), per t che va da 0 a 2.

```
>> t = 0:0.01:2
>> y = sin(2*pi*t)
```

>> plot(t,y) → apre una finestra e ci disegna sopra il grafico: t sulle ascisse, y sulle ordinate.

>> plot(t,sin(2*pi*t) → stessa cosa

Provare a disegnare qualcosa, a piacere.

Un po' di grafica

Vogliamo sovrapporre i grafici di due funzioni.

```
>> t = 0:0.01:2
>> y = sin(2*pi*t)
>> plot(t,y)
```

→ primo grafico

```
>> z = cos(2*pi*t);
>> hold on;
```

→ dice a MATLAB di non cancellare il plot precedente. Con questo comando possiamo sovrapporre diversi grafici.

```
>> plot(t,z,'r')
```

→ 'r' significa rosso. Provare 'g', 'b', 'm', 'k', '-.', 'x', 'o', *help plot*.

Aggiungiamo una griglia, una didascalia sull'asse orizzontale e verticale e una legenda.

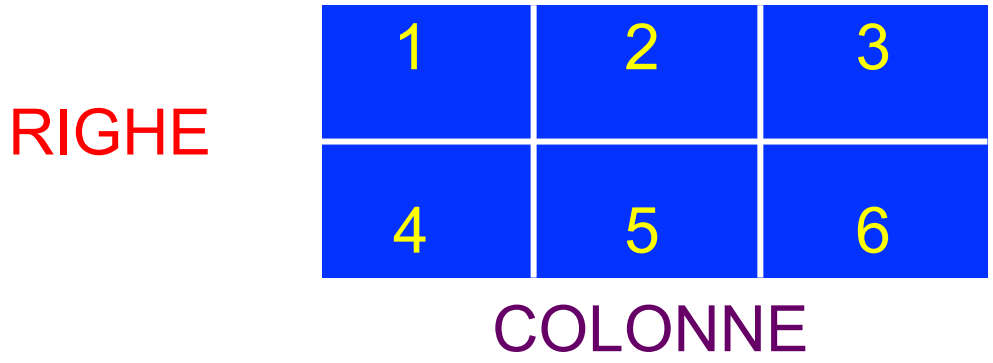
```
>> grid on
>> xlabel('tempo [secondi]'); ylabel('funzioni'); legend('seno','coseno');
```

Ancora grafica

Possiamo inserire più grafici su una stessa finestra. Provare:

```
>> close all → chiude tutte le finestre aperte
>> figure → apre una nuova finestra
>> subplot(2,1,1);
>> plot(t,y);
>> subplot(2,1,2);
>> plot(t,z);
```

La sintassi del comando subplot è: subplot(RIGHE,COLONNE,QUALE)



Ancora grafica: zoom da linea di comando, testo con click

>> figure(2) → comando per aprire una nuova finestra

>> t = 0:0.01:2;

>> y = log(t);

>> plot(t , y , t , sin(2*pi*t)); → modo più veloce di sovrapporre grafici

>> plot(t , y , 'k' , t , sin(2*pi*t) , 'r-.'); → colore

>> grid on;

>> axis([t(1) t(end) -1 1]); → argomento di axis: [xmin xmax ymin ymax]

>> gtext(' Testo ') → Argomenti: stringa di testo da posizionare su di un grafico in una posizione scelta con un click del mouse.

Per mettere il testo su più righe bisogna scriverlo tra parentesi graffe

>> gtext({ 'For those about to plot' , 'we salute you' }) → Argomento: { stringariga1, stringariga2, ecc }

Comodità (oggi) e necessità (in futuro): l'editor di testo

Col comando

>> edit

o cliccando su File → New → Script si apre l'editor di testo di Matlab.

Da qui si possono definire elenchi di comandi (algoritmi). Inoltre è possibile:

- inserire commenti: a=2; % questo è un commento
- eseguire lo script digitando Ctrl + Invio (Cmd + Invio su OS X) o cliccando sulla freccina verde (o digitando F5).
- e salvare il file scritto con estensione .m.

Salvo altre necessità, è comodo iniziare ogni script coi comandi:

>> clear all; close all; clc; % clc cancella la schermata dei comandi

Si possono inoltre salvare le variabili presenti nel workspace col comando:

>> save nomefile.mat % .mat è l'estensione per i file di dati di matlab

>> load nomefile.mat % carica le variabili del file nel workspace

>> save nomefile.mat nomevar % salva solo la variabile nomevar nel file piuttosto che l'intero workspace

→ Provare ad usare l'editor, salvare variabili e caricarle. (5 minuti)

Riferimenti

Per i più curiosi:

- Dal sito: http://home.dei.polimi.it/colaneri/ trovate il link a materiale in uso presso altri corsi (Leva, Lovera, Prandini), dal quale sono state tratti alcuni esempi in queste slides.

- MATLAB PER L'INGEGNERIA, di Holly Moore, edito da Pearson in italiano. Ci sono molti esempi e funzionalità più avanzate (definizione di funzioni, calcolo simbolico). E' un di più, in quanto questo ed altri corsi successivi sono pensati per insegnarvi quello di cui avete bisogno.

help e il WWW sono pieni di esempi interessanti.

Parte 2: sistemi dinamici LTI a tempo continuo con MATLAB

NB: si consiglia l'uso dell'editor da questo punto in poi
per velocizzare i tempi di esecuzione degli esercizi

Come si definisce un sistema dinamico in MATLAB

Rappresentazione nello spazio di stato:

>> S1 = ss(A,B,C,D) → prende come argomenti le matrici del sistema.

Funzione di trasferimento:

>> S2 = tf(num,den) → prende come aromenti i coefficienti del polinomi al numeratore e al denominatore della funzione di trasferimento del sistemo.

Esercizio: definire i due sistemi seguenti usando i comandi ss e tf.

```
>> A = [1 0; 2 1];
>> B = [1; 0];
>> C = [1 1];
>> D = 0;
>> num = 1;
>> den = [1 1];
```

ss e tf sono rappresentazioni equivalenti (a meno di cancellazioni nel passaggio da spazio di stato a funzione di trasferimento, vedremo poi cosa significa):

```
>> tf(S1)        >> ss(tf(S1))
>> ss(S2)        >> tf(ss(S2))
```

Studio della stabilità di un sistema dinamico con i criteri polinomiali: criterio di Routh

Per sistemi di ordine n <= 2, un sistema LTI è asintoticamente stabile se e solo se i coefficienti del polinomio caratteristico hanno tutti lo stesso segno (>0 o <0, non == 0).

>> A = [-3 -2; 1 0]

>> eig(A) → autovalori di A tutti a parte reale negativa, sistema as. stab.

Verifichiamo con i coefficienti del polinomio caratteristico:

>> coeff = poly(eig(A)) → coefficienti del polinomio caratteristico di A

1 3 2 → tutti dello stesso segno → sistema asintoticamente stabile

Un esempio per un sistema instabile di ordine 2:

>> phi = [1 -2 1] → coefficienti polinomio caratteristico cambiano di segno → instabile

>> S = ss(tf(1,phi))

>> eig(S.a) → autovalori a parte reale positiva: sistema instabile.

Studio della stabilità di un sistema dinamico con i criteri polinomiali: criterio di Routh

Per sistemi di ordine n > 2, la condizione precedente è solo necessaria.

Esercizio:

>> phi = [1 1 1 1] → calcolare gli autovalori del sistema che ha questo polinomio caratteristico e verificare che il sistema sia instabile.

Bisogna aggiungere una condizione sufficiente per essere sicuri di avere un sistema asintoticamente stabile quando n>2 → Criterio di Routh.

In MATLAB il criterio di Routh non è necessario, dal momento che siamo in grado di calcolare le radici dei polinomi semplicemente col comando roots o gli autovalori col comando eig.

There is no learning without doing

Extra per casa: scrivere una funzione che, data la matrice A di un sistema LTI, ne calcola gli autovalori e mostra a schermo un messaggio che dice se il sistema è stabile o meno.

Suggerimenti:

- La sintassi per creare le funzioni è:

```
function nomefunzione(argomento)
 % … calcoli …
end
```

Bisogna salvare il file come nomefunzione.m (obbligatorio altrimenti si ha errore).

- Le istruzioni condizionali in MATLAB sono simili a quelle del linguaggio C:

```
if condizione
        istruzioni
end
```

Esempio:

```
>> a=1;
>> if a>0
 disp('a positivo');  % disp mostra a schermo una stringa di testo
 end
```

Risposta allo scalino di sistemi dinamici

>> step(S) → visualizza la risposta a scalino

>> step(S,t) → risposta a scalino, con il vettore dei tempi che si desidera

>> [y,t] = step(S) → restituisce nei vettori y e t la risposta e il tempo.

>> y = step(S,t) → restituisce la risposta sull'asse dei tempi dato da t.

