Risoluzione Esercizio Lab. Informatica Grande 15 maggio 2017

Viene assegnato il sistema descritto dalla funzione di trasferimento:

$$G(s) = \frac{1}{(s+1)^3}$$

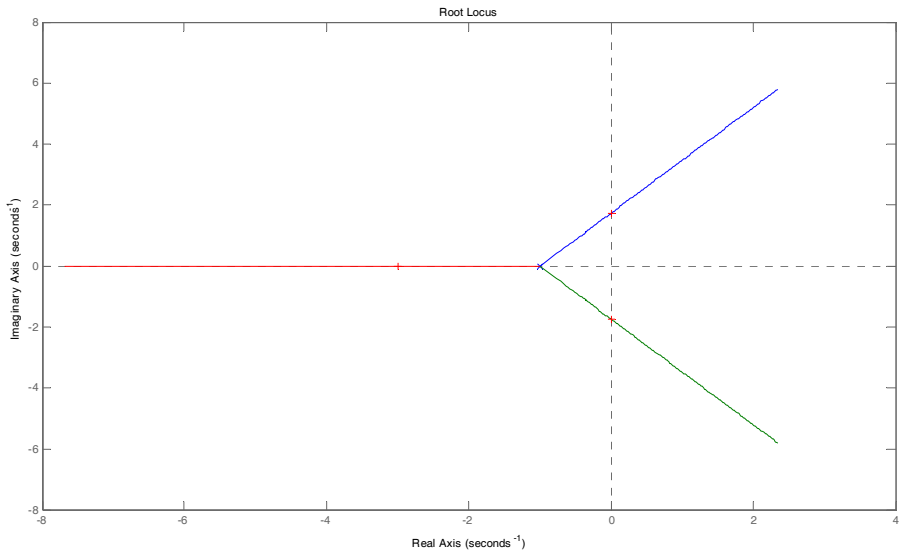
e viene successivamente definito in Matlab attraverso i seguenti comandi:

```
>> s=tf('s')

Transfer function:
s

>> Gs=1/(s+1)^3

Transfer function:
          1
---------------------
s^3 + 3 s^2 + 3 s + 1

>> [numGs,denGs]=tfdata(Gs,'v')

numGs =

     0     0     0     1


denGs =

     1     3     3     1

>>
```

Attraverso lo strumento del luogo delle radici in Matlab per la funzione di trasferimento $G(s)$ col comando `rlocus` si determina il guadagno critico K_c del sistema mediante il comando

`rlocfind` andando a determinare col puntatore del mouse uno dei due punti di intersezione del luogo delle radici stesso con l'asse delle ordinate:

