# Automatica I (Laboratorio)

**di Silvio Simani**

**Dipartimento di Ingegneria, Università di Ferrara**

Versione 1.0, Marzo, 2004

# Indice

# Elenco delle figure

# Capitolo 1

# Introduzione a Matlab

Il *Matlab*, prodotto dalla *Mathworks*[1] Inc. [1, 2] è un programma per l'elaborazione di dati numerici e la presentazione grafica dei risultati [3, 4]. Questo programma è utilizzato estensivamente da ingegneri dell'automazione per l'analisi di sistemi e per il progetto di controllori. Questo capitolo presenta alcune caratteristiche di base del programma. Per approfondimenti su *Mathworks* e sulle relative istruzioni si fa riferimento al manuale in formato *Acrobat reader*.

## 1.1 Funzioni usate nel capitolo

In questo capitolo verranno utilizzati i seguenti files *Maltab*:

`CreaMatrice.m`, esempio di creazione di una matrice in ambiente *Maltab*.

`controllo.m`, esempio di utilizzo delle istruzioni di controllo in *Maltab*.

`matrix.mat`, esempio di memorizzazione di una matrice in *Maltab*.

## 1.2 Istruzioni di Base del Matlab

L'aspetto principale del programma è la semplicità concettuale con cui vengono rappresentati i dati. I dati vengono introdotti nel programma in maniera molto semplice, mediante assegnamento. Ad esempio, con l'istruzione:

```
>> a = 4
```

definiamo la variabile `a` assegnandole il valore `4`. Occorre notare che il programma ribadisce il risultato della istruzione precedente, visualizzandolo sullo schermo:

```
a =

     4
```

[1] Sito web `http://www.mathworks.com`

Il programma non richiede definizioni particolari di tipo durante l'inizializzazione di variabili, ma il tipo viene assegnato automaticamente in funzione del dato inserito. Ad esempio, l'istruzione:

```
>> b = 4+5i
```

definisce ed inizializza la variabile `b` al valore complesso $4 + 5i$. A seguito dell'assegnazione, il programma esegue l'echo del dato introdotto:

```
b =

   4.0000 + 5.0000i
```

### 1.2.1 Vettori e matrici

La dichiarazione e l'inizializzazioni di variabili particolari quali *vettori* e *matrici* avviene nella stessa maniera. In particolare il programma *Matlab* è orientato alla gestione di **matrici**. Infatti in *Matlab* ogni variabile è una *matrice*, gli scalari non sono altro che particolari matrici $1 \times 1$, e le operazioni fondamantali sono definite direttamente sulle matrici le cui dimensioni devono soddisfare determinate regole.

Ad esempio, si consideri la matrice:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

questa può essere definita ed inizializzata in *Matlab* attraverso l'assegnamento:

```
>> A = [1 2 3;4 5 6;7 8 9]
```

a cui il programma risponde con:

```
A =

     1     2     3
     4     5     6
     7     8     9
```

La matrice `A` (si noti che il programma è *case sensistive*, distingue, cioè fra maiuscole e miniscole) viene inserita per righe, il separatore di riga è il punto e virgola (`;`). La matrice è racchiusa tra parentesi quadre. Due elementi contigui della matrice sono separati da uno spazio oppure da una virgola (`,`).

### 1.2.2 Operazioni elementari sulle matrici

Nel seguente sottoparagrafo verranno elencate con esempi le principali operazioni sulle matrici: *addizione, sottrazione, trasposizione, moltiplicazione e divisione*.

Data la matrice `A`

```
A =

     1     2     3
     4     5     6
     7     8     9
```

e definita nella stessa maniera una seconda matrice, `b`:

```
>> b = [2, 4, 5; 6, 9, 11; 4, 56, -2];
```

la *somma* `A + b` delle matrici si ottiene digitando:

```
>> A + b
```

*Matlab* risponde con il risultato:

```
ans =

     3     6     8
    10    14    17
    11    64     7
```

dove `ans` è l'abbreviazione di "answer", ovvero "risposta", vale a dire la variabile che contiene il risultato della elaboraizone richiesta. Volendo conservare tale risultato si può scrivere:

```
>> D = A + b
```

inizializzando una nuova variabile, `D`, contenente il risultato della operazione precedente.

Come si è potuto notare l'operazione di somma è definita in modo matriciale. Questa è una caratteristica costante del linguaggio, se volessimo, ad esempio, calcolare il *seno* dei valori della matrice `D`, sarebbe sufficiente digitare:

```
>> sin(D)

ans =

    0.1411   -0.2794    0.9894
   -0.5440    0.9906   -0.9614
   -1.0000    0.9200    0.6570
```

I vettori sono particolari matrici con 1 colonna e $n$ righe (oppure $n$ colonne ed 1 riga), introducibili in modo analogo a quanto fatto per le matrici:

```
>> v = [1 ; 2; 56; 7]

v =

     1
     2
    56
     7

>> p = [1 2 56 7]

p =

     1     2    56     7
```

La *differenza* di matrici può essere calcolata con il comando

```
>>D-b
```

che produrrà il seguente risultato

```
ans =

     1     2     3
     4     5     6
     7     8     9
```

coincidente con la matrice `A` definita in precedenza.

L'operazione di *trasposizione* (sia di vettori che di matrici) è l'apice (`'`):

```
>> A'
```

```
ans =

     1     4     7
     2     5     8
     3     6     9

>> v'

ans =

     1     2    56     7
```

L'elemento $i$, $j$ della matrice `A` è identificato da `A(i,j)`:

```
>> A(2,3)

ans =

     6
```

È inoltre possibile indentificare un intero vettore (riga o colonna) di una matrice, ed assegnare tale valore ad una nuova variabile:

```
>> vettore = A(1,:)

vettore =

     1     2     3

>> altro_vettore=A(:,2)

altro_vettore =

     2
     5
     8
```

dove con `A(1,:)` si intende "seleziona la prima riga e tutte le colonne", e con `A(:,2)` "seleziona la seconda colonna e tutte le righe". Per selezionare sottomatrici è possibile usare l'istruzione:

```
>> A(1:2,2:3)

ans =
```

```
     2     3
     5     6
```

L'operazione prodotto è definita per matrici di opportune dimensioni. Ad esempio per le matrici

```
>>A=[2,3;5,6;8,9]'

A =

     2     5     8
     3     6     9

>> B=[-1,7,2,-3;-2,2,0,1;-3,1,0,0]

B =

    -1     7     2    -3
    -2     2     0     1
    -3     1     0     0
```

il comando

```
>>C=A*B
```

porta al risultato

```
C =

   -36    32     4    -1
   -42    42     6    -3
```

In *Matlab* è possibile invertire matrici quadrate e non singolari con l'istruzione `inv(.)`. Ad esempio, data la matrice

```
D =

     0     1     0
     0     0     1
     1     0     0
```

si ha

```
>>inv(D)

ans =

     0     0     1
     1     0     0
     0     1     0
```

*Matlab* prevede due simboli per la divisione: / e \. Supponendo che `A` sia una matrice quadrata e non singolare, con

```
A =

    10    37    64
    13    37    61
    22    61   100
```

e

```
B =

     1
     2
     3
```

l'istruzione

```
>>X = B' / A
```

fornisce come risultato

```
X =

   -0.0333    0.4667   -0.0333
```

che è soluzione di `X * A = B'`. Infatti

```
>>E = X * A

E =

    1.0000    2.0000    3.0000
```

coincide con B'. Mentre la divisione `X = A \ B`,

```
>>X = A \ B

X =

    0.0500
    0.3000
    0.0500
```

è soluzione di `B = A * X`. Infatti

```
>>A*X

ans =

    1.0000
    2.0000
    3.0000
```

coincide con `B`.

### 1.2.3 Funzioni di matrice

In seguito verranno elencate le principali funzioni che hanno come argomento le matrici: autovalori ed autovettori di matrice, potenza di matrice, determinante, rango, norma e pseudoinversa.

Se `A` è una matrice quadrata e **p** uno scalare, l'espressione *potenza di matrice* $A^p$ eleva la matrice `A` alla potenza **p**. Se **p** è intero, l'espressione viene calcolata mediante iterazioni ripetute (e.g. $A^3 = A * A * A$).

Data la matrice `A` tale che

```
A =

     0     1     0
     0     0     1
     0     0     0
```

si ottengono i seguenti risultati

```
>>A^2

ans =

     0     0     1
     0     0     0
```

```
     0     0     0
```

```
>>A^3

ans =

     0     0     0
     0     0     0
     0     0     0
```

Nel caso in cui **p** non sia un numero naturale, `A`$^p$ viene calcolato utilizzando *autovalori ed autovettori*. La funzione `[V,D] = eig(A)` restituisce autovalori (`D`) e autovettori (`V`) della matrice `A`.

```
A =

     1     2
     3     4

>>[V,D] = eig(A)

V =

   -0.8246   -0.4160
    0.5658   -0.9094


D =

   -0.3723         0
         0    5.3723
```

D'altra parte

```
>>A^3

ans =

    37    54
    81   118


>>V*D^3*inv(V)

ans =
```

```
   37.0000   54.0000
   81.0000  118.0000
```

L'*esponenziale di matrice* viene calcolato attraverso la funzione `exp(.)`

```
A =

     1     2
     3     4

>>exp(A)

ans =

    2.7183    7.3891
   20.0855   54.5982
```

E come verifica, se

```
A =

     1     2
     3     4


>>log(exp(A))

ans =

     1     2
     3     4
```

*Determinante e rango di una matrice* vengono calcolati attraverso le funzioni `det(.) rank(.)`.

Come si può facilmente verificare, data la matrice diagonale

```
A =

     1     0
     0     4
```

si ottiene che

```
det(A)
```

```
ans =

     4
```

ed inoltre

```
rank(A)

ans =

     2
```

La *norma*-2 *di una matrice* viene calcolata attraverso il comando `norm(B)`. Essa coincide con la radice quadrata del massimo autovalore della matrice simmetrica `B'*B`. Ad esempio,

```
B =

     1     2
     2     5


>>eig(B)

ans =

    0.1716
    5.8284



>>norm(B)

ans =

    5.8284
```

La *pseudoinversa di matrice* viene calcolata attraverso la funzione `pinv(.)`. Estende il concetto di inversa di matrice, definibile anche per matrici singolari o non quadrate. Data la matrice `C` di dimensioni $m \times n$, in generale la pseudoinversa avrà dimensione $n \times m$.

```
A =

     1     2     3
```

```
>>pinv(A)

ans =

    0.0714
    0.1429
    0.2143
```

Nel caso di matrice quadrata, diagonale e singolare

```
A =

    0.5000         0
         0    1.0000


>>pinv(A)

ans =

     2     0
     0     1
```

Si noti che la pseudoinversa di una matrice non singolare coincide con l'inversa.

La pseudoinversa di una matrice può essere calcolata attraverso una procedura computazionalmente meno onerosa rispetto quella utilizzata dalla funzione `pinv(.)`. La pseudoinversa è descritta dalla relazione

```
>>pinv(A) = inv(A'*A)*A'
```

quando la matrice `A'*A` risulta non singolare ovvero se A è di rango massimo.

### 1.2.4 Generazione automatica di una matrice

Una matrice può anche essere generata utilizzando le funzioni *built-in* di *Matlab*. Per esempio l'istruzione

```
>> B = magic(3)
```

produce la matrice `B` di dimensioni $3 \times 3$ costituita da numeri naturali compresi tra 1 e $3^2$ con righe e colonne che presentano tutte la stessa somma,

```
B =
```

```
     8     1     6
     3     5     7
     4     9     2
```

Le istruzioni *Matlab* possono essere raccolte in un file di testo, un M-*file*, con suffisso .m. Le istruzioni in esso contenute vengono eseguite mediante l'istruzione costituita dal nome dell' M-*file* stesso.

Se, per esempio, il file `CreaMatrice.m` contiene le istruzioni

```
C = [1,2,3;4,5,6;7,8,9]
```

il comando

```
>> CreaMatrice
```

produce il seguente risultato

```
C =

     1     2     3
     4     5     6
     7     8     9
```

Una matrice può essere anche generata caricandola da un file generato in precedenza da *Matlab* stesso, attraverso il comando `save`.

Per esempio, mediante le istruzioni:

```
>>D = [1,2,3;4,5,6];
>>save matrix D
>>clear D
```

viene inizialmente generata la matrice `D`, successivamente salvata nel file `matrix.mat` (comando `save matrix D`) con lo stesso nome `D` con cui è stata definita in precedenza e successivamente eliminata dall'ambiente di lavoro (definito *workspace*) mediante l'istruzione `clear`.

Con l'istruzione `load`, come nell'esempio seguente,

```
>>load matrix
```

viene caricata la matrice `D` nel *workspace* leggendone il contenuto dal file `matrix.mat`. L'istruzione di caricamento non richiede la conoscenza del nome con cui è stata salvata la matrice.

Il file `matrix.mat` è in formato codificato in binario, cioé non di testo. È possibile però *importare* od *esportare* files di dati in formato *ASCII*. Si possono quindi scambiare dati con programmi esterni a *Matlab*.

Per verificare effettivamente che è stato generato il file `matrix.mat`, basta digitare il comando DOS-*like*

```
>>dir
```

(od equivalentemente il comando UNIX-*like* `ls`) e verrà mostrato il contenuto della directory di lavoro. Nella lista dei files comparirà anche il nome `matrix.mat`. Per visualizzare la directory corrente di lavoro, si utilizza il comando

```
>>pwd
```

### 1.2.5 Istruzioni DOS-*like*

Le istruzioni DOS più utilizzate in ambiente *Matlab* sono

| | |
|---|---|
| `dir` | elenca i files contenuti nella directory corrente. |
| `type` *filename* | visualizza il contenuto del file *filename*. |
| `delete` *filename* | elimina il file *filename* dalla directory corrente. |
| ↑ (freccia sù della tastiera) | richiama le istruzioni digitate in precedenza. |
| `!` *command* | invia l'istruzione *command* al sistema operativo. |

### 1.2.6 *Script-files* e *function-files*

Le istruzioni in linguaggio *Matlab* possono essere raggruppate in file di testo in modo da poter essere salvate e richiamate in un secondo momento. I file che le racchiudono possono ossere di due tipi:

- *Script-file*, che racchiudono in modo semplice una sequenza di istruzioni *Matlab*.
- *Function-file*, che consentono, in ambiente *Matlab*, la definizione di funzione simili a quelle previste nei linguaggi di programmazione standard. Le variabili vengono passate per valore.

Uno *Script-file* viene eseguito semplicemente richiamando il suo nome (senza il suffisso `.m`). Le istruzioni contenute in uno *script-file* lavorano sulle variabili contenute nello *workspace* globale. Tutte le variabili utilizzate dallo *script-file* rimangono disponibili una volta terminata l'esecuzione (si ricordi l'esempio con `CreaMatrice.m`).

Un *function-file* inizia con un'istruzione che contiene la parola `function`. Nella stessa riga vengono dichiarati i parametri di uscita, il nome della *function* e i parametri di ingresso. Una *function* differisce da uno *script* perché lavora su variabili locali e per il fatto che non accede alle variabili globali.

Un esempio di funzione è il seguente e può essere visualizzato utilizzando il comando `type`

```
>>type rank.m
```

cioè vedere il listato della funzione `rank` definita in *Matlab*. Si ottiene il seguente output

```
function r = rank(A,tol)
%RANK   Matrix rank.
%   RANK(A) provides an estimate of the number of linearly
%   independent rows or columns of a matrix A.
%   RANK(A,tol) is the number of singular values of A
%   that are larger than tol.
%   RANK(A) uses the default tol = max(size(A)) * norm(A) * eps.

%   Copyright (c) 1984-97 by The MathWorks, Inc.
%   $Revision: 5.6 $  $Date: 1997/04/08 06:28:04 $

s = svd(A);
if nargin==1
   tol = max(size(A)') * max(s) * eps;
end
r = sum(s > tol);

>>
```

in cui $A$ e $tol$ sono le variabli di ingresso e $r$ la variabile di uscita. La spiegazione sintetica dell'impiego della funzione avviene con il testo preceduto da "%" subito dopo la definizione dell'header della funzione. L'istruzione

```
>>help rank
```

fornisce il seguente risultato

```
    RANK   Matrix rank.
    RANK(A) provides an estimate of the number of linearly
    independent rows or columns of a matrix A.
    RANK(A,tol) is the number of singular values of A
    that are larger than tol.
    RANK(A) uses the default tol = max(size(A)) * norm(A) * eps.
```

ovvero visualizza il contenuto del testo delimitato da "%".

La funzione calcola i valori singolari della matrice di ingresso `A`. Se il parametro di ingresso coincide con la sola matrice `A` (`nargin==1`), la tolleranza viene

calcolata in base alle dimensioni della matrice `A`, al più grande dei valori singolari e all'`eps` di *matlab* (eps=2.2204e-016). Il rango della matrice coinciderà con il numero di valori singolari maggiori di `tol`.

### 1.2.7 Istruzioni di controllo.

| | |
|---|---|
| `for` | ripetizione di un insieme di istruzioni per un numero predeterminato di iterazioni. Deve terminare con `end`. |
| `while` | ripetizione di un insieme di istruzioni fino a quando una condizione rimane vera. Deve terminare con `end`. |
| `if` | istruzione condizionale. Deve terminare con `end`. Si possono utilizzare anche **else** e **elseif**. |
| `break` | interruzione di un ciclo. |

Il seguente esempio illustra l'utilizzo delle istruzioni di controllo

```
%Esempio di utilizzo delle istruzioni di controllo
while 1
    n = input('Introduci un numero intero n (valore negativo per uscire)');
    if n <= 0, break, end
    y = 1;
    for i=1:n,
       if n == 1, y = 2; disp(y);
       else y = 3 * y + 1; disp(y);
       end
    end;
end
```

contenute nello *script-file* `controllo.m`.

## 1.3 Approfondimenti ed ulteriori dettagli.

Questo capitolo ha avuto come scopo quello di famigliarizzare con l'ambiente *Matlab* fornendo alcuni concetti di base sull'utilizzo del programma. Le istruzioni qui presentate sono necessarie e sufficienti per eseguire gli esercizi proposti[2].

Per ulteriori approfondimenti è possibile consultare:

- L'**help** in linea del programma.
- I **manuali** in formato PDF e cartaceo[3].

[2] Anche nei capitolo successivi, ulteriori istruzioni e commenti sul *Matlab* e *Simulink*, verranno presentati quando sarà necessario al fine di svolgere le esercitazioni

[3] i due formati sono completamante equivalenti

- L'help in formato **Ipertestuale**, consultabile con un normale "WEB browser", come *Netscape* o *Internet Explorer*.

### 1.3.1 L'help in linea di Matlab.

Il programma *Matlab* ha una funzione di "help" in linea, organizzato in mniera gerarchica. Digitando la parola chiave:

```
>> help
```

si ottiene una schermata del tipo[4]:

```
HELP topics:

matlab\general       -  General purpose commands.
matlab\ops           -  Operators and special characters.
matlab\lang          -  Programming language constructs.
matlab\elmat         -  Elementary matrices and matrix manipulation.
matlab\elfun         -  Elementary math functions.
matlab\specfun       -  Specialized math functions.
matlab\matfun        -  Matrix functions - numerical linear algebra.
matlab\datafun       -  Data analysis and Fourier transforms.
matlab\polyfun       -  Interpolation and polynomials.
matlab\funfun        -  Function functions and ODE solvers.
matlab\sparfun       -  Sparse matrices.
matlab\graph2d       -  Two dimensional graphs.
matlab\graph3d       -  Three dimensional graphs.
matlab\specgraph     -  Specialized graphs.
matlab\graphics      -  Handle Graphics.
matlab\uitools       -  Graphical user interface tools.
matlab\strfun        -  Character strings.
matlab\iofun         -  File input/output.
matlab\timefun       -  Time and dates.
matlab\datatypes     -  Data types and structures.
matlab\dde           -  Dynamic data exchange (DDE).
matlab\demos         -  Examples and demonstrations.
simulink\simulink    -  Simulink
simulink\blocks      -  Simulink block library.
simulink\simdemos    -  Simulink demonstrations and samples.
simulink\dee         -  Differential Equation Editor
toolbox\local        -  Preferences.

For more help on directory/topic, type "help topic".
```

volendo, ad esempio, saperne di più sulle operazioni elementari matematiche ("Elementary math functions."), è possibile digitare:

[4] La scehrmata può essere leggermente diversa a seconda dei *toolboxes* installati.

```
>> help elfun
```

ottenendo:

```
  Elementary math functions.

  Trigonometric.
    sin         - Sine.
    sinh        - Hyperbolic sine.
    asin        - Inverse sine.
    asinh       - Inverse hyperbolic sine.
    cos         - Cosine.
    cosh        - Hyperbolic cosine.
    acos        - Inverse cosine.
    acosh       - Inverse hyperbolic cosine.
    tan         - Tangent.
    tanh        - Hyperbolic tangent.
    atan        - Inverse tangent.
    atan2       - Four quadrant inverse tangent.
    atanh       - Inverse hyperbolic tangent.
    sec         - Secant.
    sech        - Hyperbolic secant.
    asec        - Inverse secant.
    asech       - Inverse hyperbolic secant.
    csc         - Cosecant.
    csch        - Hyperbolic cosecant.
    acsc        - Inverse cosecant.
    acsch       - Inverse hyperbolic cosecant.
    cot         - Cotangent.
    coth        - Hyperbolic cotangent.
    acot        - Inverse cotangent.
    acoth       - Inverse hyperbolic cotangent.

  Exponential.
    exp         - Exponential.
    log         - Natural logarithm.
    log10       - Common (base 10) logarithm.
    log2        - Base 2 logarithm and dissect floating point number.
    pow2        - Base 2 power and scale floating point number.
    sqrt        - Square root.
    nextpow2    - Next higher power of 2.

  Complex.
    abs         - Absolute value.
    angle       - Phase angle.
    conj        - Complex conjugate.
```

```
    imag        - Complex imaginary part.
    real        - Complex real part.
    unwrap      - Unwrap phase angle.
    isreal      - True for real array.
    cplxpair    - Sort numbers into complex conjugate pairs.

  Rounding and remainder.
    fix         - Round towards zero.
    floor       - Round towards minus infinity.
    ceil        - Round towards plus infinity.
    round       - Round towards nearest integer.
    mod         - Modulus (signed remainder after division).
    rem         - Remainder after division.
    sign        - Signum.
```

A questo punto, volendo sapere come utilizzare la funzione `sin` è sufficiente digitare

```
>> help sin
```

### 1.3.2 Manuali in formato PDF o cartaceo.

Il programma *Matlab* è corredato da una serie di manuali disponibili sia in versione elettronica (in formato PDF o "Portable Document Format") che cartacea.

Il formato PDF è un formato standard per la distribuzione di documenti elettronici. Il programma di visualizzazione, l'*Acrobat Reader* è distribuito gratuitamente ed è disponibile per tutti i principali sistemi operativi[5]. Il numero di manuali disponibili è molto grande, per cui è meglio procedere con ordine. Sono consigliati:

- **Getting Started with MATLAB**. Il manuale da cui partire. Vi sono descritte le funzioni di base, necessarie ad un utilizzatore principiante.
- **Using MATLAB**. Ulteriori informazioni su *Matlab*. Adatto ad un utilizzatore esperto.
- Using MATLAB Graphics. Descrive come utilizzare l'interfaccia grafica di *Matlab* per ottimizzare l'utilizzo della grafica. Ancora un manuale adatto per un utente esperto.
- **Language Reference Manual** e **Graphics Reference Manual**, descrivono tutte le funzioni di base del *Matlab*. Da usare solo come riferimento.

### 1.3.3 Help in formato ipertestuale.

Digitando `helpdesk` all'interno del programma *Matlab*, si aprirà automaticamente il "WEB browser" predefinito sul sistema operativo che si sta usando su di una pagina di help dal contenuto autoesplicativo (si veda Fig. 1.1).

[5]Sito internet `http://www.adobe.com/prodindex/acrobat/readstep.html`, da cui è possibile scaricare il programma.

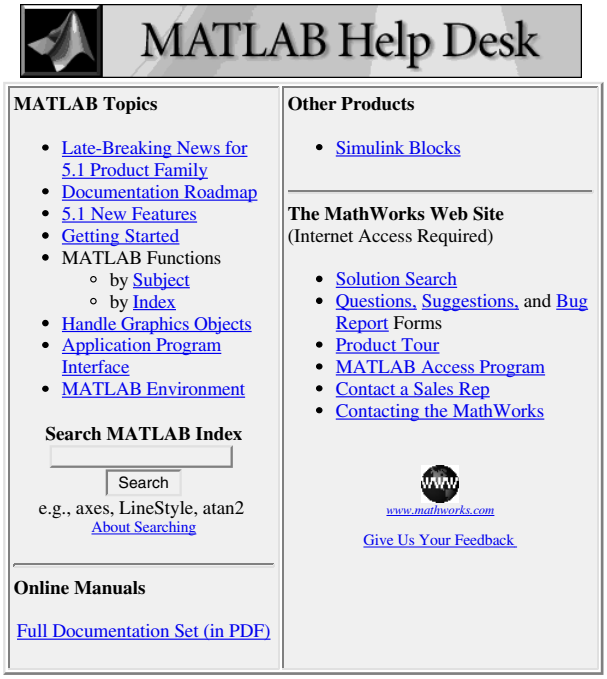



Figura 1.1: Helpdesk di *Matlab*

## 1.4 Esercizi proposti in aula didattica.

1. Scrivere la funzione $H = my\_hankel(X,NrH,NcH,shift)$, in cui X é un vettore di $L$ elementi, NrH é il numero di righe di H, NcH é il numero di colonne di H, e shift é un intero maggiore od uguale a 0. La matrice H deve essere costruita in modo tale che

$$H = \begin{bmatrix} X(1+shift) & \dots & X(shift+NcH) \\ \vdots & \ddots & \vdots \\ X(shift+NrH) & \dots & X(shift+NcH+NrH-1) \end{bmatrix} \quad (1.1)$$

   con l'ipotesi che $L \geq$ shift+NcH+NrH-1.

2. Scrivere un programma che, date le matrici $A_{n\times n}$ e $B_{n\times r}$, costruisca la matrice $P = [B,\ A*B,\ \dots,\ A^{n-1}*B]$. Successivamente effettuare il test del rango.

3. Scrivere un programma che, date le matrici $A_{n\times n}$ e $C_{m\times n}$, costruisca la matrice $Q = [C^T,\ A^T{*}C^T,\ \dots,\ A^{T^{n-1}}{*}C^T]^T$. Successivamente effettuare il test del rango.

4. Data la terna $(\mathrm{A}_{n\times n}, \mathrm{B}_{n\times 1}, \mathrm{C}_{1\times n})$, eseguire un cambiamento di base per determinare la parte raggiungibile (controllabile) del sistema. Successivamente determinare la forma minima.

5. Data la terna $(\mathrm{A}_{n\times n}, \mathrm{B}_{n\times 1}, \mathrm{C}_{1\times n})$, calcolare la matrice $P = [B,\ A * B,\ \ldots,\ A^{n-1} * B]$. Successivamente calcolare le matrici `T1 = im(P)` e `T2`, con `T2` tale che `T = [T1, T2]` sia quadrata e invertibile. Si esegua la trasformazione `Ac = inv(T) * A * T`, `Bc = inv(T) * B` e `Cc = C * T`. Infine, detto $n_c$ il numero di colonne di `T1`, estrarre le matrici `Ac1`, avente le prime $n_c$ righe e colonne di `Ac`, `Bc1` dalle prime $n_c$ righe di `Bc` e `Cc1`, le prime $n_c$ colonne di `Cc`. In maniera analoga, calcolare la matrice $Q = [C;\ (C * A);\ \ldots;\ C * A^{n-1}]$ e effettuare la trasformazione `T` ricavata, come in precedenza, dall'immagine di `Q'` e dal suo complemento ortogonale. Si estraggano quindi le matrici $(\mathrm{A}_o, \mathrm{B}_o, \mathrm{C}_o)$.

   Si esegua l'esercizio con le matrici seguenti:

$$A = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 2 & -2 & 1 & 0 & 0 \\ 3 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 2 & -1 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, C = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{1.2}$$

# Capitolo 2

# Introduzione a Simulink

*Simulink*, prodotto dalla *Mathworks Inc.* è un programma per la simulazione di sistemi dinamici [5]. Estende le potenzialità di *Matlab*, aggiungendo molte funzioni specifiche e mantenendo le caratteristiche generali.

*Simulink* viene utilizzato attraverso due fasi: quella di definizione del modello da simulare e quella di analisi del sistema stesso. Spesso questi due passi vengono eseguiti sequenzialmente modificando i parametri del sistema al fine di ottenere il comportamento desiderato.

Affinché la definizione del modello possa essere immediata, *Simulink* utilizza un ambiente a finestre, chiamate *Block diagram windows* attraverso cui creare i modelli semplicemente impiegando il mouse.

L'analisi del modello avviene sia scegliendo le opzioni dai menu di *Simulink* che riutilizzando i comandi *Matlab* attraverso la *Matlab Command Windows*. I risultati della simulazione sono disponibili durante la fase di simulazione stessa e l'esito finale disponibile nello spazio di lavoro di *Matlab*.

## 2.1 Istruzioni base di Simulink

Per aprire *Simulink* si deve digitare all'interno della *Matlab Command Window* il comando

```
>>simulink
```

che provoca la visualizzazione della finestra (Library: Simulink) contenente le icone delle librerie standard di *Simulink* (vedi Figura 2.1) ed una seconda finestra in cui costruire il modello del sistema da simulare.
I blocchi possono essere copiati dalla prima finestra alla seconda trascinandoli col mouse nella posizione desiderata. Tali blocchi possono essere connessi da linee disegnate sempre col mouse: tenendo premuto il tasto sinistro, partendo dall'uscita di un blocco, col puntatore si crea una nuova connessione all'ingresso ad un altro blocco, mentre premendo il tasto destro posizionati su una connessione preesistente, si genera una diramazione per collegare un altro blocco.

Lo schema viene salvato utilizzando le istruzioni *Save* e *Save as* della tendina *File*. L'istruzione *New* apre un nuovo file *Simulink*, mentre *Open* carica un file *Simulink* salvato precedentemente.

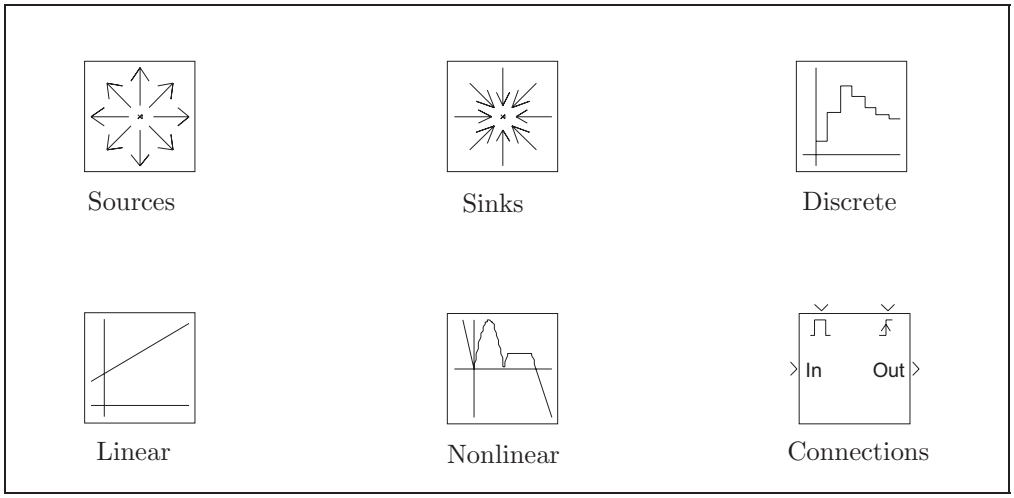



Figura 2.1: *Simulink* block library.