Sostituendo alle righe precedenti impulse a step si ottiene la risposta all'impulso.

Esercizio 1: provare i comandi elencati sopra ponendo S pari al sistema la cui funzione di trasferimento ha **guadagno unitario**, **nessuno zero** e **due poli in -5 e -10**. Porre **t=0:0.01:1**. Usare plot per visualizzarla se step non dovesse farlo.

Esercizio 2: ripetere il tutto nel caso in cui i poli della funzione di trasferimento di S siano complessi coniugati e pari a 0.1±j0.4. Porre **t=0:0.2:80**.

Movimento libero e movimento forzato

Movimento libero dallo stato iniziale x0 (per sistemi specificati nello spazio di stato):

```
>> initial(S,x0);
>> initial(S,x0,t);
>> [y,t,x]=initial(S,x0);    → c'è anche il movimento di x
>> [y,t,x]=initial(S,x0,t); → stessa cosa
```

Esercizio: calcolare la risposta del sistema definito da

```
>> A=[-1 0;1 -2]; b=[1 0]'; c=[1 1]; d=0;
```

allo scalino unitario, con stato iniziale x0'=[1 1], per t da 0 a 10s a passi di 0.01s.

```
>> S=ss([-1 0;1 -2],[1;0],[1 1],0);
>> x0=[1 1]'; t=0:0.01:10;
>> [yL,t]=initial(S,x0,t); yF=step(S,t);
>> y=yL+yF;
>> plot(t,yL,t,yF,t,y);
>> grid on; legend('risposta libera','risposta forzata','integrale completo')
```

Esponenziale di matrice

Esercizio: data una matrice A di dimensione n x n, per $t \geq 0$ la matrice esponenziale e^{At}

è definita come:

$$e^{At} = \sum_{i=0}^{\infty} \frac{1}{i!}(At)^i = I_n + At + \frac{(At)^2}{2!} + \ldots$$

In MATLAB la matrice e^{M} si calcola col comando expm(M):

```
>> A=[-1 0;1 -2]
>> expm(A)
```

→ usa una approssimazione di Padè di ordine elevato.

Al solito, chi fosse interessato può dare help expm per i dettagli dell'algoritmo.

Esercizio: confronto tra initial e expm. Soluzione


```
t=0:0.01:2;
x0 = [1 2]';
y = zeros(2,length(t));
A = [-1 0; 1 -2];
B = [ 1 0]';
C = [ 1 0; 0 1];
D = 0;
S = ss(A,B,C,D);
for ii=1:length(t)
    y(:,ii) = expm(A*t(ii)) * x0;
end
figure
plot(t,y,'k',t,initial(S,x0,t),'r-.')
grid on
xlabel('tempo [s]')
ylabel('stato')
legend('risposta libera (x1) con expm','risposta libera (x2) con expm ','risposta
    libera (x1) con initial','risposta libera (x2) con initial')
```

Movimento libero di un sistema dinamico:

$$x_l(t) = e^{A(t-t0)} x_{t_0}$$

$$y_l(t) = C e^{A(t-t0)} x_{t_0}$$

Risposta ad un ingresso qualunque specificato dall'utente

Risposta a ingresso u e stato iniziale x0 generici:

```
>> y=lsim(S,u,t);          (cond. iniz. nulle)
>> y=lsim(S,u,t,x0);       (cond. iniz. x0)
>> [y,t,x]=lsim(S,u,t);
>> [y,t,x]=lsim(S,u,t,x0);
```

Esercizio:

```
>> S1=tf(1,[1 1 1]);
>> S2=ss(-2,1,1,1);
>> t=0:0.01:20;
>> u=sin(2*pi*t)+0.2*sin(2*pi*5*t);
>> y1=lsim(S1,u,t);
>> y2=lsim(S2,u,t,2);
```

Risposta a scalino

$y(t)=G(s)\,u(t)$

Dato un ingresso u(t) trasformabile secondo Laplace e con *condizioni iniziali nulle*, si ha Y(s) = G(s)U(s)

E' interessante analizzare il comportamento di un sistema sollecitato da ingressi particolari (canonici), tra cui

lo scalino (vedremo oggi) ma anche l'impulso, la rampa e altri segnali con trasformata del tipo U/s^n, ...

la sinusoide (prossima volta).

Parleremo di risposta sottintendendo sempre l'ipotesi di condizioni iniziali nulle.

Importanza della risposta a scalino

La risposta a scalino rappresenta abitualmente il passaggio da un valore di regime ad un altro (transitorio).

E' relativamente facile da ottenere in pratica (cambio l'ingresso al sistema da un valore costante ad un altro valore costante).

Altre risposte canoniche possono essere dedotte da essa (ad esempio poiché l'impulso è "la derivata" dello scalino, si dimostra che la risposta all'impulso è la derivata della risposta a scalino); la risposta all'impulso non è affatto facile da ottenere in pratica, anzi a rigore non è ottenibile.

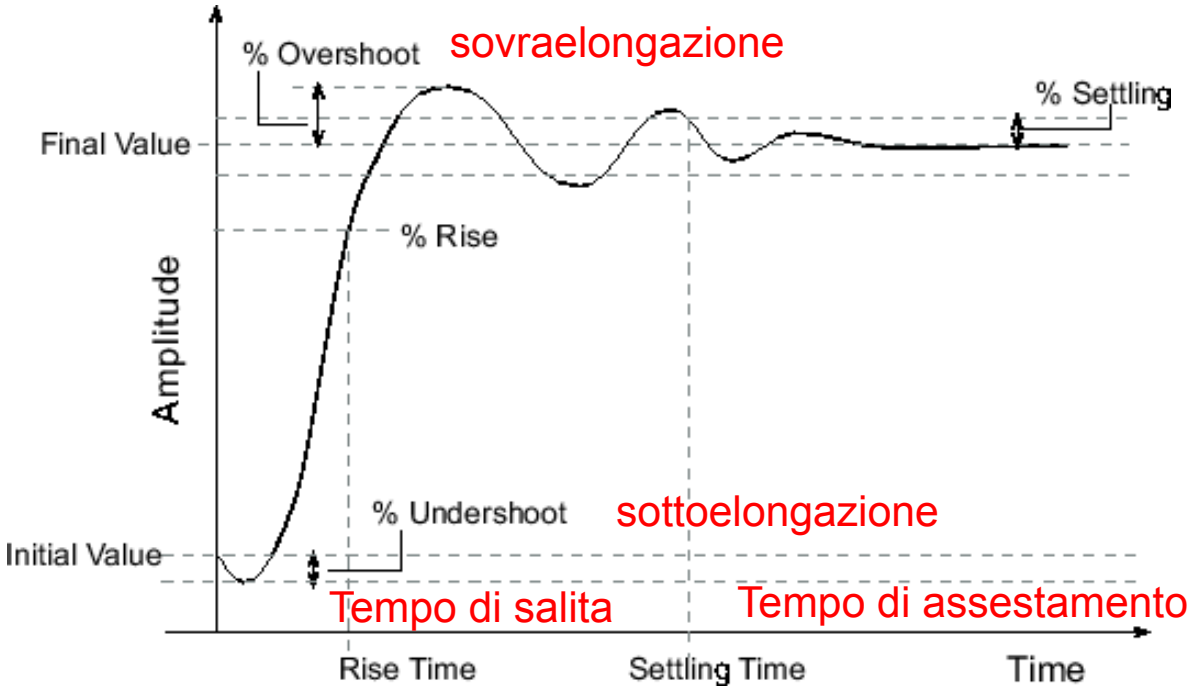
La sua conoscenza è quindi equivalente alla conoscenza della funzione di trasferimento (che coincide con la trasformata della risposta all'impulso)

Risposta a scalino di sistemi asintoticamente stabili

L'uscita, dopo un certo transitorio, raggiunge un nuovo valore di regime y, associato all'ampiezza dell'ingresso a scalino u.

Il transitorio è caratterizzato da:

- durata (velocità di risposta) → tempo di assestamento o settling time
- presenza di oscillazioni attorno al valore finale
- presenza di sovraelongazioni o sottoelongazioni → overshoot, undershoot.



Parte 2b: risposta a scalino di sistemi notevoli

Per un confronto analitico si può fare riferimento al libro di testo, paragrafo 4.4

Sistemi del 1° ordine: l'integratore

$G(s) = \mu/s$

$Y(s) = G(s)U(s) = (\mu/s)(1/s) = \mu/s^2$

$y(t) = \mu\ ram(t)$

La risposta del sistema è, a meno di una costante, l'integrale dell'ingresso:

- la **risposta allo scalino è una rampa**
- la **risposta all'impulso è uno scalino**.

