Fondamenti di Automatica

Risposta di Funzioni di Trasferimento
(con Control Systems e Symbolic Math toolbox)

Prof. Marcello Bonfè

Dipartimento di Ingegneria - Università di Ferrara

Tel. +39 0532 974839

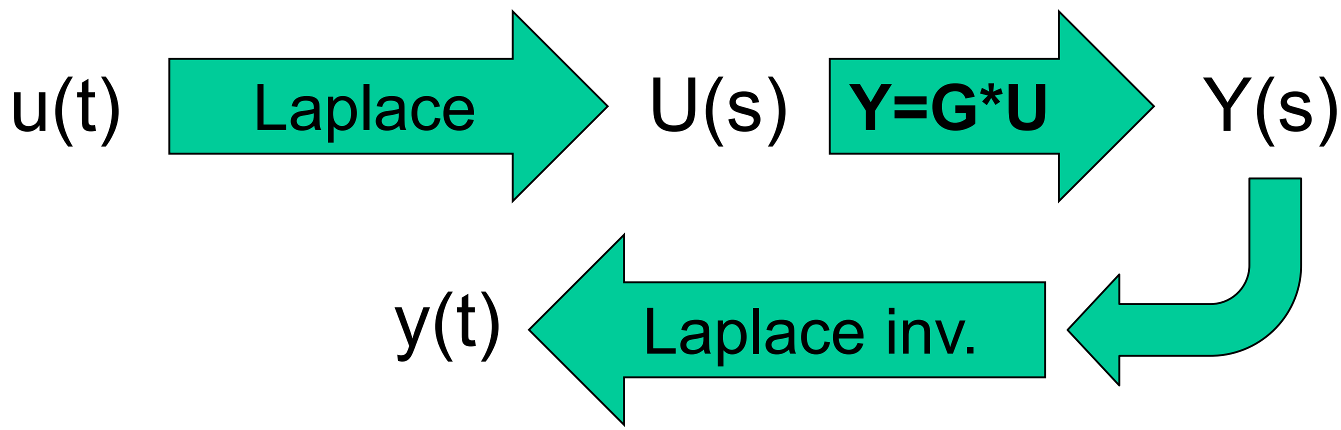
E-mail: marcello.bonfe@unife.it

Department of Engineering Ferrara

Università degli Studi di Ferrara

Matlab: Laplace e risposta del sistema

- Trasformata e anti-trasformata di Laplace permettono di calcolare **l’espressione analitica della risposta** di un sistema rispetto a qualunque segnale, senza svolgere integrali di convoluzione (necessari invece nel dominio del tempo):



Matlab: risposta di Funzioni di Trasferimento

- Lo svolgimento manuale dei calcoli necessari per ottenere l'antitrasformata di Laplace di una funzione razionale (caso tipico delle Funzioni di Trasferimento, FdT, di un Sistema LTI) richiede la **scomposizione in fratti semplici** della funzione:

$$F(s) = \frac{N(s)}{D(s)} = \frac{N(s)}{(s-p_1)(s-p_2)\dots(s-p_n)} = \sum_{i=1}^{n} \frac{k_i}{s-p_i}$$

$$\boxed{k_i = \left[(s-p_i)\frac{N(s)}{D(s)}\right]_{s \to p_i}}$$

Residuo di F(s) nel polo p^i

$$f(t) = \sum_{i=1}^{n} k_i e^{p_i t}$$

Matlab: risposta di Funzioni di Trasferimento

- La scomposizione in fratti semplici è supportata da Matlab sia per operazioni numeriche che simboliche
- **Matlab** (standard):
  - `[R,P,k]=residue(Num,Den)` restituisce il vettore dei residui R, quello dei poli P e l'eventuale termine costante k
- **Symbolic Toolbox**:
  - `partfrac(F,s)` restituisce la funzione F(s) scomposta nelle sue frazioni parziali (i.e. fratti semplici)