Ciascuna icona della Figura 2.1 contiene i blocchi relativi alla libreria a cui si riferisce. In seguito verranno descritti brevemente i blocchi contenuti in ciascuna libreria

1. **Sources**: (Library: simulink/Sources) contiene alcuni generatori di segnale e vengono visualizzati nella Figura 2.2

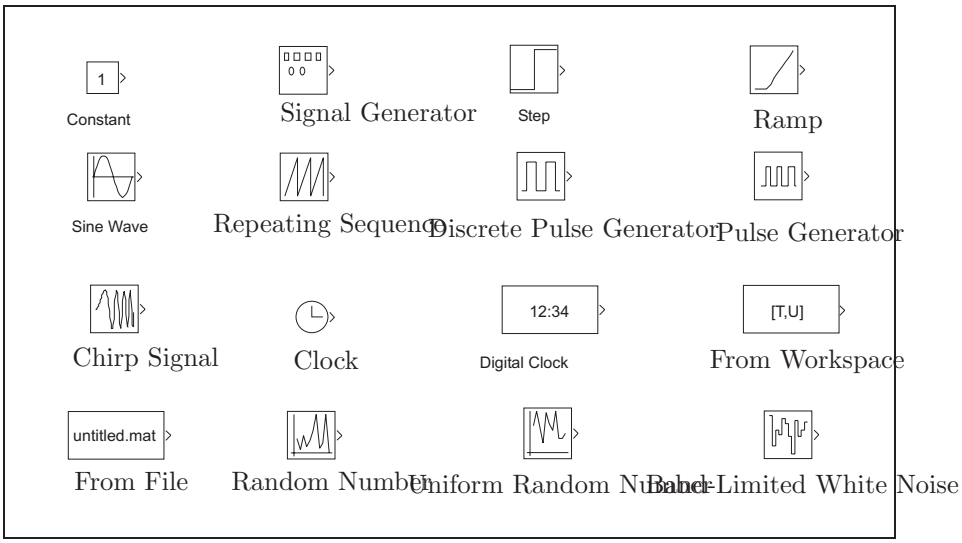



Figura 2.2: *Simulink* Signal Source library.

- **Constant**: genera un valore costante programmabile.
- **Signal Generator**: generatore di segnali sinusoidali, onde quadre,

denti di sega e segnali casuali. Si possono impostare ampiezza e frequenza.

- **Step**: genera un gradino di ampiezza prefissata, specificando il valore iniziale e quello finale.
- **Sine Wave**: genera un'onda sinusoidale di ampiezza, frequenza e fase determinate.
- **Repeating Sequence**: ripete una sequenza di valori e ad istanti predeterminati.
- **Discrete Pulse Generator**: genera impulsi ad intervalli regolari, specificando l'ampiezza, il periodo e ritardo di fase come interi multipli del tempo di campionamento.
- **Pulse Generator**: genera impulsi, specificando il periodo in secondi, il duty cicle (percentuale del periodo), l'ampiezza e l'istante di partenza.
- **Chirp Signal**: genera un segnale sinusoidale con frequenza crescente. Si devono specificare la frequenza iniziale e dopo quanti secondi deve essere raggiunta una certa frequenza predeterminata.
- **Clock**: generatore della base dei tempi.
- **Digital Clock**: genera il tempo di simulazione secondo il tempo di campionamento impostato. Durante il periodo di campionamento vengono mantenuti i valori della simulazione fino al successivo istante di campionamento.
- **From File**: legge il contenuto di una matrice specificata dal `<file>.mat`. La prima riga della matrice deve contenere i valori degli istanti di campionamento e in quelle successive sono memorizzati i corrispondenti valori delle uscite.
- **From Workspace**: legge i valori specificati in una matrice presente nel *WorkSpace* di *Matlab*. La matrice deve contenere nella prima colonna i valori corrispondenti agli istanti di campionamento. Le successive colonne rappresentano i valori delle uscite.
- **Random Number**: genera valori con distribuzione normale gaussiana, dati il valore medio, la varianza e un valore iniziale per il seme.
- **Uniform Random Number**: genera numeri aventi distribuzione uniforme tra due valori prefissati. Si deve specificare anche il seme.
- **Band-Limited White Noise**: genera rumore bianco per sistemi continui. Si specifica la potenza del rumore, istante di campionamento e il seme.

2. **Sinks**: (Library: simulink/Sinks) contiene alcuni rivelatori di segnale, come si può vedere nella Figura 2.3

   - **Scope**: visualizza in funzione del tempo il segnale di ingresso applicato.
   - **XY Graph**: visualizza un grafico $(x, y)$ utilizzando la finestra grafica di *Matlab*. Il primo ingresso corrisponde all'ascissa del grafico e generalmente coincide con la base dei tempi. Si possono introdurre i valori del *range* del grafico.

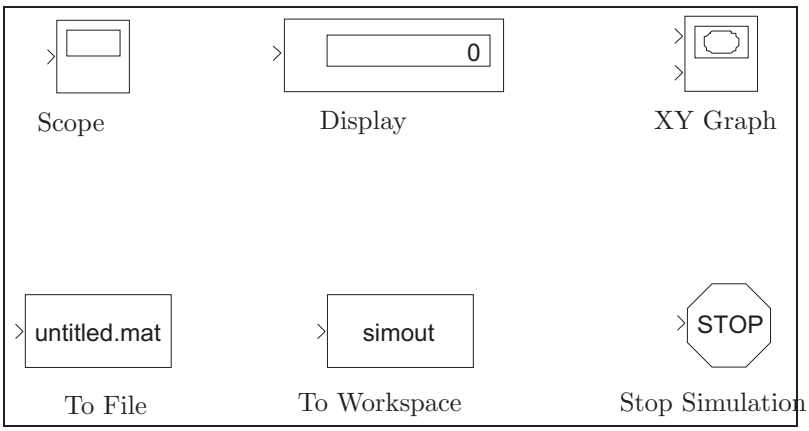



Figura 2.3: *Simulink* Signal Sinks library.

- **Display**: display numerico dei valori dell'ingresso. Si specifica il formato del parametro da visualizzare.
- **To File**: salva gli ingressi applicati all'interno di una matrice in un file `<untitled>.mat`. Si specifica il nome del file e il nome della variabile. I valori vengono salvati per righe. La prima riga della matrice contiene la base dei tempi.
- **To Workspace**: vengono scritti gli ingressi applicato nel *WorkSpace* di *Matlab*. La matrice ha una colonna per ciascun ingresso ed una riga per ogni istante della simulazione. Il dato si perde se la simulazione viene interrotta o messa in pausa. Si specifica il nome della variabile di ingresso e il massimo numero di righe.
- **Stop**: arresta la simulazione quando l'ingresso applicato è diverso da zero.

3. **Discrete**: (Library: simulink/Discrete) sono contenuti i blocchi necessari all'analisi dei sistemi lineari tempo-discreti e vengono raccolti nella Figura 2.4

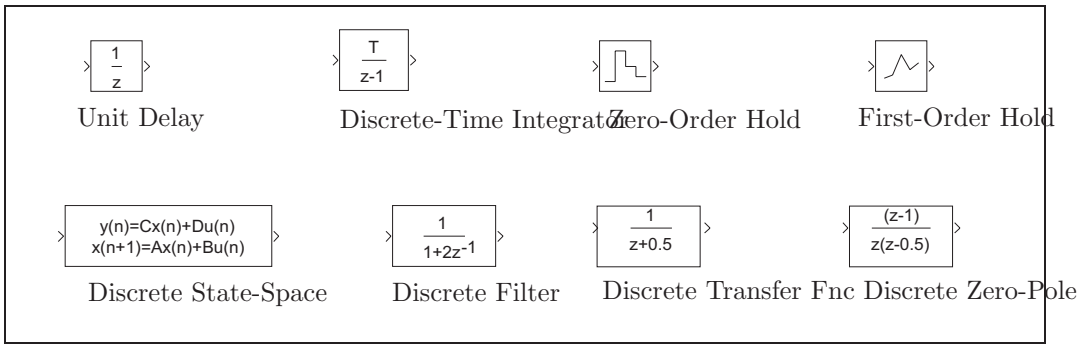



Figura 2.4: *Simulink* Discrete-Time library.

- **Unit Delay**: campiona e mantiene il valore all'ingresso per un periodo di campionamento. Ha come parametri le condizioni iniziali e il tempo di campionamento.
- **Discrete-Time Integrator**: integrazione a tempo-discreto del segnale di ingresso. Utilizza diversi metodi di integrazione, fornite le condizioni iniziali.
- **Zero-Order Hold**: dispositivo di tenuta di ordine zero. Mantiene costante in uscita il valore all'ingresso nell'intervallo di campionamento. Deve essere fornito il tempo di campionamento.
- **First Order Hold**: dispositivo di tenuta di ordine uno. L'uscita cresce linearmente rispetto il valore dell'ingresso nell'intervallo di campionamento. Deve essere fornito il tempo di campionamento.
- **Discrete State-Space**: modello discreto nello spazio degli stati. Vengono fornite le matrici del modello `(A,B,C,D)`, le condizioni iniziali e il tempo di campionamento.
- **Discrete Zero-Pole**: rappresentazione un modello FIR (Finite Impulse Response) o IIR (Infinite Impulse Response) secondo guadagno, poli e zeri, forniti come matrici e vettori. Il numero delle uscite coincide con il numero di colonne della matrice degli zeri. L'uscita è uno scalare se gli zeri sono in un vettore.
- **Discrete Filter**: implementa un filtro tempo-discreto FIR o IIR. Il numeratore e il denominatore sono vettori, i cui elementi sono i coefficienti del polinomio secondo potenze crescenti di $z^{-1}$.
- **Discrete Transfer Fnc**: implementa una funzione di trasferimento discreta. Il numeratore è una matrice, mentre il denominatore un vettore. Il numero delle uscite coincide con il numero delle righe del numeratore, i cui elementi sono i coefficienti del polinomio secondo potenze decrescenti di $z$. Anche il vettore a denominatore contiene i coefficienti del relativo polinomio secondo potenze decrescenti di $z$.

4. **Linear**: (Library: simulink/Linear) contiene i blocchi necessari all'analisi dei sistemi lineari tempo-continui evidenziati nella Figura 2.5

   - **Gain**: guadagno scalare o vettoriale. Si imposta il guadagno $k$ e il blocco calcola l'uscita $y$ dato l'ingresso $u$ secondo l'espressione $y = k. * u$.
   - **Sum**: effettua la somma o la differenza degli ingressi. Si deve inserire la lista dei segni con cui ogni ingresso entra nel blocco.
   - **Integrator**: calcola l'integrazione tempo continua del segnale di ingresso, stabilite le condizioni iniziali ed eventuali limiti superiore ed inferiore di saturazione.
   - **Transfer Fnc**: espressione per la funzione di trasferimento, in cui il numeratore viene rappresentato da una matrice e il denominatore da un vettore. Il numero delle uscite eguaglia il numero dellle righe della matrice al numeratore, i cui elementi sono i coefficienti del polinomio secondo potenze decrescenti di $s$. Anche il vettore al denominatore rappresenta i coefficienti del polinomio secondo potenze decrescenti di $s$.

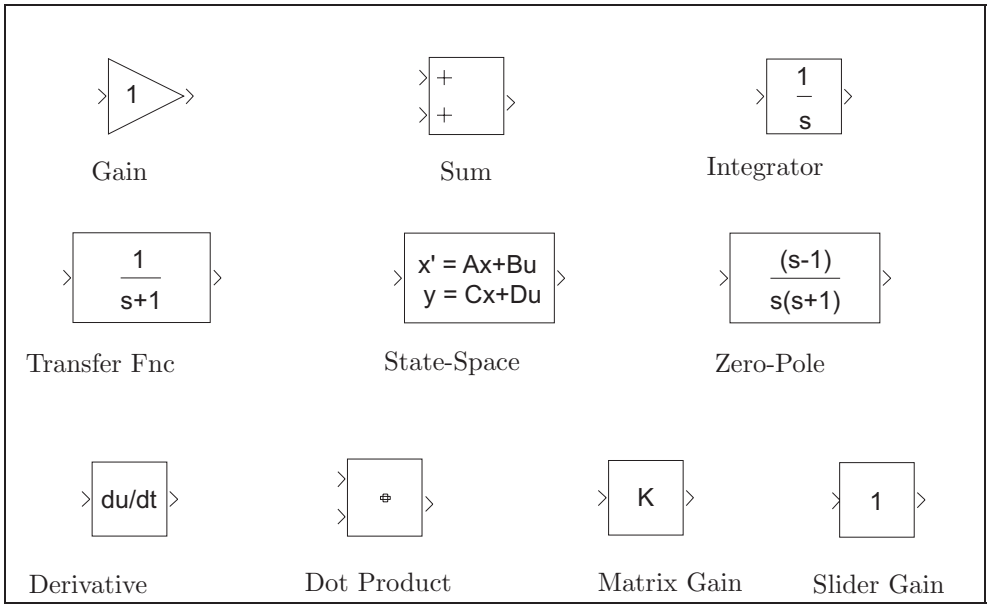



Figura 2.5: *Simulink* Linear library.

- **State-Space**: modello nello spazio degli stati. Occorre inserire le matrici del modello `(A,B,C,D)` e le relative condizioni iniziali.
- **Zero-Pole**: funzione Guadagno, Zeri e Poli. Gli zeri vengono rappresentati da una matrice, mentre i poli da un vettore. Il numero delle uscite coincide con il numero delle colonne della matrice degli zeri.
- **Derivative**: effettua la derivata numerica dell'ingresso.
- **Dot Product**: effettua il prodotto (prodotto scalare) elemento per elemento degli ingressi $u_1$ e $u_2$ secondo l'espressione `y = sum(`$u_1$ `.*` $u_2$`)`.
- **Matrix Gain**: restituisce in uscita l'ingresso moltiplicato per una matrice predefinita.
- **Slider Gain**: guadagno regolabile tra un valore superiore ed uno inferiore.

5. **Nonlinear**: (Library: simulink/Nonlinear) contiene i blocchi che svolgono funzioni non lineari e sono riportati nella Figura 2.6

   - **Abs**: dato l'ingresso $u$, calcola l'uscita $y = |u|$.
   - **Trigonometric Function**: implementa diverse funzioni trigonometriche ed iperboliche: `sin`, `cos`, `tan`, `asin`, `acos`, `atan`, `atan2`, `sinh`, `cosh` e `tanh`.
   - **Math Function**: implementa funzioni matematiche come quelle logaritmiche, esponenziali, potenze e modulo: `exp`, `log`, $10^u$, `log10`, `square`, `sqrt`, `pow`, `reciprocal`, `hypot`, `rem` e `mod`.

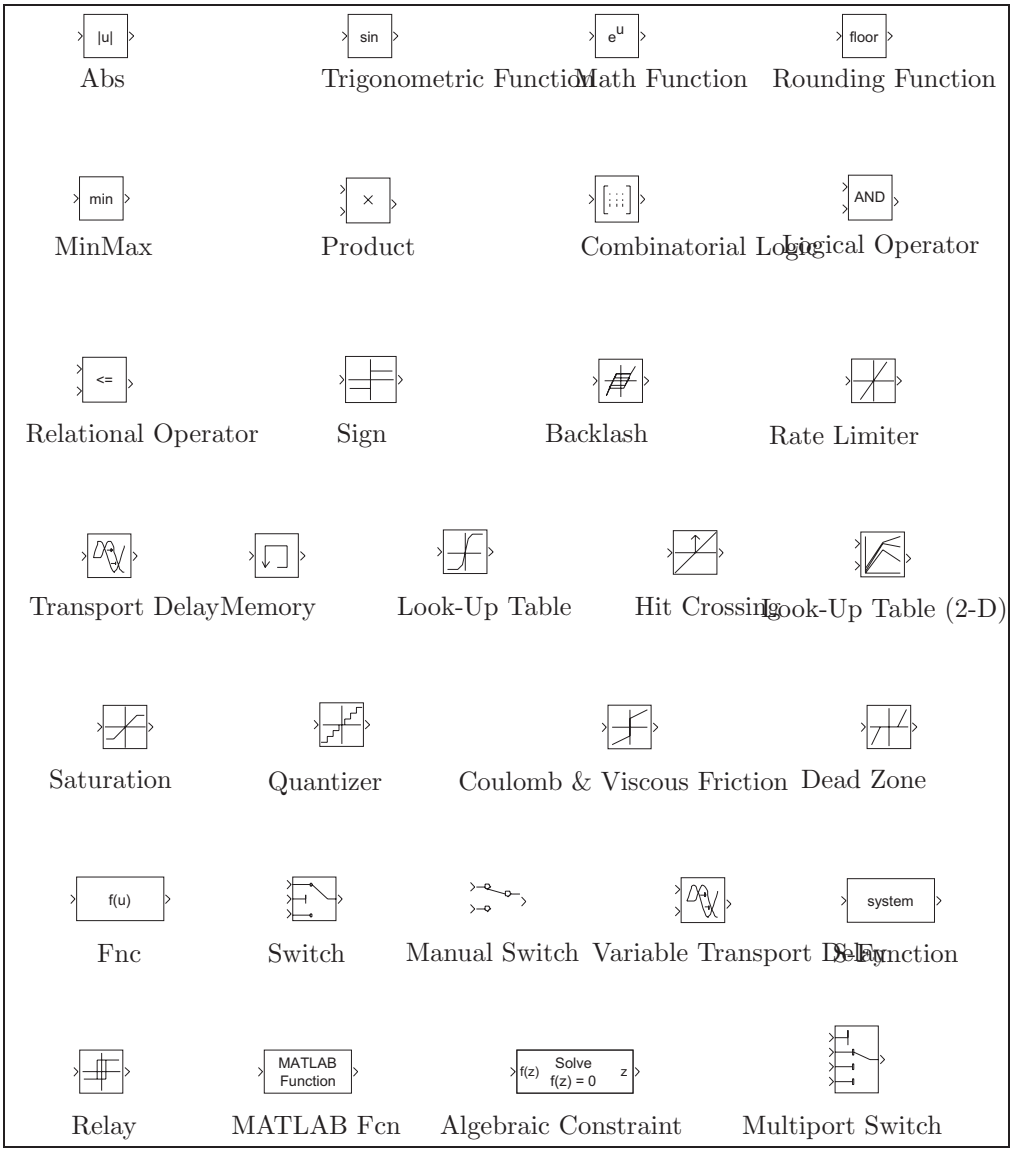



Figura 2.6: *Simulink* Nonlinear library.

- **Rounding Function**: contiene le operazioni di arrotondamento: `floor`, `ceil`, `round` e `fix`.
- **MinMax**: restituisce il minimo od il massimo dell'ingresso. Prevede la scelta del numero degli ingressi e quale operazione deve essere svolta su ogni ingresso.
- **Product**: Moltiplica o divide gli ingressi. Occorre specificare il numero degli ingressi.
- **Combinatorial Logic**: ricerca gli elementi specificati nel vettore

d'ingresso (trattati come valori booleani) nella tabella della verità impostata e restituisce le righe della tabella della verità stessa.

- **Logical Operator**: effettua una operazione logica per un prefissato numero di ingressi: AND, OR, NAND, NOR, XOR, NOT. Per un singolo ingresso, l'operazione viene effettuata tra tutti i valori dell'ingresso memorizzati in un vettore. Per ingressi multipli, l'operazione logica viene eseguita sugli elementi dei diversi vettori di ingresso che occupano la stessa posizione.
- **Relational Operator**: effettua confronti tra gli ingressi: `==`, `=`, `>`, `>=`, `<` e `<=`.
- **Sign**: signum. Restituisce il valore 1 se l'ingresso è positivo, $-1$, per ingresso negativo e 0 per ingresso nullo.
- **Rate limiter**: limita lo slew-rate (velocità di variazione) del segnale di ingresso. Si imposta lo slew-rate positivo e negativo.
- **Saturation**: limita superiormente ed inferiormente il segnale di ingresso secondo due limiti prefissati.
- **Quantizer**: quantizza l'ingresso all'interno di un intervallo prefissato.
- **Coulomb & Viscous Friction**: funzione di attrito viscoso e forza di Coulomb. La forza coulombiana è modellata da una discontinuità nello zero (`y=sign(x)`) mentre l'attrito viscoso è rappresentato da una relazione lineare (`Gain*abs(x)+Offset`), Complessivamente l'uscita risulta `y=sign(x)*(Gain*abs(x)+Offset)`. `Gain` e `Offset` sono parametri del blocco.
- **Backlash**: simula una zona d'isteresi o un certo "gioco"di ampiezza prefissata. Ad esempio, due ruote dentate i cui denti sono abbastanza spaziati.
- **Dead Zone**: l'uscita rimane a zero per valori interni alla "deadzone". Si specifica l'inizio e la fine dell'intervallo.
- **Look-Up Table**: effettua una interpolazione monodimensionale dei valori dell'ingresso usando quelli nella tabella specificata. I valori esterni a quelli della tabella vengono estrapolati.
- **Look-Up Table (2D)**: effettua una interpolazione bidimensionale dei valori dell'ingresso usando quelli nella tabella specificata. I valori esterni a quelli della tabella vengono estrapolati.
- **Memory**: rappresenta un ritardo di durata unitaria. L'uscita coincide con il valore assunto precedentemente dall'ingresso. Occorre specificare le condizioni iniziali.
- **Transport Delay**: ritarda di una quantita specificata il segnale di ingresso. Il ritardo deve essere più grande del passo utilizzato nella simulazione.
- **Variable Transport Delay**: ritarda il primo segnale di ingresso di una quantita specificata dal secondo ingresso. Il ritardo deve essere più grande del passo utilizzato nella simulazione.

- **Hit Crossing**: segnala quando il segnale di ingresso attraversa lo zero secondo un certo margine prefissato. Si può specificare la direzione di attraversamento dello zero.
- **Fnc**: permette di specificare una funzione $f$ arbitraria dell'ingresso $u$, `y = f(u)`.
- **MATLAB function**: passa i valori dell'ingresso ad una funzione *Matlab* affinché possa essere valutata. La funzione `Matlab` deve restituire un vettore la cui lunghezza deve essere definita.
- **S-Function**: blocco che può essere progettato dall'utente in *Matlab*, C, Fortran o usando le funzioni di *Simulink* standard. I parametri `t`, `x`, `u` e `flag` sono passati automaticamente alla funzione di *Simulink*. Possono essere specificati anche altri parametri.
- **Switch**: l'uscita coincide con il primo ingresso quando il secondo ingresso è maggiore od uguale ad una certa soglia, altrimenti assume i valori del terzo ingresso.
- **Manual Switch**: commutatore regolabile col mouse senza parametri.
- **Multiport Switch**: coincide con gli ingressi secondo i valori arrotondati assunti dal primo di questi.
- **Relay**: l'uscita assume due valori impostati se l'ingresso è maggiore dell'estremo superiore o minore dell'estremo inferiore di un certo intervallo specificato attraverso due parametri. Lo stato del Relay non dipende dall'ingresso quando questo assume un valore interno dell'intervallo.
- **Algebraic Contraint**: vincola il segnale d'ingresso `f(z)` a zero e restituisce il corrispondente valore algebrico `z`. Quindi il blocco fornisce il valore `z` tale per cui `f(z) = 0`. L'uscita deve influenzare l'ingresso attraverso una certa retroazione. Occorre fornire un valore di tentativo per `z`.

6. **Connections**: (Library: simulink/Connections) contiene i blocchi necessari ad effettuare connessioni come mostra la Figura 2.7

   - **In**: fornisce una porta d'ingresso per un modello. Occorre specificare il tempo di campionamento.
   - **Out**: fornisce una porta d'uscita per un modello. Quando il modello non è disabilitato, occorre fornire il corrispondente valore dell'uscita.
   - **Mux**: raggruppa scalari o vettori in un vettore di dimensioni maggiori.
   - **Demux**: disaggrega i vettori d'ingresso in scalari o vettori di dimensioni inferiori.
   - **From**: riceve i segnali dal blocco **Goto** secondo l'etichetta (*tag*) specificata.
   - **Goto**: invia i segnali al blocco **From** avente l'etichetta specificata. Permette di definire la visibilità dell'etichetta.
   - **Goto Tag Visibility**: viene usato con i blocchi **From** e **Goto** e permette di specificare la visibilità di una etichetta.

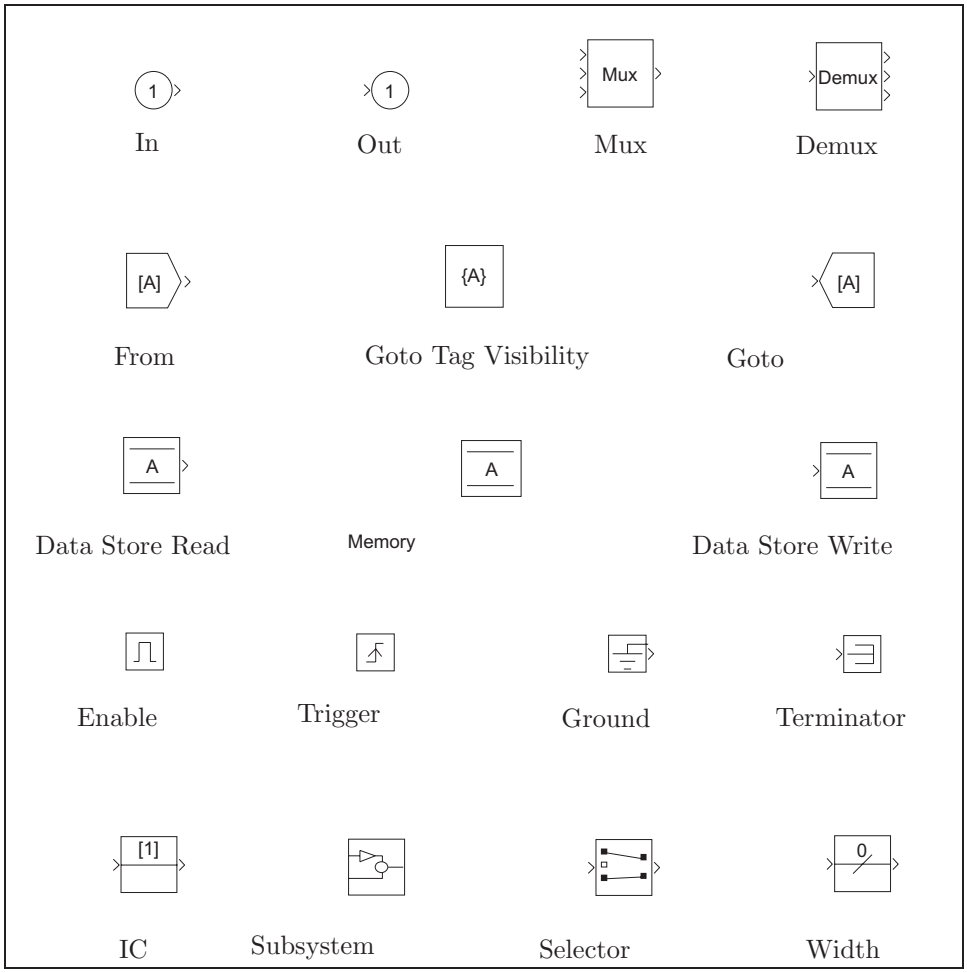



Figura 2.7: *Simulink* Connection library.

- **Data Store Read**: legge i dati memorizzati in una certa regione definita dal blocco **Data Store Memory** secondo un nome prefissato. Occorre definire il nome della zona di memoria e il tempo di campionamento.
- **Data Store Memory**: permette di definire nome e valore iniziale di una regione di memoria utilizzata dai blocchi **Data Store Read** e **Data Store Write**.
- **Data Store Write**: scrive la zona di memoria specificata dal nome. Viene definito anche il tempo di campionamento.
- **Enable**: il blocco viene posto all'interno di un modello affinché sia abilitato.
- **Trigger**: il blocco fornisce una porta di trigger predefinito
- **Ground**: viene utilizzato per mettere a zero i segnali di ingresso.

Si evitano i problemi dovuti agli ingressi non collegati. Fornisce una uscita nulla.

- **Terminator**: usato per isolare un segnale di uscita e per prevenire così i problemi provocati dalle uscite non connesse.
- **IC**: permette di specificare le condizioni iniziali per un segnale.
- **Subsystem**: fornisce una finestra in cui costruire un modello di subsystem.
- **Selector**: seleziona e riordina gli elementi specificati del vettore d'ingresso.
- **Width**: fornisce in uscita l'ampiezza del segnale d'ingresso.

È possibile assegnare ad ogni variabile o intero blocco un nome che verrà evidenziato sia nello schema a blocchi, sia nei grafici che riportano gli andamenti delle variabili. Fino ad ora si sono mantenuti i nomi di default per i blocchi predefiniti da *Simulink*.

Una volta costruito uno schema a blocchi, utilizzando l'apposita finestra fornita da *Simulink* e i blocchi necessari, si passa alla fase di simulazione, ovvero all'integrazione delle equazioni differenziali che descrivono il sistema costruito.

Per utilizzare il simulatore offerto da *Simulink* occorre utilizzare l'apposita finestra richiamabile dal *Simulink Control Panel* alla quale si accede dal menu principale selezionando in sequenza le opzioni *Simulation* e successivamente *Parameters*.

Le informazioni principali del menu *Solver* all'interno della finestra *Simulation Parameters* sono analoghe a quelle relative ai parametri definibili nelle funzioni *Matlab* utilizzate per integrare sistemi di equazioni differenziali ordinarie, introdotte nel Capitolo 3 e in particolare nel Paragrafo 3.3:

1. **Simulation time: Start time**: istante iniziale della simulazione.
2. **Simulation time: Stop time**: istante finale della simulazione.
3. **Solver Options: Type**: permette di definire se si vuole utilizzare un passo di integrazione fisso (Fixed-step) o variabile (Variable-step).
4. **Solver Options**: permette di scegliere la funzione di integrazione ottimale. Sono disponibili `ode45`, `ode23`, `ode113`, `ode15s`, `ode23s` e un metodo per sistemi discreti (discrete).
5. **Solver Options**: Si possono inoltre definire ampiezza massima e iniziale (Max step size e Initial step size) del passo di integrazione, nonché le tolleranze relative ed assolute (Relative e Absolute tolerance) legate all'accuratezza della soluzione.