Esercizio: visualizzare la risposta a scalino e all'impulso del sistema G(s) = 1/s.

```
>> G = tf(1,[1 0])
```

>> impulse(G) → la risposta all'impulso è uno scalino

>> step(G) → la risposta allo scalino è una rampa

Sistemi del 1° ordine senza zeri

$G(s) = \mu/(1+sT)$ con $\mu, T>0$

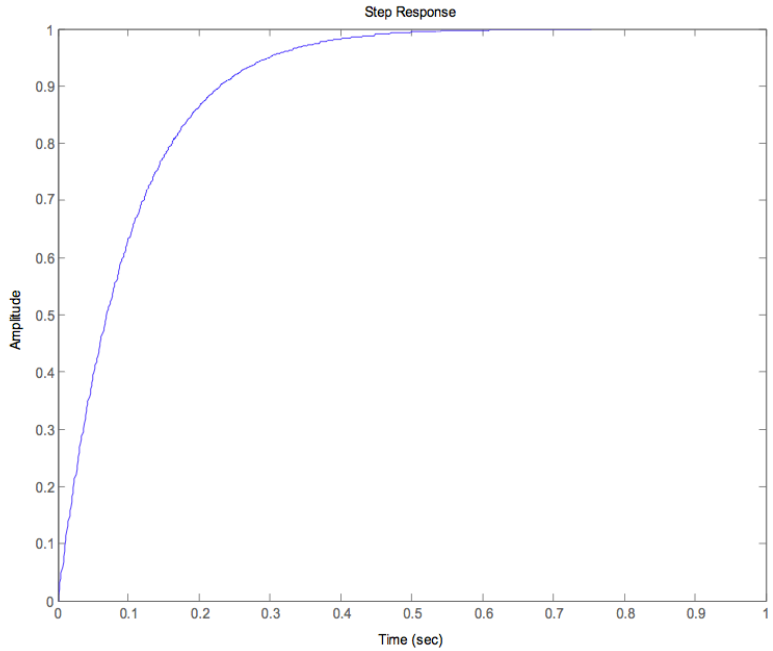
$Y(s) = G(s)U(s) = (\mu/(1+sT))(1/s) = \mu/(s(1+sT))$

Teorema del valore iniziale è utile per il calcolo della pendenza iniziale della risposta:

$y(0) = 0$

$dy/dt|_{t=0} = \mu/T$

Antitrasformando: $y(t) = \mu(1-e^{-t/T})$



Esercizio:

```
>>T=0.1;
>>mu=1;
>>t=0:0.001:1;
>>step(tf(mu,[T 1]),t);
```

NB: la costante di tempo dà informazioni su quanto tempo occore affinché si esaurisca il fenomeno transitorio, dunque quanto deve durare l'asse dei tempi per vedere tutto il fenomeno fino alla nuova condizione di regime.

Sistemi del 1° ordine senza zeri

Valutiamo il tempo di assestamento (al 99%) t_a:

è il minimo valore di t per cui $0.99\,\mu \leq y(t) \leq 1.01\,\mu$, per ogni $t \geq t_a$

In questo caso:

$\mu(1-e^{-t/T}) = 0.99\mu$, quindi $e^{-t/T} = 0.01$, quindi $t_a \approx 4.6T$

Nella pratica si assume che il tempo di assestamento (al 99%) valga 5T;

T prende il nome di costante di tempo;

la velocità di risposta dipende dalla posizione del polo -1/T (più il polo è vicino all'asse immaginario, cioè più T è grande, più il transitorio è lento).

La risposta è monotona.

Sistemi del 1° ordine con uno zero

$G(s) = \mu(1+sTz)/(1+sTp)$ $\mu, Tp>0$

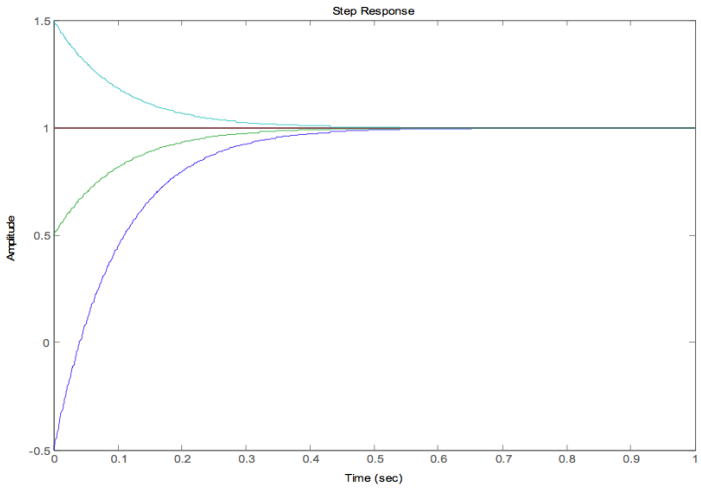
Teorema del valore iniziale:

$y(0) = \mu Tz/Tp$

$dy/dt|t=0 = (\mu/Tp)(1-Tz/Tp)$

Studiamo con MATLAB l'andamento della risposta al variare della posizione dello zero:

```
>>vecTz=T*[-0.5 0.5 1 1.5];
>>for k=1:length(vecTz)
    step(tf(mu*[vecTz(k) 1],[T 1]),t); hold on,
    end
>>hold off
```

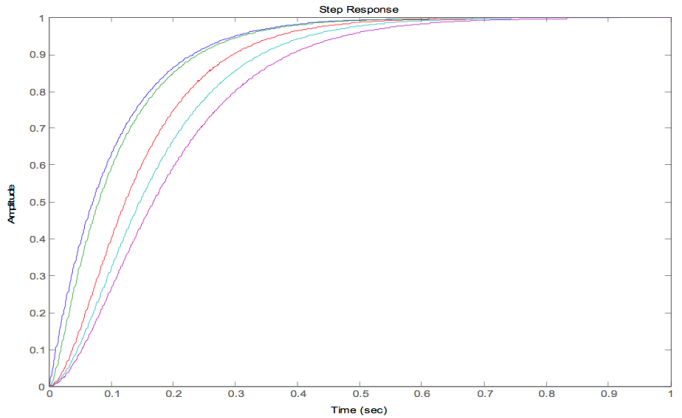


NB: comando for simile al C.

Si conclude con l'istruzione end.

Sistemi del 2° ordine con poli reali e senza zeri

$G(s) = \mu/((1+sT_1)(1+sT_2))$ $\quad\quad$ $\mu, T_1, T_2 > 0$



Teorema del valore iniziale:

$y(0) = 0$

$dy/dt|_{t=0} = 0$

$d^2y/dt^2|_{t=0} = \mu/(T_1T_2)$

Studiamo con MATLAB l'andamento della risposta al variare della seconda costante di tempo:

```
>>vecT2=T*[0.01 0.1 0.5 0.75 1];
>>for k=1:length(vecT2)
    step(tf(mu,conv([T 1],[vecT2(k) 1])),t), hold on,
    end
>>hold off
```

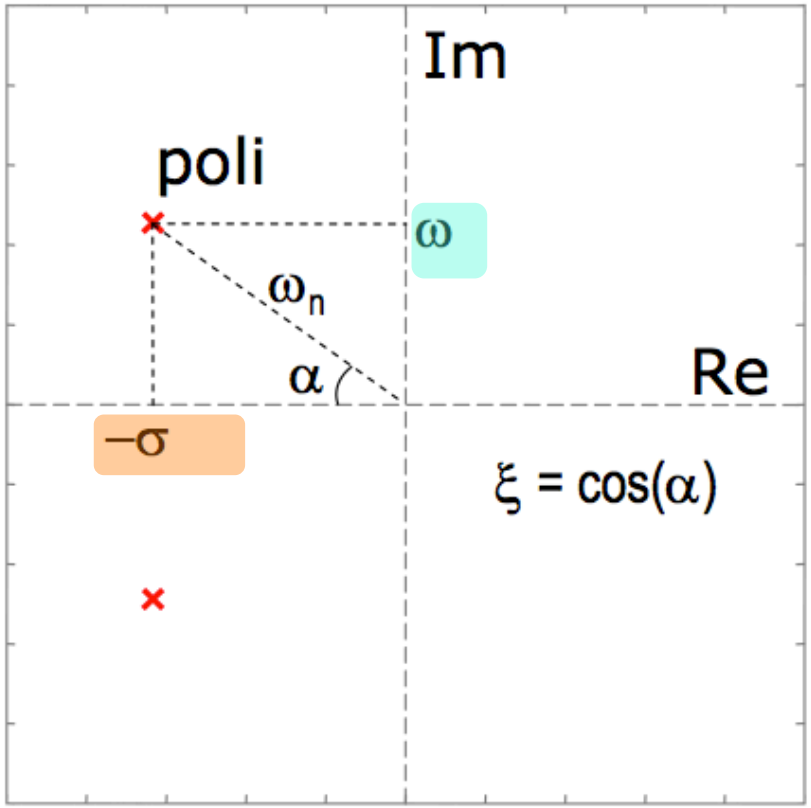
Sistemi del 2° ordine con poli reali e senza zeri

Osservazioni:

- la risposta parte con pendenza nulla;
- il tempo di assestamento è pari a circa 5 volte la costante di tempo maggiore, la quale ***domina*** il comportamento più lento:
- $t_a \approx 5 \max(T_1, T_2)$; la velocità di risposta dipende quindi dalla posizione del polo più vicino all'asse immaginario (***polo dominante***)
- la costante di tempo minore produce il suo effetto essenzialmente nei primi istanti del transitorio.
- la risposta è monotona.