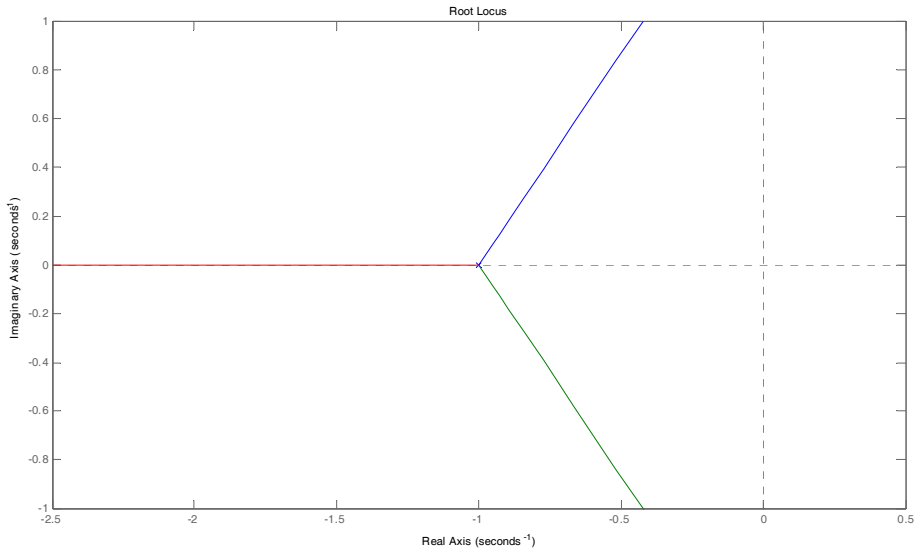
```
>>
>> rlocus(Gs,[0:0.1:300])
>>
```



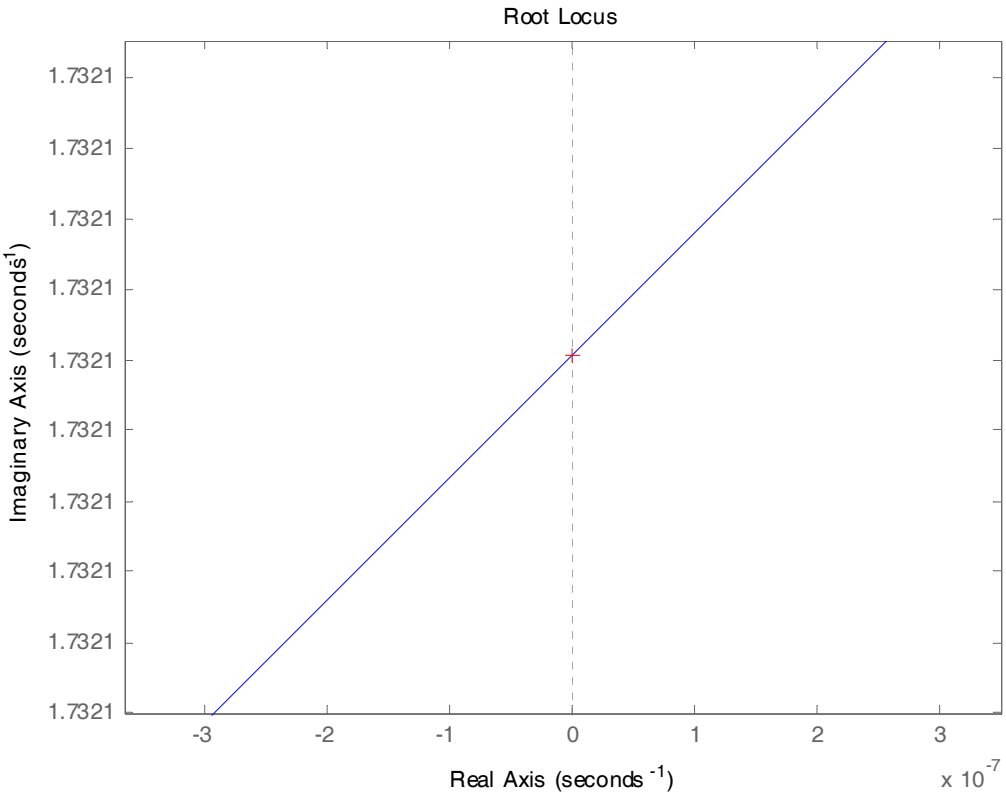
Si osservi come al comando `rlocus` sia stato assegnato un valore predefinito di punti in cui calcolare il luogo delle radici, altrimenti, con il solo comando:

```
>>
>> rlocus(Gs)
>>
```

si sarebbe ottenuto il grafico seguente:



che non consentiva di definire i punti di intersezione del luogo con l'asse delle ordinate. Successivamente, ingrandendo in maniera opportuna il grafico in prossimità di uno dei due punti di intersezione, si ottiene:




```
>> Kc=rlocfind(Gs)
```

```
Select a point in the graphics window

selected_point =

   0.0000 + 1.7321i


Kc =

    8.0000
```