## 2.2 Analisi di un circuito non lineare.

Si riprenda l'esempio del circuito non lineare utilizzato nel Capitolo 3 Paragrafo 3.2. Utilizzando *Simulink*, il modello non lineare è rappresentato in Figura 2.8

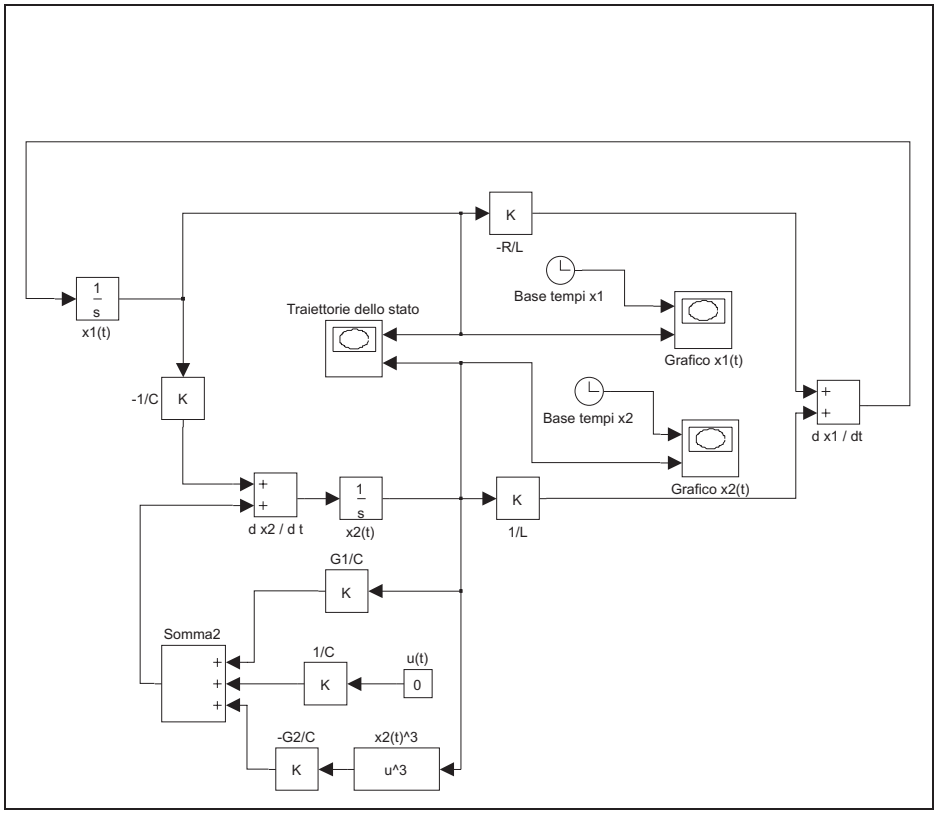



Figura 2.8: Circuito non lineare in *Simulink*.

da cui si ottengono i seguenti risultati delle simulazioni, già presentati nel Capitolo 3, impostando i parametri ai valori $G_1 = 0.8$, $G_2 = 0.05$, $R = 0.5$, $L = 1$ e $C = 1$, con condizioni iniziali $x(0) = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$.

In particolare, le traiettorie degli stati sono presentate in Figura 2.9
e si confronti con la Figura 3.7.

Analogamente, l'andamento delle variabili di stato $x_1(t)$ e $x_2(t)$ nel tempo è graficato nelle Figure 2.10
Si confronti l'andamento con quello analogo mostrato nella Figura 3.8 del Capitolo 3.

Si conclude quindi che si può equivalentemente integrare un modello differenziale costruito in *Matlab*, usando le relative funzioni, oppure progettare e simulare lo stesso modello in ambiente *Simulink*.

## 2.3 Modello di un motore in corrente continua

Si consideri un motore in corrente continua controllato sull'armatura e con l'avvolgimento di eccitazione alimentato a corrente e tensione costante. Sull'asse del motore è presente, oltre al carico inerziale ($J$), una coppia resistente ($f$)

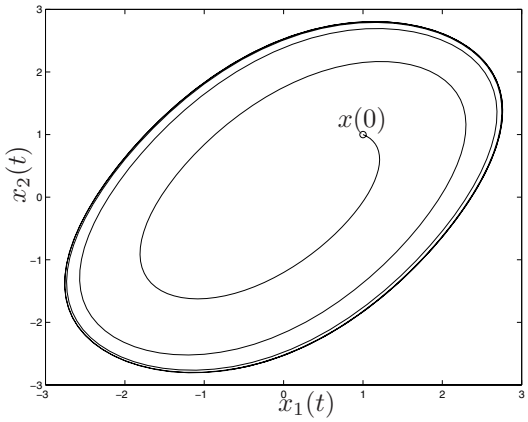



Figura 2.9: Traiettorie dello stato in *Simulink*.

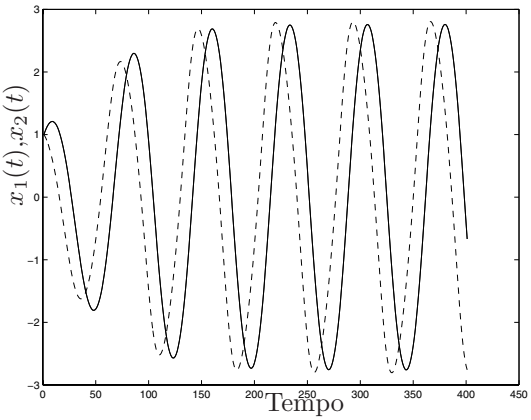



Figura 2.10: Andamento delle variabili di stato $x_1(t)$ e $x_2(t)$ nel tempo calcolata in *Simulink*.

dovuta all'attrito dei cuscinetti ed alle perdite di ventilazione (proporzionali alla velocità di rotazione) ed una coppia di carico $C_c$. Lo schema è riportato in Figura 2.11.

I parametri del motore sono $R_a = 3$ Ohm, $L_a = 30$ mH, $k_m = 2$ N m/A, $J = 3\ kg\ m^2$ e $f = 5 \times 10^{-3}$ N m s / rad. $R_a$ è la resistenza di armatura, $L_a$ la relativa induttanza e $k_m$ una costante che lega la forza controelettromotrice sviluppata dal motore alla velocità angolare $\omega(t)$. Se si assumono come ingressi la tensione di alimentazione di armatura $V_a(t)$ e la coppia assorbita dal carico $C_c(t)$ e come uscite la corrente di armatura $i_a(t)$ e la velocità angolare $\omega(t)$, si

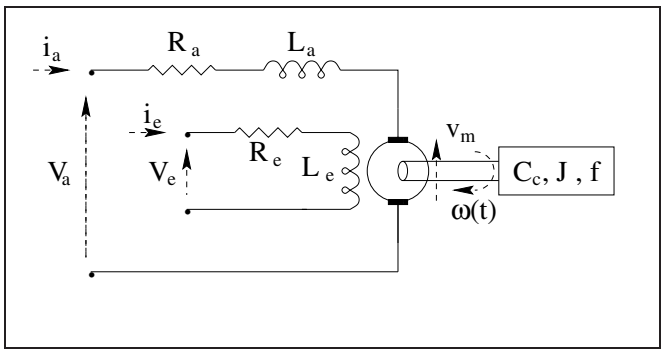



Figura 2.11: Motore in corrente continua.

ottiene il modello nello spazio degli stati $(A,B,C)$

$$\begin{aligned}\begin{bmatrix} \dot{i}_a(t) \\ \dot{\omega}(t) \end{bmatrix} &= \begin{bmatrix} -\frac{R_a}{L_a} & -\frac{k_m}{L_a} \\ \frac{k_m}{J} & -\frac{f}{J} \end{bmatrix} \begin{bmatrix} i_a(t) \\ \omega(t) \end{bmatrix} + \begin{bmatrix} \frac{1}{L_a} & 0 \\ 0 & -\frac{1}{J} \end{bmatrix} \begin{bmatrix} V_a(t) \\ C_c(t) \end{bmatrix} \\ \omega(t) &= \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} i_a(t) \\ \omega(t) \end{bmatrix}\end{aligned} \tag{2.1}$$

Pertanto, le matrici del sistema risultano

$$A = \begin{bmatrix} -\frac{R_a}{L_a} & -\frac{k_m}{L_a} \\ \frac{k_m}{J} & -\frac{f}{J} \end{bmatrix}, B = \begin{bmatrix} \frac{1}{L_a} & 0 \\ 0 & -\frac{1}{J} \end{bmatrix} \text{ e } C = \begin{bmatrix} 0 & 1 \end{bmatrix}.$$

se si pone $y(t) = \omega(t)$ e

$$x(t) = \begin{bmatrix} i_a(t) \\ \omega(t) \end{bmatrix} \text{ e } u(t) = \begin{bmatrix} V_a(t) \\ C_c(t) \end{bmatrix}.$$

Utilizzando lo schema del motore in corrente continua in ambiente *Simulink* e con i parametri forniti precedentemente, si alimenta il motore come in Figura 2.12 con un gradino di tensione di armatura che si mantiene al valore di 0V da 0 a 50s, per poi portarsi al valore di 5V per altri 50s.
La Figura 2.13 riporta il grafico relativo alla velocità angolare $w(t)$ del motore e quella del segnale di ingresso.
Successivamente, si è alimentato il motore con un impulso di tensione di ampiezza $V_a(t) = 10$V e durata $\tau = 40$s. I grafici della posizione del rotore $\alpha(t)$ in radianti e della relativa velocità angolare sono rappresentati nelle Figure 2.14 e 2.15.
Per ottenere l'ulteriore uscita $\alpha(t)$ a partire dal modello del secondo ordine descritto nell'Equazione 2.1 occorre notare che $\dot{\alpha}(t) = \omega(t)$ e, pertanto, ponendo

$$x(t) = \begin{bmatrix} i_a(t) \\ \omega(t) \\ \alpha(t) \end{bmatrix} \text{ e } y(t) = \begin{bmatrix} \omega(t) \\ \alpha(t) \end{bmatrix}$$

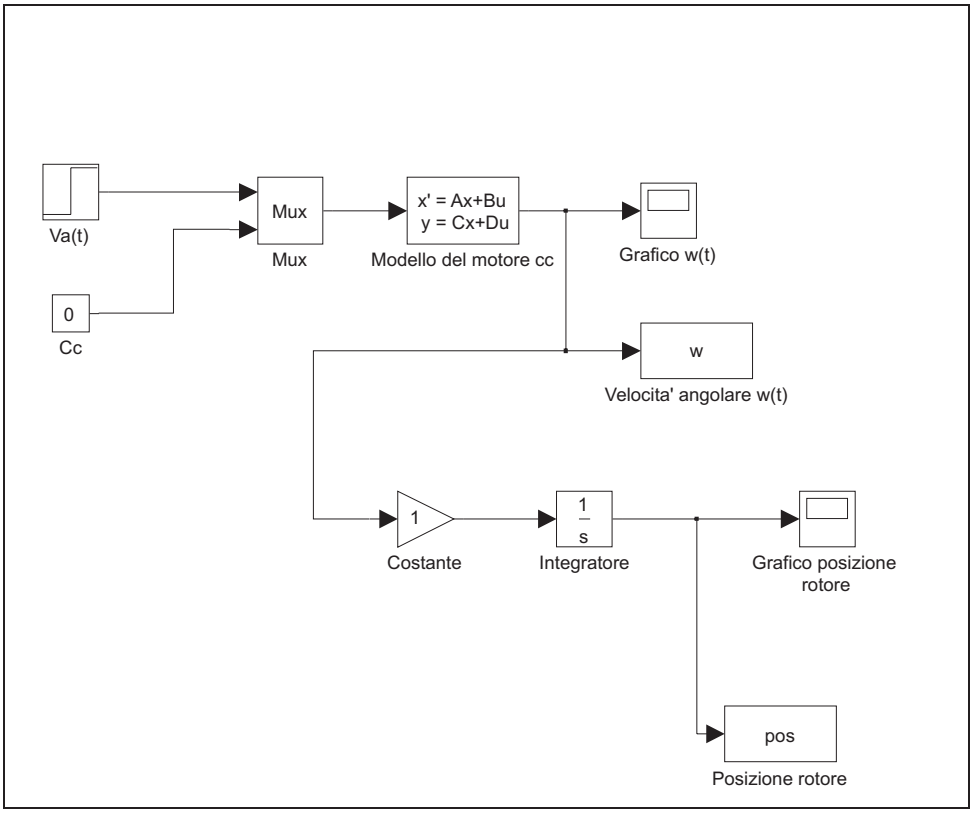
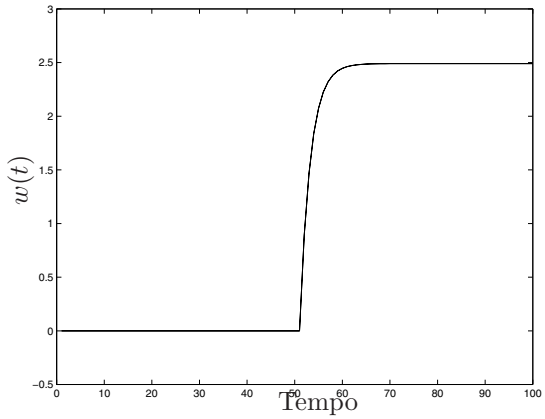


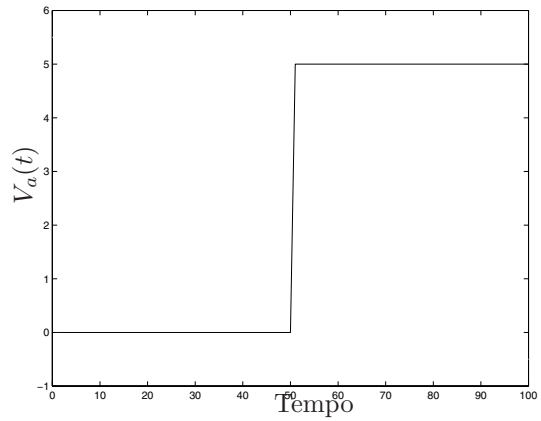

Figura 2.12: Modello *Simulink* del motore in corrente continua.

le matrici del sistema ($A$,$B$,$C$) si modificano in

$$A = \begin{bmatrix} -\frac{R_a}{L_a} & -\frac{k_m}{L_a} & 0 \\ \frac{k_m}{J} & -\frac{f}{J} & 0 \\ 0 & 1 & 0 \end{bmatrix}, B = \begin{bmatrix} \frac{1}{L_a} & 0 \\ 0 & -\frac{1}{J} \\ 0 & 0 \end{bmatrix} \text{ e } C = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (2.2)$$




(a)



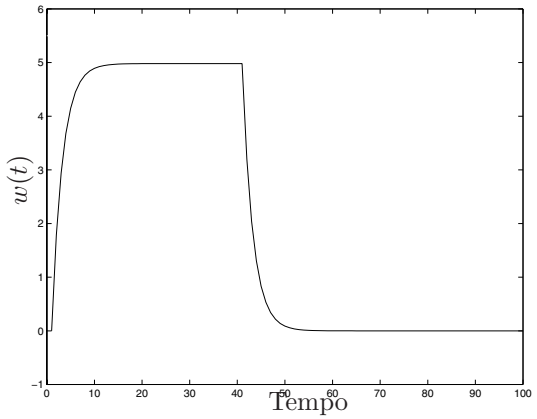

(b)

Figura 2.13: Velocità angolare del motore in cc (a) soggetto ad un gradino di tensione (b).

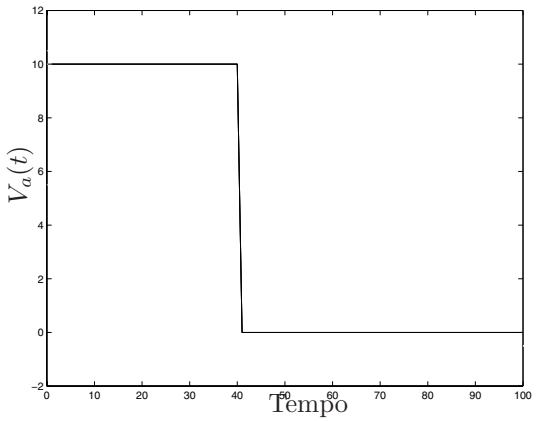
## 2.4 Esercizi proposti in aula didattica.

1. Si realizzi in ambiente *Simulink* il sistema di equazioni differenziali relative ai modello del motore in corrente continua (2.1) e (2.2). Se ne verifichi successivamente la correttezza confrontandolo con le realizzazioni equivalenti nello spazio degli stati (2.1) e (2.2).

2. Utilizzando gli stessi i valori dei parametri del motore in corrente continua, determinare l'ampiezza del gradino $V_a(t)$ necessaria a raggiungere una velocità angolare *di regime* pari a $\omega(t) = 10$rad/s, nelle ipotesi di assenza del carico $C_c = 0$ e con il modello del motore del secondo ordine. Si verifichi analiticamente il risultato ottenuto.




(a)

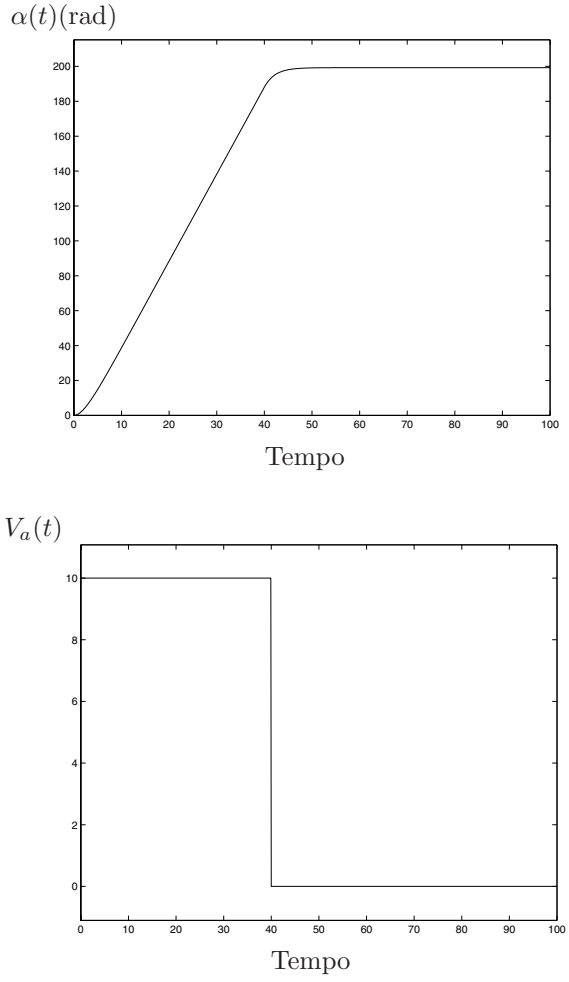



(b)

Figura 2.14: Velocità angolare del motore (a) soggetto ad un gradino di 10V. e durata 5s (b).

3. Fissata l'ampiezza della tensione di armatura a $V_a = 10$V., progettare la durata dell'impulso $\tau$ in modo da raggiungere una posizione assegnata $\alpha = 20$rad., sempre nelle ipotesi di assenza di carico.

4. Fissato $\tau$, graficare l'andamento temporale della posizione del rotore per una tensione pari alla metà e al doppio della tensione fissata al punto precedente.




(b)

Figura 2.15: Posizione in radianti del rotore dell motore (a) soggetto ad un impulso di tensione (b).

# Capitolo 3

# Simulazione di sistemi dinamici

L'analisi dei sistemi non lineari presenta analogie e differenze con quella dei sistemi lineari. Le similitudini derivano dal fatto che una delle tecniche principali di analisi dei sistemi non lineari consiste nella loro approssimazione per mezzo di un sistema lineare e quindi nell'applicazione di metodologie relative a questi ultimi. Le differenze risiedono nel fatto che i sistemi non lineari possono presentare comportamenti completamente nuovi. L'analisi è diversa dal momento che le soluzioni esplicite sono raramente disponibili e quindi devono essere utilizzati metodi particolari per identificare le caratteristiche di comportamento.

## 3.1 Funzioni e modelli usati nel capitolo

In questo capitolo verranno utilizzati i seguenti files *Maltab*:

`IntegraCircuito.m`, integrazione di un modello differenziale.

`LinNonlinCompare.m`, confronto di un sistema col modello linearizzato.

`LinNonlinCompare long.m`, versione estesa di `LinNonlinCompare.m`.

`PassoBase.m`, confronto dei diversi sistemi di integrazione.

`circuito nl impulso.m`, integrazione di circuito non lineare con ingresso non nullo.

`circuito non lineare.m`, integrazione di circuito non lineare e generazione di grafici per diverse condizioni iniziali.

`tunnel1.m`, implementazione del circuito con diodo tunnel.

`tunnel2.m`, implementazione del circuito con diodo tunnel e ingresso non nullo.

## 3.2 Analisi di un circuito non lineare.

Si consideri il circuito rappresentato nella Figura 3.1 nel quale l'elemento non lineare T è caratterizzato da una relazione tensione corrente del tipo $i = -G_1 * v + G_2 * v^3$ essendo $G_1$ e $G_2$ due costanti a valori positivi.

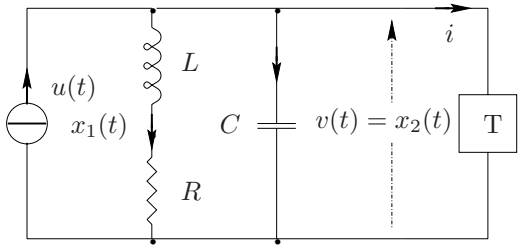



Figura 3.1: Circuito non lineare.

Assumendo come variabile di ingresso la corrente $u(t)$ erogata dal generatore, come uscita la tensione $v(t)$ ai capi dell'elemento non lineare e come variabili di stato la corrente $x_1(t)$ nell'induttore $L$ e la tensione $x_2(t)$ sul condensatore $C$, il modello del sistema nello spazio degli stati è del tipo

$$\begin{aligned} \dot{x}_1(t) &= -\frac{R}{L}\, x_1(t) + \frac{1}{L}\, x_2(t) \\ \dot{x}_2(t) &= -\frac{1}{C}\, x_1(t) + \frac{1}{C}\big(G_1 x_2(t) - G_2 x_2^3(t)\big) + \frac{1}{C} u(t) \end{aligned} \tag{3.1}$$

Con ingresso nullo, $u(t) = 0$, gli stati di equilibrio del sistema si ottengono come soluzioni delle equazioni $x_2 = R\ x_1$ e $x_1 = G_1\ x_2 - G_2\ x_2^3$, ovvero

$$\bar{x}' = \begin{bmatrix} 0 \\ 0 \end{bmatrix},\ \bar{x}'' = \begin{bmatrix} \sqrt{\frac{G_1 R - 1}{G_2 R^3}} \\ \sqrt{\frac{G_1 R - 1}{G_2 R}} \end{bmatrix} \text{ e } \bar{x}''' = \begin{bmatrix} -\sqrt{\frac{G_1 R - 1}{G_2 R^3}} \\ -\sqrt{\frac{G_1 R - 1}{G_2 R}} \end{bmatrix}. \tag{3.2}$$

Si noti nelle ipotesi che $G_1 * R - 1 < 0$ soltanto l'origine dello spazio degli stati è punto di equilibrio del circuito.

Il circuito è stato simulato con i seguenti valori dei parametri

$$G_1 = 0.8\ ,\ G_2 = 0.05\ ,\ R = 2\ ,\ L = 1 \text{ e } C = 1$$

in assenza di ingresso e a partire da diverse condizioni iniziali. Con le seguenti funzioni, `tunnel1.m`

```
function xd = tunnel1(t,x,flag,param)

% Funzione che implementa il circuito Di Eq.(2.1) con u(t) = 0.
%
% d x1(t) / dt = - R/L x1(t) + 1/L x2(t)
```

```
% d x2(t) / dt = - 1/C x1(t) + 1/C ( G1 x2(t) - G2 x2(t)^3 ) + 1/C u(t)
%

R = param(1);
L = param(2);
C = param(3) ;
G1 = param(4);
G2 = param(5);

x1d = - (R/L) * x(1) + (1.0/L) * x(2);
x2d = - (1.0/C) * x(1) + (1.0/C)*( G1 * x(2) - G2 * x(2)^3 );

xd = [x1d; x2d];

return
```

per la realizzazione del sistema di equazioni differenziali e `IntegraCircuito.m`

```
% Script-file che integra il sistema differenziale (2.1)
% e grafica i risultati.

options = odeset('RelTol',1e-6); % Opzioni per la funzione di integrazione

R = 2; % Parametri fisici del modello
L = 1;
C = 1;
G1 = 0.8;
G2 = 0.05;

param = [R,L,C,G1,G2]; % Parametri fisici del modello

ci = [2 2]; % Condizioni iniziali per l'integrazione

time = [0 40]; % Tempo di integrazione

[t,x] = ode45('tunnel1',time,ci,options,param); % Funzione che effettua l'integrazione

%% Grafico delle traiettorie dello stato

figure
plot(x(:,1),x(:,2),'-') % Disegna i vettori passati come argomenti
title('Traiettorie dello stato') % Titolo del grafico
xlabel('x1') % Etichetta dell'asse delle ascisse
ylabel('x2') % Etichetta dell'asse delle ordinate


%% Grafico del moto dello stato

figure
plot(t,x(:,1),'-',t,x(:,2),'--')
```

```
title('Andamento nel tempo')
xlabel('Tempo')
ylabel('x1(t) e x2(t)')
```

per l'integrazione del sistema e la visualizzazione dei risultati, si ottengono i seguenti grafici rappresentati nelle Figure 3.2 e 3.3. La Figura 3.3 è stata ottenuta partendo dallo stato iniziale $x_1(0) = 2$ e $x_2(0) = 2$.

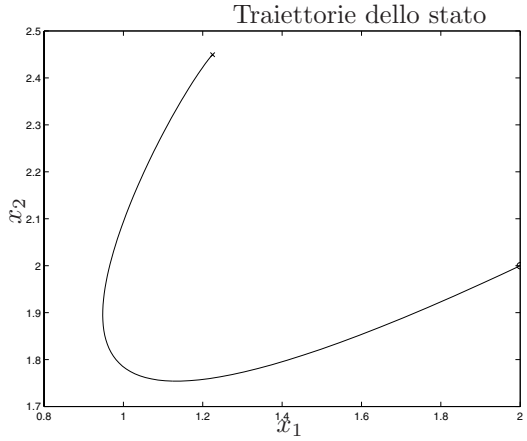



Figura 3.2: Traiettorie dello stato.

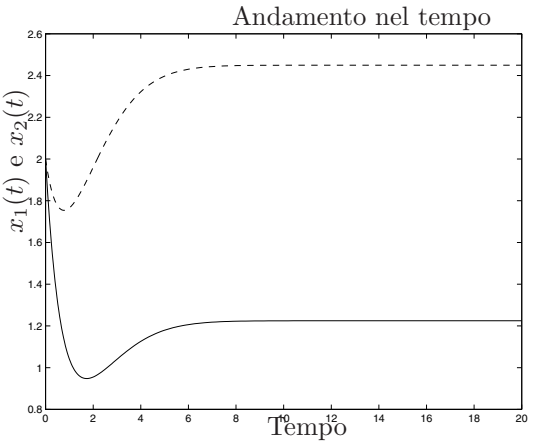



Figura 3.3: Andamento nel tempo delle variabili di stato.

L'istruzione di *Matlab* `ode45` consente di integrare sistemi di equazioni e verrà descritta nel paragrafo 3.3.

Il comando `plot(x,y,'z')` di *Matlab*, con `x,y` vettori riga o colonna costituiti da elementi reali, produce un grafico con le ascisse costituite dagli elementi del vettore `x` e con le ordinate costituite dagli elementi del vettore `y`. L'ulteriore argomento `'z'` impone l'impiego di un determinato stile di linea per la visualizzazione del grafico. Le istruzioni `title('text')`, `xlabel('text')` e

`ylabel('text')` inseriscono la stringa 'text', rispettivamente, come titolo del grafico, come etichetta dell'asse delle ordinate e delle ascisse.

La figura 3.4 rappresenta le traiettorie percorse dalle variabili di stato del sistema, nel caso in cui $G_1R - 1 > 0$, per diversi valori dello stato iniziale. La retta tratteggiata ha equazione $x_2 = R\ x_1$ mentre la curva tratteggiata ha equazione $x_1 = G_1\ x_2 - G_2\ x_2^3$. Appare evidente il comportamento stabile del sistema nell'intorno dei due punti di equilibrio diversi da zero.

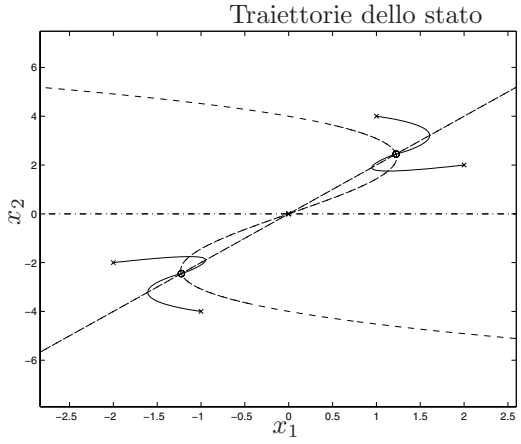



Figura 3.4: Traiettorie dello stato.

Nella situazione per cui $G_1 * R - 1 < 0$, ad esempio ponendo $G_1 = 0.8$ ed $R = 1$, l'origine è l'unico punto di equilibrio per il sistema. La Figura 3.5 mostra l'andamento nel tempo delle traiettorie a partire da diversi stati iniziali, mentre in Figura 3.6 è riportato l'andamento smorzato delle due variabili di stato a partire dalla condizione $x_1(0) = 3$ e $x_2(0) = -3$.

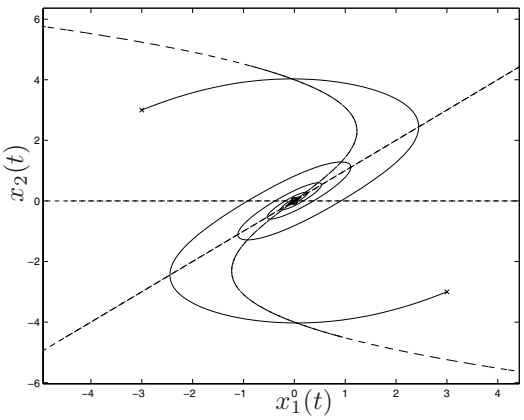



Figura 3.5: Traiettorie dello stato.

Sempre nelle condizioni in cui, $G_1 * R - 1 < 0$, ad esempio ponendo $G_1 = 0.8$ ed $R = 0.5$, l'origine rimane l'unico punto di equilibrio per il sistema. La Figura 3.7

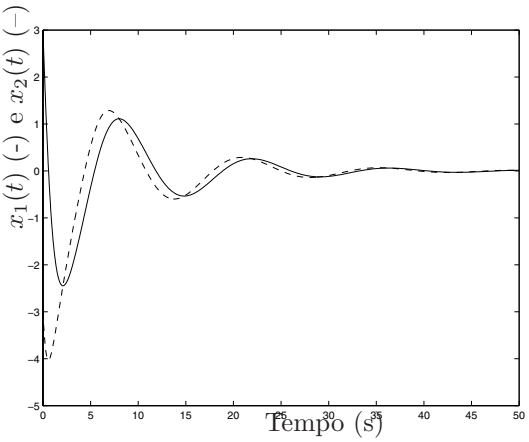



Figura 3.6: Andamento delle variabili di stato.

mostra l'andamento nel tempo delle traiettorie a partire da diversi stati iniziali, mentre in Figura 3.8, è riportato l'andamento oscillatorio delle due variabili di stato a partire dalla condizione $x_1(0) = 1$ e $x_2(0) = 1$.

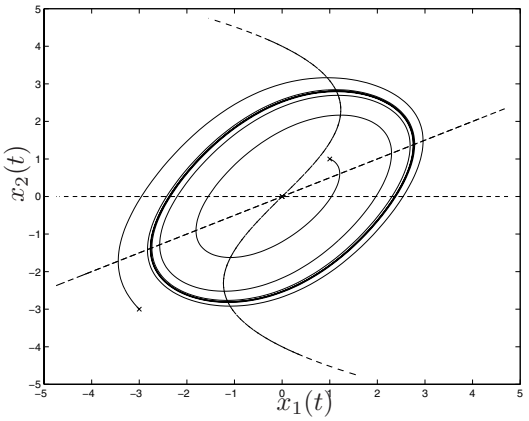



Figura 3.7: Traiettorie dello stato con un punto di equilibrio.