Sistemi del 2° ordine con poli complessi coniugati

$$G(s) = \frac{\mu\omega_n^2}{s^2 + 2\xi\omega_n s + \omega_n^2}$$



ω_n e ξ si dicono rispettivamente "pulsazione naturale" e "fattore di smorzamento"

Formule di conversione:

$\omega = \omega_n(1 - \xi^2)^{1/2}$

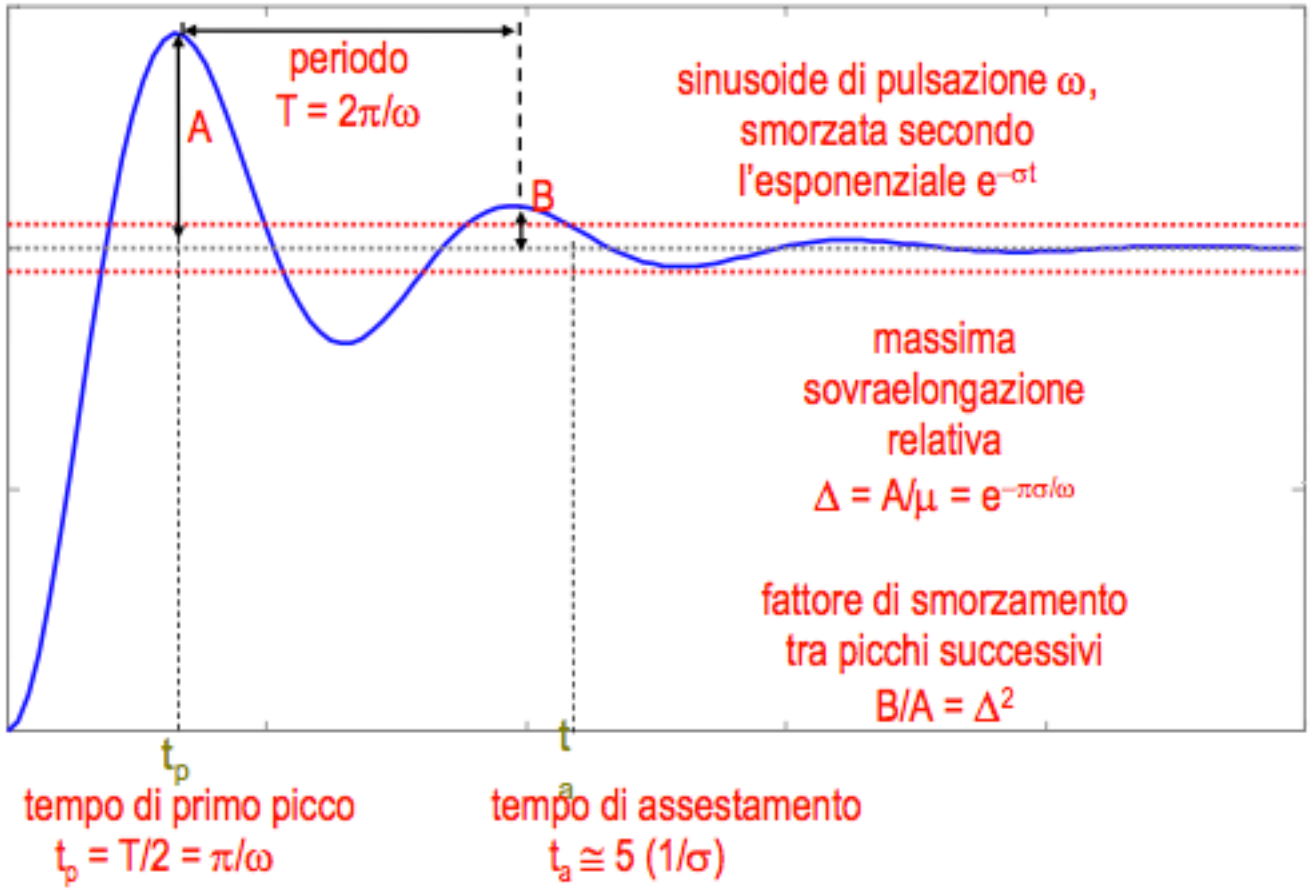
$\sigma = \xi\omega_n$

$\xi = \cos(\alpha) = \cos(\text{arctg}(\omega/\sigma))$

$\omega_n = (\omega^2 + \sigma^2)^{1/2}$

Risposta tipica di un sistema del secondo ordine con poli complessi coniugati

Parametri caratteristici della risposta a scalino;



Sistemi del 2° ordine con poli complessi coniugati

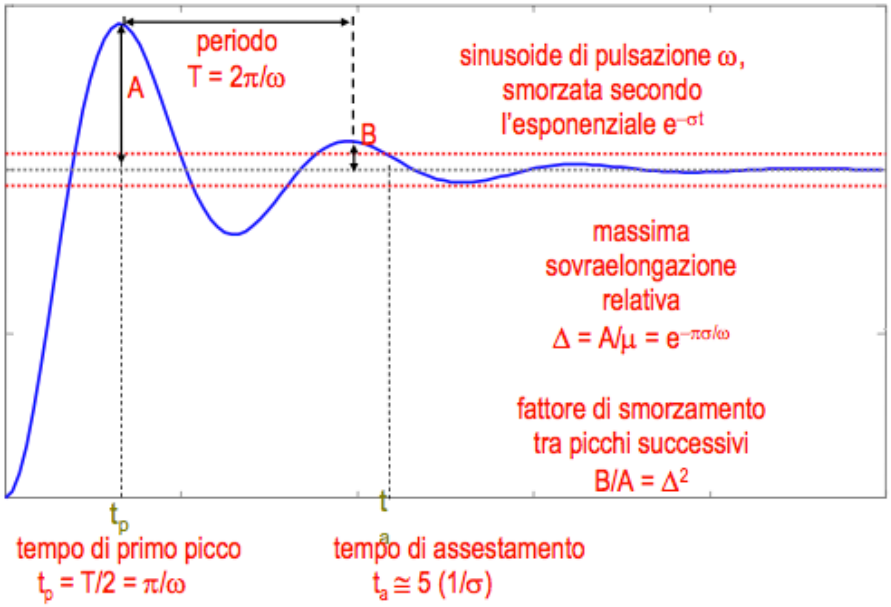
Valori:

$ta \approx 5\,(1/\sigma) = 5\,(1/\xi\omega_n)$

$T = 2\pi/\omega = 2\pi/\omega_n(1-\xi 2)^{1/2}$

$t_p = T/2 = \pi/\omega = \pi/\omega_n(1-\xi 2)^{1/2}$

$\Delta = A/\mu = e^{-\pi\sigma/\omega} = e^{-\pi\xi/(1-\xi)}$



Osservazioni:

- Δ dipende solo da ξ e non da ω_n
- per $\xi = 0$ (poli immaginari puri) si hanno oscillazioni non smorzate
- per $\xi = 1$ (poli reali coincidenti) la risposta non oscilla

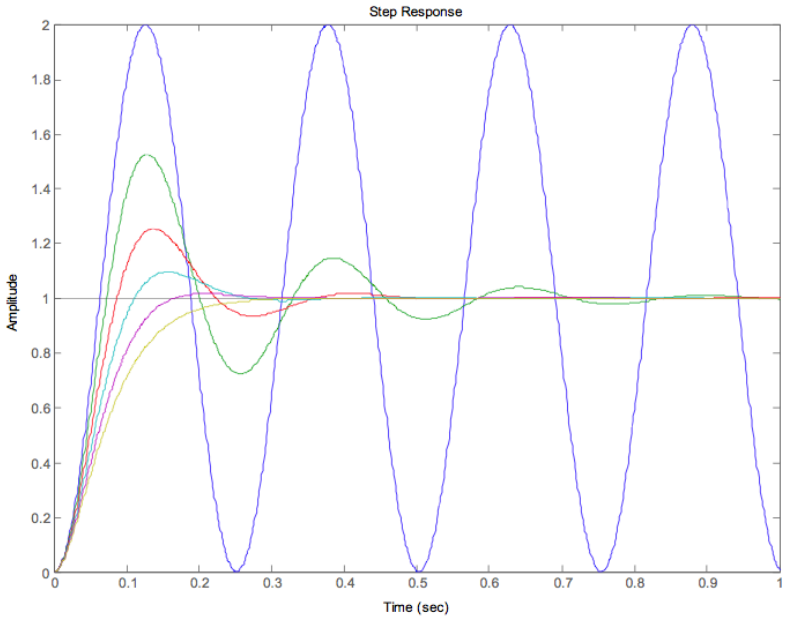
Sistemi del 2° ordine con poli complessi coniugati