Matlab: scomposizione in fratti semplici

- Esempio (**Matlab standard**):

$$F(s) = \frac{6s+26}{s^2+8s+15}$$

```
>> [R,P,k]=residue([6 26],[1 8 15])
R =
     2
     4
P =
    -5
    -3
k = []
```

Matlab: scomposizione in fratti semplici

- Esempio (**Symbolic toolbox**):

$$F(s) = \frac{6s+26}{s^2+8s+15}$$

```
>> syms s
>> F=(6*s + 26)/(s^2 + 8*s + 15)
F =
(6*s + 26)/(s^2 + 8*s + 15)
>> partfrac(F)
ans =
4/(s + 3) + 2/(s + 5)
```

Matlab: trasformate e antitrasformate di Laplace

- **NOTA**: il risultato ottenuto è ovviamente propedeutico all'espressione dell'antitrasformata della F(s), che si ottiene immediatamente ricordando che:

$$\mathcal{L}^{-1}\left[\frac{k_i}{(s-p_i)}\right] = k_i e^{p_i t}$$

e quindi per l'esempio della slide precedente:

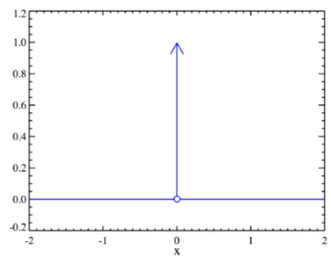
$$f(t) = 4e^{-3t} + 2e^{-5t}$$

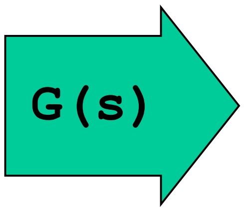
- Il risultato finale corrisponde a quello calcolabile direttamente con **`ilaplace(F)`** (per **`F`** simbolica)

Matlab: risposta della FdT con Control Systems Tlbx

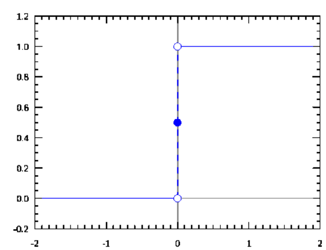
- La F(s) precedente può la Funzione di Trasferimento di un sistema (la cui antitrasformata corrisponde alla risposta impulsiva del sistema) oppure la risposta del sistema ad un certo ingresso (es. FdT*1/s = risposta al gradino!)
- Come già visto (vedi FdA-M.2) il Control Systems Toolbox ha due funzioni specifiche per le risposte *canoniche*:

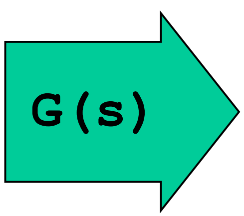
```
>> impulse(G)
```





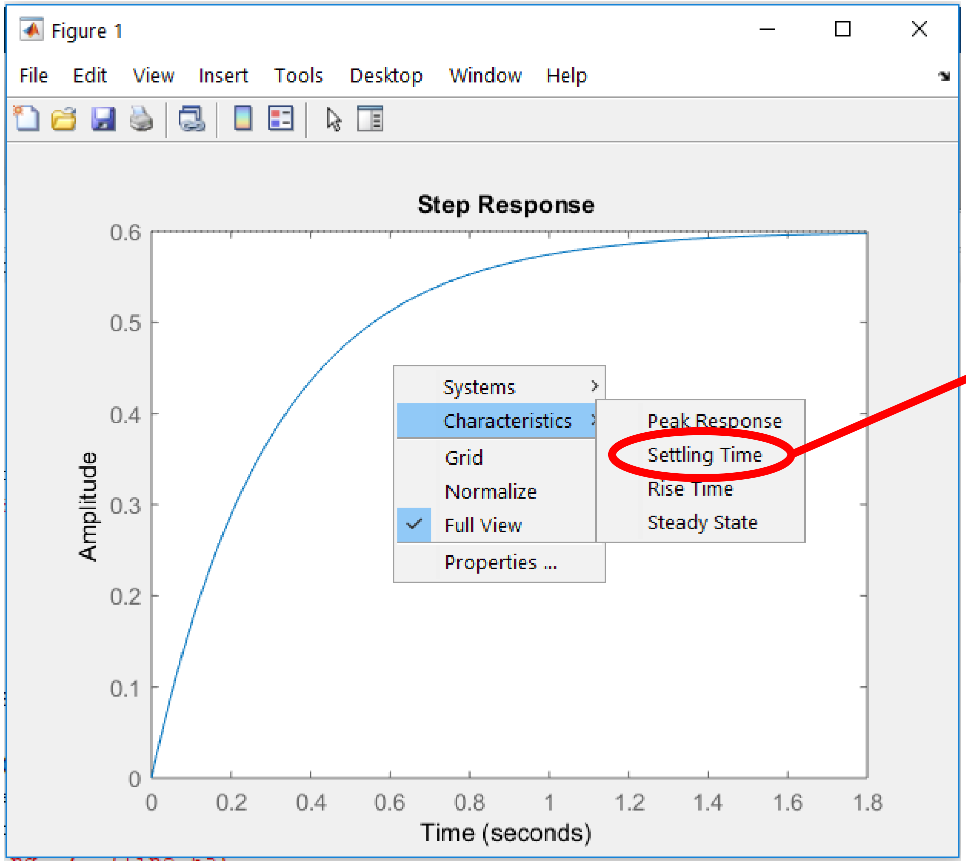

```
>> step(G)
```





Matlab: risposta della FdT con Control Systems Tlbx

- Il grafico ottenuto con il Control Systems Toolbox è interattivo e ricco di funzionalità, supportate tramite il click con tasto destro del mouse…



Tempo di assestamento

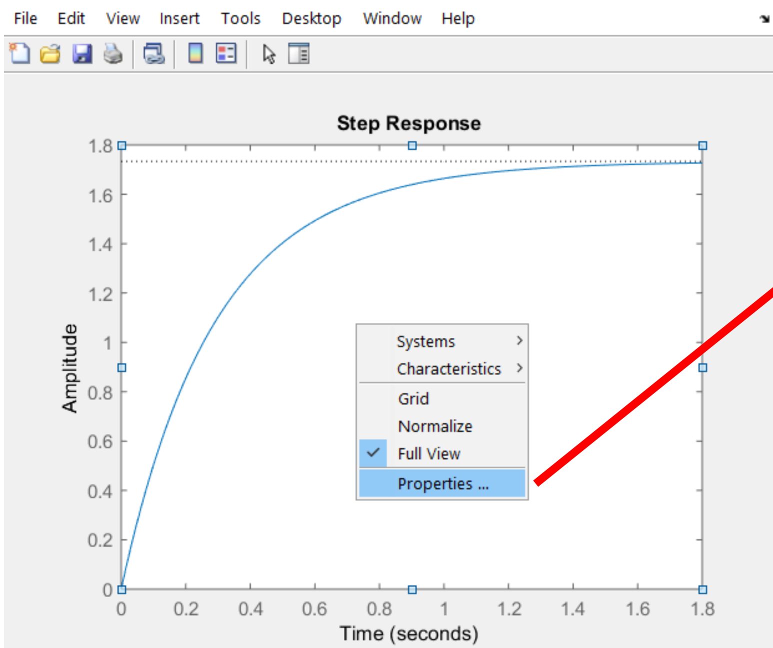
Matlab: risposta al gradino e tempo di assestamento