Il periodo delle oscillazioni critiche T_u è definito come la parte reale del valore dell'ordinata determinato precedentemente, ovvero 1.7321 i, cioè $\omega_c \; i$, attraverso la formula:

$$T_u = \frac{2\ \pi}{\omega_c}$$

implementati secondo i seguenti comandi:

```
>>

>> Wc=1.7321

Wc =

    1.7321

>> Kp=0.6*Kc;
>>
>> Tu=2*pi/Wc

Tu =

    3.6275
```

Le formule di Ziegler-Nichols usate per questo progetto sono definite come segue, e dipendono appunto dal guadagno critico K_c, e dal periodo delle oscillazioni critiche T_u:

$$\begin{cases} K_p = 0.6 K_c \\ K_i = 2\dfrac{K_p}{T_u} \\ K_d = K_p \dfrac{T_u}{8} \end{cases}$$

```
>> Ki = 2*Kp/Tu

Ki =

    2.6465

>> Kd = Kp*Tu/8

Kd =

    2.1765

>>
```

Si osservi come le stesse relazioni, calcolabili facilmente a linea di comando, possono essere invece definite mediante una funzione Matlab, in maniera più pratica, e dipendente proprio dai parametri di ingresso K_c e T_u. Le variabili di uscita saranno invece i parametri da calcolare del PID nella sua forma parallela, K_p, K_i, e K_d.

```
function [Kp,Ki,Kd] = PID_tune_ZN(Kc,Tu)
% La funzione [Kp,Ki,Kd] = PID_tune_ZN(Kc,Tu) calcola il guadagno
% proporzionale, il guadagno integrale e quello derivativo,
% rispettivamente Kp, Ki e Kd, per il PID % implementato nella sua
% forma parallela, quando Kc è il guadagno critico, mentre Tu è il
% periodo delle oscillazioni critiche. I guadagni vengono
% calcolati secondo le seguenti formule di Ziegler-Nichols:
%
% Kp = 0.6 * Kc;
% Ki = 2 * Kp / Tu;
% Kd = Kp * Tu / 8;
%

Kp = 0.6 * Kc;
Ki = 2 * Kp / Tu;
Kd = Kp * Tu / 8;
```