Studiamo con MATLAB l'andamento della risposta al variare dello smorzamento:

```
>>vecCsi=[0:0.2:1];
>>wn = 25;
>>for k=1:length(vecCsi);
step(tf(mu,[1/wn^2 2*vecCsi(k)/wn 1]),t), hold on,
end
>>hold off
```

A parità di ω_n, al crescere di ξ da 0 a 1

- Δ diminuisce
- T e t_p aumentano ($\rightarrow \infty$ per $\xi \rightarrow 1$)



Sistemi del 2° ordine con poli reali e uno zero

$G(s) = \mu(1+sT_z)/((1+sT_1)(1+sT_2))$ $\mu, T_1, T_2 > 0$

Teorema del valore iniziale:

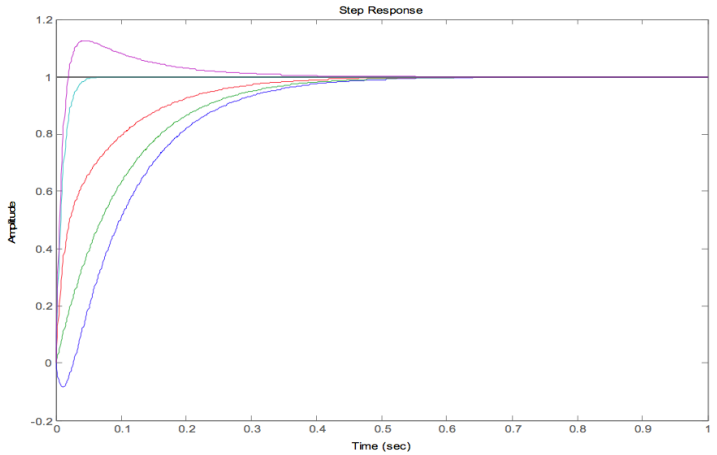
$y(0) = 0$

$dy/dt|t=0 = 0$

$d2y/dt2|t=0 = (\mu Tz)/(T1T2)$

Studiamo con MATLAB l'andamento della risposta al variare della costante di tempo dello zero:

```
>>vecTz=T*[-0.2 0.1 0.5 1 1.2]; >>for k=1:length(vecTz)
step(tf(mu*[vecT2(k) 1],conv([T 1],[0.1*T 1])),t),
hold on, end
>>hold off
```



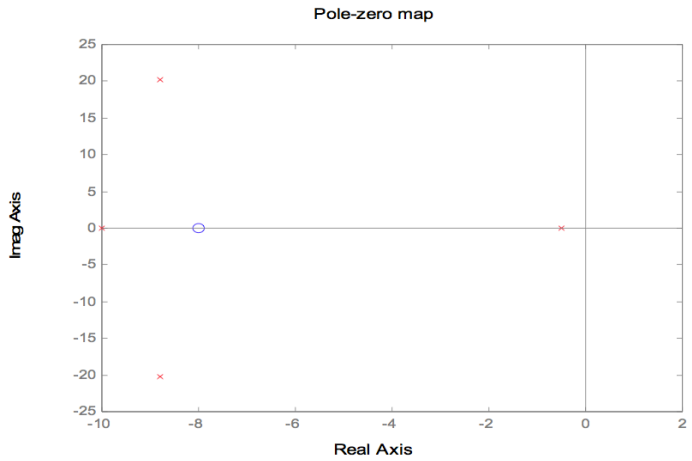
Notare la presenza di sottoelongazioni o sovraelongazioni a seconda della posizione dello zero rispetto a quella dei poli. → comando pzmap per mostrare la mappa poli/zeri. Si veda la slide successiva per un esempio.

Sistemi con ordine maggiore di 2

Consideriamo un sistema (asintoticamente stabile) del 4° ordine e vediamo dove sono nel piano complesso i suoi poli e zeri (si usa il comando pzmap):

```
>>mu=1; Tz=1/8; csi=0.4; wn=22; T1=0.1; T2=2;
>>S=tf([Tz 1],conv(conv([T1 1],[T2 1]),[1/wn^2 2*csi/wn 1]));
>>pzmap(S);
```

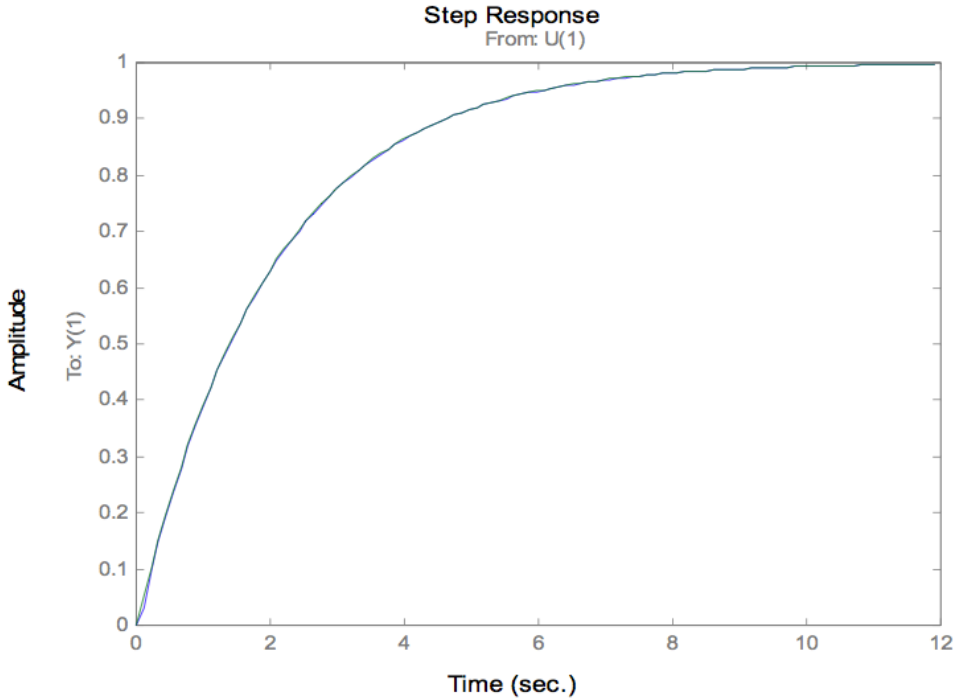
Chiediamoci da cosa dipende la sua dinamica dominante, ovvero la dinamica più lenta dei suoi transitori (ad esempio, di risposta a scalino).



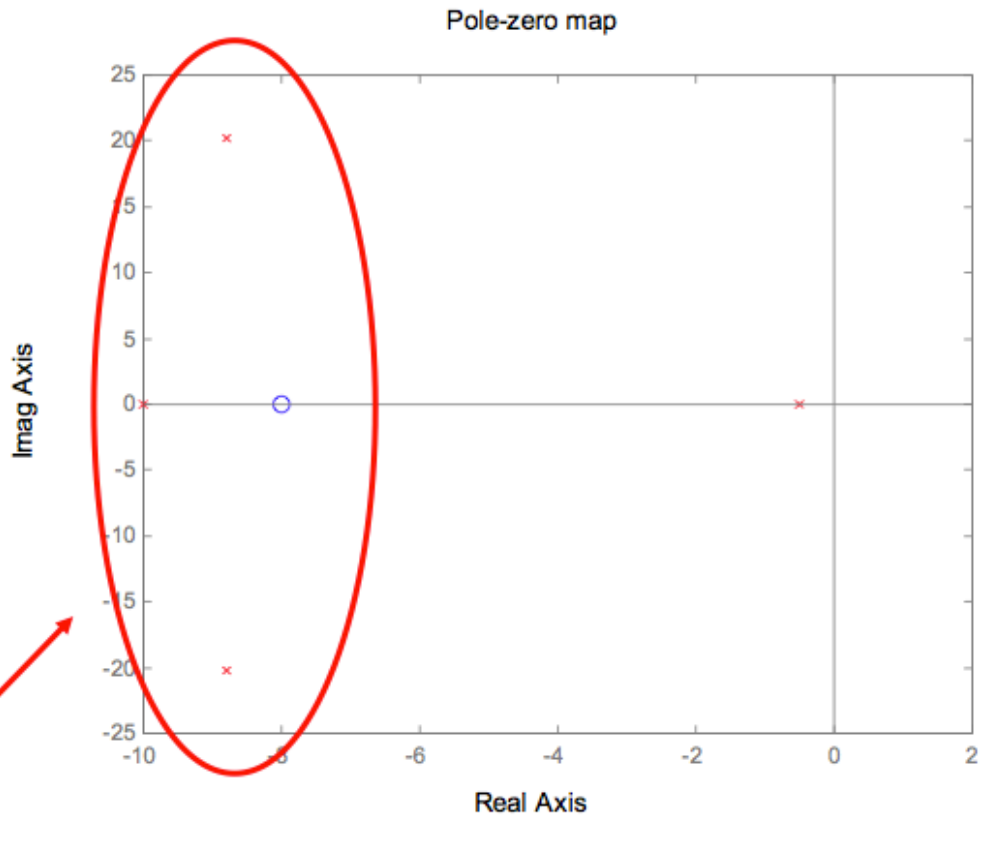
Sistemi con ordine maggiore di 2

Verifichiamo che la sua risposta a scalino è praticamente uguale a quella di un sistema con lo stesso guadagno (ovvio) e il solo polo più lento, ossia quello con la costante di tempo più grande:

```
>>step(S,tf(mu,[T2 1]));
```