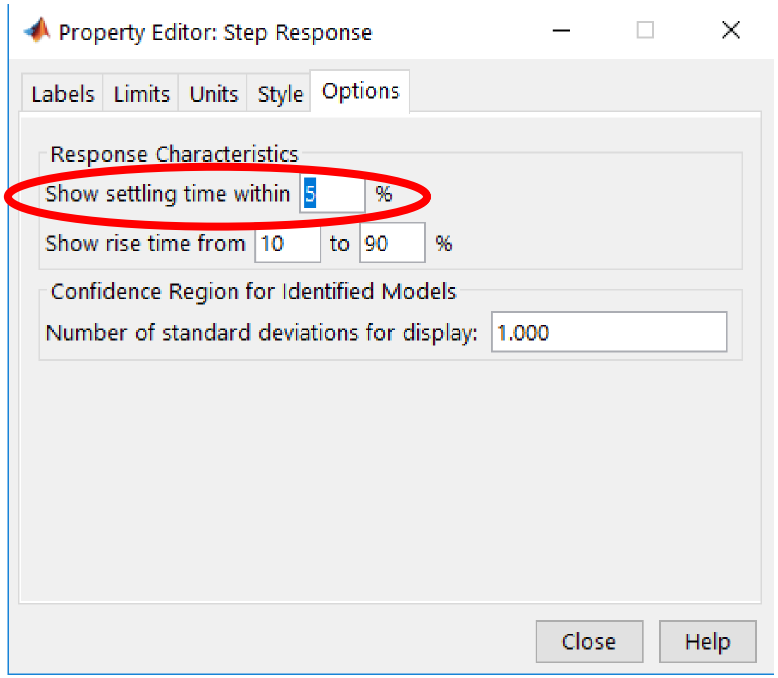
- **NOTA**: è importante osservare che il tempo di assestamento (*settling time*) calcolato da Matlab corrisponde al raggiungimento della risposta di una fascia del +/- 2% (**di default**) rispetto al valore a regime (steady state), cioè per t→∞
- Nelle dispense di questo corso e negli esercizi d'esame si considerano invece **formule per il tempo di assestamento** valide rispetto ad una **fascia del +/- 5%**
- Le impostazioni del grafico ottenuto con **`step()`** si possono (devono!) modificare coerentemente…

Matlab: risposta al gradino e tempo di assestamento

- Modifica delle impostazioni sulla soglia per il tempo di assestamento:

1. Tramite “**Properties” del plot specifico**:





Matlab: risposta al gradino e tempo di assestamento

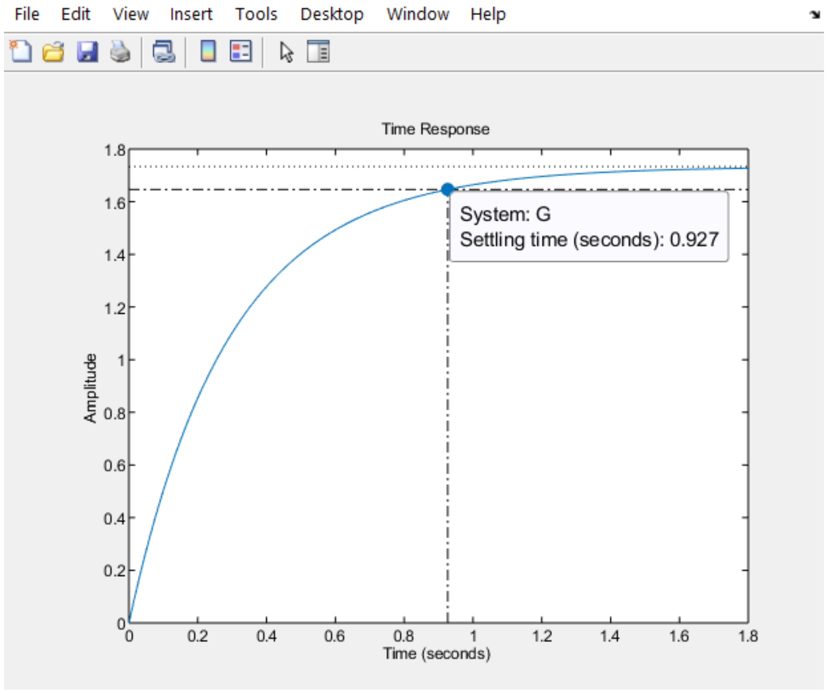
- Modifica delle impostazioni sulla soglia per il tempo di assestamento:

2. Tramite parametro con struttura **`timeoptions`** (riutilizzabile per ogni chiamata successiva):

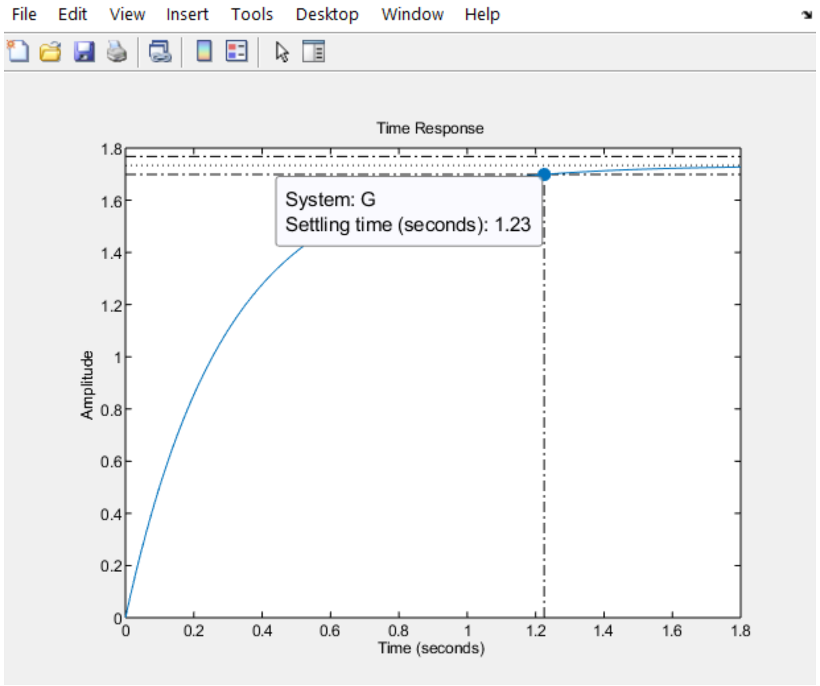
```
>> Popts = timeoptions
>> Popts.SettleTimeThreshold = 0.05
>> step(G,Popts)
```

Matlab: risposta al gradino e tempo di assestamento

T_a al +/-5%



T_a al +/-2%



Matlab: risposta al gradino e tempo di assestamento

- **NOTA**: le caratteristiche importanti della risposta al gradino si possono anche ottenere da prompt, senza passare dal grafico, e con impostazione diretta della soglia per il tempo di assestamento:

```
>> stepinfo(G,'SettleTimeThreshold',0.05)
     RiseTime: 0.6732
     SettlingTime: 0.9265
     SettlingMin: 1.5613
     SettlingMax: 1.7319
     ...
```

Matlab: altri comandi utili per l'analisi di FdT

- **Calcolo di poli e zeri:**

```
>> G=tf([4 5 6],[1 3 2 3])
>> pole(G)
ans =
   -2.6717 + 0.0000i
   -0.1642 + 1.0469i
   -0.1642 - 1.0469i
>> zero(G)
ans =
   -0.6250 + 1.0533i
   -0.6250 - 1.0533i
```

Matlab: altri comandi utili per l'analisi di FdT

- Calcolo di **τ, δ, ω_n per ogni polo** (v. FdA-2.2, slide 33 e 41)

```
>> damp(G)
 Pole                  Damping      Frequency       Time Constant
                                   (rad/seconds)   (seconds)
-1.64e-01+1.05e+00i   1.55e-01     1.06e+00         6.09e+00
-1.64e-01-1.05e+00i   1.55e-01     1.06e+00         6.09e+00
-2.67e+00             1.00e+00     2.67e+00         3.74e-01
```

NOTE:

1. `Damping` è sempre 1 per poli reali,
2. `Frequency` è ω_n per poli complessi e $1/\tau$ per quelli reali
3. `Time Constant` è $1/(\delta\omega_n)$ per poli complessi e τ per quelli reali