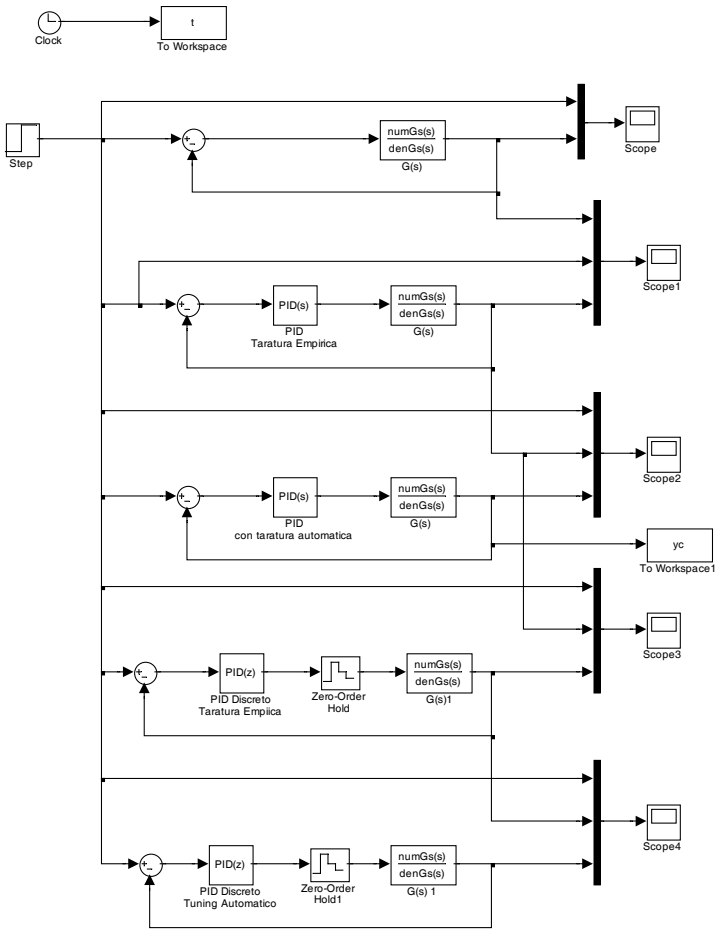
```
return
```

Una volta definito quindi il function file col nome "`PID_tune_ZN.m`", nella stessa directory di lavoro in cui si sta lavorando con Matlab, si effettua la chiamata alla funzione stessa con i parametri calcolati sopra:

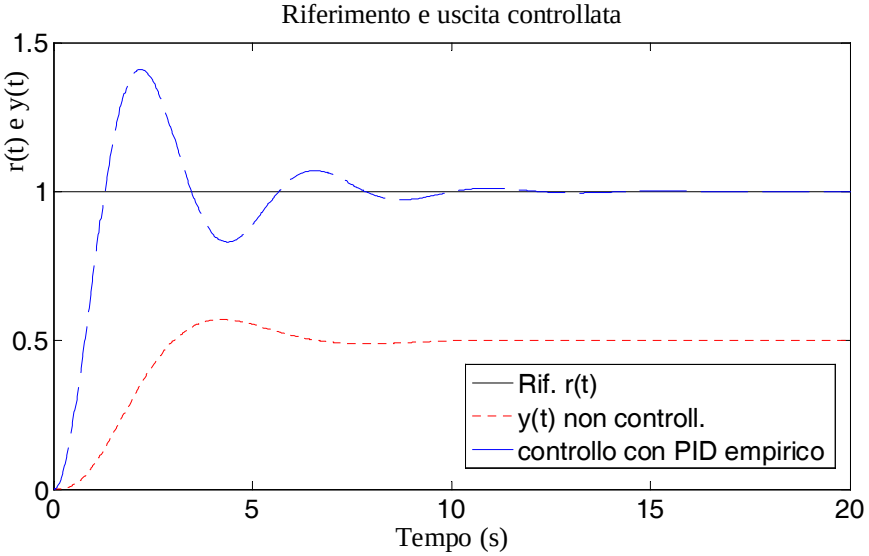
```
>> [Kp,Ki,Kd] = PID_tune_ZN(Kc,Tu)

Kp =

    4.8000


Ki =

    2.6465


Kd =

    2.1765

>>
```

Tali parametri vengono inseriti nello schema Simulink come segue, che contiene già il PID i cui paramerti saranno quelli calcolati empiricamente, e le cui prestazioni verranno confrontate con quelle del PID progettato con il metodo di taratura automatica definito dal blocco stesso Simulink del PID.

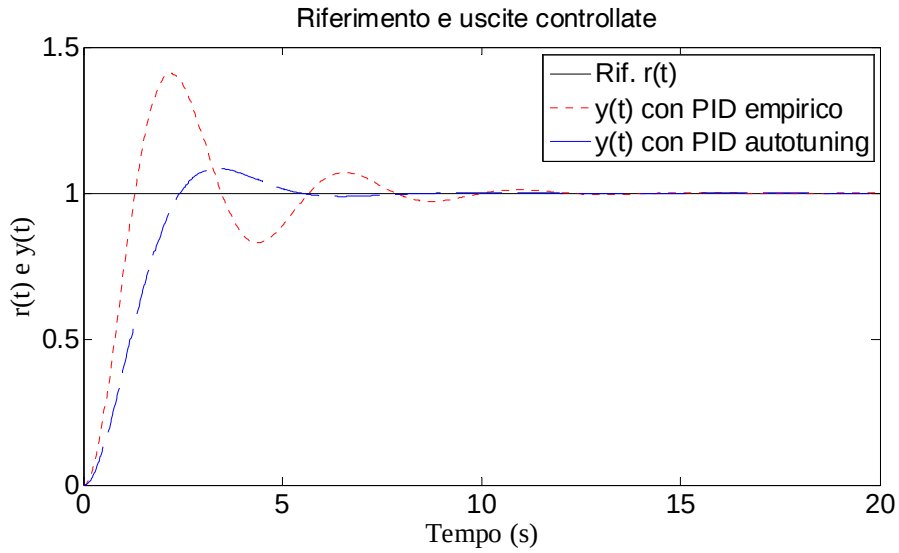
In particolare, nella figura successiva, partendo dall'alto, si costruisce il primo schema in retroazione senza regolatore; successivamente, si inserisce il regolatore standard PID, i cui parametri sono quelli definiti in maniera empirica attraverso le formule di Ziegler-Nichols; il terzo schema riporta invece lo stesso blocco PID i cui parametri vengono calcolati col metodo di taratura automatico definito all'interno stesso del blocco Simulink del PID.