Sistemi con ordine maggiore di 2

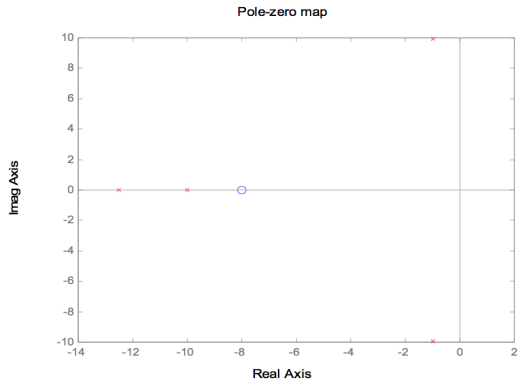


Le singolarità più lontane dall'asse immaginario hanno un effetto trascurabile sull'andamento della risposta: il loro effetto si esaurisce rapidamente nei primi istanti del transitorio.

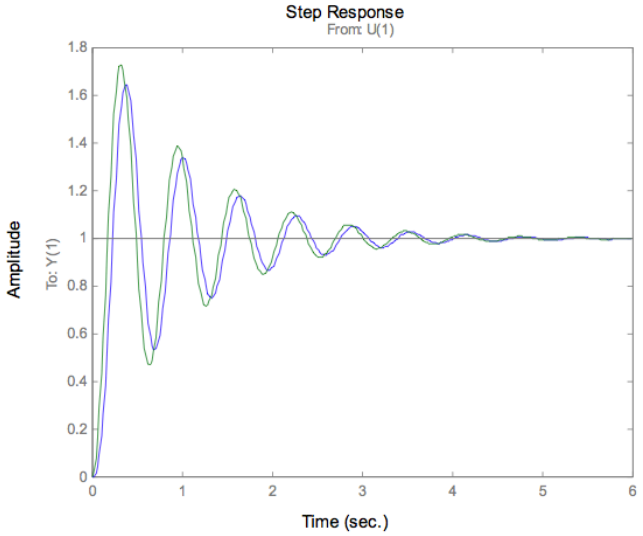
Sistemi con ordine maggiore di 2

Consideriamo un altro sistema (asintoticamente stabile) del 4° ordine, dove le singolarità “dominanti”, ovvero più vicine all'asse immaginario, sono una coppia di poli complessi coniugati, e verifichiamo che la sua risposta a scalino è praticamente indistinguibile da quella di un sistema del 2° ordine con lo stesso guadagno e la sola coppia di poli complessi:

```
>>mu=1; Tz=1/8; csi=0.1; wn=10; T1=0.1; T2=0.08;
>>S=tf([Tz 1],conv(conv([T1 1],[T2 1]),[1/wn^2 2*csi/wn 1]));
>>pzmap(S);
```




```
>>step(S,tf(mu,[1/wn^2 2*csi/wn 1]));
```



Sistemi con ordine maggiore di 2

La ***dinamica dominante*** di un sistema S di ordine > 2 è approssimabile con un sistema del 1° o 2° ordine con

- lo stesso guadagno di S,
- i poli (e gli zeri) di S più vicini all'asse immaginario (purché non vicini rispettivamente ad altri zeri o poli).

Tale approssimazione consente di valutare le caratteristiche "macroscopiche" del transitorio e in particolare la sua durata t_a.

La qualità dell'approssimazione è tanto migliore quanto più rilevante è la separazione tra le singolarità incluse nel modello approssimante e quelle eliminate.

Parte 3: connessione di sistemi con MATLAB. Problemi di cancellazione, raggiungibilità e osservabilità.

Cancellazioni, raggiungibilità e osservabilità nel passaggio da rappresentazione interna a esterna

E’ noto dalla teoria che nel passare da una rappresentazione in spazio di stato ad una in funzione di trasferimento possono avvenire cancellazioni tra poli e zeri. Vediamo cosa significa con l’aiuto del calcolatore.

Sistema non completamente raggiungibile


```
>> A = [-1 0 0; 1/2 -1 0; 1/2 0 -1];
>> B = [1; 0; 0];
>> C = [1 1 0; 0 0 1]; → nota bene: 2 uscite (*)
>> D = 0;
>> S = ss(A,B,C,D);
```

Confrontare gli autovalori della matrice A con i poli della funzione di trasferimento:

```
>> eig(A)
>> F=tf(S) → (*) 2 funzioni di trasferimento
```

Osservare che NON TUTTI I POLI DELLA FDT SONO AUTOVALORI DI A.

Cosa è successo? C'è stata una cancellazione polo/zero!

Calcoliamo il rango della matrice di raggiungibilità:

```
>> Mr = [B A*B A*A*B]
>> Mr = ctrb(A,B) → stessa cosa della riga precedente
>> rank(Mr) → il rango è 2, cioè minore di n=3. SISTEMA NON RAGGIUNGIBILE.
```

Sistema non completamente raggiungibile

Cosa significa che il sistema non è raggiungibile? Supponiamo di voler portare il sistema nello stato di equilibrio $\bar{x}$

Corrispondentemente si avrà: $\bar{x} = -A^{-1}B\bar{u} = -\Gamma\bar{u}$

La matrice Γ nell'esempio vale:

```
>> GAM = inv(A)*B
   -1.0000
   -0.5000
   -0.5000
```

di conseguenza l'equazione all'equilibrio diventa:

$$\begin{aligned} \bar{x}_1 &= \bar{u} \\ \bar{x}_2 &= 0.5\bar{u} \\ \bar{x}_3 &= 0.5\bar{u} \end{aligned}$$

E' evidente come non si è in grado di far assumere al vettore x un valore di regime qualsiasi.

→ Spunto di riflessione: cosa significa in termini di come si è modellizzato/ strumentato il sistema (A,B,C,D) e quale potrebbe essere una soluzione?

Sistema non completamente osservabile

```
>> A = [-1 0 ; 1 -1]
>> B = [1; 0 ]
>> C = [1 0]
>> D = 0
>> S = ss(A,B,C,D)
```

>> tf(S) → si vede che c'è un solo polo! C'è stata una cancellazione.

Calcoliamo il rango della matrice di osservabilità:

```
>> Mo = [C; C*A]
```

>> Mo = obsv(A,C) → stessa cosa della riga precedente

>> rank(Mo) → rango 1 < 2 → Il sistema non è completamente osservabile.

Sistema non completamente osservabile

Cosa significa che il sistema non è osservabile? Supponiamo di voler osservare la risposta $\bar{y}$ di un sistema al regime (ingresso costante).

Corrispondentemente si avrà: $\bar{y} = C\bar{x}$

La matrice C nell'esempio vale:

```
>> C = [1   0]
```

di conseguenza gli stati di equilibrio $\bar{x} = [1 \;\; 0]'$ $\bar{x} = [1 \;\; 1]'$ $\bar{x} = [1 \;\; -0.5]'$

sono indistinguibili dal punto di vista dell'uscita, cioè producono tutti lo stesso segnale di uscita $\bar{y}$.

→ Riflettere: cosa significa in termini di come si è modellizzato/strumentato il sistema (A,B,C,D) e quale potrebbe essere una soluzione?