Volendo analizzare la risposta del circuito ad un impulso di corrente di ampiezza pari a $u(t) = 8$ e di durata di $10s$. ($u(t) = 0$ per $t > 10$), la funzione `tunnel1.m` viene modificata nel modo seguente (`tunnel2.m`)

```
function xd = tunnel2(t,x,flag,param)
% Funzione che implementa il circuito di Eq. (2.1). E' presente
% anche una funzione dell'ingresso udt funzione del tempo.
%
% d x1(t) / dt = - R/L x1(t) + 1/L x2(t)
% d x2(t) / dt = - 1/C x1(t) + 1/C ( G1 x2(t) - G2 x2(t)^3 ) + 1/C u(t)
%
```

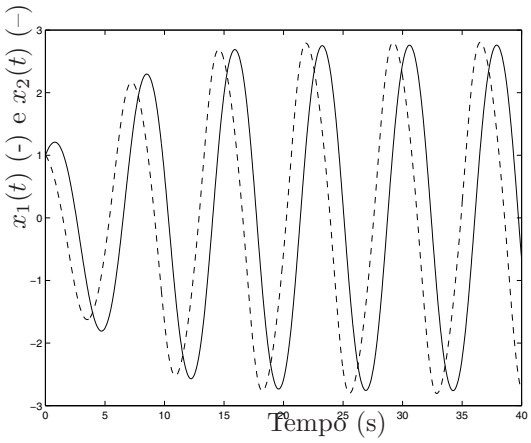



Figura 3.8: Andamento oscillatorio delle variabili di stato.

```
R = param(1); % Parametri del circuito non lineare
L = param(2);
C = param(3);
G1 = param(4);
G2 = param(5);
Tstart = param(6); % Istante di inizio del gradino
Tstop = param(7); % Istante finale del gradino
Value = param(8); % Ampiezza del gradino

if((t>=Tstart)&(t<Tstop)),udt=Value; % Definizione del gradino
 else udt=0.0;
end;

x1d = - (R/L) * x(1) + (1.0/L) * x(2);
x2d = - (1.0/C) * x(1) + (1.0/C)*( G1 * x(2) - G2 * x(2)^3 );

xd = [x1d; x2d + (1.0/C)*udt];

return
```

e il programma di simulazione fornisce a partire dallo stato zero i seguenti risultati riportati nelle Figure 3.9 e 3.10.

Si voglia ora linearizzare il circuito caratterizzato dagli stessi parametri elettrici dell'esempio precedente, con ingresso nullo e nell'intorno dell'origine dello spazio degli stati. Il modello linearizzato assume la struttura

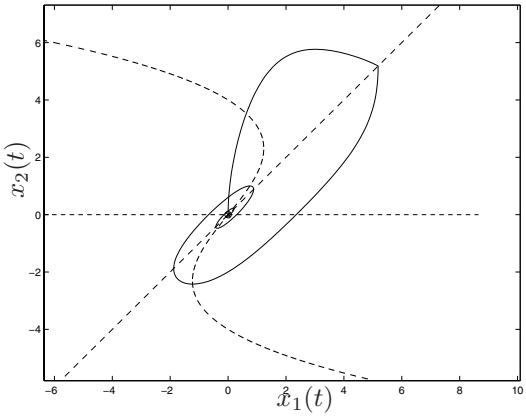



Figura 3.9: Traiettorie dello stato con un punto di equilibrio e impulso di corrente.

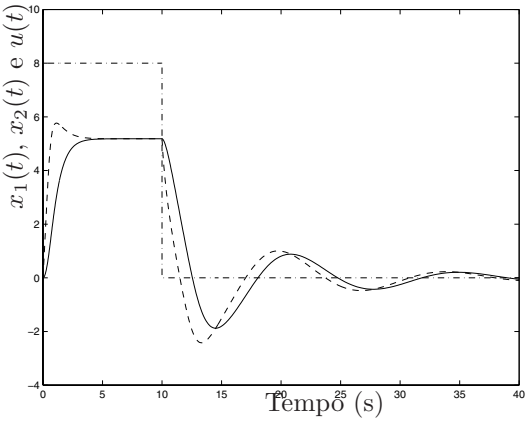



Figura 3.10: Andamento smorzato delle variabili di stato con impulso di corrente.

$$\begin{aligned} \delta\dot{x}(t) &= A\delta x(t) + B\delta u(t) \\ \delta y(t) &= C\delta x(t) \end{aligned} \tag{3.3}$$

avendo posto

$$\begin{aligned} \dot{x}_1(t) &= f_1(x(t), u(t)) \\ \dot{x}_2(t) &= f_2(x(t), u(t)) \\ y(t) &= g(t) \end{aligned}$$

dove $\delta x(t)$, $\delta u(t)$ e $\delta y(t)$ rappresentano, rispettivamente, gli scostamenti dello

stato, dell'ingresso e dell'uscita dai valori di equilibrio e le matrici

$$A = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} \end{bmatrix}, B = \begin{bmatrix} \frac{\partial f_1}{\partial u} \\ \frac{\partial f_2}{\partial u} \end{bmatrix} \text{ e } C = \begin{bmatrix} \frac{\partial g}{\partial x_1} & \frac{\partial g}{\partial x_2} \end{bmatrix},$$

vanno calcolate in corrispondenza del punto di equilibrio scelto.

Nel caso in esame risulta

$$A = \begin{bmatrix} -\frac{R}{L} & \frac{1}{L} \\ -\frac{1}{C} & \frac{G_1}{C} \end{bmatrix}, B = \begin{bmatrix} 0 \\ \frac{1}{C} \end{bmatrix} \text{ e } C = \begin{bmatrix} 0 & 1 \end{bmatrix}$$

Il programma seguente `LinNonlinCompare.m` mette a confronto la simulazione del circuito, in assenza di ingresso e a partire da assegnate condizioni iniziali, utilizzando il modello non lineare e quello linearizzato.

```
%%
%% Script-file per il confronto della risposta del sistema lineare
%% e quella del sistema non lineare.
%%

%%
%% Modello non lineare
%%


options = odeset('RelTol',1e-6);

R = 1; % Parametri del circuito non lineare.
L = 1;
C = 1;
G1 = 0.8;
G2 = 0.05;

param = [R,L,C,G1,G2]; % Parametri del circuito non lineare.

ci = [0.5 0.5]; % Condizioni iniziali

ti = 0;
tf = 40;

time = [ti tf]; % Istante iniziale e finale di integrazione

[t,x] = ode45('tunnel1',time,ci,options,param); % Integrazione del sistema

y = x(:,2); % Uscita del sistema


%%
%% Modello lineare nello spazio degli stati
%%
```

```
A = [-R/L   1.0/L;
     -1.0/C G1/C ];

B = [1.0/C 0]';

C = [0 1];

D = 0;

Sys = ss(A,B,C,D); % Crea il modello nello spazio degli stati

tl = ti:0.01:tf;   % Tempo di simulazione
Ul = zeros(size(tl)); % Ingresso del sistema lineare
[yl,tl,xl] = lsim(Sys,Ul,tl,ci);
                      % Simula la risposta nel tempo
                      % per un sistema lineare tempo-invariante

figure
plot(tl,yl,'-'), hold on
plot(t,y,'--')
title('Risposte')
xlabel('Tempo')
ylabel('y')


figure
plot(x(:,1),x(:,2),'-'), hold on
plot(xl(:,1),xl(:,2),':'), hold on
title('Traiettorie dello stato')
xlabel('x1')
ylabel('x2')
```

La Figura 3.11 riporta l'andamento temporale dell'uscita, calcolata sia con il modello lineare che con il modello linearizzato a partire dallo stato iniziale $x(0) = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$, vicino all'origine. La linea continua rappresenta la risposta del sistema lineare, mentre la curva tratteggiata, quella del sistema non lineare.
La Figura 3.12 riporta lo stesso grafico a partire da un nuovo stato iniziale ($x(0) = \begin{bmatrix} -2 \\ 2 \end{bmatrix}$), lontano dall'origine.
Le Figure 3.13 e 3.14 riportano le traiettorie dello stato calcolate nelle stesse condizioni.
La linea continua rappresenta la traiettoria dello stato del sistema lineare, mentre la curva punteggiata, quella del sistema non lineare.

Si noti che mentre ci si allontana al punto rispetto al quale si è effettuata la linearizzazione, i modelli forniscono risposte significativamente diverse.

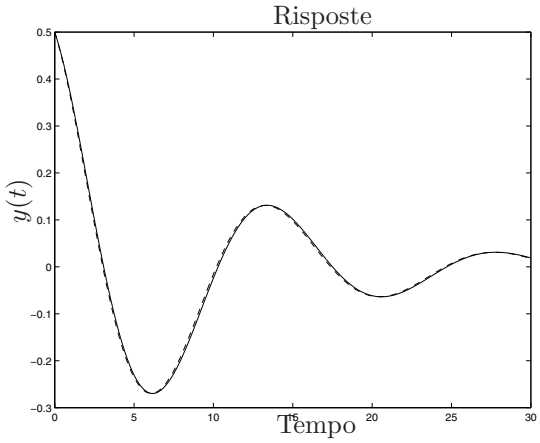



Figura 3.11: Risposta del sistema lineare e del sistema non lineare.

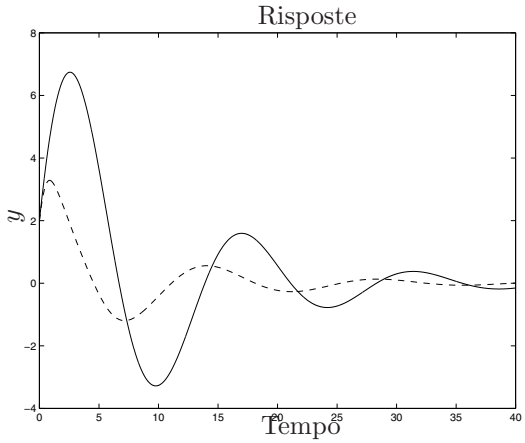



Figura 3.12: Risposta dei sistemi per una condizione iniziale distante dall'origine.

## 3.3 Metodi numerici per integrazione di equazioni differenziali in Matlab.

La simulazione di sistemi in *Matlab* generalmente richiede l'integrazione di sistemi di equazioni differenziali ordinarie. *Matlab* fornisce un insieme di funzioni per risolvere numericamente queste funzioni differenziali. I risultati in termini di velocità ed accuratezza dipendono dal tipo di modelli e di condizioni. Sono riportati nel seguito alcune caratteristiche dei metodi di integrazione.

1. `ode15s`: risolve equazioni differenziali ordinarie per sistemi stiff autonomi e non (generalmente sistemi non lineari e "smooth") implementando un metodo di integrazione numerica di ordine variabile fino al $5^o$ con un passo di integrazione quasi fisso. Riesce a risolvere efficientemente anche problemi relativi a sistemi non stiff.

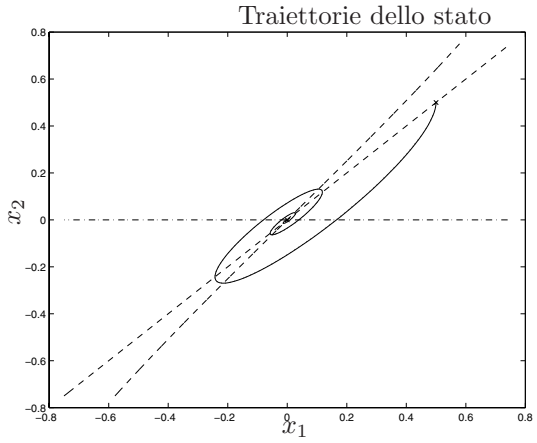



Figura 3.13: Traiettorie dello stato per condizioni iniziali vicino all'origine.

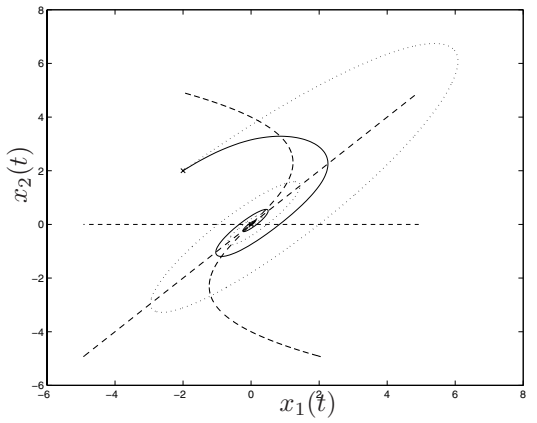



Figura 3.14: Traiettorie dello stato per condizioni iniziali lontane all'origine.

2. `ode23s`: risolve equazioni differenziali per sistemi stiff utilizzando metodi di ordine non elevato.

3. `ode23`: questo programma è un'alternativa a `ode15s` per la soluzione di problemi stiff ed implementa un metodo di Runge-Kutta del $2^o$ ordine fisso. Permette di ottenere soluzioni con tolleranze non troppo spinte ed è indicato per sistemi con autovalori vicini all'asse immaginario, quindi sistemi con memoria a dinamica lenta. Riesce comunque a gestire sistemi con dinamiche veloci, anche non autonomi.

4. `ode45`: come `ode23` implementa il metodo di Runge-Kutta ed è quindi adatta per sistemi fortemente non lineari. Questa funzione non è consigliabile per sistemi che manifestino dinamiche lente e veloci contemporaneamente.

5. `ode113`: è stata introdotta per superare i problemi che presentano le fun-

zioni `ode23` e `ode45`. Permette di ottenere una accuratezza sia moderata che elevata pur mantenendo una complessità accettabile. Utilizzando formule risolutive di ordine elevato fornisce soluzioni con una risoluzione adeguata anche per applicazioni grafiche.

Le funzioni per l'integrazione delle equazioni differenziali ordinarie richiedono tutte la stessa sintassi. Nell'esempio seguente verrà perció utilizzata la generica funzione `ode45`. Nel paragrafo 3.2, si è utilizzato il comando

```
[t,x] = ode45('tunnel1',[ti tf],x0,options);
```

in cui '`tunnel1`' è l'equazione differenziale da integrare, espressa attraverso un *function-file* di *Matlab*, `[ti tf]` rappresenta l'intervallo di integrazione, a partire dalle condizioni iniziali `x0`. La funzione prevede la possibilità di utilizzare un insieme di opzioni.

```
options = odeset('RelTol',1e-4,'AbsTol',[1e-4 1e-4],'Maxstep',10);
```

Si guardi in proposito l'help in linea per la funzione '`odeset`'. Il vettore '`options`' viene costruito attraverso la suddetta funzione che accetta coppie valore e funzione chiave (e.g. '`RelTol`' e `1e-4`).

Le opzioni più utilizzate sono appunto '`RelTol`' e '`AbsTol`', le tolleranze relativa ed assoluta associate all'errore per il controllo della convergenza. Per la funzione `ode45`, l'errore $\varepsilon_i$ sull'$i$-esima componente relativa alla stima della soluzione $y_i$ dell'equazione differenziale, soddisfa la disequazione

$$|\varepsilon_i| \leq r|y_i| + a_i \tag{3.4}$$

in cui $r =$ `RelTol` e $a_i =$ `AbsTol(i)`. L'errore relativo scalare ha il valore di default di $10^{-3}$, mentre quello assoluto di $10^{-6}$. Un ultimo parametro che si può utilizzare è la massima ampiezza del passo di integrazione, per garantire che il metodo di integrazione utilizzato riconosca fenomeni che avvengono in quell'intervallo di tempo prefissato. Viene indicato con '`MaxStep`' e il valore di default coincide con 1/10 dell'intervallo di integrazione.

## 3.4 Problematiche relative all'integrazione di sistemi dinamici.

Si riprenda l'esempio precedente relativo all'integrazione del sistema implementato attraverso la funzione `tunnel1.m` e si utilizzi `ode45` con tolleranze e passi di integrazione diversi, come appare nel listato del programma `PassoBase.m`

```
%%
%% Script-file per il confronto delle risposta del sistema non lineare
%% utilizzando diversi metodi di integrazione.
%%

R = 1; % Parametri del circuito non lineare.
L = 1;
C = 1;
```

```
G1 = 0.8;
G2 = 0.05;

param = [R,L,C,G1,G2];

ci = [0.5 0.5];

ti = 0;
tf = 40;

time = [ti tf];

options = odeset('RelTol',1.0,'AbsTol',[1e-1 1e-1],'MaxStep',1000);
[t1,x1] = ode45('tunnel1',time,ci,options,param);
y1 = x1(:,2);

options = odeset('RelTol',1e-1,'AbsTol',[1e-2 1e-2],'MaxStep',100);
[t2,x2] = ode45('tunnel1',time,ci,options,param);
y2 = x2(:,2);

options = odeset('RelTol',1e-3,'AbsTol',[1e-6 1e-6],'MaxStep',10);
[t3,x3] = ode45('tunnel1',time,ci,options,param);
y3 = x3(:,2);


figure
plot(t1,y1,'-'), hold on
plot(t2,y2,'--'), hold on
plot(t3,y3,'-.')
title('Risposte')
xlabel('Tempo')
ylabel('y')


return
```

I risultati di tale simulazione sono riportati nella Figura 3.15. Si noti che i risultati ottenuti sono diversi per i diversi valori assegnati alle tolleranze e ai passi di integrazione. Più piccola è la tolleranza, più passi di integrazione sono richiesti dal metodo utilizzato, che porta a risultati più accurati.

Analogamente, risultati diversi si ottengono integrando le stesse equazioni differenziali con tecniche differenti. Per una descrizione più dettagliata di queste tecniche e per un utilizzo più appropriato si rimanda all'help in linea di *Matlab* e alla consultazione dell'articolo [6].

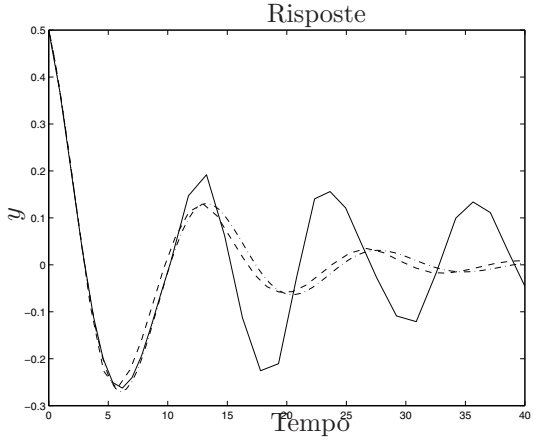



Figura 3.15: Risposta del sistema con diversi passi di integrazione.

## 3.5 Esercizi proposti in aula didattica.

1. Si consideri il modello matematico di Lotka-Volterra (3.5) che descrive la dinamica di due popolazioni interagenti

$$\begin{aligned} \dot{x}_1(t) &= a_1(1 - x_1(t)/k)x_1(t) - a_2 x_1(t)x_2(t) + u(t)(\ prede) \\ \dot{x}_2(t) &= -a_3 x_2(t) + a_4 x_1(t)x_2(t)(\ predatori) \end{aligned} \tag{3.5}$$

dove $x_1(t)$ e $x_2(t)$ rappresentano rispettivamente il numero di prede e il numero di predatori presenti nell'ecosistema all'istante $t$ ed $u(t)$ l'apporto esterno di cibo per le prede introdotto nell'unità di tempo. Il coefficiente $k$ rappresenta il numero massimo di prede presenti nell'ecosistema in assenza di predatori e in assenza di apporto di cibo esterno ($u(t) = 0$). Il parametro $a_3$ ($> 0$) è il tasso di crescita del predatore, in assenza di prede, mentre $a_1$ ($> 0$) il tasso di crescita delle prede. Il termine $-a_2 x_1(t)x_2(t)$ modella il decremento nella popolazione delle prede per la presenza dei predatori, con $a_2 > 0$, mentre il termine $a_4 x_1(t)x_2(t)$ rappresenta l'incremento della popolazione dei predatori dovuto alla presenza delle prede.

Nelle ipotesi di assegnare ai parametri i valori $a_1 = 20$, $a_2 = 1$, $a_3 = 7$, $a_4 = 0.5$ e $k = 30$, si determinino

   (a) l'andamento nel tempo del numero di prede e predatori, supponendo nullo l'ingresso u(t) e nelle ipotesi di partire da un ecositema contenente 10 prede e 10 predatori. Si calcoli anche la traiettoria percorsa dal sistema nello spazio degli stati.

   (b) gli stati di equilibrio del sistema in assenza di ingresso.

   (c) i valori di regime raggiunti dal numero di prede e predatori nelle ipotesi che $u(t)$ sia un gradino di ampiezza $u(t) = 20$ e a partire dalle stesse condizioni proposte al punto 1). Si determini per tentativi l'ampiezza del gradino che consente di mantenere a regime un numero di predatori pari a 15.

2. (Facoltativo) Si definisce *modello ibrido* un sistema composto da diversi modelli, ciascuno valido in una particolare condizione di funzionamento del sistema stesso. Tale condizione dipende dallo stato del sistema e dai suoi ingressi.

   Si consideri quindi il sistema ibrido descritto dal seguente modello matematico:

$$\dot{\mathbf{x}}(t) = \begin{cases} \mathrm{A}_1\mathbf{x}(t) & \text{se } x_1(t) * x_2(t) < 0 \\ \\ \mathrm{A}_2\mathbf{x}(t) & \text{se } x_1(t) * x_2(t) \geq 0 \end{cases},$$

   in cui

$$\mathbf{x}(t) = \begin{bmatrix} x_1(t) \\ x_2(t) \end{bmatrix}$$

   e

$$\mathrm{A}_1 = \begin{bmatrix} -0.1 & 1.0 \\ -10.0 & -0.1 \end{bmatrix}, \mathrm{A}_2 = \begin{bmatrix} -0.1 & 10 \\ -1.0 & -0.1 \end{bmatrix}.$$

   (a) Si discuta la stabilità dei sistemi singoli.

   (b) Si disegnino le traiettorie dello stato per i sistemi singoli e per il sistema completo considerando un tempo di simulazione di 10s e le condizioni iniziali $(0, 1)$, $(1, 0)$ e $(10^{-6}, 10^{-6})$.

   (c) Si disegni l'andamento nel tempo delle variabili di stato dei sistemi singoli e di quello ibrido nelle stesse condizioni.

# Capitolo 4

# Osservatori e retroazione uscita-stato-ingresso

In questo capitolo verranno richiamate le problematiche relative ai sistemi dinamici lineari a tempo continuo riguardanti l'assegnabilità degli autovalori, la retroazione algebrica dell'uscita, il progetto di un osservatore identità per la stima dello stato e la retroazione stato stimato-ingresso realizzata mediante osservatore.

In relazione ai sistemi del secondo ordine, saranno utilizzati i parametri che caratterizzano la risposta del sistema stesso ad un gradino (massima sovraelongazione $S$, tempo di assestamento $T_a$ ed errore a regime) in relazione alla posizione dei poli nel piano complesso.

Verrà infine presentata una funzione *Matlab* che permette di assegnare arbitrariamente gli autovalori di un sistema.

## 4.1 Assegnabilità degli autovalori e retroazione dello stato

Si consideri il sistema dinamico lineare e stazionario nella forma

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned} \tag{4.1}$$

in cui $x(t) \in \Re^n$, $u(t) \in \Re^r$ e $y(t) \in \Re^m$. Se lo stato è accessibile si realizzi una retroazione algebrica dello stato attraverso l'equazione $u(t) = -\mathrm{H}x(t) + v(t)$, con $\mathrm{H}_{r\times n}$, da cui ottenere il sistema

$$\begin{aligned} \dot{x}(t) &= \big(A - BH\big)x(t) + Bv(t) \\ y(t) &= Cx(t) \end{aligned} \tag{4.2}$$

Si può dimostrare che gli autovalori di $(A - BH)$ sono posizionabili arbitrariamente in funzione della matrice di retroazione H se e solo se il sistema

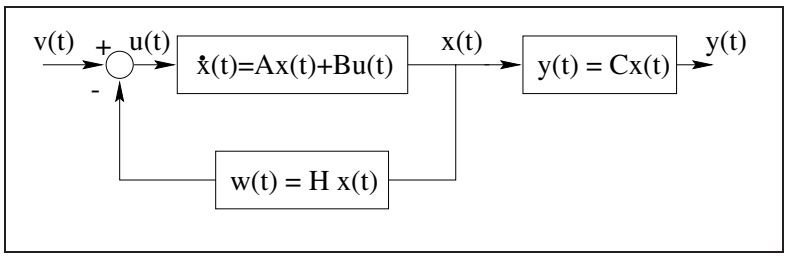



Figura 4.1: Schema a blocchi del sistema in retroazione ingresso-stato

$(A, B, C)$ è completamente raggiungibile e controllabile. Lo schema a blocchi della retroazione è rappresentato nella Figura 4.1
In *Matlab* è possibile effettuare l'operazione di assegnamento degli autovalori in maniera automatica attraverso la funzione `place`. L'istruzione `H = PLACE(A,B,P)` calcola la matrice di retroazione dello stato H in modo tale che gli autovalori di $(A - BH)$ siano quelli specificati nel vettore P. Gli autovalori non devono avere molteplicità maggiore del numero degli ingressi. Attraverso gli ulteriori parametri d'uscita `[H,PREC,MESSAGE] = PLACE(A,B,P)` viene restituita in `PREC` una stima di quanto gli autovalori sono stati assegnati vicino a quelli specificati (l'accuratezza è espressa in cifre decimali). `MESSAGE` contiene un messaggio di *warning* se qualche autovalore risulta lontano più del 10% rispetto alla posizione desiderata.

## 4.2 Retroazione algebrica dell'uscita

Il sistema continuo dinamico lineare stazionario

$$\begin{aligned} \dot{x}(t) &= Ax(t) + Bu(t) \\ y(t) &= Cx(t) \end{aligned} \tag{4.3}$$

con una retroazione algebrica dell'uscita nella forma $u(t) = -Ky(t) + v(t)$ si modifica in

$$\begin{aligned} \dot{x}(t) &= \big(A - BKC\big)x(t) + Bv(t) \\ y(t) &= Cx(t) \end{aligned} \tag{4.4}$$

come illustrato nella Figura 4.2.

## 4.3 Luogo delle radici

Per un sistema SISO, la funzione `RLOCUS(SYS)` calcola e grafica il luogo delle radici relativo all'equazione

$$H(s) = D(s) + KN(s) = 0 \tag{4.5}$$

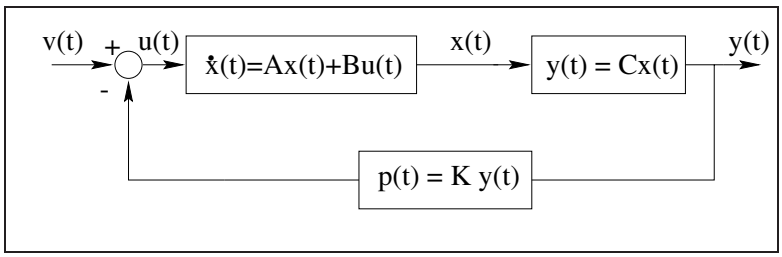



Figura 4.2: Schema a blocchi del sistema in retroazione uscita-ingresso

per un insieme di valori del guadagno $K$ tale da assicurare un grafico sufficientemente regolare. Il sistema `SYS` è ottenuto utilizzando la funzione *Matlab* `ss`, tale che `SYS = ss(A,B,C,D)` oppure `SYS = tf(N,D)` con `N` e `D` contenenti i coefficienti dei polinomi in $s$ del numeratore $N(s)$ e denominatore $D(s)$ della funzione di trasferimento $C(sI - A)^{-1}B + D$.

`[R,K] = rlocus(SYS,K)` restituisce in `R` le posizioni nel piano complesso degli autovalori ottenute specificando il valore del guadagno `K`.

La funzione `rlocfind` determina il guadagno relativo al luogo delle radici per un insieme prefissato di autovalori. L'istruzione `[K,POLES] = rlocfind(SYS)` apre una finestra grafica che permette di selezionare in maniera interattiva col mouse il guadagno relativo ad una collocazione di poli, basandosi sul luogo delle radici prodotto dalla funzione `rlocus`. Il guadagno che produce quel posizionamento dei poli viene restituito col parametro `K` e i corrispondenti poli, nel vettore `POLES`.

## 4.4 Osservatore identità

L'osservatore identità per un sistema nella forma 4.1 risulta

$$\dot{\hat{x}}(t) = \big(A-KC\big)\hat{x}(t)+Bu(t)+Ky(t) = A\hat{x}(t)+Bu(t)-K\big(C\hat{x}(t)-y(t)\big). \quad (4.6)$$

Se si definisce l'errore di stima $e(t) = \hat{x}(t) - x(t)$ si ha che la dinamica di $e(t)$ risulta descritta dalla relazione

$$e(t) = e^{(A-KC)t}e(0). \quad (4.7)$$

Si può dimostrare che l'osservatore ha dinamica arbitraria se il sistema di partenza è completamente osservabile e ricostruibile. Lo schema dell'osservatore è illustrato in Figura 4.3

## 4.5 Retroazione stato stimato-ingresso

In riferimento allo schema a blocchi di Figura 4.4
utilizzando le seguenti equazioni

$$\begin{aligned} \dot{x}(t) &= Ax(t) - BH\hat{x}(t) + Bv(t) \\ \dot{\hat{x}}(t) &= \big(A - KC\big)\hat{x}(t) + Bv(t) - BH\hat{x}(t) + KCx(t) \end{aligned}$$

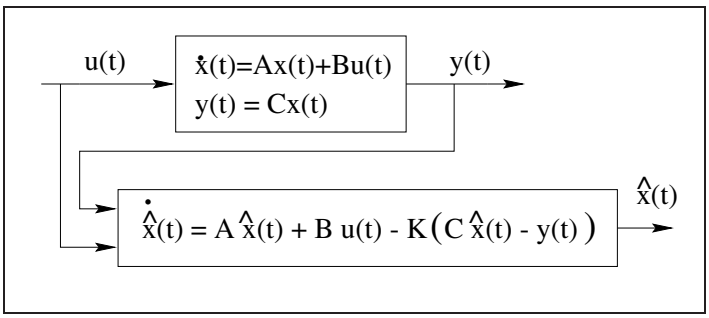



Figura 4.3: Schema a blocchi dell'osservatore identità

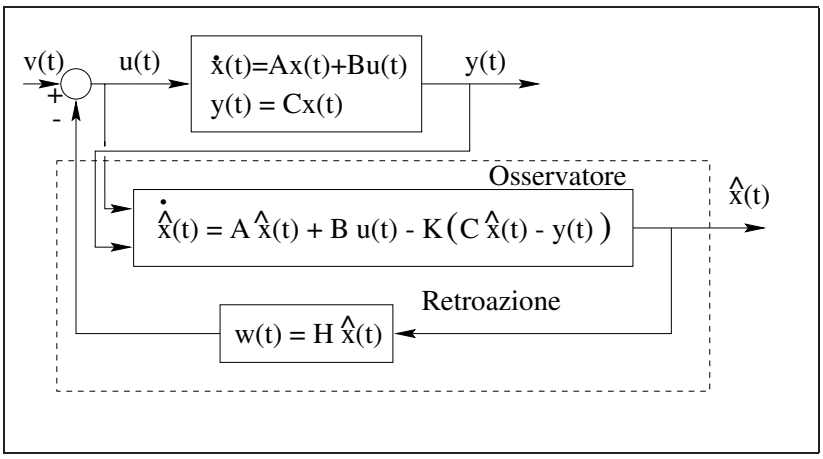



Figura 4.4: Schema a blocchi della retroazione dello stato stimato.

da cui, dalla definizione dell'errore di stima

$$\begin{aligned} \dot{x}(t) &= (A - BH)x(t) - BHe(t) + Bv(t) \\ \dot{e}(t) &= (A - KC)e(t). \end{aligned}$$

Considerando il sistema di ordine $2n$ avente vettore di stato $\begin{bmatrix} x \\ e \end{bmatrix}$, risulta

$$\begin{bmatrix} \dot{x}(t) \\ \dot{e}(t) \end{bmatrix} = \begin{bmatrix} A - BH & -BH \\ 0 & A - KC \end{bmatrix} \begin{bmatrix} x(t) \\ e(t) \end{bmatrix} + \begin{bmatrix} B \\ 0 \end{bmatrix} v(t). \qquad (4.8)$$

Dalla forma della matrice di stato del sistema di ordine $2n$ risulta evidente il "principio di separazione degli autovalori".