La figura successiva riporta il contronto tra il riferimento, l'uscita del sistema non controllato, e l'uscita del sistema controllato dal PID i cui parametri sono stati ottenuti empiricamente.



La figura successiva riporta il confronto tra uscita del sistema controllato con PID empirico e lo stesso regolatore i cui parametri sono stati calcolati in maniera automatica dallo stesso blocco Simulink del PID.



Si osservino i notevoli miglioramenti di prestazioni che possono essere ottenuti con il PID i cui parametri sono stati ottimizzati dal procedimento di taratura automatico implementato nel blocco Simulink del PID.

Si passa ora al progetto dei regolatori PID a tempo discreto. Occorre definire un tempo di campionamento opportuno. Se si calcola il tempo di assestamento del regolatore PID ottenuto dal procedimento di taratura automatica:

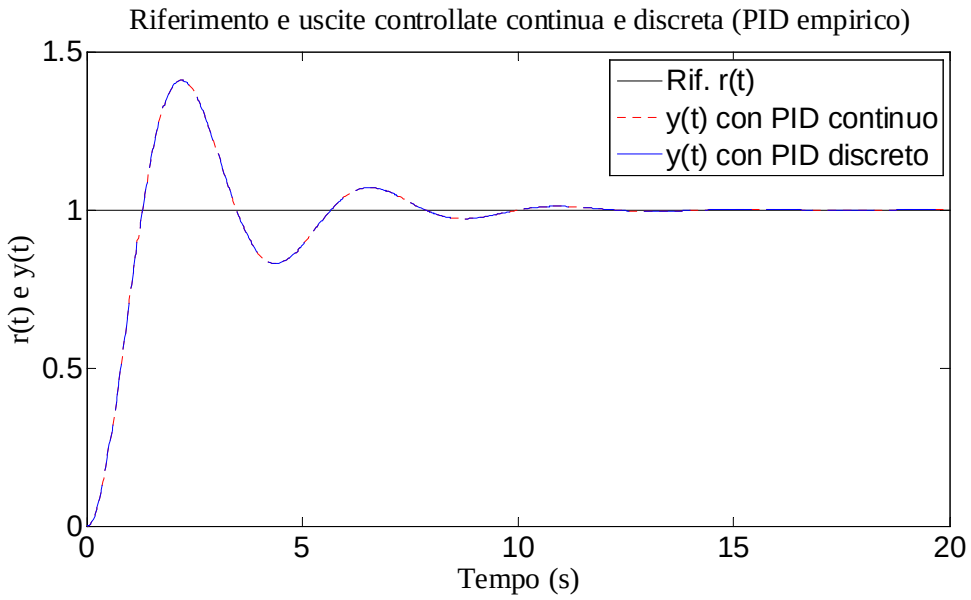
```
>>
>> lsiminfo(yc,t)

ans =

    SettlingTime: 4.9011
             Min: 0
         MinTime: 0
             Max: 1.0843
         MaxTime: 3.3868
```