Parte 4: il circuito RLC serie

Circuito RLC serie

Esercizio:

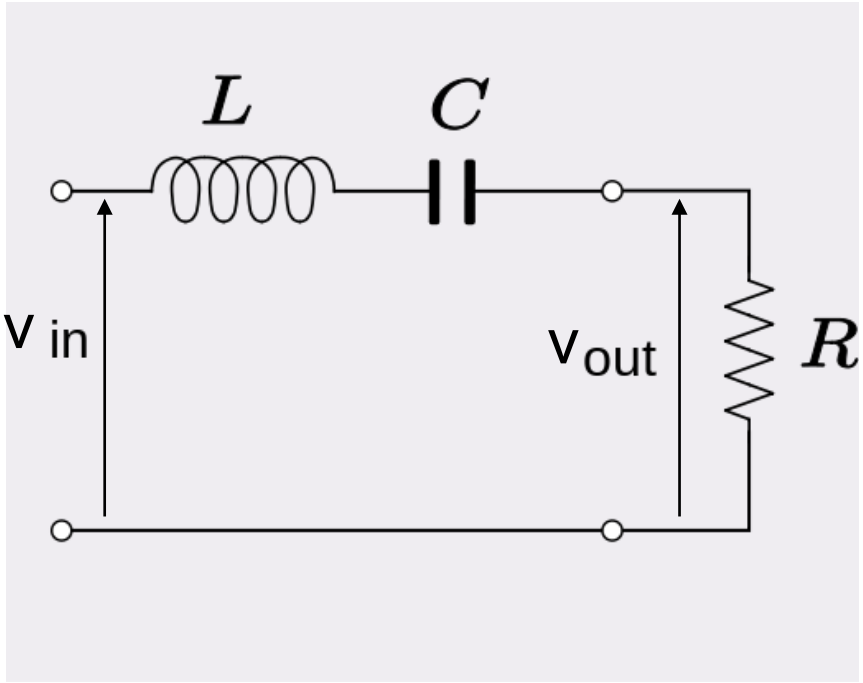
Scrivere il modello del sistema con la convenzione dei segni illustrata in figura chiamando:

X1 = tensione ai capi di C

X2 = corrente che scorre in L

U = tensione di ingresso

Y = tensione ai capi di R



Circuito RLC serie

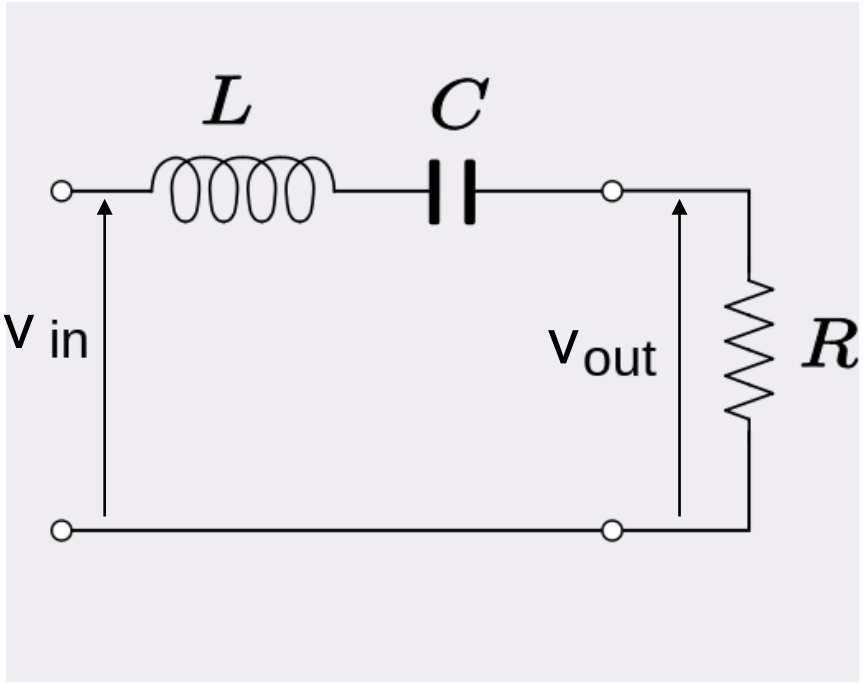
Modello del sistema:

x_1 = tensione ai capi di C

x_2 = corrente che scorre in L

u = tensione di ingresso

y = tensione ai capi di R



$$\begin{aligned} \dot{x}_1(t) &= \frac{1}{C}x_2(t) \\ \dot{x}_2(t) &= -x_1(t) - \frac{R}{L}x_2(t) + u(t) \\ y(t) &= Rx_2(t) \end{aligned}$$

Circuito RLC serie

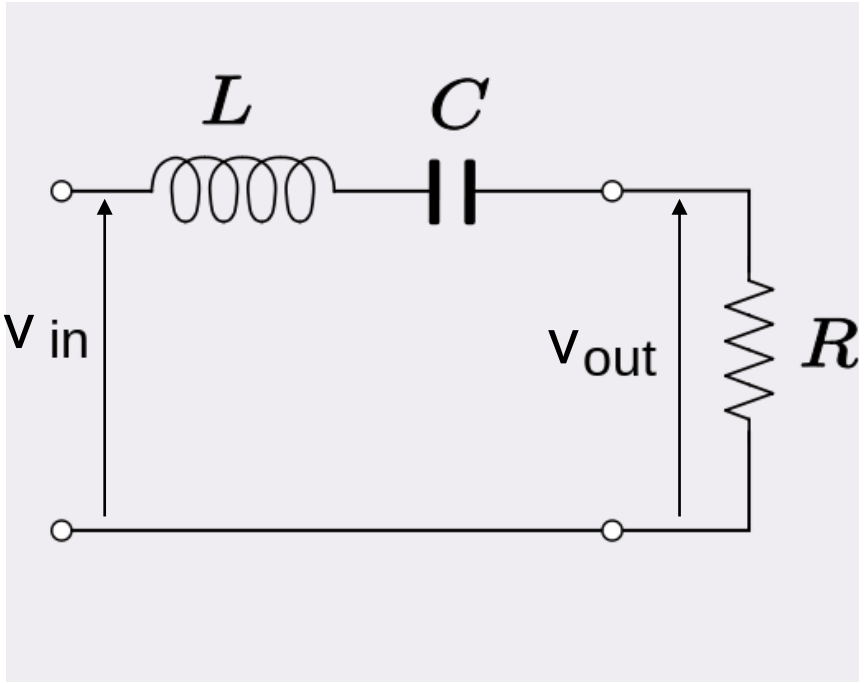
Modello del sistema:

x_1 = tensione ai capi di C

x_2 = corrente che scorre in L

u = tensione di ingresso

y = tensione ai capi di R



$$\begin{aligned} \dot{x}_1(t) &= \frac{1}{C}x_2(t) \\ \dot{x}_2(t) &= -x_1(t) - \frac{R}{L}x_2(t) + u(t) \\ y(t) &= Rx_2(t) \end{aligned}$$

Definire le matrici A B C D del sistema e verificarne controllabilità e raggiungibilità al variare dei parametri.

Circuito RLC serie

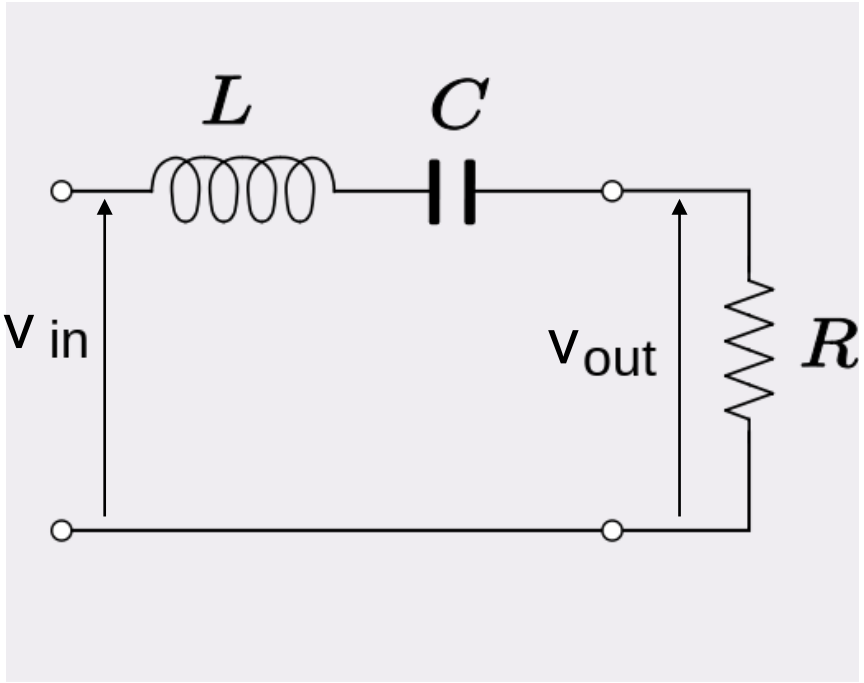
Modello del sistema:

x_1 = tensione ai capi di C

x_2 = corrente che scorre in L

u = tensione di ingresso

y = tensione ai capi di R



$$\begin{aligned} \dot{x}_1(t) &= \frac{1}{C}x_2(t) \\ \dot{x}_2(t) &= -x_1(t) - \frac{R}{L}x_2(t) + u(t) \\ y(t) &= Rx_2(t) \end{aligned}$$

Definire le matrici A B C D del sistema e verificarne controllabilità e raggiungibilità con R = 10, L=1, C=1e-3.