## 4.6 Parametri caratteristici di sistemi del secondo ordine

Dato il sistema del secondo ordine nella forma

$$G(s) = \frac{\omega_n^2}{s^2 + 2\delta\omega_n s + \omega_n^2} \tag{4.9}$$

nelle ipotesi per cui $0 < \delta < 1$, $\omega_n$ viene definita pulsazione naturale mentre $\delta$ coefficiente di smorzamento. Nel piano complesso $(\sigma, \omega)$, con $\sigma + \omega i = s$, $\omega_n\sqrt{1-\delta^2}$ rappresenta la parte immaginaria dei poli complessi coniugati e $-\delta\omega_n$ la parte reale. La risposta $y(t)$ al gradino unitario di tale sistema ha andamento oscillatorio e può essere caratterizzata dai seguenti parametri

- $S$, *massima sovraelongazione*, definita come $S = \frac{y(T_m)-y_\infty}{y_\infty} \times 100$ , in cui $T_m$ è l'istante di massima sovraelongazione e $y_\infty = \lim_{t\to\infty} y(t)$.

- $T_a$, *tempo di assestamento*, definito come il minimo valore del tempo $t$ tale che $|y(t) - y_\infty| \leq 0.05 y_\infty$, con $t \geq T_a$.

Per i sistemi del secondo ordine del tipo (4.9) si può dimostrare che

$$T_a \approx \frac{3}{\delta\omega_n} \tag{4.10}$$

mentre

$$S = 100e^{-\frac{\pi\delta}{\sqrt{1-\delta^2}}}. \tag{4.11}$$

## 4.7 Errore a regime

Si consideri il sistema con funzione di trasferimento $G(s)$ chiuso in retroazione unitaria come in Figura 4.5.

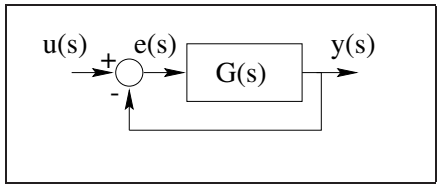



Figura 4.5: Sistema $G(s)$ chiuso in retroazione unitaria.

Definito $e(s) = u(s) - y(s)$, si ha che $e(s) = \frac{1}{1+G(s)}u(s)$ e per il Teorema del Valore Finale, $e_r = \lim_{t\to\infty} e(t) = \lim_{s\to 0} se(s)$ . L'errore a regime $e_r$ ha significato quando $u(s)$ coincide con i segnali di test, come il gradino, la rampa o la parabola.

## 4.8 Progetto di una retroazione

Nel seguito si utilizzeranno i precedenti richiami per risolvere in ambiente *Simulink* un problema di assegnamento di autovalori mediante retroazione e di progetto di osservatori.

In riferimento al sistema tempo-continuo, lineare e stazionario nello spazio degli stati descritto dalle matrici $(A, B, C, D)$

$$A = \begin{bmatrix} -9 & -26 & -24 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, B = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, C = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \text{ e } D = 0 \quad (4.12)$$

1. si determinino gli autovalori del sistema assegnato e si progetti per tentativi (graficando l'andamento nel tempo dell'uscita) una retroazione uscita-ingresso, come in Figura 4.2, tale che il sistema retroazionato abbia una risposta al gradino (di ampiezza 10) caratterizzata da un tempo di assestamento pari a 3s. ed un errore a regime $(v(t) - Ky(t))$ non superiore al 30% del valore dell'ampiezza del gradino.

   Calcolare successivamente gli autovalori dello stesso sistema chiuso in retroazione.

   Si faccia riferimento allo schema *Simulink* in Figura 4.6

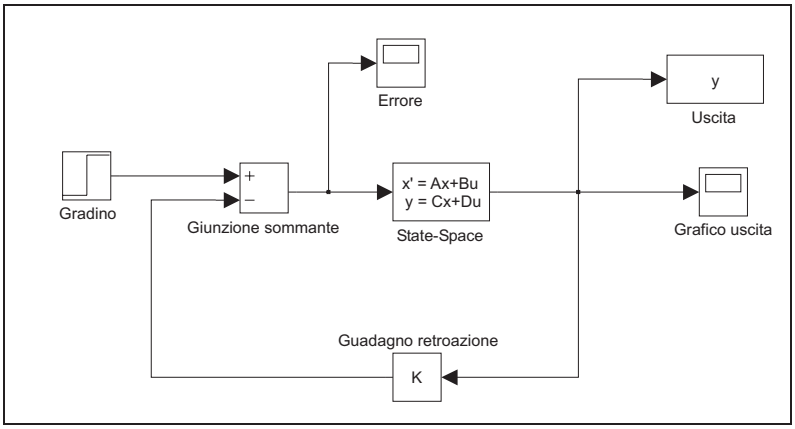



Figura 4.6: Sistema $(A, B, C, D)$ chiuso in retroazione.

Gli autovalori del sistema $(A, B, C, D)$ risultano

```
>>eig(A)

ans =

   -4.0000
   -3.0000
   -2.0000
```

Dalle specifiche sul tempo di assestamento $T_a$ (relativo ad un sistema del secondo ordine), si ottiene che la parte reale degli autovalori deve essere uguale a $-1$. Il procedimento di approssimazione del sistema in esame del terzo ordine con uno del secondo è possibile perché aumentando il guadagno di retroazione positivo, il più piccolo degli autovalori ($-4$) tende a spostarsi verso sinistra nel piano complesso, mentre gli altri due ($-3$ e $-2$) diventano complessi coniugati con parte reale sempre più grande, fino a diventare positiva. Conseguentemente, il tempo di assestamento peggiora mentre l'errore a regime si abbassa. Attraverso la funzione `rlocfind`, si può determinare una possibile soluzione ($K = 60$) del problema. L'andamento della risposta del sistema ad un gradino di ampiezza pari a 10 a partire dall'istante $t = 0$ è riportato in Figura 4.7.

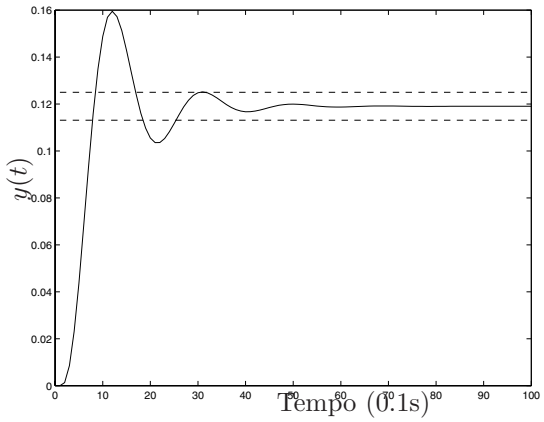



Figura 4.7: Risposta del sistema $(A, B, C, D)$ ad un gradino.

Viene evidenziato anche l'intervallo ottenibile dalla definizione di $T_a$ per verificare che il tempo di assestamento risulta approssimativamente uguale a $30 \times 10^{-1}$ secondi.

L'errore a regime viene riportato in Figura 4.8

L'errore a regime, che vale 2.8571, risulta soddisfare la specifica imposta.

La funzione `rlocus` permette inoltre di verificare il corretto posizionamento degli autovalori

```
>>R=rlocus(A,B,C,D,60)

R =

  -7.0000          -1.0000 + 3.3166i          -1.0000 - 3.3166i
```

Si osservi ancora che la risposta del sistema del terzo ordine è assimilabile a quella di uno del secondo ordine per la presenza di un autovalore reale inferiore rispetto alla parte reale dei due autovalori complessi coniugati. I due autovalori complessi rappresentano i *poli dominanti* per il sistema in retroazione, che si comporta quindi come un modello del secondo ordine.

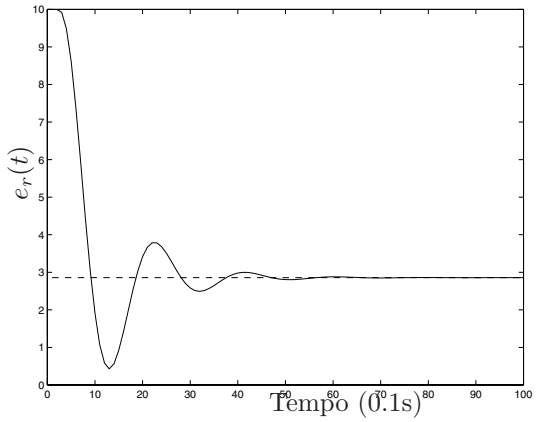



Figura 4.8: Errore a regime del sistema $(A, B, C, D)$ in riferimento ad un gradino.

È perciò giustificato l'utilizzo della relazione 4.10 relativa ad un sistema del secondo ordine con poli complessi coniugati senza zeri.

Gli autovalori del sistema $(A, B, C, D)$ in retroazione si ottengono

```
>>eig(A-B*K*C)

ans =

  -7.0000
  -1.0000 + 3.3166i
  -1.0000 - 3.3166i
```

coincidenti con quelli determinati dalla funzione `RLOCUS`, per un valore di $K = 60$ e contenuti nel vettore `R`.

2. Progettare per tentativi un osservatore identità per il sistema di partenza con autovalori reali e inferiori a $-4$ in modo da assicurare che l'errore di stima $e(t)$ scenda sotto al 2.5% in un tempo non superiore ad 1 secondo. Inoltre, i valori assoluti dei guadagni della matrice di retroazione dell'osservatore non devono risultare maggiori di 6. Si supponga che lo stato iniziale del sistema valga $[12, 20, -30]$ e che l'osservatore parta dallo stato zero.

   Dopo alcuni tentativi si sono determinati i valori corretti degli autovalori tali da soddisfare le specifiche richieste. La procedura è contenuta nel file `place eig.m`

```
%%
%% Funzione place_eig: assegna gli autovalori per ottenere
```

```
%% un errore di stima e(t) inferiore a 2.5% dopo 1 secondo.
%% Gli autovalori consentono di ottenere guadagni dell'ossevatore
%% inferiori a 6.
%% Le matrici A,B,C,D sono caricate dal file 'osserv.mat'.
%%

load osserv

AUTOV = [-4.85 , -4.9 , -4.82];
%AUTOV = [-5.0 , -4.5 , -5.5];
%AUTOV = [-2.0 , -3.0 , -4.0];
%AUTOV = [-4.5 , -5.0 , -5.5];

K = place(A',C',AUTOV)';

Ao = A - K*C;
Bo = [B K];
Co = eye(3);
Do = zeros(3,2);
```

mentre l'osservatore è implementato nel file *Simulink* rappresentato in Figura 4.9.

Si osservi come anche per l'assegnamento degli autovalori dell'osservatore sia possibile utilizzare la funzione `PLACE`, dal momento che $(A - KC) = (A^T - C^T K^T)^T$.

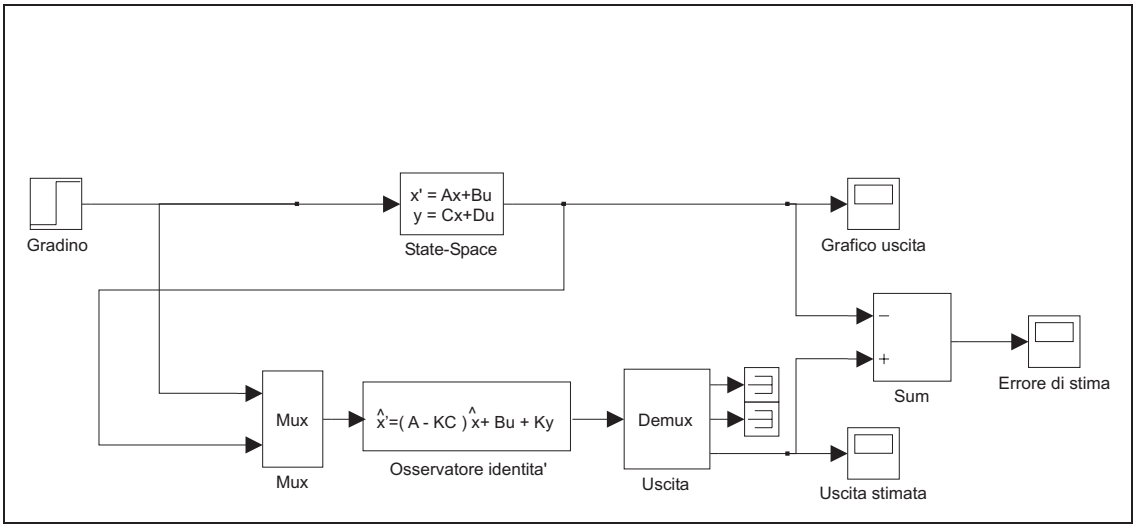



Figura 4.9: Osservatore identità per il sistema $(A, B, C, D)$.

Assegnando gli autovalori

```
AUTOV =
```

```
   -4.8500   -4.9000   -4.8200
```

si sono ottenuti i seguenti guadagni

```
K =

   -5.9427
   -5.3700
    5.5700
```

che soddisfano le specifiche richieste. L'errore di stima vale $e(t) = 0.2282$ per $t = 1s$ e l'andamento di $e(t)$ è riportato in Figura 4.10.

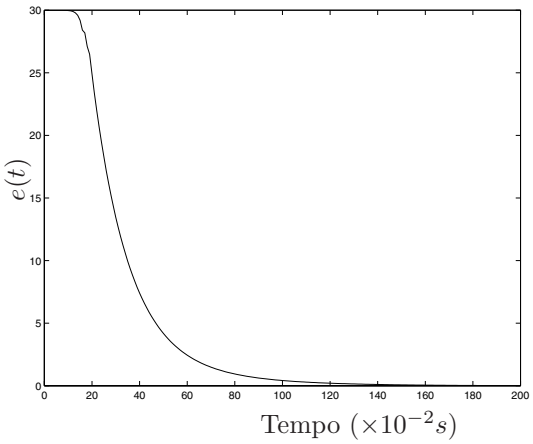



Figura 4.10: Andameno dell'errore di stima per il sistema $(A, B, C, D)$.

3. Per il sistema iniziale si progetti una retroazione dello stato stimato dall'osservatore calcolato al passo precedente in modo che vengano assegnati al sistema gli stessi autovalori che ha l'osservatore. Si grafichi l'andamento dell'uscita del sistema globale eccitato da un gradino di ampiezza unitaria.

Si possono assegnare al sistema di partenza gli autovalori dell'ossevatore identità utilizzando il file `place eig stato.m`

```
%%
%% Assegna gli autovalori dell'osservatore identita' al sistema
%% di partenza. Ks e' il guadagno della retroazione stato
%% stimato-ingresso.
%%


AUTOV = [-4.85 , -4.9 , -4.82];
```

```
Ks = place(A,B,AUTOV);

As = A - B*Ks;
```

e implementando lo schema a blocchi relativo ad una retroazione dello stato stimato-ingresso della Figura 4.11.

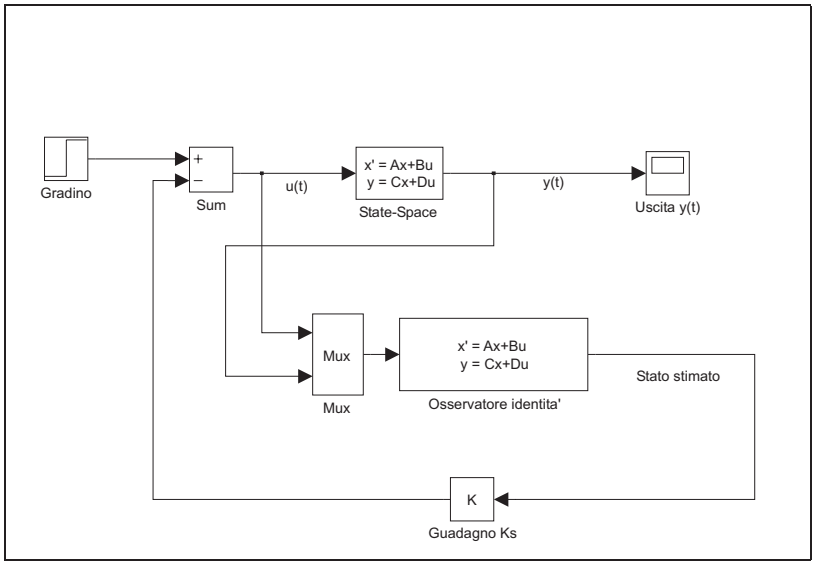



Figura 4.11: Retroazione dello stato stimato per il sistema $(A, B, C, D)$.

Gli autovalori $\begin{bmatrix} -4.8500 & -4.9000 & -4.8200 \end{bmatrix}$ vengono assegnati con una matrice di guadagno di retroazione pari a

```
Ks =

    5.5700   44.7600   90.5473
```

La risposta del sistema ad un gradino unitario chiuso in retroazione con lo stato stimato da un osservatore identità è rappresentata nella Figura 4.12.

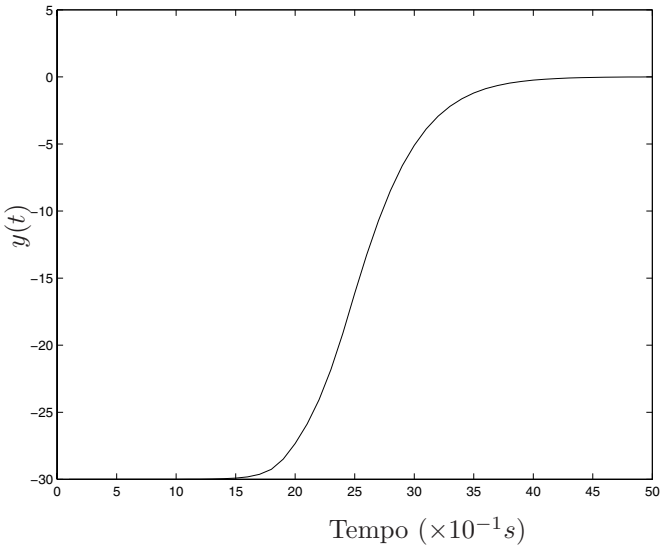



Figura 4.12: Risposta del sistema in retroazione con lo stato stimato.

## 4.9 Esercizi proposti in aula didattica.

È dato il sistema dinamico descritto dal modello

$$\begin{cases} \dot{x}(t) &= \mathrm{A}x(t) + \mathrm{B}u(t) \\ y(t) &= \mathrm{C}x(t) \end{cases} \tag{4.13}$$

con $x(t) \in \Re^n$, $u(t) \in \Re^r$ e $y(t) \in \Re^m$ e $n = 4$, $r = 1$, $m = 1$.

Le matrici del sistema siano

$$\mathrm{A} = \begin{bmatrix} -13 & -46 & -48 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 3 & 4 \end{bmatrix}, \qquad \mathrm{B} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \text{ e } \mathrm{C} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \end{bmatrix}. \tag{4.14}$$

In riferimento al sistema 4.13:

1. effettuare una realizzazione del sistema in ambiente *Simulink* e calcolarne gli autovalori;

2. determinare in ambiente *Matlab* il valore della matrice di retroazione `Ks` per una retroazione uscita-ingresso che assegni ad un autovalore il valore $-9$. Successivamente realizzare lo schema *Simulink* per il sistema in retroazione eccitato da un gradino di ampiezza 10; graficare la risposta al gradino $y(t)$ e l'errore $e(t)$. Calcolare graficamente il tempo di assestamento e la massima sovraelongazione.

3. Dato in ingresso un gradino di ampiezza 10, determinare una matrice di retroazione uscita-ingresso `Kr` affinché il tempo di assestamento sia $T_a =$

3s. In tali condizioni, in seguito determinare graficamente la massima sovraelongazione.

4. Determinare per il sistema di partenza il guadagno `Ko` di un osservatore dello stato che assegni gli autovalori $[-3, -3, -4]$. Si grafichi l'errore di stima nelle ipotesi che lo stato iniziale del sistema parta da $[12, -20, -30]$.

# Capitolo 5

# Progetto di Reti Correttrici

Il progetto di un sistema di controllo in retroazione si riconduce alla scelta della funzione di trasferimento del regolatore in modo che il sistema retroazionato abbia determinate caratteristiche. Dato che in molti casi le proprietà di un sistema retroazionato possono essere facilmente accertate a partire dalla risposta in frequenza associata alla funzione di trasferimento d'anello, un possibile approccio della sintesi è quello di individuare una funzione d'anello che permetta di soddisfare tutti i requisiti, ricavando poi da questa la funzione di trasferimento incognita del regolatore. Generalmente si utilizza un procedimento per tentativi, in cui si prendono in considerazione inizialmente regolatori con struttura molto semplice, complicandoli se necessario successivamente in base all'analisi delle prestazioni rispetto alle diverse specifiche di progetto.

Questo Capitolo è dedicato ad illustrare tramite esempi le principali tecniche di progetto basate sui diagrammi di Bode e Nyquist e sul luogo delle radici, limitandosi a considerare il caso del controllo di sistemi privi di poli nel semipiano destro del piano complesso. Saranno utilizzati alcuni regolatori, come le cosiddette reti anticipatrici e ritardatrici, e il progetto avverrà nell'ambiente *TFI*.

## 5.1 L'interprete TFI

L'Interprete di Funzioni di Trasferimento (*TFI*) è un pacchetto che lavora sotto *Matlab* creando un ambiente specifico di progettazione assistita rivolto all'analisi e alla sintesi interattiva di sistemi di controllo [7]. Tale ambiente si distingue per l'uso diretto delle funzioni di trasferimento, che vendono memorizzate nella directory di lavoro dell'hard disk come files `.mat`. Quando si accede a *TFI*, chiamando dalla *Matlab Command Window* il programma `tfi`, sullo schermo appare un diverso prompt (`>` invece di `>>`), per informare che la sintassi della *Command Window* è stata cambiata.

Il *TFI* è stato sviluppato per l'analisi e il progetto in modo interattivo veloce di sistemi di controllo automatico ad un ingresso ed una uscita. È utile per illustrare i concetti fondamentali dei Controlli Automatici, permette di effettuare l'analisi nel dominio dei tempi e delle frequenze. Mette a disposizione funzioni grafiche per la verifica della stabilità e la connessione di sistemi in retroazione,

per graficare luoghi delle radici e per il progetto di reti correttrici e regolatori. Possono essere simulati anche sistemi non lineari e a tempo discreto.

### 5.1.1 Modalità d'uso ed esempi

Se dalla *Matlab Command Window* si lancia il comando

```
>>tfi
```

verrà visualizzata la seguente schermata

```
**** passaggio all'ambiente TFI - attendere, prego

   **** la directory di lavoro di Matlab e' stata cambiata !

   INFORMAZIONI SULL'AMBIENTE TFI :

   la directory di lavoro di Matlab e' c:\matlab\toolbox\local
   la directory di lavoro di TFI e'    c:\matlab\toolbox\local
   il matlabpath NON E' RIDOTTO
   lo sfondo delle figure e' NERO
   la legenda nelle figure NON E' ATTIVATA

   usare "startint" da TFI per cambiare tali impostazioni

 **** premere un tasto per proseguire
```

e di seguito

```
TRANSFER FUNCTION INTERPRETER - A.Civolani e G.Marro - ver 3.2, 22-1-1998

 NOTA: sono disponibili da TFI i comandi Matlab:
       cd, clc, what, help file, print file [opzioni], grid, degrid,
       delete file, delete(n), dir, shg, zoom on, zoom off, figure;
       si possono valutare espressioni, come 3*6/(2+7) o [pi/6];
       "shg" fa passare da Command Window alla figura corrente;
       il tasto escape fa passare dalla figura a Command Window;
       i comandi "new" e "figure" creano una nuova finestra grafica;
       la sessione termina (uscita a Matlab) con "exit" o "quit".

 e' necessario definire un tempo di campionamento per la conversione da
 tempo continuo a tempo discreto (programmi convert e wplane):

   il tempo di campionamento attuale e' 0.1 sec

 per cambiarlo, introdurre il nuovo valore usando "sam[ptime]".
 Per maggiori informazioni sui comandi, inviare "help tfi" o "tfi".
```

Una funzione di trasferimento viene definita da tastiera con una stringa di caratteri composta dal nome della funzione di trasferimento, seguita dal segno `=` e da altri caratteri corrispondenti a numeri, parentesi, operatori e simboli (`s` o `z`).

Se viene rilevata un'operazione non ammessa compare un messaggio di errore. Consideriamo il seguente esempio.

Se si vuole inserire la funzione

$$g(s) = \frac{1+s}{1+5s} \tag{5.1}$$

usando la tastiera si digita

```
> g = (1 + s)/(1 + 5*s)

g=(1+s)/(1+5*s)
..............
--------------------
.....................

     0.2 (s + 1)
g = ------------
     (s + 0.2)

 >
```

quindi la funzione viene memorizzata nel file `g.mat`.

Sono ammesse le operazioni di somma, sottrazione, prodotto, divisione e potenza già definite in *Matlab*.

La visualizzazione della funzione di trasferimento avviene semplicemente chiamando il relativo nome

```
> g

    0.2 (s + 1)
g = ----------
    (s + 0.2)

 >
```

oppure usando le notazioni

```
> g =

    0.2 (s + 1)
g = ----------
   (s + 0.2)

 > g:

    1 (s + 1)
g = ---------
    (5*s + 1)

 >
```

per ottenere la funzione di trasferimento nella forma zeri-poli o con costanti di tempo.

Infine, se si invia il comando

```
> g;
```

viene visualizzata la mappa grafica degli zeri e poli del sistema, come in Figura 5.1.

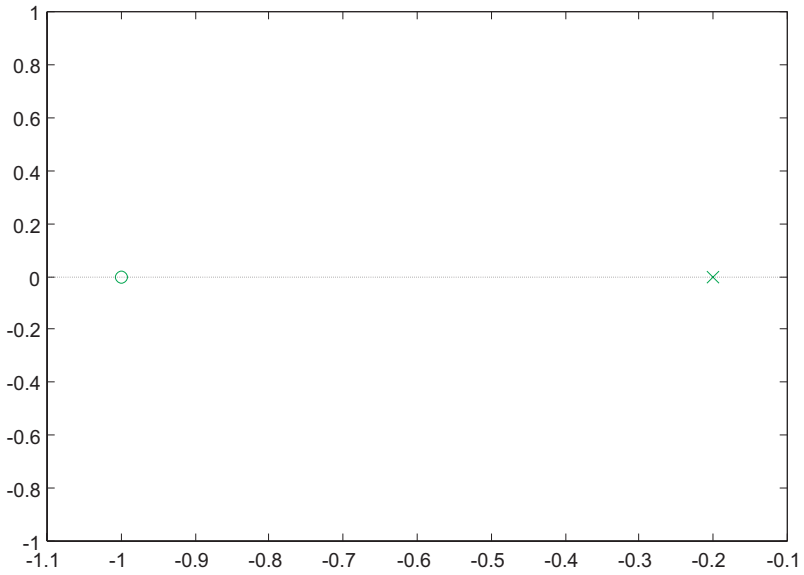



Figura 5.1: Polo (×) e zero (○) della funzione `g`.

*TFI* consente l'uso della sintassi del *Matlab* anche per calcoli numerici. L'espressione da calcolare va posta entro parentesi quadre, a meno che essa non inizi con un numero. In questi casi l'espressione viene passata automaticamente all'interprete dei comandi di *Matlab*.

### 5.1.2 Passaggio di funzioni di trasferimento tra *TFI* e *Matlab*

*TFI* salva le funzioni di trasferimento in file di tipo `.mat` e lavora perciò con oggetti che possono essere caricati da *Matlab* e viceversa. È quindi possibile convertire la forma di *Matlab* a quella di *TFI* e da *TFI* a *Matlab* usando le funzioni *Matlab* `sys=exportf('g',[1])` per convertire la funzione di trasferimento $g$ al corrispondente sistema *TFI sys* e `importf(sys,'g',[1])` per caricare in *Matlab* una funzione di trasferimento di *TFI* nella forma polinomiale se si utilizza l'opzione `[1]`. Per gli ulteriori comandi si rimanda al manuale [7].

### 5.1.3 Comandi *Matlab* in ambiente *TFI*

Da *TFI* è possibile usare alcuni dei comandi comuni all'ambiente *Matlab*. I più utilizzati sono:

```
cd

clc

clear

delete

dir

help file

fig[ure]

new

print

shg

zoom

what
```

### 5.1.4 *TFI* e le sue applicazioni

Nel seguito verrannno richiamate ed illustrate in ordine alfabetico le funzioni utilizzate per la risoluzione degli esercizi proposti nel seguito.

- **defactf**: `defactf,gi,gj` visualizza e salva nella directory corrente di lavoro con il nome `gj` la forma polinomiale di una data funzione di trasferimento `gi` relativa ad un sistema a tempo continuo o tempo discreto. Se perciò `gi` ha termini in forma fattorizzata, `gj` sarà espressa nella forma non fattorizzata.

- **factf**: `factf,gi,gj` visualizza e salva nella directory corrente di lavoro con il nome `gj` la forma fattorizzata (con un fattore di primo grado per ogni radice reale e uno di secondo grado per ogni coppia di radici complesse) di una funzione di trasferimento `gi` relativa ad un sistema, a tempo continuo o a tempo discreto, data in forma polinomiale non completamente fattorizzata.

- **fresp**: `fresp,gi` traccia la funzione di risposta in frequenza del sistema a tempo continuo o a tempo discreto `gi`. L'interattivo di cui il programma è dotato consente di ottenere, in colori diversi, piu' grafici nella stessa figura, relativi a diverse funzioni. Una volta invocato, il programma propone le seguenti scelte:

```
1 - Diagramma di Bode dell'ampiezza
2 - Diagramma di Bode della fase
3 - Diagrammi di Bode di ampiezza e fase - una sola figura
4 - Diagrammi di Bode di ampiezza e fase - due figure
5 - Diagramma di Nichols
6 - Diagramma di Nyquist

operare una scelta (0 per uscire) :
```

- **lagc**: `lagc,gi,gj` realizza il progetto per tentativi di una rete ritardatrice `gj` per il sistema controllato `gi` utilizzando i diagrammi di Bode.

  Al momento dell'invocazione del comando, viene visualizzata la seguente schermata e la Figura del sistema a blocchi 5.2

```
> lagc,gi,gj

lagc (progetto interattivo di rete ritardatrice)

**** premere invio per proseguire

informazioni sul progetto della rete ritardatrice ? (1) : 1


RETE RITARDATRICE

                                    1 + alfa tau s
Funzione di trasferimento: gc(s) = -------------- .
                                      1 + tau s

Pulsazione di centro banda: om_0 = 1/(tau sqrt(alfa)) .

                                                      1 - alfa
Massimo ritardo di fase (ad om_0): phi_0 = -arcsen -------- .
                                                      1 + alfa


**** premere un tasto per vedere i diagrammi di Bode
```

  Per il progetto si seguono le istruzioni del programma.

- **leadc**: `leadc,gi,gj` realizza il progetto per tentativi di una rete anticipatrice `gj` per il sistema controllato `gi` utilizzando i diagrammi di Bode.

  L'output è simile a quello mostrato per la rete ritardatrice.

- **rootl**: `rootl,gi` traccia il luogo delle radici di $1 + K\texttt{gi} = 0$ per sistemi a tempo continuo o discreto con $K \in [0, \infty)$.