Si deve quindi scegliere un tempo di campionamento T il cui valore è circa $1/100$ del tempo di assestamento, che vale circa $T_a = 4.9s$. Appare quindi ragionevole la scelta di un tempo di campionamento $T = 0.05s$, arrotondato al valore al valore del primo decimale più vicino (anziché $0.049s$):

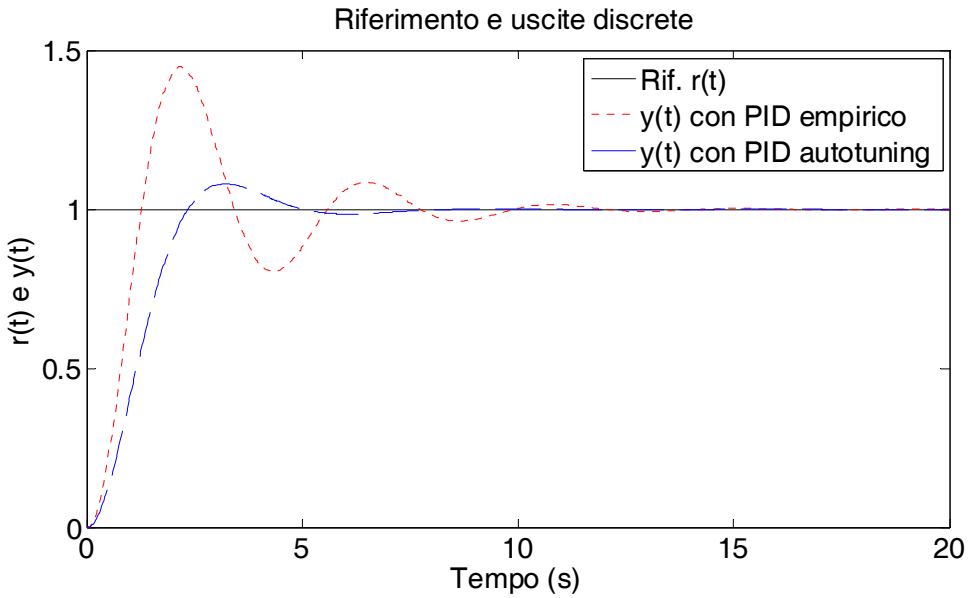
```
>>
```

```
>> T=0.05
```

Nel quarto schema Simulink viene dunque implementato il PID a tempo discreto i cui parametri sono gli stessi di quello ottenuto con taratura empirica, inserendo in cascata il dispositivo di tenuta di ordine zero. Si ricordi che il regolatore PID digitale deve essere discretizzato con il metodo di Eulero in Avanti (EA) per la parte integrale, mentre si utilizza il metodo di Eulero all'Indietro (EI) per la parte derivativa. Si lascia al suo valore di default invece il valore del filtro per l'implementazione del filtro del blocco derivativo. La figura seguente riporta il confronto tra uscita del regolatore a tempo continuo e la corrispondente a tempo discreto, secondo la metodologia di taratura empirica attraverso le formule di Ziegler-Nichols. Si ha quindi la conferma qualitativa della scelta adeguata del tempo di campionamento, che fa mantenere anche la stabilità della realizzazione del controllo digitale: le uscite infatti risultano praticamente indistinguibili.



La figura successiva mostra invece il confronto tra le uscite dei PID continuo e discreto i cui parametri sono stati ottimizzati direttamente dal blocco Simulink del PID.



Si osservi infine che il miglioramento rimane anche nel confronto dei due PID a tempo discreto, i cui parametri sono stati ottenuti con le formule empiriche di Ziegler-Nichols e dal blocco Simulink PID a tempo discreto in maniera automatica.

L'ultima figura sotto riporta invece il confronto tra l'uscita del sistema controllato dal PID a tempo continuo con autotuning e l'uscita del sistema controllato dal PID a tempo discreto sempre con autotuning. Rimane la conferma della scelta appropriata del tempo di campionamento.