Circuito RLC serie

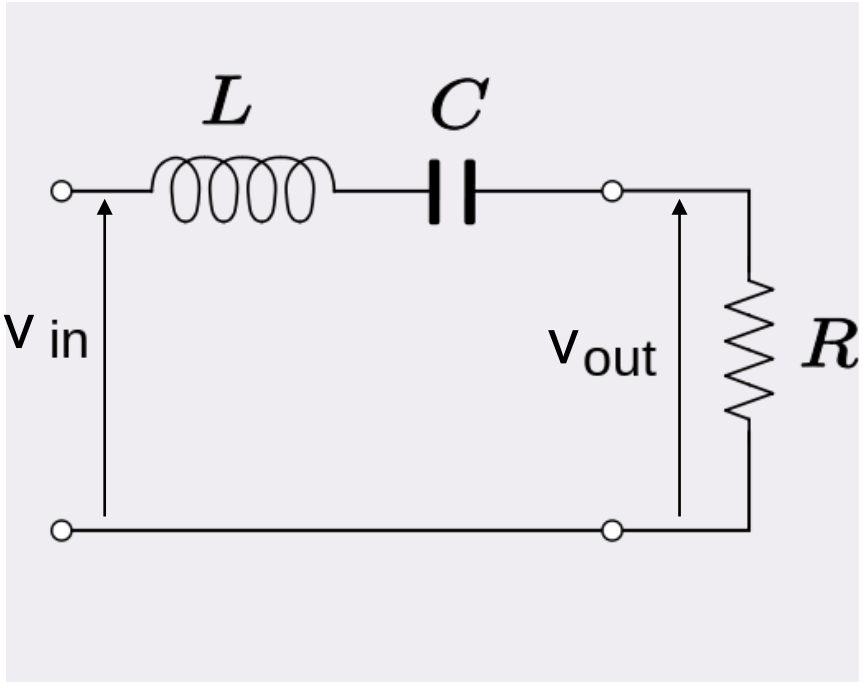
Modello del sistema:

x_1 = tensione ai capi di C

x_2 = corrente che scorre in L

u = tensione di ingresso

y = tensione ai capi di R



Mostrare a schermo la risposta allo scalino con R = 10, L=1, C=1e-3.

Circuito RLC serie: effetto delle variazioni dei parametri

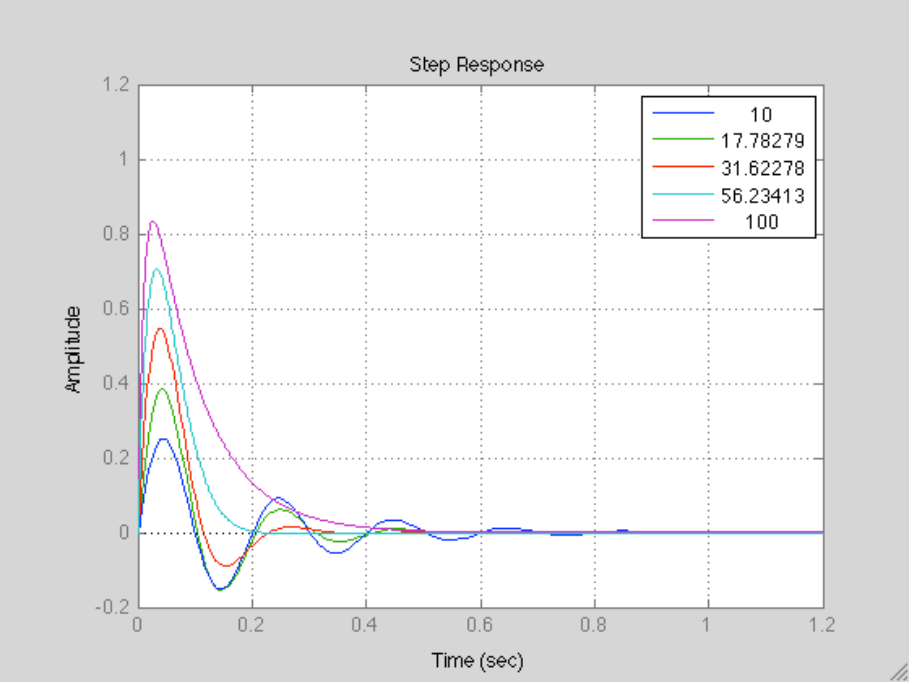
Prendere:

```
>> vecC = 1e-3;
>> vecL = 1;
>> vecR = logspace(1,2,5)
```

Mostrare la risposta a scalino per tutti i valori di R in vecR:

```
>> vecC=1e-3;
>> vecL=1;
>> vecR = logspace(1,2,5)
>> for ii=1:length(vecR)
    A = [0 1/vecC; -1 -vecR(ii)/vecL];
    B = [0; 1];
    C = [0 vecR(ii)];
    D = 0;
    S = ss(A,B,C,D);
    step(S),hold on;
end
>> hold off
>> legend(num2str(vecR'))
```

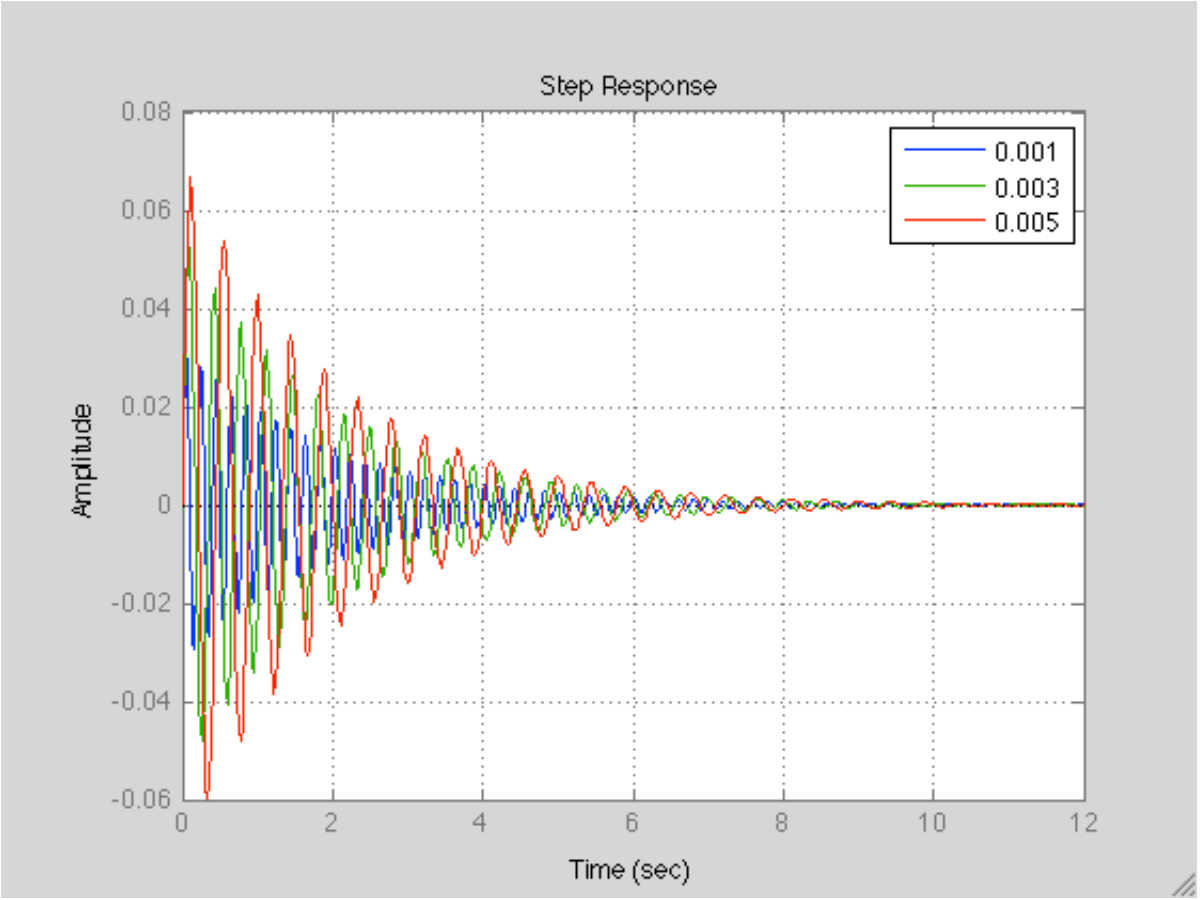
→ num2str converte i numeri in stringhe di testo



Circuito RLC serie: effetto delle variazioni dei parametri

Ripetere l'esercizio per diversi valori di capacità C, fissati R=10 e L=1.

```
>> vecC = 1e-3*(1:2:6);
```



Circuito RLC serie: effetto delle variazioni dei parametri

Ripetere l'esercizio per diversi valori di induttanza, fissati R=10 e C=1e-3.

```
>> vecL = 1:2:6
```