  Il programma visualizza il seguente output e la Figura 5.3

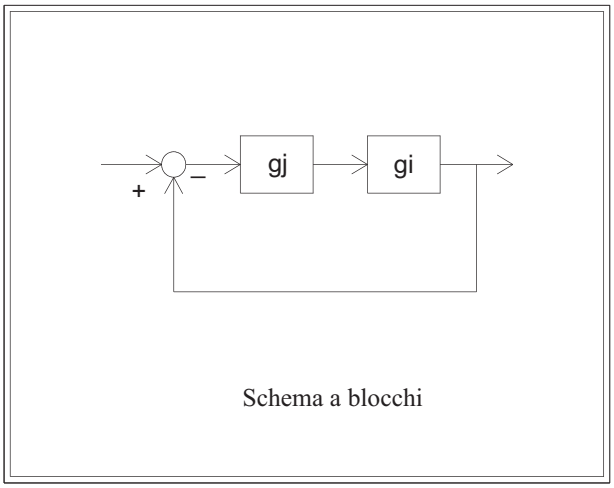



Figura 5.2: Schema in retroazione con una rete ritardatrice.

```
> rootl,gi

rootl (luogo delle radici)

**** premere invio per proseguire

LUOGO DELLE RADICI :

x poli ad anello aperto (= poli ad anello chiuso per K=0)
o zeri ad anello aperto (= poli ad anello chiuso per K=Inf)
+ poli ad anello chiuso per K=1

il luogo viene tracciato passo passo per K crescente
o per K decrescente se il sistema e' non causale

scegliere il colore del grafico: w=bianco, g=verde,
b=blu, r=rosso, y=giallo, m=magenta, c=celeste, default verde :
```

- **tresp**: `tresp,gi` traccia la risposta al gradino o all'impulso del sistema con funzione di trasferimento `gi`, a tempo continua o a tempo discreto, con scelta fra la risposta ad anello aperto o in retroazione unitaria. Il programma infatti propone le seguenti opzioni

```
1 - risposta al gradino ad anello aperto
2 - risposta al gradino ad anello chiuso
3 - risposta all'impulso ad anello aperto
4 - risposta all'impulso ad anello chiuso
```

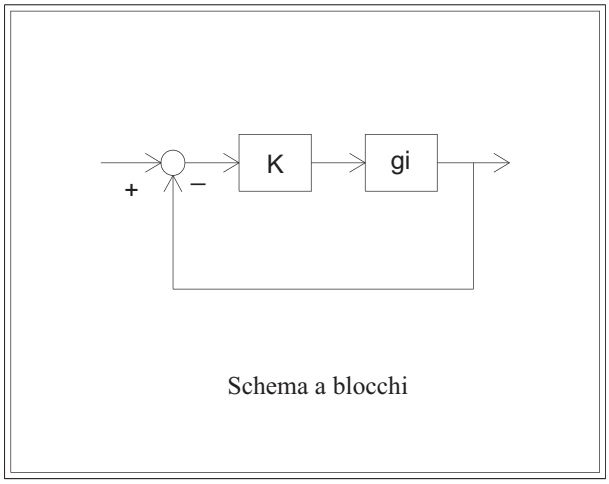



Figura 5.3: Schema per il calcolo del luogo delle radici.

```
operare una scelta (default 1, 0 per uscire) :
```

## 5.2 Progetto di una rete anticipatrice con i diagrammi di Bode

Si consideri il sistema chiuso in retroazione unitaria caratterizzato dalla funzione di trasferimento $G_p(s)$

$$G_p(s) = \frac{1000}{s(s+10)}. \tag{5.2}$$

1. si progetti una rete anticipatrice $G_c(s)$ da collegare in serie al sistema $G_p(s)$, con funzione di trasferimento

$$G_c(s) = \frac{1+\tau s}{1+\alpha\tau s} \tag{5.3}$$

   con $\alpha < 1$, in modo che il sistema risultante sia caratterizzato da un margine di fase $M_f > 75^o$ e da un picco di risonanza minore di 1.1.

2. Si illustri l'intervento della rete progettata con i diagrammi di Bode e di Nyquist.

3. Determinare le caratteristiche (massima sovraelongazione, tempo di assestamento, errore a regime) della risposta al gradino del sistema compensato chiuso in retroazione unitaria.

## Risoluzione

Viene inserita la funzione del sistema da controllare

```
> Gp=1000/(s*(s+10))

Gp=1000/(s*(s+10))
..................
--------------------
.....................

     1000
Gp = ---------
     s (s + 10)
```

e successivamente si richiama la funzione `leadc` per il progetto interattivo della rete anticipatrice.

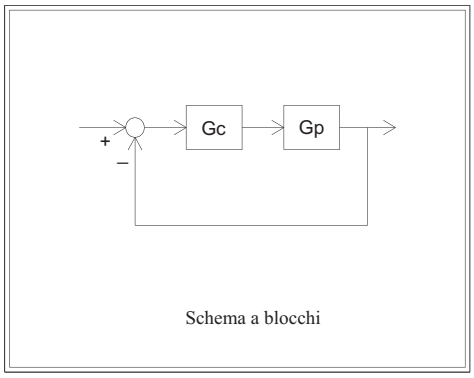
```
leadc,Gp,Gc
   leadc (progetto interattivo di rete anticipatrice)

   **** premere invio per proseguire
```

che genera il seguente output (Figure 5.4)
e l'interattivo a testo

```
PROCEDIMENTO DI PROGETTO DELLA RETE ANTICIPATRICE:

   1 - vengono calcolati e mostrati il margine di fase phm e la
       corrispondente pulsazione omm del sistema controllato;
   2 - specificato dall'utente il margine di fase richiesto phd,
       viene calcolato e mostrato il valore alfa_0 corrispondente
       ad un anticipo di fase a centro banda pari a phd-phm;
   3 - per il primo tentativo si assume alfa=(alfa_0)/2;
   4 - tau viene variato fra sqrt(alfa)/omm e 1/(omm*sqrt(alfa))
       (100 passi) e viene scelto il valore corrispondente al
       massimo margine di fase del sistema complessivo.

   In interattivo si puo' modificare il valore di alfa.


   inviare 1 per proseguire con il progetto, 0 per uscire : 1
```

Nelle Figure 5.5 vengono visualizzati i diagrammi di Bode del sistema senza rete correttrice e dopo l'applicazione della rete correttrice progettata sulla base delle seguenti specifiche

```
   margine di fase senza correzione: 17.96 gradi
                   alla pulsazione: 30.84 rad/sec

   specificare il margine di fase voluto : 75

   anticipo di fase necessario: 57.04 gradi
   valore di alfa di primo tentativo: 0.04377

   **** premere un tasto per proseguire
```



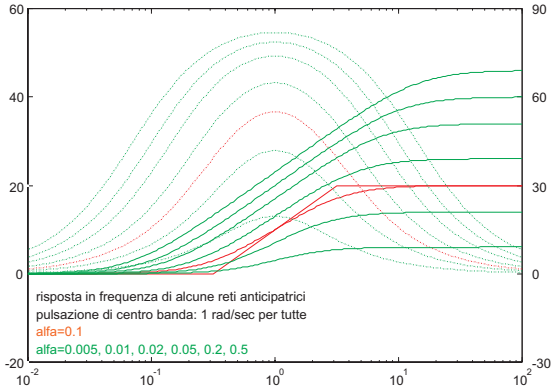




Figura 5.4: Output grafico della funzione `leadc`.

da cui calcolare i seguenti valori e le risposte del sistema in retroazione

```
**** premere un tasto per proseguire

**** figura 1 ****

**** figura 2 ****

**** figura 3 ****

colore di riferimento: g ; f.d.t. del sistema controllato: Gp

100 passi di ricerca di tau nell'intervallo scelto:
margine di fase massimo trovato al passo: 37

margine di fase senza rete correttrice: 17.96
margine di fase con rete correttrice  : 75
```

```
   la rete correttrice ottenuta:
   alfa = 0.04377 , tau = 0.06201 sec

   si puo' cambiare alfa; verra' determinato il tau
   corrispondente al margine di fase massimo

   specificare alfa (min .005, max 0.1751), invio per uscire :
```

`**** figura 1 ****` si riferisce alla Figura 5.5, mentre `**** figura 2 ****` e `**** figura 3 ****` alle Figure 5.6
Con il valore di primo tentativo per $\alpha$ e $\tau$ corriespondente al margine di fase massimo, si ottiene il seguente messaggio

```
LA RETE CORRETTRICE OTTENUTA :

   alfa = 0.04377 , tau = 0.06201 sec

     22.85 (s + 16.13)
Gc = ----------------
     (s + 368.4)

>
```

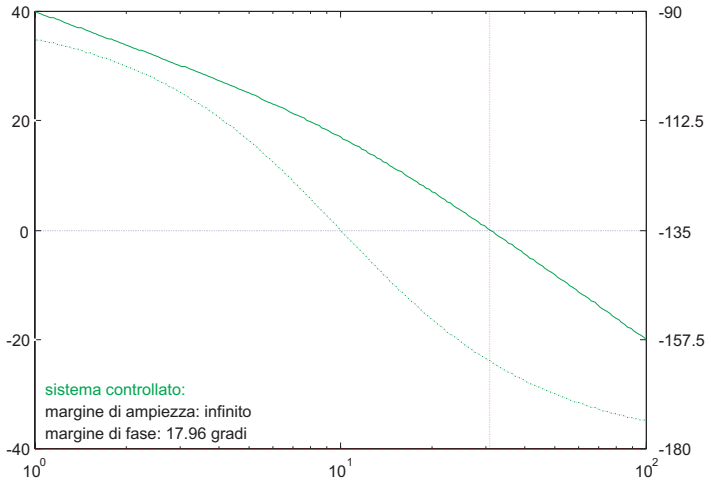
oppure

```
> Gc:

     1 (0.06201*s + 1)
Gc = ----------------
     (0.002714*s + 1)
```

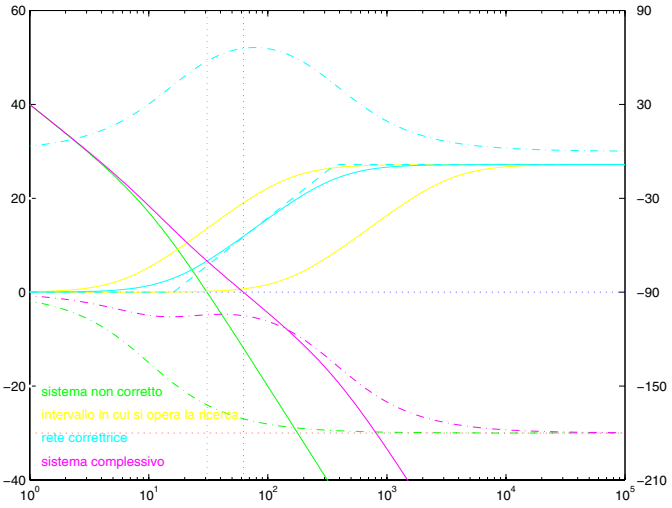
Per visualizzare i diagrammi di Bode e determinare il picco di risonanza e verificare il margine di fase ottenuti, si utilizza la funzione `fresp`, secondo la sintassi

```
> G=Gp*Gc

G=Gp*Gc
.......
--------------------
.....................

    2.285e+004 (s + 16.13)
G = ----------------------
    s (s + 10) (s + 368.4)

 > fresp,G
```

Si ottengono i seguenti diagrammi di Bode dell'ampiezza a della fase visualizzati nella Figura 5.7



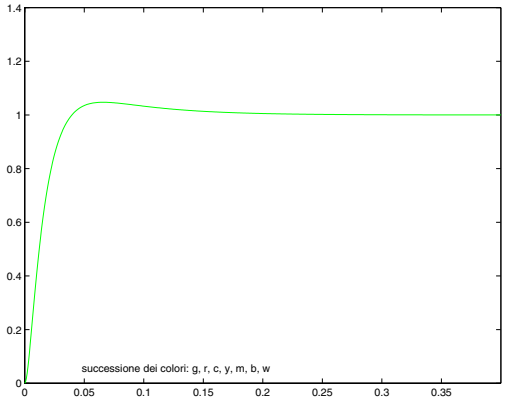

(a)



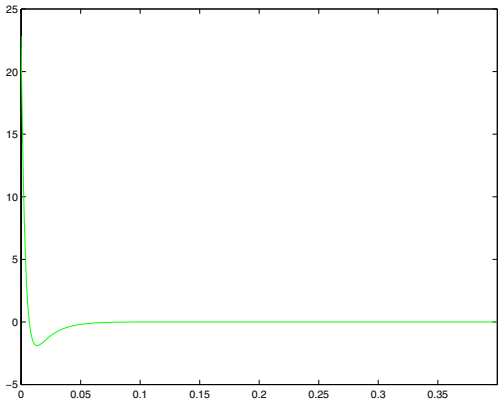

(b)

Figura 5.5: Diagrammi di Bode del sistema con (a) e senza (b) rete correttrice.

```
RISPOSTA IN FREQUENZA AD ANELLO APERTO :
   margine di ampiezza non determinabile
   margine di fase: 75 gradi per omega= 62.36 rad/sec
   ascissa dell'asintoto verticale del diagramma polare: -4.071
```




(a)




(b)

Figura 5.6: Risposta al gradino del sistema compensato (a) e uscita della rete correttrice (b).

```
RISPOSTA IN FREQUENZA AD ANELLO CHIUSO :
risonanza assoluta: 1.038 (0.3239 db) per omega= 18.1 rad/sec
guadagno statico: 1 (0 db); risonanza relativa: 1.038 (0.3239 db)
banda passante (-3db): 81.29 rad/sec

**** premere un tasto per proseguire
```

Per evidenziare l'effetto dell'intervento della rete anticipatrice, si riportano i diagrammi di Bode (Figura 5.8) e Nyquist (Figura 5.9) del sistema non compensato e compensato con le rispettive caratteristiche

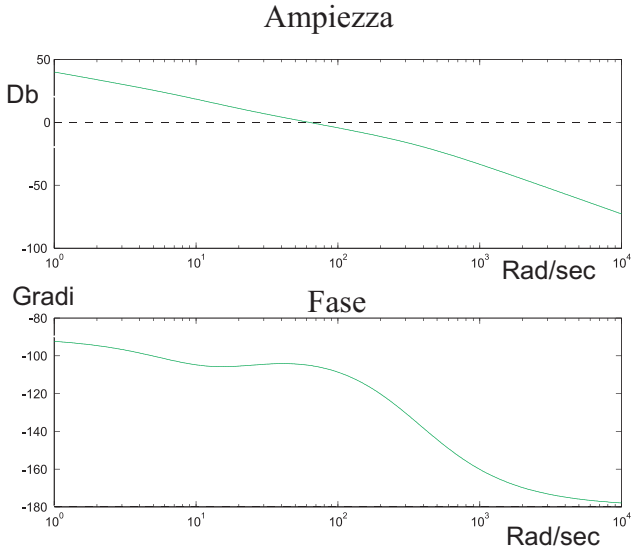



Figura 5.7: Diagramma di Bode della fase e dell'ampiezza per il sistema compensato $G(s) = G_c(s) * G_p(s)$.

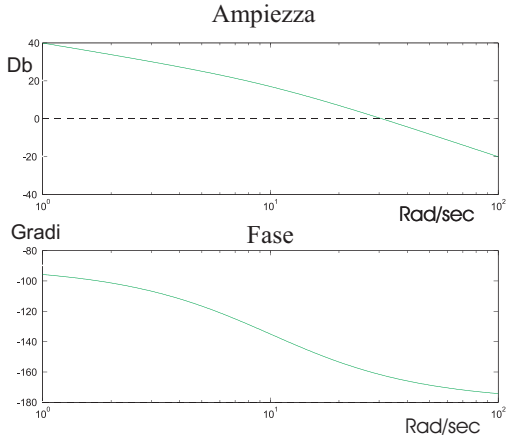
Si può notare anche l'effetto della rete anticipatrice confrontando i parametri della risposta in frequenza per il sistema non compensato

```
RISPOSTA IN FREQUENZA AD ANELLO APERTO :
   margine di ampiezza non determinabile
   margine di fase: 17.96 gradi per omega= 30.84 rad/sec
   ascissa dell'asintoto verticale del diagramma polare: -10

   RISPOSTA IN FREQUENZA AD ANELLO CHIUSO :
   risonanza assoluta: 3.203 (10.11 db) per omega= 30.82 rad/sec
   guadagno statico: 1 (0 db); risonanza relativa: 3.203 (10.11 db)
   banda passante (-3db): 48.27 rad/sec
```

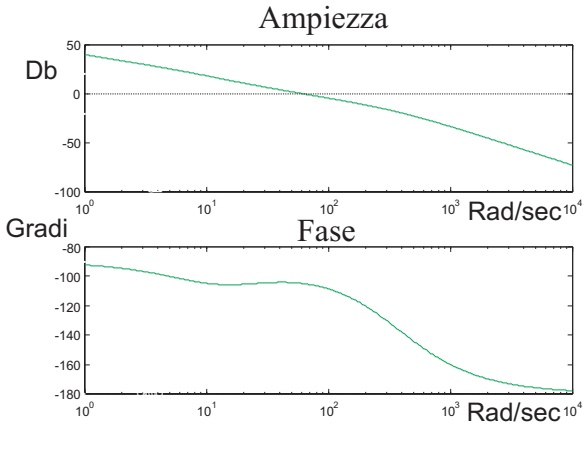
e quelli del sistema compensato

```
RISPOSTA IN FREQUENZA AD ANELLO APERTO :
   margine di ampiezza non determinabile
   margine di fase: 75 gradi per omega= 62.36 rad/sec
   ascissa dell'asintoto verticale del diagramma polare: -4.071

   RISPOSTA IN FREQUENZA AD ANELLO CHIUSO :
   risonanza assoluta: 1.038 (0.3239 db) per omega= 18.1 rad/sec
   guadagno statico: 1 (0 db); risonanza relativa: 1.038 (0.3239 db)
   banda passante (-3db): 81.29 rad/sec
```

Si possono perciò notare l'aumento del margine di fase (per cui la rete è stata progettata), la diminuzione del picco di risonanza e l'aumento della larghezza di banda dovuto all'incremento del guadagno alle alte frequenze.




(a)

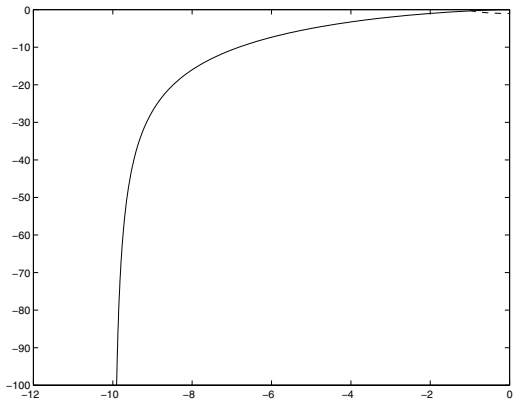



(b)

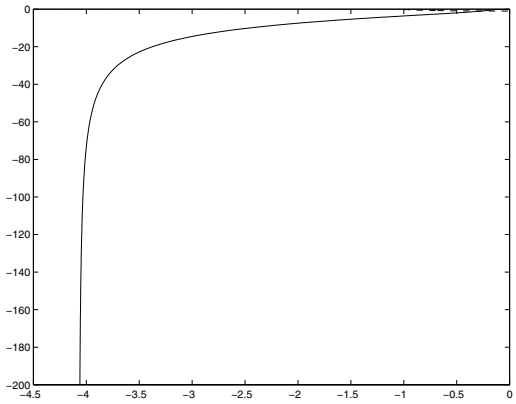
Figura 5.8: Diagrammi di Bode della fase e dell'ampiezza per il sistema non compensato (a) e compensato (b) $G_p(s)$.

I diagrammi di Nyquist del sistema con e senza rete anticipatrice sono visualizzati nella Figura 5.9
I parametri della risposta al gradino del sistema compensato chiuso in retroazione unitaria sono detrminabili attraverso la funzione `tresp`. Digitando infatti il comando

```
> tresp,G
   tresp (risposta nel tempo)

   1 - risposta al gradino ad anello aperto
```




(a)




(b)

Figura 5.9: Diagrammi di Nyquist per il sistema non compensato (a) e compensato (b) $G_p(s)$.

```
   2 - risposta al gradino ad anello chiuso
   3 - risposta all'impulso ad anello aperto
   4 - risposta all'impulso ad anello chiuso

   operare una scelta (default 1, 0 per uscire) : 2
```

si ottiene il seguente output

```
scegliere il colore del grafico: w=bianco, g=verde,
   b=blu, r=rosso, y=giallo, m=magenta, c=celeste, default verde :

   **** figura 1 ****
```

```
MENU :

1 - cambiare gli assi di riferimento
2 - inserire reticolo
3 - informazioni (solo sulla risposta al gradino)
4 - aggiungere un grafico in colore diverso
5 - rivedere la figura
6 - informazioni sui grafici con il mouse

operare una scelta (premere invio per uscire) : 3

RISPOSTA AL GRADINO :
massima sovraelongazione: 4.741 percento per t=0.06586 sec
tempo di ritardo (al 50 percento): 0.0122 sec
tempo di salita (dal 10 al 90 percento): 0.02456 sec
tempo di assestamento (al piu'/meno 5 percento): 0.03265 sec

ERRORI A REGIME (SOLO AD ANELLO CHIUSO) :
errore a regime in risposta al gradino: 0
errore a regime in risposta alla rampa: 0.01

premere un tasto per proseguire
```

La risposta al gradino nella Figura 5.10.

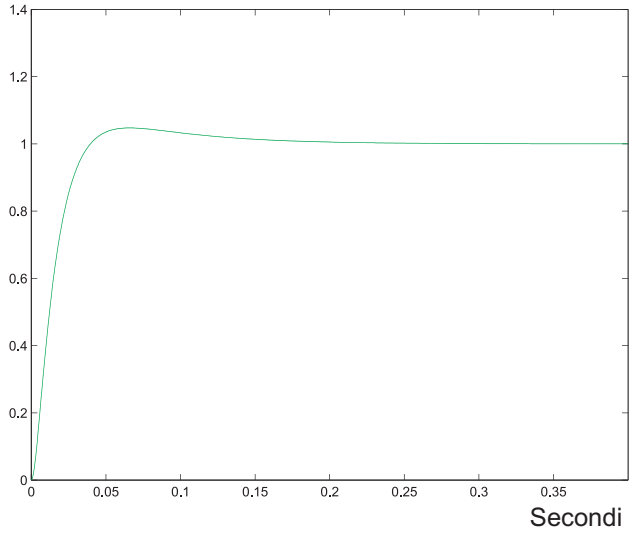



Figura 5.10: Risposta al gradino del sistema chiuso in retroazione unitaria.

## 5.3 Progetto di una rete correttrice con il luogo delle radici

Si consideri il sistema descritto dalla funzione di trasferimento

$$G_p(s) = \frac{K}{s(s+10)(s+50)} \tag{5.4}$$

chiuso in retroazione unitaria

Utilizzando il luogo delle radici, si progetti la rete anticipatrice, con uno zero localizzato ad $s = -15$, che consenta di soddisfare le seguenti specifiche

- massima sovraelongazione percentuale approssimativamente uguale a 7.5%
- tempo di assestamento pari a 0.4 secondi.

### Soluzione

Utilizzando la funzione `rootl`, si grafica il luogo delle radici (Figura 5.11) relativo alla funzione di trasferimento $G_p(s)$ del sistema da controllare come in Figura 5.12.

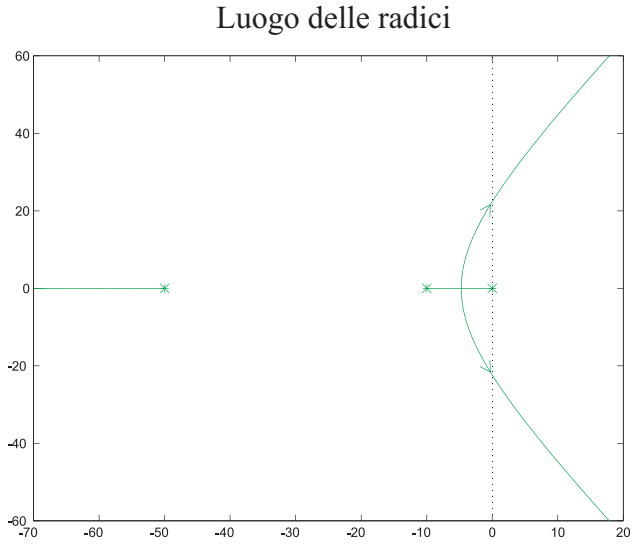



Figura 5.11: Luogo delle radici di $G_p(s)$.

Viene generato in seguente output

```
> Gp = 1/(s*(s+10)*(s+50))

Gp=1/(s*(s+10)*(s+50))
.......................
--------------------
.....................
```

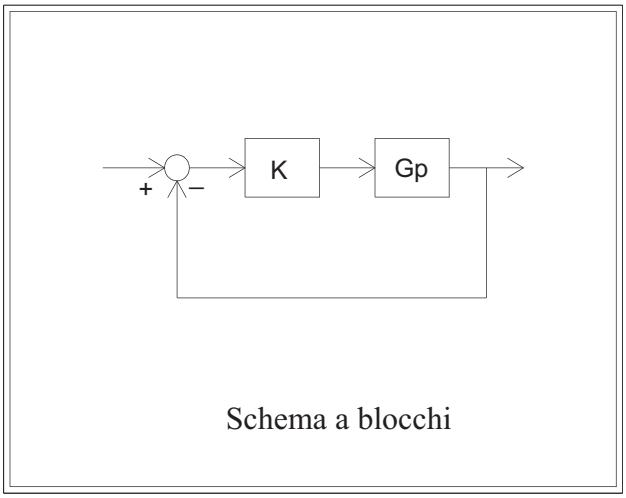



Figura 5.12: Schema a blocchi relativo al luogo delle radici per $G_p(s)$.

```
     1
Gp = ------------------
     s (s + 10) (s + 50)
```

```
> rootl,Gp
   rootl (luogo delle radici)

   **** premere invio per proseguire

   LUOGO DELLE RADICI :

   x poli ad anello aperto (= poli ad anello chiuso per K=0)
   o zeri ad anello aperto (= poli ad anello chiuso per K=Inf)
   + poli ad anello chiuso per K=1

   il luogo viene tracciato passo passo per K crescente
   o per K decrescente se il sistema e' non causale

   scegliere il colore del grafico: w=bianco, g=verde,
   b=blu, r=rosso, y=giallo, m=magenta, c=celeste, default verde :

   **** figura 1 ****

   MENU :

   1 - cambiare gli assi di riferimento e ripetere l'ultimo grafico
   2 - inserire reticolo
```

```
   3 - informazioni su punti di diramazione e asintoti
   4 - aggiungere un grafico in colore diverso
   5 - rivedere la figura
   6 - informazioni sul luogo con il mouse
   7 - luoghi a delta costante (on/off)
   8 - tracciare gli asintoti
   9 - mostrare la disposizione dei poli per un dato valore di K
  10 - cambiare il passo e ripetere l'ultimo grafico

   operare una scelta (premere invio per uscire) : 6

   **** premere invio per abilitare la selezione

   **** figura 1 ****


   conservare le frecce ? (1) : 1
```

Le intersezioni del luogo con l'asse delle ordinate corrispondono ad un $K = 2.99 \times 10^4$.

Si introduce la rete anticipatrice $G_c(s)$ con lo zero fissato e il polo scelto a $-100$ corrispondente ad una dinamica veloce.

$$G_c(s) = \frac{1 + \frac{s}{15}}{1 + \frac{s}{100}} \tag{5.5}$$

```
> Gc=(1+s/15)/(1+s/100)

Gc=(1+s/15)/(1+s/100)
.....................
--------------------
.....................

     6.667 (s + 15)
Gc = --------------
     (s + 100)

>
```

L'inserimento della rete a monte del blocco $G_p$ nella Figura 5.12 porta ad un nuovo luogo delle radici, riportato in Figura 5.13.

```
> G=Gc*Gp

G=Gc*Gp
.......
--------------------
.....................

    6.667 (s + 15)
```

```
G = ----------------------------
     s (s + 10) (s + 50) (s + 100)

>rootl,G
   rootl (luogo delle radici)

   **** premere invio per proseguire

   LUOGO DELLE RADICI :

   x poli ad anello aperto (= poli ad anello chiuso per K=0)
   o zeri ad anello aperto (= poli ad anello chiuso per K=Inf)
   + poli ad anello chiuso per K=1

   il luogo viene tracciato passo passo per K crescente
   o per K decrescente se il sistema  non causale

   scegliere il colore del grafico: w=bianco, g=verde,
   b=blu, r=rosso, y=giallo, m=magenta, c=celeste, default verde :

   **** figura 2 ****
```

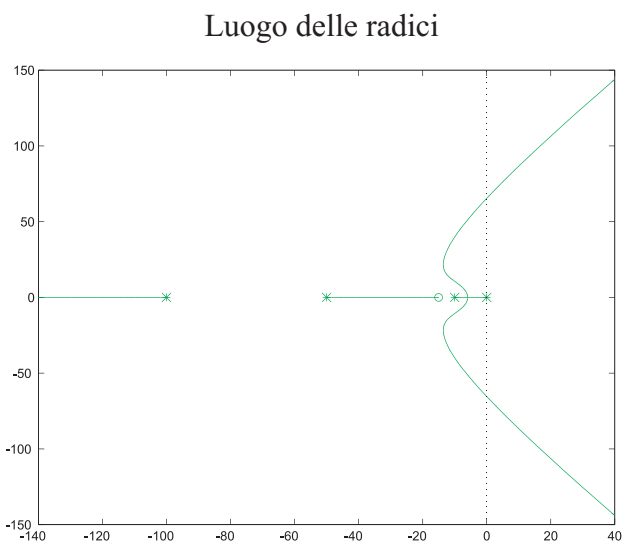



Figura 5.13: Luogo delle radici per $G_c(s)G_p(s)$.

I punti di intersezione del luogo con l'asse delle ordinate corrispondono ad un valore di $K$ pari a $8.95 \times 10^4$. Si osservi come l'introduzione di uno zero generi una deformazione nel luogo delle radici, provocando un effetto "attrattivo".

Per rispettare la prima specifica di $S < 7.5\%$, se il sistema fosse del secondo ordine e privo di zeri, sarebbe necessario un $\delta \approx 0.65 \div 0.66$ (Figura 5.14). Usando ancora il menu di `rootl`, si graficano le zone a $\delta$ costante e, utilizzando il mouse, si determina un valore di $K$ di primo tentativo.

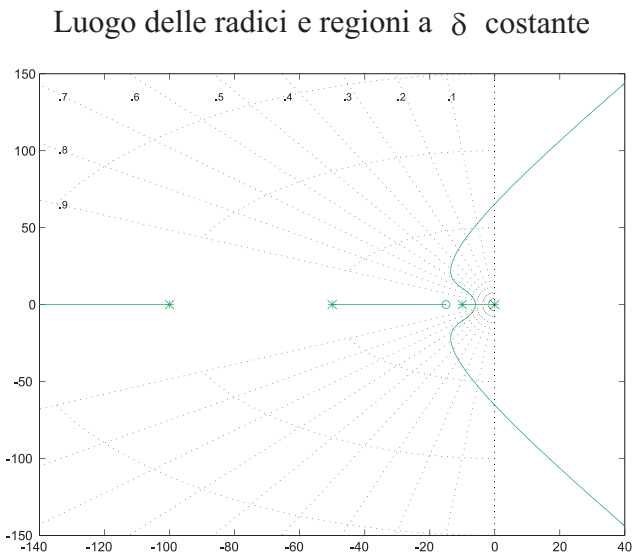



Figura 5.14: Regioni a $\delta$ per $G_c(s)G_p(s)$.