PROGETTO CON SPECIFICHE SU τ, δ, ω_n

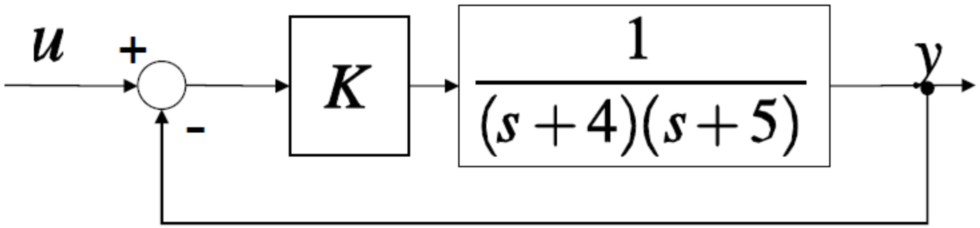
Closed-loop del secondo ordine e vincoli di progetto

- Se il sistema ad anello aperto è del secondo ordine, tale risulterà anche ad anello chiuso!
- In una FdT del secondo ordine:
  - l'***overshoot*** è direttamente e solamente correlato al **coefficiente di smorzamento δ**
  - il **tempo di assestamento** è correlato **sia a δ che alla pulsazione naturale ω_n**

$$\boxed{T_a = \frac{3}{\delta\omega_n}} \quad \text{SE} \quad 0 < \delta < 0.7 \qquad \boxed{T_a = \frac{4.5\delta}{\omega_n}} \quad \text{SE} \quad 0.7 \leq \delta < 1$$

Esempio: progetto di un controllore proporzionale

- Si progetti **K** affinchè il sistema ad anello chiuso abbia δ=0,5:




```
>> syms K s
>> G=K/(s+4)/(s+5)
>> Gcl=G/(1+G)
>> Gcl=collect(Gcl)
Gcl=
K/(s^2 + 9*s + K + 20)
```

Esempio: progetto di un controllore proporzionale

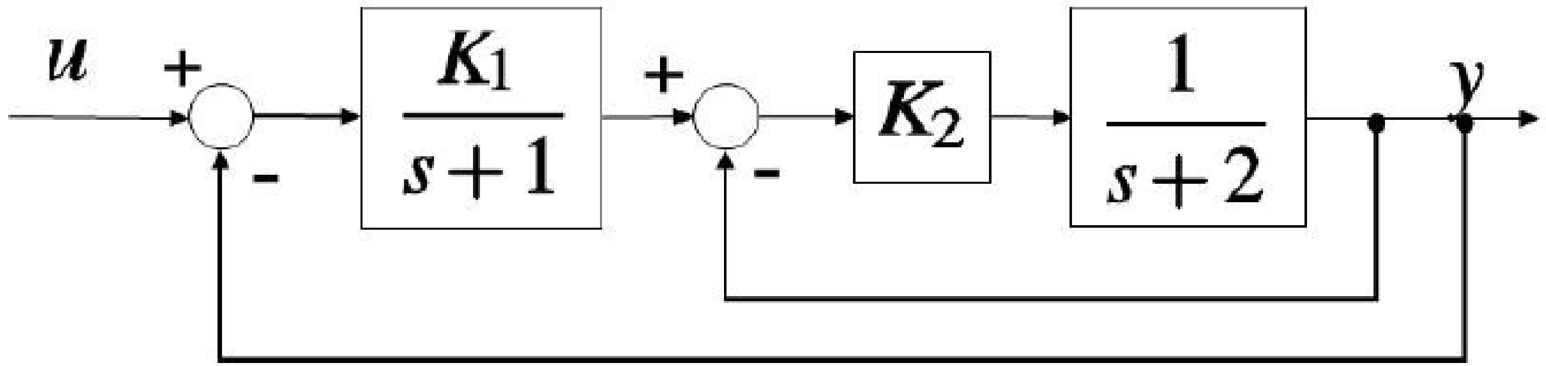
- Ora il denominatore del sistema ad anello chiuso va confrontato con quello di riferimento per un sistema del secondo ordine: dalle uguaglianze tra i coefficienti si ottengono i vincoli per **K**

$$s^2+2\delta\omega_n s+\omega_n^2 = s^2+9s+K+20$$

```
>> syms omega_n
>> delta=0.5
>> solve([9==2*delta*omega_n;
            K+20==omega_n^2],[K,omega_n])
```