Valori ammissibili per $K$ sono $5000 < K < 10000$. Una procedura per tentativi richiede di verificare i risultati ottenuti con la funzione `tresp` applicata a $G(s) = KG_c(s)G_p(s)$ per la risposta al gradino. Tale procedura porta a scegliere il valore di $K = 6000$, corrispondente ai seguenti risultati.

```
> G=6000*Gc*Gp


G=6000*Gc*Gp
.............
--------------------
.....................


     4e+004 (s + 15)
G = ----------------------------
     s (s + 10) (s + 50) (s + 100)


> tresp,G
   tresp (risposta nel tempo)

   1 - risposta al gradino ad anello aperto
   2 - risposta al gradino ad anello chiuso
   3 - risposta all'impulso ad anello aperto
   4 - risposta all'impulso ad anello chiuso
```

```
   operare una scelta (default 1, 0 per uscire) : 2

   scegliere il colore del grafico: w=bianco, g=verde,
   b=blu, r=rosso, y=giallo, m=magenta, c=celeste, default verde :

   **** figura 1 ****
```

Si ottiene la seguente risposta al gradino (Figura 5.15) corrispondente ai seguenti risultati.

```
RISPOSTA AL GRADINO :
   massima sovraelongazione: 7.477 percento per t=0.2913 sec
   tempo di ritardo (al 50 percento): 0.09172 sec
   tempo di salita (dal 10 al 90 percento): 0.1309 sec
   tempo di assestamento (al piu'/meno 5 percento): 0.3759 sec

   ERRORI A REGIME (SOLO AD ANELLO CHIUSO) :
   errore a regime in risposta al gradino: 0
   errore a regime in risposta alla rampa: 0.08333
```

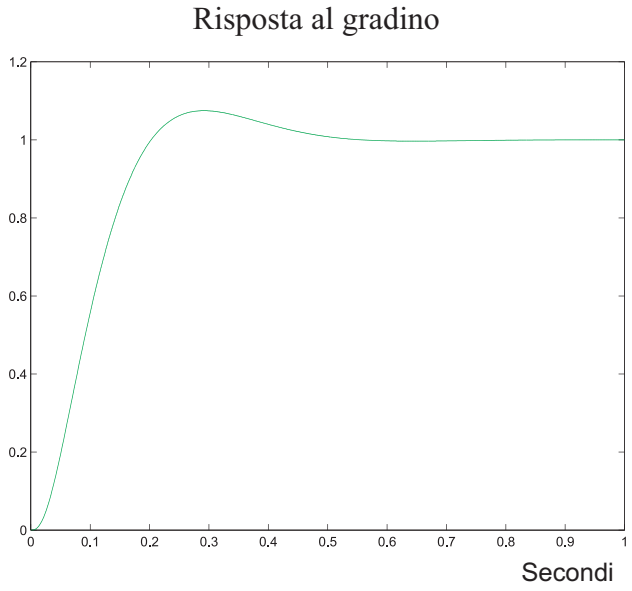



Figura 5.15: Risposta al gradino del sistema $KG_c(s)G_p(s)$ chiuso in retroazione con $K = 6000$.

Le specifiche risultano soddisfatte.

## 5.4 Progetto di una rete ritardatrice

Data la funzione di trasferimento

$$Gp(s) = \frac{K}{s(s+5)^2} \tag{5.6}$$

progettare la rete ritardatrice e calcolare il valore di $K$ tali che il sistema chiuso in retroazione unitaria soddisfi le seguenti specifiche

- massima sovraelongazione minore dell'1%
- tempo di salita minore di 2 secondi
- tempo di assestamento di 2.5 secondi

Confrontare le prestazioni delle diverse soluzioni ottenute in termini di larghezza di banda, margine di ampiezza e margine di fase.

## Soluzione

La Figura 5.16 riporta il luogo delle radici per il sistema $G_p(s)$

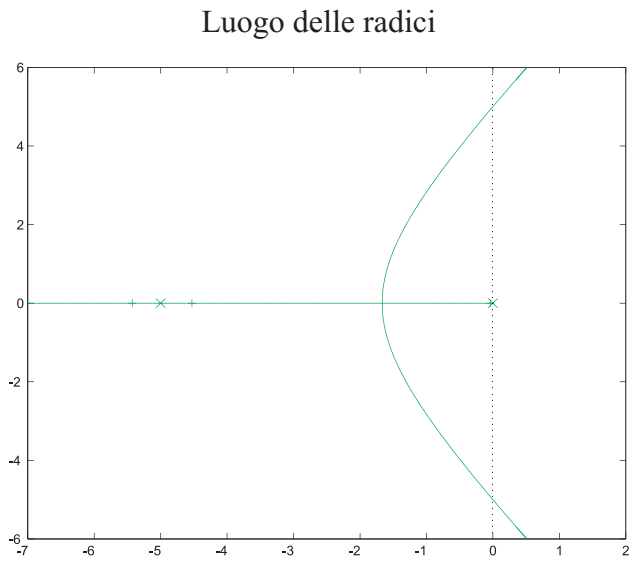



Figura 5.16: Luogo delle radici per $G_p(s)$.

precedentemente introdotto in *TFI*

```
> Gp=1.0/(s*(s+5)^2)

Gp=1.0/(s*(s+5)^2)
...................
--------------------
......................

      1
```

```
Gp = ----------
     s (s + 5)^2

 >
```

Le tacche nel luogo corrispondono al valore di $K = 1$. La risposta al gradino del sistema non compensato risulta avere le seguenti caratteristiche

```
RISPOSTA AL GRADINO :
   nessuna sovraelongazione
   tempo di ritardo (al 50 percento): 17.46 sec
   tempo di salita (dal 10 al 90 percento): 54.04 sec
   tempo di assestamento (al pi/meno 5 percento): 74.09 sec

   ERRORI A REGIME (SOLO AD ANELLO CHIUSO) :
   errore a regime in risposta al gradino: 0
   errore a regime in risposta alla rampa: 25
```

Come primo tentativo si introduce una rete ritardatrice $G_{c1}(s)$ con lo zero vicino al polo $-5$, secondo l'equazione

$$G_{c1} = \frac{1 + \frac{s}{4.8}}{1 + \frac{s}{4.5}} \tag{5.7}$$

Il luogo si modifica secondo la Figura 5.17.

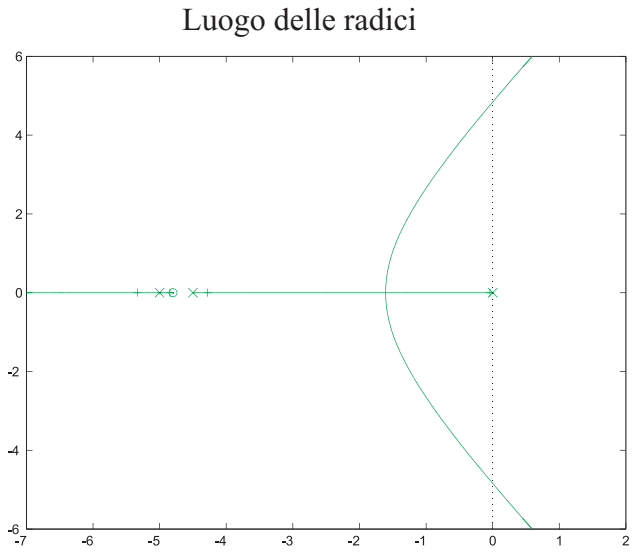



Figura 5.17: Luogo delle radici per $G_{c1}G_p(s)$.

La specifica di $S < 1\%$ richiede, se il sistema fosse del secondo ordine e senza zeri, un $0.85 < \delta < 1$. Utilizzando lo stesso luogo delle radici con le regioni a $\delta$ costante, si ottiene un valore di $K$ circa uguale a 21. In corrispondenza di tale valore si verificano le specifiche relative alla risposta al gradino per il sistema $G_1(s) = 21G_p(s)G_{c1}(s)$.

```
> G1=21*G1

G1=21*G1
........
--------------------
.....................

     19.69 (s + 4.8)
G1 = --------------------
     s (s + 4.5) (s + 5)^2
```

Si utilizza la funzione `tresp` applicata a $G_1(s)$ e si ottengono i seguenti risultati

```
RISPOSTA AL GRADINO :
   massima sovraelongazione: 0.1618 percento per t=4.313 sec
   tempo di ritardo (al 50 percento): 1.075 sec
   tempo di salita (dal 10 al 90 percento): 1.707 sec
   tempo di assestamento (al piu'/meno 5 percento): 2.487 sec

   ERRORI A REGIME (SOLO AD ANELLO CHIUSO) :
   errore a regime in risposta al gradino: 0
   errore a regime in risposta alla rampa: 1.19
```

corrispondenti alla risposta di Figura 5.18. Le caratteristiche della risposta al gradino soddisfano le specifiche richieste.

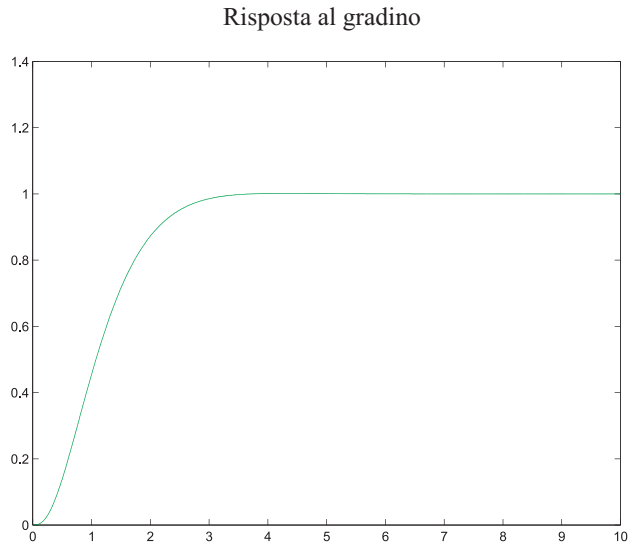



Figura 5.18: Risposta al gradino per il sistema $G_1(s) = 21G_{c1}(s)G_p(s)$.

Una seconda possibile rete correttrice è ottenuta dalla funzione di trasferi-

mento $G_{c2}(s)$

$$G_{c2}(s) = \frac{1 + \frac{s}{9}}{1 + \frac{s}{8}} \tag{5.8}$$

secondo cui, il luogo delle radici della rete complessiva $G_{c2}(s)G_p(s)$ sono rappresentati nella Figura 5.19

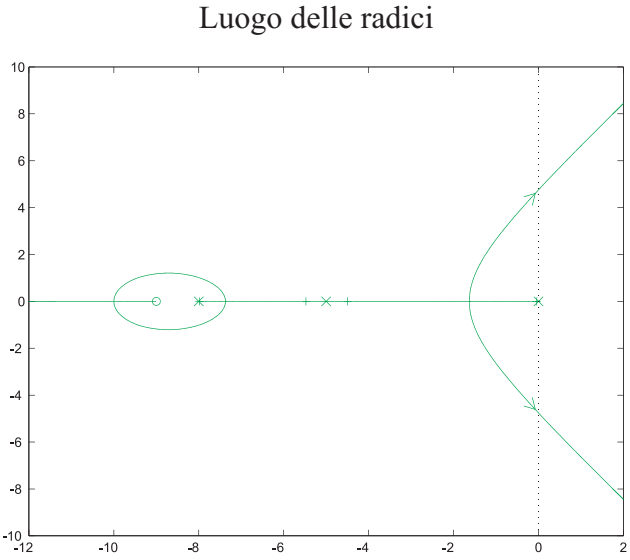



Figura 5.19: Luogo delle radici della funzione $G_{c2}(s)G_p(s)$.

precedentemente introdotta in *TFI*

```
> Gc2=(1+s/9)/(1+s/8)

Gc2=(1+s/9)/(1+s/8)
...................
--------------------
...................

      0.8889 (s + 9)
Gc2 = --------------
      (s + 8)
>
```

e la funzione complessiva risulta

```
> G2=Gc2*Gp

G2=Gc2*Gp
.........
--------------------
.....................
```

```
     0.8889 (s + 9)
G2 = ------------------
     s (s + 5)^2 (s + 8)
 >
```

Disegnando il luogo a $\delta$ costante si può determinare il valore di primo tentativo per $K$. Tale diagramma è rappresentato in Figura 5.20

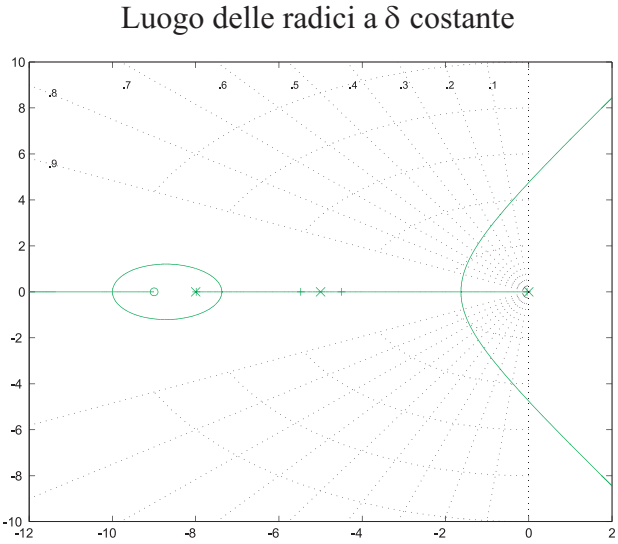



Figura 5.20: Luogo delle radici a $\delta$ costante per la funzione $G_{c2}(s)G_p(s)$.

Si determina un valore iniziale di 20 che porta alle seguenti prestazioni

```
RISPOSTA AL GRADINO :
   massima sovraelongazione: 0.03215 percento per t=5.246 sec
   tempo di ritardo (al 50 percento): 1.111 sec
   tempo di salita (dal 10 al 90 percento): 1.819 sec
   tempo di assestamento (al piu'/meno 5 percento): 2.662 sec

   ERRORI A REGIME (SOLO AD ANELLO CHIUSO) :
   errore a regime in risposta al gradino: 0
   errore a regime in risposta alla rampa: 1.25
```

Non essendo rispettata la specifica del tempo di assestamento ($< 2.5$ secondi), $K$ deve essere modificato per tentativi fino al valore 22, al fine di raggiungere la prestazione richiesta. Si ottiene perciò la seguente risposta al gradino della rete rappresentata in Figura 5.21.
La rete complessiva ha funzione di trasferimento

```
      19.56 (s + 9)
>G2 = ------------------
      s (s + 5)^2 (s + 8)

>
```

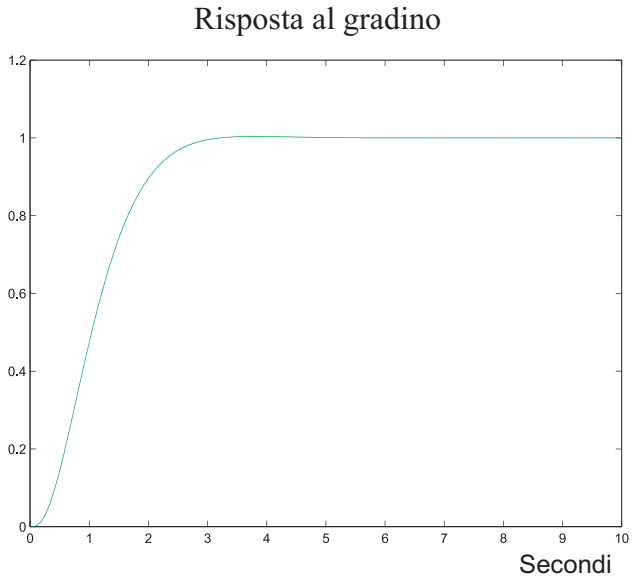



Figura 5.21: Risposta al gradino per la funzione $KG_{c2}(s)G_p(s)$.

le cui prestazioni al gradino risultano

```
RISPOSTA AL GRADINO :
   massima sovraelongazione: 0.3624 percento per t=3.797 sec
   tempo di ritardo (al 50 percento): 1.041 sec
   tempo di salita (dal 10 al 90 percento): 1.598 sec
   tempo di assestamento (al pi/meno 5 percento): 2.328 sec

   ERRORI A REGIME (SOLO AD ANELLO CHIUSO) :
   errore a regime in risposta al gradino: 0
   errore a regime in risposta alla rampa: 1.136
```

che verificano quelle assegnate dall'esercizio.

Per finire, si possono confrontare le prestazioni delle due reti correttrici rappresentate dalle funzioni (5.7) e (5.8). Si utilizza la funzione `fresp`, per determinare le caratteristiche della risposta in frequenza dei sistemi compensati. Per la prima rete (Equazione 5.7) si ha

```
RISPOSTA IN FREQUENZA AD ANELLO APERTO :
   margine di ampiezza:  11.55 (21.25 db) per omega= 4.842 rad/sec
   margine di fase: 70.82 gradi per omega= 0.8167 rad/sec
   ascissa dell'asintoto verticale del diagramma polare: -0.3477

   RISPOSTA IN FREQUENZA AD ANELLO CHIUSO :
   risonanza ad anello chiuso non determinabile
   guadagno statico ad anello chiuso: 1 (0 db)
   banda passante (-3db): 1.259 rad/sec
```

mentre per la seconda (Equazione 5.8),

```
RISPOSTA IN FREQUENZA AD ANELLO APERTO :
   margine di ampiezza:  10.58 (20.49 db) per omega= 4.755 rad/sec
   margine di fase: 69.94 gradi per omega= 0.8541 rad/sec
   ascissa dell'asintoto verticale del diagramma polare: -0.3642

   RISPOSTA IN FREQUENZA AD ANELLO CHIUSO :
   risonanza ad anello chiuso non determinabile
   guadagno statico ad anello chiuso: 1 (0 db)
   banda passante (-3db): 1.344 rad/sec
```

La seconda rete, avendo il polo più negativo, corrisponde ad un sistema ad anello chiuso "più pronto", cioé con una banda passante maggiore.

## 5.5 Esercizi proposti in aula didattica.

Un motore in corrente continua è rappresentato dalla seguente funzione di trasferimento

$$G_m(s) = \frac{6.087 \times 10^{10}}{s(s^3 + 423.42s^2 + 2.6667 \times 10^6 s + 4.2342 \times 10^8)} \tag{5.9}$$

A causa dell'accoppiamento con l'albero motore, tale funzione contiene due poli poco smorzati che generano oscillazioni nella risposta. Devono essere soddisfatte le seguenti specifiche

- Massima elongazione $< 1\%$,
- Tempo di salita $< 0.15$ secondi,
- Tempo di assestamento $< 0.15$ secondi,
- La risposta non deve presentare oscillazioni.

Progettare una rete anticipatrice con funzione di trasferimento $G_c(s)$

$$G_c(s) = \frac{1 + \alpha\tau s}{1 + \tau s} \tag{5.10}$$

affinché risultino verificate le specifiche relative alla risposta al gradino.

# Capitolo 6

# Sintonizzazione di Controllori PID

Lo schema del controllo in retroazione di Figura 6.1 è comunemente usato nelle applicazioni industriali della teoria del controllo automatico. L'obiettivo dello schema consiste nel forzare la *variabile controllata* $y(t)$ (l'uscita di un certo sistema da controllare) a seguire il più fedelmente possibile una *variabile di riferimento* $r(t)$ definita da un generatore di riferimento [8].

Il sistema di controllo viene progettato per raggiungere questa specifica tenendo conto di un modello matematico del sistema e dell'attuatore, vale a dire del sistema di potenza che trasforma il *segnale di controllo* $v(t)$ generato dal controllore in un *segnale di potenza* $u(t)$ in grado agire sul sistema da controllare.

Il sistema di controllo acquisisce la misura della variabile da controllare tramite un sensore che fornisce una variabile di misura $\tilde{y}(t)$ proporzionale alla variabile misurata $y(t)$, quindi confronta tale valore con il riferimento $r(t)$, ottenendo una variabile errore $e(t)$. La variabile errore viene infine elaborata dal controllore per calcolare un opportuno valore per il segnale di controllo $v(t)$.

Nelle applicazioni industriali sono spesso usati controllori con una struttura fissa costituita da un termine proporzionale (P), un termine integrale (I) e uno derivativo (D). Tali controllori, detti regolatori standard o PID, sono particolarmente apprezzati per la loro semplicità ed efficacia. In questi appunti verrà presentata e discussa la struttura di base di questi regolatori, mostrando metodi analitici di sintesi e affinamenti della struttura di base necessari nelle applicazioni pratiche.

Nel corso della trattazione lo schema di controllo in retroazione di Figura 6.1 potrà essere considerato in forma semplificata per favorire la comprensione dei concetti esposti. In particolare potrà essere trascurata la presenza dell'attuatore ($v(t) = u(t)$), ovvero trascurata la presenza del sensore ($\tilde{y}(t) = y(t)$).

## 6.1 Struttura di un PID

Il regolatore PID è un sistema dinamico che elabora il segnale di ingresso errore come differenza fra il riferimento e la variabile controllata $e(t) = r(t) - y(t)$ per ottenere un segnale di controllo $u(t)$ fornito in ingresso al sistema da controllare.

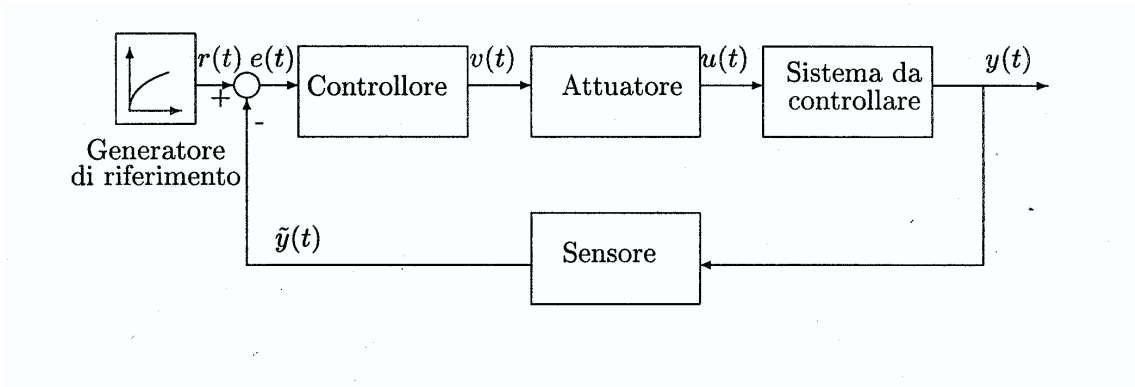



Figura 6.1: Diagrammi a blocchi di un sistema di controllo.

Il regolatore PID nella sua forma base, ha una struttura fissa ed è parametrizzabile da tre coefficienti $K, T_i, T_d$:

$$u(t) = K\left(e(t) + \frac{1}{T_i}\int_0^t e(\tau)d\tau + T_d\frac{de(t)}{dt}\right) \tag{6.1}$$

Analizziamo separatamente nel seguito il significato di questi tre termini.

1. In termini statici, un $K$ elevato riduce l'errore a regime, diminuisce eventuali effetti di disturbo e del rumore sulla variabile controllata. Dal punto di vista dinamico, l'introduzione di un termine proporzionale (P) nell'anello di controllo generalmente produce un incremento della larghezza di banda del sistema retroazionato, per cui si ha un *aumento* della velocità di risposta del sistema, e quindi una maggiore prontezza nell'inseguimento del riferimento, ma, allo stesso tempo, una *diminuzione* dei margini di stabilità del sistema.

2. Dal punto di vista statico, la funzione principale del termine integrale consiste nell'annullare l'errore a regime per un ingresso a gradino. Infatti tale termine integra l'errore nel tempo e quindi un errore costante provoca un incremento dell'azione di controllo fino a che l'errore non si è completamente annullato. In termini dinamici, l'introduzione del termine integrale porta ad un aumento del ritardo di fase della catena diretta di controllo pari a $\pi/2$, e quindi determina un peggioramento dei margini di fase ed ampiezza del sistema retroazionato.

3. La funzione principale del termine derivativo consiste nel migliorare i margini di stabilità del sistema fornendo un anticipo di fase all'anello di controllo in retroazione. D'altra parte, ll termine derivativo ha la caratteristica di amplificare i segnali con contenuto armonico a frequenze elevate.

## 6.2 Modifiche alla struttura del PID

Nel paragrafo 6.1 è stata introdotta la struttura di base del regolatore PID. Il regolatore PID utilizzato nelle applicazioni industriali ha una struttura più complessa in quanto vi sono problemi di natura pratica che non possono venire risolti utilizzando semplicemente la (6.1). In questo capitolo vedremo una riformulazione della (6.1) utilizzata in applicazioni industriali e l'introduzione di un dispositivo di anti-saturazione del termine integrale nella legge di controllo.

### 6.2.1 Limitazione di banda del termine derivativo

Il termine derivativo del regolatore PID migliora i margini di stabilità dell'anello di controllo ed introduce una azione di correzione dell'errore di tipo anticipativo. Dal punto di vista applicativo vi sono però alcuni importanti aspetti da valutare:

- Il termine derivativo amplifica ed esalta i segnali a larga banda come il rumore elettromagnetico sulla misura.
- La funzione di trasferimento corrispondente al termine derivativo del PID è non propria e quindi non fisicamente realizzabile.
- Il contributo del termine derivativo diviene teoricamente infinito nel caso in cui venga applicato un ingresso di riferimento $r(t)$ a gradino, introducendo sollecitazioni potenzialmente dannose per gli organi di attuazione.

Per risolvere i problemi citati è possibile filtrare con un sistema del primo ordine l'uscita del termine derivativo, sostituendo quindi tale contributo con un termine con funzione di trasferimento propria:

$$D(S) = \frac{sKT_d}{1 + sT_d/N} \tag{6.2}$$

dove la costante di tempo del filtro $(T_d/N)$ è solitamente fissata tra un decimo ed un ventesimo della costante di tempo del termine derivativo $T_d$.

Per evitare che segnali di riferimento a banda troppo larga, ad esempio un segnale a gradino, producano delle sollecitazioni troppo forti, è possibile modificare lo schema di base del PID (6.1) come mostrato in Figura 6.2. In tale schema viene fornito all'ingresso del termine derivativo la sola misura dell'uscita $y(t)$.

### 6.2.2 "Anti-Windup" del termine integrale.

Gran parte dello studio dei sistemi dinamici viene condotto utilizzando l'approssimazione di linearità del sistema in esame, sia pure nell'intorno di un punto di lavoro. Una delle principali cause di non linearità consiste nella limitazione fisica degli attuatori: la coppia fornita da un motore elettrico è limitata, una valvola non può essere più che completamente chiusa o completamente aperta.

Questo tipo di non-linearità può incidere profondamente sul comportamento del controllore PID. Infatti consideriamo lo schema di figura 6.3,
A causa della saturazione dell'attuatore, il valore in uscita dal regolatore PID $(v(t))$ può essere differente da quello della variabile di controllo $(u(t))$. Vi può

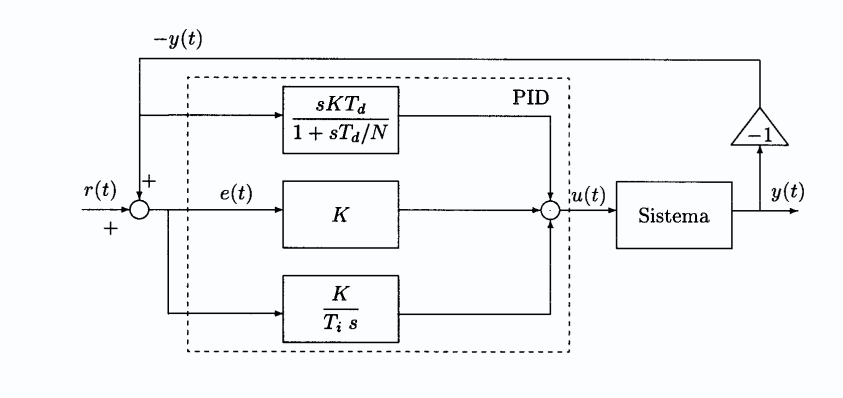



Figura 6.2: Schema a blocchi del PID con limitazione di banda del termine derivativo.

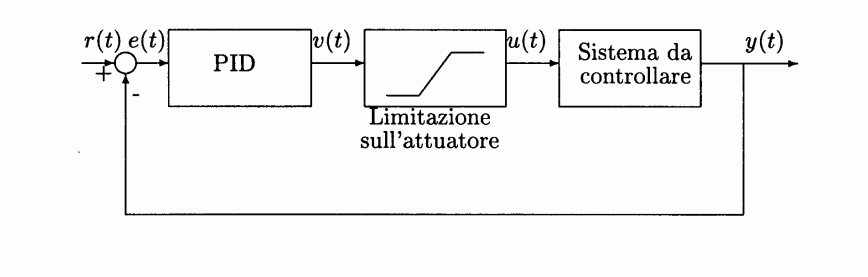



Figura 6.3: Schema a blocchi di un sistema di controllo con limitazione sull'attuatore.

essere saturazione sul valore della variabile di controllo ovvero sulla sua derivata temporale. Tali saturazioni possono venire espresse dalle relazioni:

$$u(t) = \begin{cases} u^+; \text{ if } v(t) \geq u^+ \\ v(t); \text{ if } u^- < v(t) < u^+ \\ u^-; \text{ if } v(t) \leq u^- \end{cases} \tag{6.3}$$

$$\frac{du(t)}{dt} = \begin{cases} d^+; \text{ if } \frac{dv(t)}{dt} \geq d^+ \\ \frac{dv(t)}{dt}; \text{ if } d^- < \frac{dv(t)}{dt} < d^+ \\ d^-; \text{ if } \frac{dv(t)}{dt} \leq d^- \end{cases} \tag{6.4}$$

dove $u^-$, $u^+$, $d^-$ e $d^+$ sono, rispettivamente, il limite minimo e massimo dell'uscita e della derivata temporale dell'uscita dell'attuatore.

Per capire come la saturazione dell'attuatore influisce sull'azione di controllo del sistema, consideriamo il seguente caso: supponiamo che il sistema si trovi in uno stato di equilibrio (errore nullo). Ad un certo istante viene applicato un gradino al segnale di riferimento $r(t)$, che sollecita il controllore a fornire un gradino al segnale di controllo $v(t)$. A causa della saturazione il segnale di

attuazione $u(t)$ è più basso di $v(t)$ e quindi la variazione del segnale controllato, e quindi dell'errore, sarà inferiore rispetto al caso non limitato. A causa della ridotta velocità della decrescita del segnale di errore, il termine integrale aumenta molto di più rispetto al caso privo di limitazione. Quando poi l'errore tende ad annullarsi, il segnale di controllo rimane alto a causa dell'elevato valore dell'integrale, causando quindi una elevata sovraelongazione e un tempo di assestamento più elevato. Questo fenomeno viene chiamato *windup* del PID.

Nel seguito verrano mostrati alcuni schemi per attenuare il problema della saturazione del termine integrale (dispositivi di "anti wind-up").

### Ricalcolo del termine integrale.

Questa tecnica introduce un termine di compensazione $e_s(t) = v(t) - u(t)$ all'ingresso del termine integrale, con segno opposto a quello del segnale di errore $e(t)$. Il segnale di compensazione è nullo fino a che l'uscita del PID non è saturata, mentre fornisce un'azione tendente a diminuire il termine integrale nel caso in cui l'uscita del PID si saturi. In Figura 6.4 sono mostrati due schemi di antisaturazione: il primo utilizza una misura diretta della variabile di controllo $u(t)$, il secondo impiega una stima di $u(t)$ ottenuta tramite un modello dell'attuatore.

Il segnale di compensazione del termine integrale viene filtrato da una funzione di trasferimento $F(s)$ tramite cui è possibile modulare l'azione di desaturazione. Siccome il temine di compensazione entra in un integratore, solitamente la funzione di traferimento del filtro $F(s)$ è semplicemente una costante $1/T_a$.

Per scegliere correttamente $T_a$ occorre considerare che per $T_a$ molto piccoli l'azione di desaturazione è molto veloce, però è anche facilmente soggetta all'azione di disturbi, se $T_a$ è grande l'azione di desaturazione è lenta.