Esempio: progetto con due parametri

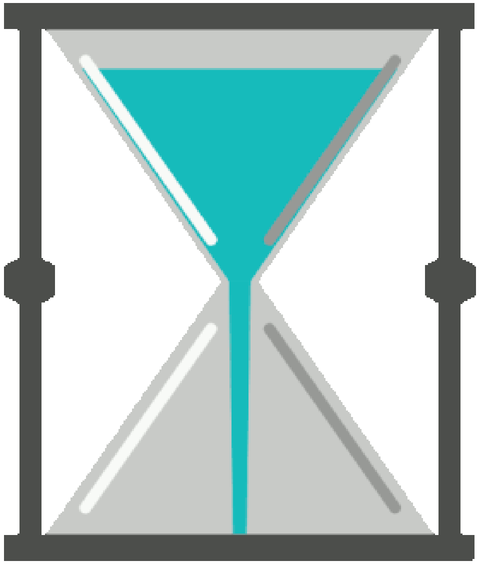
- Si progettino **K_1** e **K_2** affinchè il sistema ad anello chiuso risulti avere tempo di assestamento T_a=0,5 secondi e coefficiente di smorzamento δ=0,6:



NOTA: poichè è specificato un δ<0,7, si può fare riferimento alla formula $T_a = 3/(\delta\omega_n)$

Esempio: progetto con due parametri

Let's do it!!



PROGETTO CON SPECIFICHE SULL’ERRORE A REGIME

Matlab: altri comandi utili per l'analisi di FdT

- **Calcolo del *guadagno statico*** (i.e. G(s) per s →0)

```
>> dcgain(G)
```

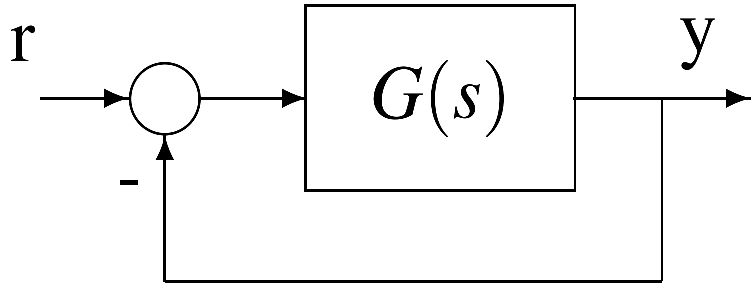
NOTA: `dcgain()` può essere di aiuto per la **verifica dell'errore a regime**, insieme alle funzioni `step()` e/o `stepinfo()`. Il risultato di queste ultime, però, può essere inutilizzabile allo scopo, SE il sistema ad anello chiuso è instabile (cosa che dal `dcgain()` non si evince..).

Esempio: errore a regime in risposta al gradino unitario del sistema con `G` posta in retroazione negativa unitaria:

```
>> e_inf=1/(1+dcgain(G))
e_inf =
     0.3333
```

Esempio: errore a regime con retroazione unitaria

- Si consideri il sistema con **`G=4/(s^2+4*s+2)`** posta in retroazione negativa unitaria:



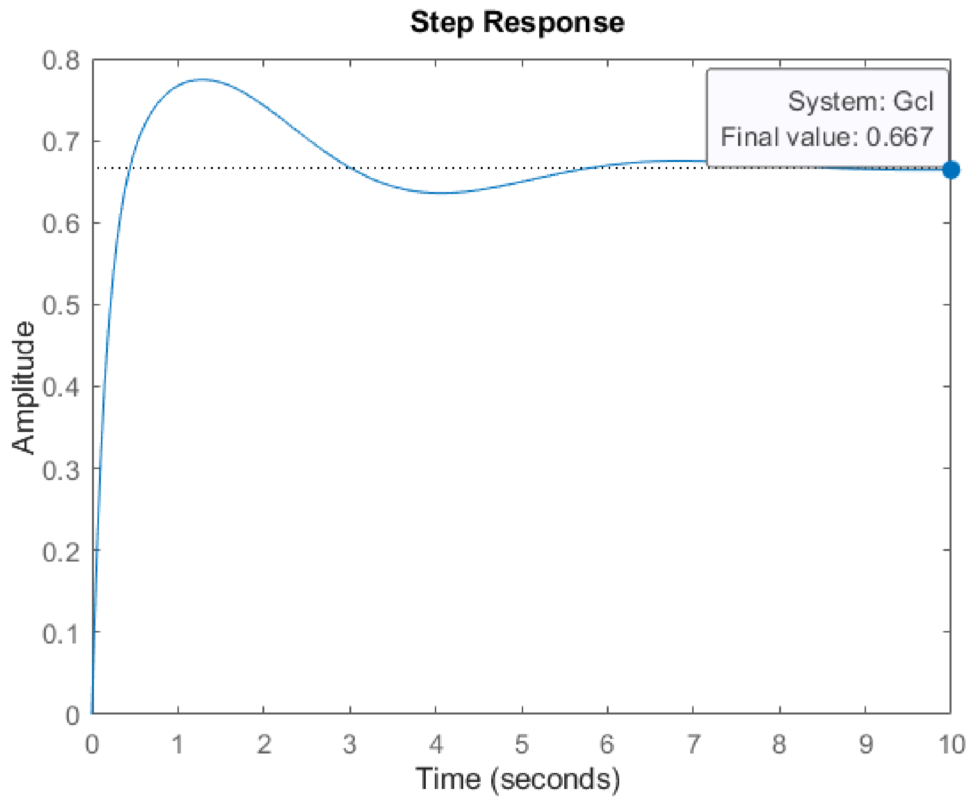

```
>> e_inf=1/(1+dcgain(G))
e_inf =
    0.3333
```

Matlab: altri comandi utili per l'analisi di FdT

- **Verifica grafica dell'errore a regime**: differenza tra 1 (ingresso a gradino unitario!) e "Steady-state"

```
>> Gcl=feedback(G,1)
>> step(Gcl)
```

Errore a regime = 1 – Final value



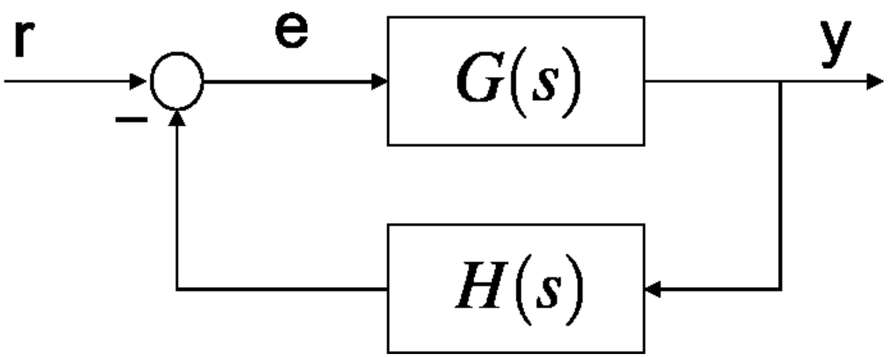
Matlab: altri comandi utili per l'analisi di FdT

- **Progetto del guadagno K** per ottenere errore a regime specificato con **Symbolic Math toolbox**

```
>> syms K
>> e_inf=1/(1+K*dcgain(G))
>> K=solve(e_inf==0.01)
>> K=double(K)
>> Gcl1=feedback(K*G,1)
>> step(Gcl1)
```

Errore a regime con retroazione NON unitaria

- **NOTA:** nel caso in cui si abbia un blocco con FdT H(s) nel ramo di retroazione:



le operazioni descritte in precedenza diventano

```
>> e_inf=1/(1+K*dcgain(G*H))
...
>> Gcl1=feedback(K*G,H)
```

Errore a regime =
1 – [Final value * dcgain(H)]

RISPOSTA DI FdT IN MATLAB

FINE