### Integrazione condizionata

In questa tecnica l'ingresso del termine integrale viene azzerato nel caso in cui il segnale di compensazione $e_s(t) = v(t) - u(t)$ sia non nullo (vedi figura 6.5). Se chiamiamo $e_i(t)$ il valore di ingresso del termine integrale, allora l'integrazione condizionata si esprime matematicamente:

$$e_i(t) = \begin{cases} e(t), & \text{se } v(t) = u(t) \\ 0, & \text{se } v(t) \neq u(t) \end{cases} \tag{6.5}$$

## 6.3 Funzioni e modelli usati nel capitolo

In questo capitolo verranno utilizzati i seguenti files *Maltab* e *Simulink*:

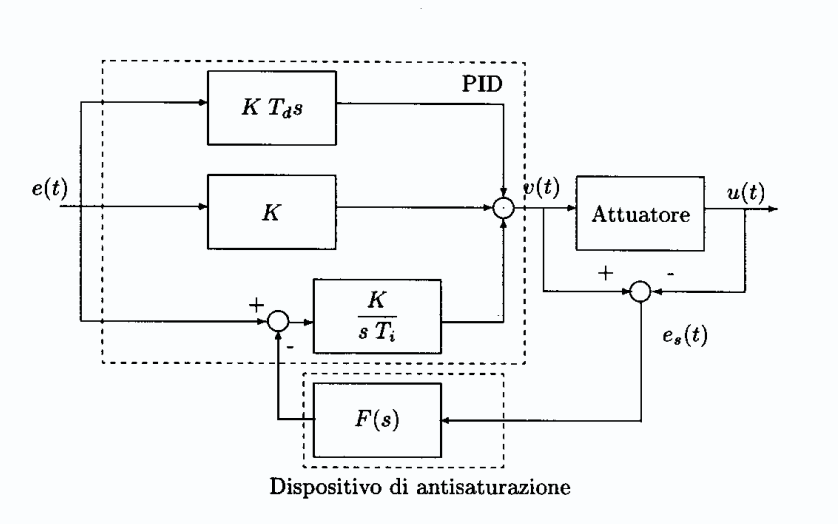
`sim1 pid.mdl`, sistema in retroazione con PID.

`PIDsat.mdl`, *subsystem* PID con saturazione.

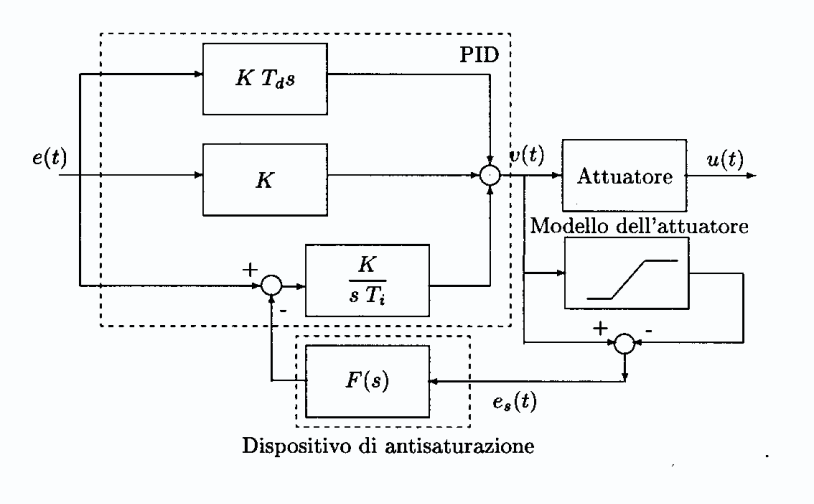
`es1 pid.mdl`, sistema PID e serbatoio.

`es2 pid.mdl`, sistema PID e serbatoio.

`es3 pid.mdl`, sistema PID anti-windup e serbatoio.




(a) Schema con misura della variabile di attuazione.




(b) Schema com stima della variabile di attuazione.

Figura 6.4: Controllore PID con dispositivo di anti-saturazione del termine integrale mediante ricalcolo del termine integrale.

`data pid.mat`, dati di ingresso per il serbatoio.

`id1 pid.mdl`, identificazione della funzione di trarsferimento.

`ziegler.m`, formule di Ziegler-Nichols.

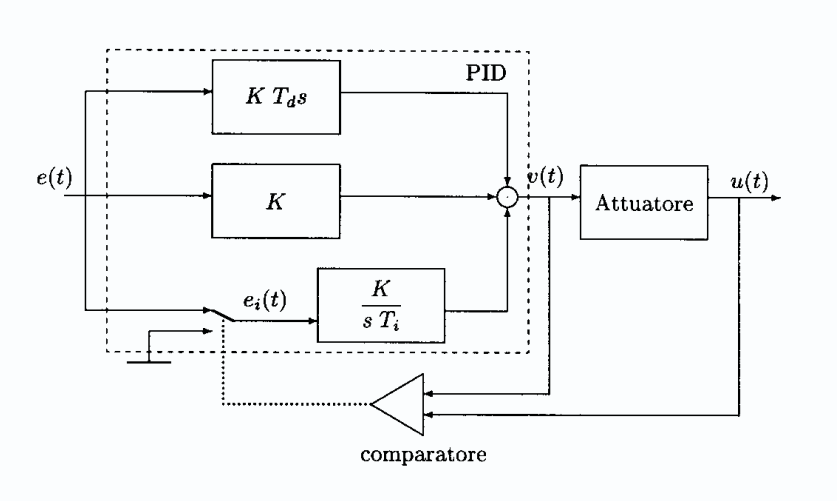



Figura 6.5: Schema di antisaturazione mediante integrazione condizionata.

`serbatoio.mdl`, modello del serbatoio.

`sim2 pid.mdl`, schema *Simulink* per l'esercizio 1).

`sim3 pid.mdl`, schema *Simulink* per l'esercizio 2).

`sim4 pid.mdl`, schema *Simulink* per l'esercizio 3).

## 6.4 Controllo di livello di un serbatoio

In questo paragrafo si presenta l'esempio relativo ad un controllo di livello effettuato attraverso un regolatore PID e progettato attraverso specifiche sulla precisione statica.

Si fa riferimento al serbatoio di sezione circolare pari ad $A$, con $A = 10m^2$ rappresentato in Figura 6.6 con un condotto di ingresso per introdurre liquido con portata $Q_i$ $[\frac{m^3}{s}]$ ed uno di uscita, per prelevarne con portata $Q_o$ $[\frac{m^3}{s}]$.

Il sistema dinamico può essere rappresentato secondo uno schema *Simulink* come in Figura 6.7, in cui la portata netta $Q_n$ viene divisa per la sezione del serbatoio ed integrata, per ottenere l'altezza del liquido rispetto il fondo, $h$.

Lo schema del sistema chiuso in retroazione attraverso un controllore PID è rappresentato in Figura 6.8. A valle del PID è stata inserita una saturazione, che limita l'uscita del PID da 0 a 0.2. L'immissione di liquido viene infatti controllata da una valvola rappresentata nello schema di Figura 6.7 e la saturazione tiene conto appunto del limite in portata dovuto alla valvola stessa.
In riferimento allo schema *Simulink* di Figura 6.8, con una portata in uscita pari a $0.05\frac{m^3}{s}$, un riferimento di livello fissato a $9m$ e un livello iniziale pari a $10m$

- progettare un PID con il solo termine proporzionale $K$ tale da garantire un errore di livello inferiore a $0.005m$.

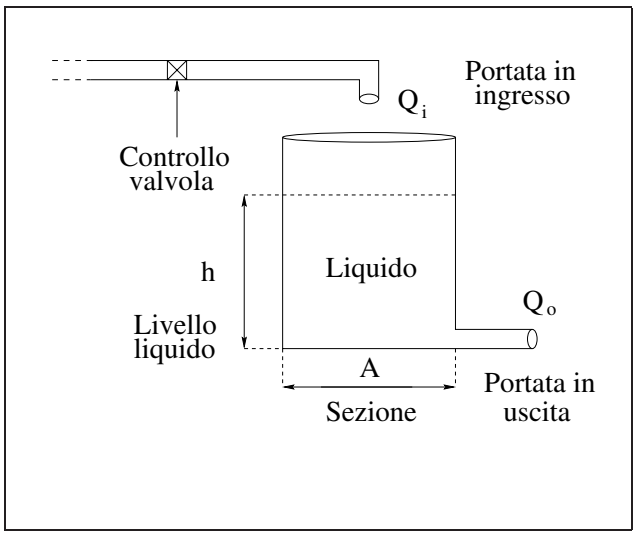



Figura 6.6: Rappresentazione del sistema controllato.

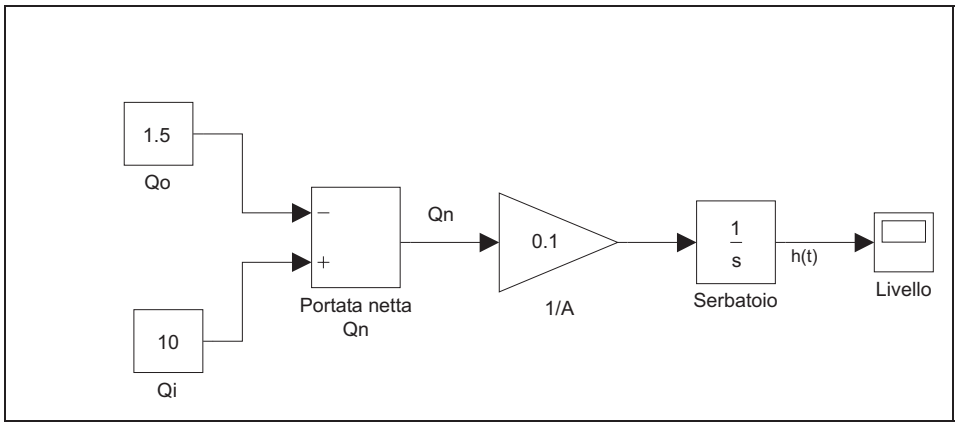



Figura 6.7: Schema *Simulink* del sistema controllato.

- introdurre il termine integrale per annullare l'errore a regime.

Come può essere ricavato dallo schema di Figura 6.9, per avere un errore a regime inferiore allo $0.005m$ occorre un guadagno superiore a 10.

Tale guadagno garantisce che il sistema lavori in regime lineare, cioé l'uscita del PID non è soggetta alle limitazioni imposte dalla saturazione. Nelle Figure 6.10 sono rappresentati il livello di riferimento $h_{rif} = 9m$ e il livello del serbatoio $h(t)$.

Si osservi l'andamento del segnale di controllo all'uscita del regolatore riportato in Figura 6.11.

Per ottenere l'annullamento dell'errore a regime, si introduce il termine in-

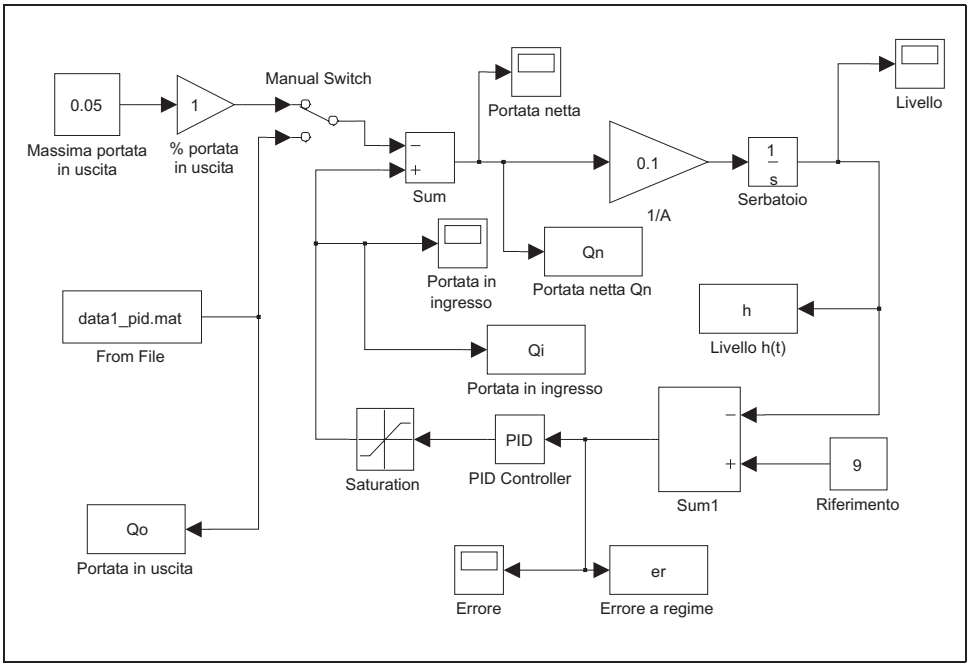



Figura 6.8: Schema *Simulink* del controllo di livello con PID.

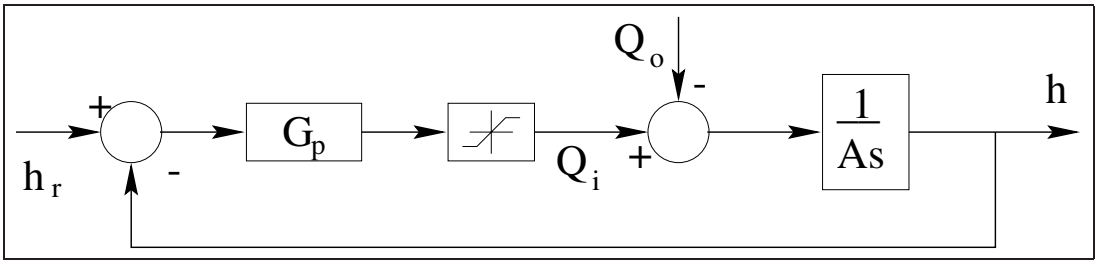


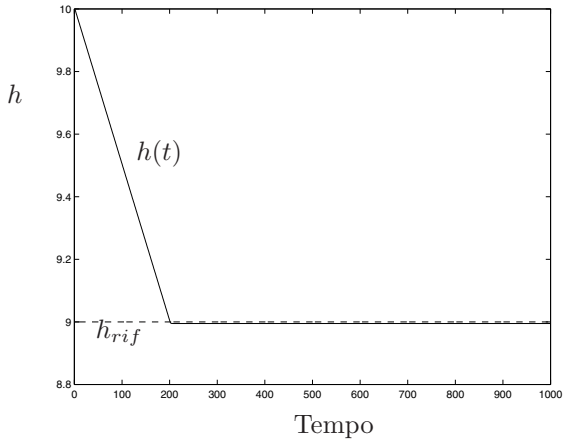

Figura 6.9: Schema a blocchi del controllo di livello con PID.

tegrale con un guadagno pari a 0.2. Anche questo valore evita l'intervento della saturazione.

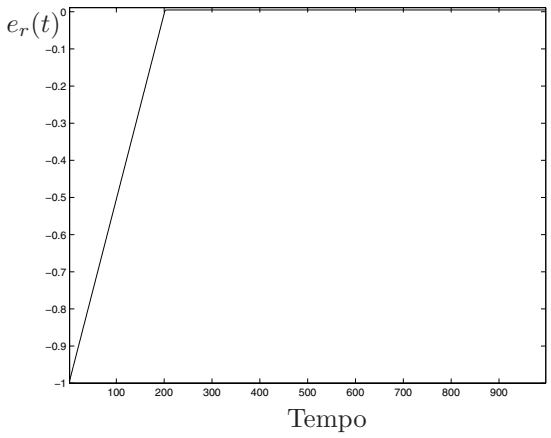
Il nuovo regolatore di tipo PI permette di ottenere un errore a regime nullo come evidenziato in Figura 6.12. L'andamento del segnale $Q_i$ ottenuto attraverso un controllore di tipo PI è riportato in Figura 6.13

## 6.5 Progetto di un PID con le formule di Ziegler-Nichols

In questo paragrafo si effettuerà il progetto di un PID per un sistema descritto da una funzione di trasferimento utilizzando le relazioni di Ziegler-Nichols




(a)




(b)

Figura 6.10: Livello del serbatoio e livello di riferimento (a) ed errore a regime (b) con regolatore P.

(modificate da Cohen e Coon) calcolate in base alla funzione di trasferimento approssimata.
Si consideri il sistema del quarto ordine descritto dalla funzione di trasferimento

$$G(s) = \frac{1}{(1+s)(1+0.25s)(1+0.5s)(1+0.75s)} \tag{6.6}$$

che può essere approssimata, utilizzando il *metodo delle aree*, dalla funzione

$$G_a(s) = \frac{e^{-1.0652s}}{(1+1.4348s)}. \tag{6.7}$$

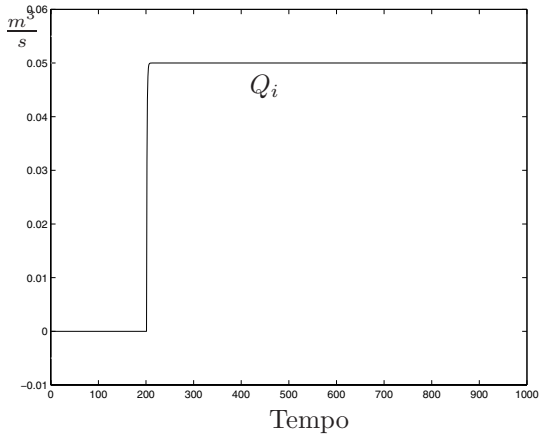


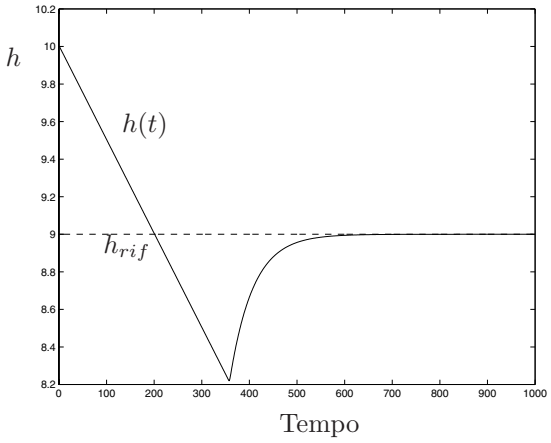

Figura 6.11: Portata in ingresso al serbatoio.

in cui sono stati sostituiti i valori di $\tau = 1.0652$ e $T = 1.4348$. Le relazioni di Ziegler-Nichols sono implementate nella funzione *Matlab* `ziegler.m` che richiede in ingresso $\tau$, $T$, $C_o$, il valore di regime della risposta del sistema eccitato da un gradino di ampiezza $M_o$.

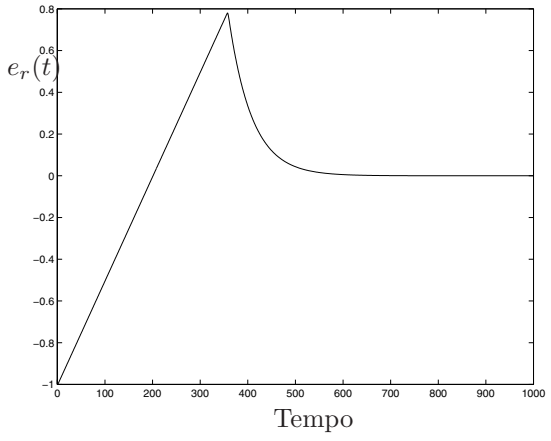
```
function [P,I,D] = ziegler(tau,T,Mo,Co)
%
% function [P,I,D] = ziegler(tau,T,Mo,Co) per il calcolo dei parametri
% del PID. tau e' il tempo di ritardo, T la costante di tempo, Mo l'ampiezza
% del gradino applicato e Co l'ampiezza a regime della risposta.
%

R = tau/T;
Mo = 1;
Co = 1;
N = Co/T;

P = Mo*(4/3+R/4)/(N*tau);
I = (tau*(32+6*R))/(13+8*R);
D = (4*tau)/(11+2*R);
```

Si ottengono i seguenti valori per i parametri del PID

```
>> P

P =

    2.0460

>> I

I =
```




(a)




(b)

Figura 6.12: Livello del serbatoio e livello di riferimento (a) ed errore a regime (b) con regolatore PI.

```
    2.0503

>> D

D =

    0.3413

>>
```

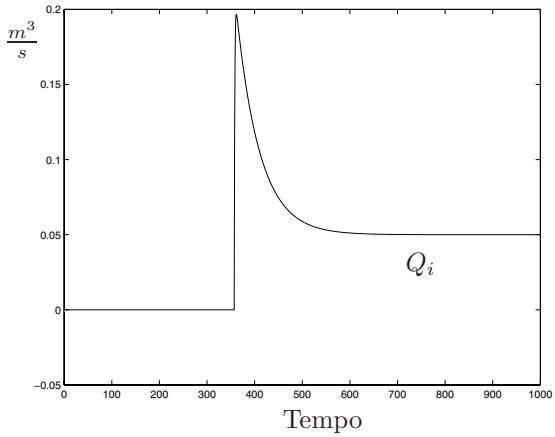



Figura 6.13: Portata in ingresso al serbatoio con regolatore PI.

Il modello *Simulink* per l'identificazione della funzione di trasferimento $G(s)$ con la $G_a(s)$ è riportato in Figura 6.14

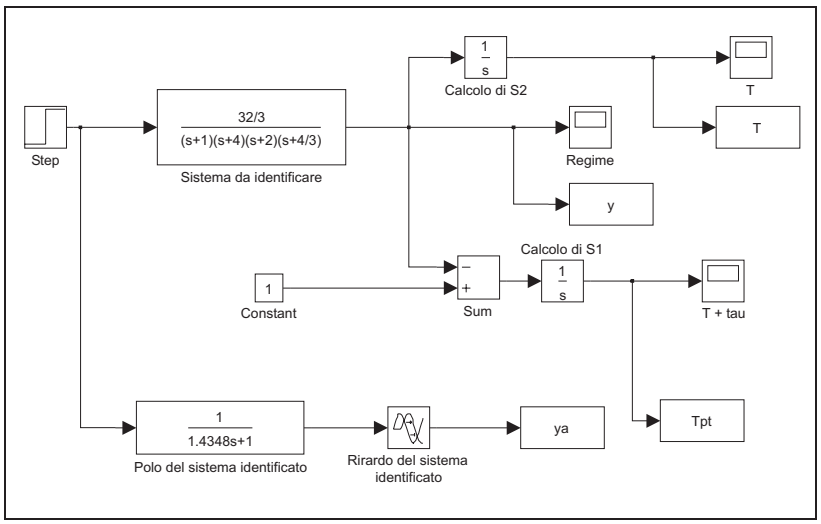



Figura 6.14: Modello *Simulink* per l'identificazione di $G(s)$.

mentre il confronto tra la risposta del sistema reale e quella del sistema approssimato è riportato in Figura 6.15.
In Figura 6.16 viene mostrata l'implementazione di un PID in ambiente *Simulink*, modificata introducendo il blocco di saturazione.

Si utilizza questa versione di PID, in cui il blocco di saturazione viene escluso mettendo rispettivamete come limiti superiore ed inferiore i valori di `+inf` e `-inf`. Con tali parametri il funzionamento del blocco in Figura 6.16 è equivalente a quello del PID semplice già presente nelle librerie di *Simulink*.

Il blocco *Subsystem* PID viene inserito nello schema di Figura 6.17.

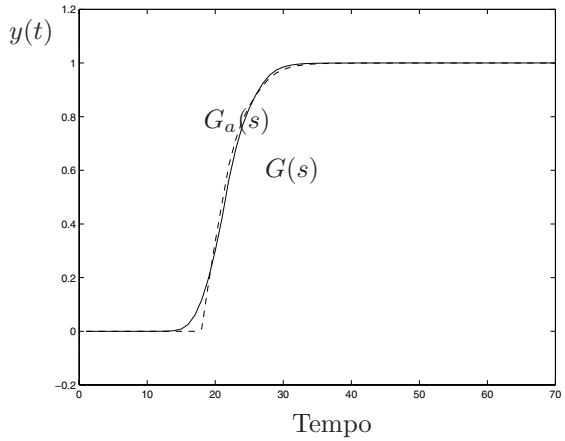



Figura 6.15: Confronto delle risposte al gradino di $G(s)$ e $G_a(s)$.

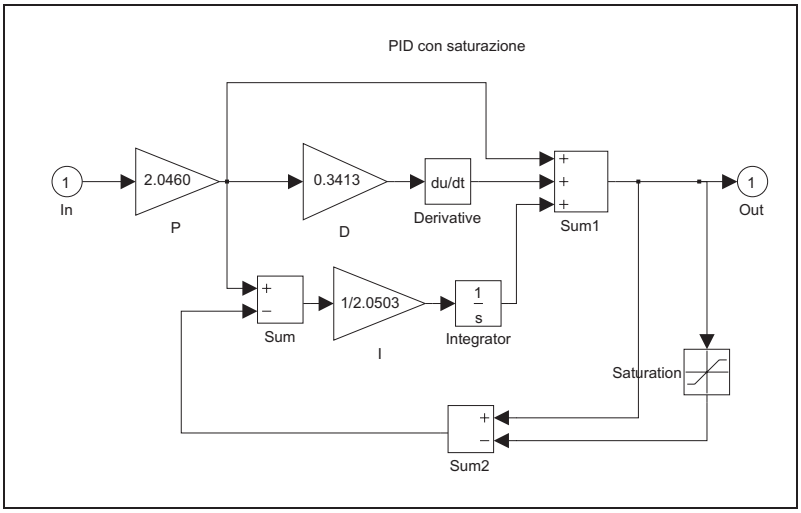



Figura 6.16: Subsystem PID con saturazione in ambiente *Simulink*.

Si confrontano quindi le risposte del sistema controllato senza regolatore e sistema in retroazione col PID. La Figura 6.18 riporta le risposte $y(t)$ del sistema $G(s)$ e $y_c(t)$ del sistema compensato $G_c(s)$.

In Figura 6.18 sono stati anche evidenziati i livelli di $10\% \div 90\%$ e $\pm 5\%$ rispettivamente per la definizione del tempo di salita e il tempo di assestamento.

Come si può osservare dalla Figura 6.18, l'introduzione del PID peggiora le caratteristiche dinamiche della risposta, aumentando la sovraelongazione, ma migliora la velocità della risposta stessa, diminuendo il tempo di salita e lasciando invariato il tempo di assestamento.

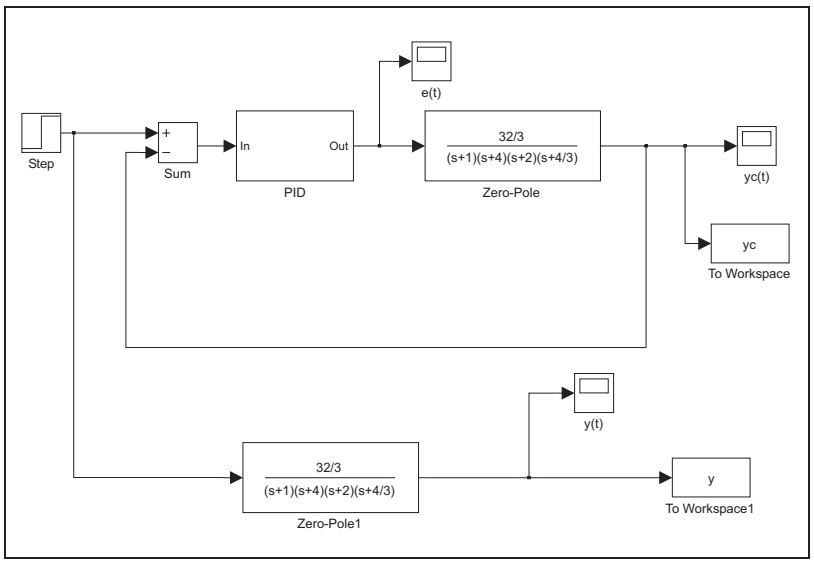



Figura 6.17: Il sistema controllato $G(s)$ in retroazione con il PID.

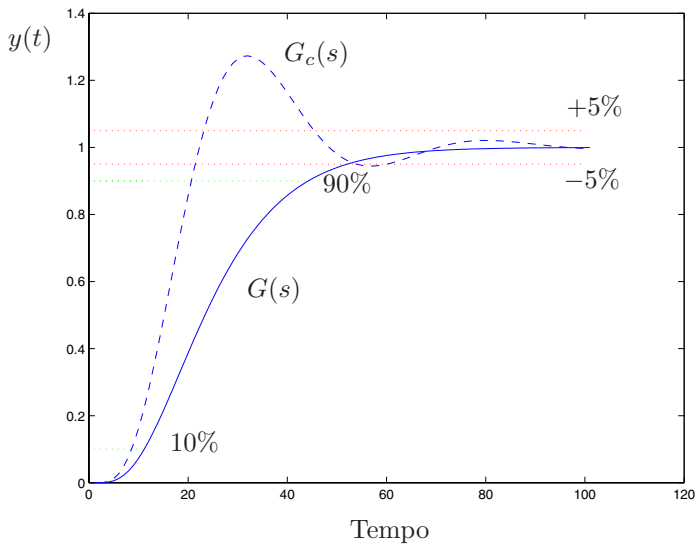
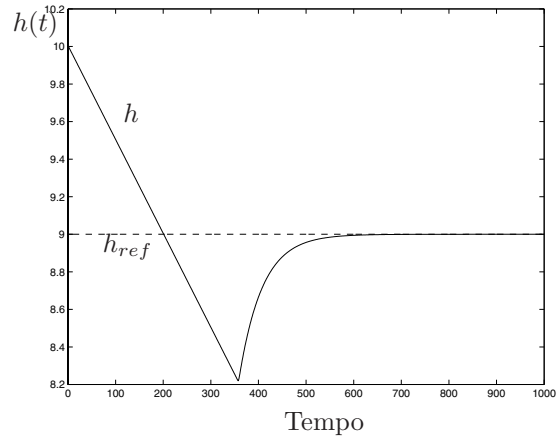


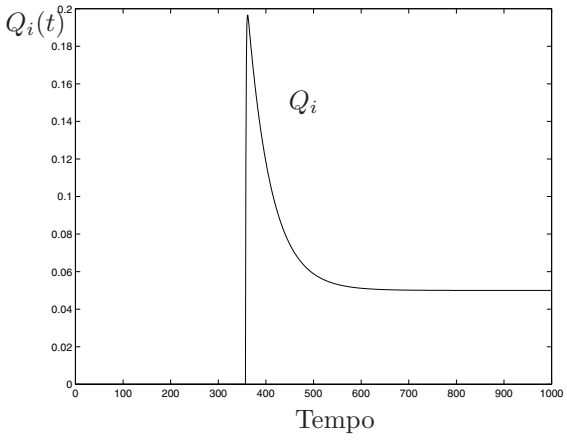

Figura 6.18: Risposte del sistema con e senza compensazione e livelli di riferimento.

## 6.6 PID con schema anti-windup

Si consideri lo schema del serbatoio per il controllo di livello utilizzato nel Paragrafo 6.4 in retroazione con un PI classico con $K_p = 10$ e $T_i = 0.2$. Imponendo un prelievo costante al serbatoio si ottengono i seguenti andamenti del livello del serbatoio $h(t)$ e della portata di ingresso $Q_i$, riportati in Figura 6.19.
Il livello di riferimento è fissato a 9 mentre il serbatoio parte da un livello iniziale




(a)




(b)

Figura 6.19: Livello del serbatoio con PID classico con livello di riferimento (a) e portata in ingresso $Q_i$ (b).

pari a 10.

Successivamente, facendo riferimento ad un PID secondo uno schema riportato in Figura 6.20, con sistema di limitazione della derivata e anti-windup, inserito nel sistema serbatoio-regolatore di Figura 6.21, si ottendono i risultati riportati in Figura 6.22.
Il PID utilizza i seguenti parametri: $K_p = 10$, $T_i = 0.2$, $T_d = 0$, $N = 5$, $T_t = 3$ e come valori di saturazione $[0, 0.2]$.

La Figura 6.22 riporta il livello del serbatoio confrontato con quello di riferimento e l'andamento del flusso di liquido in ingresso.

Si può notare la risposta più veloce del sistema che utilizza un PID con

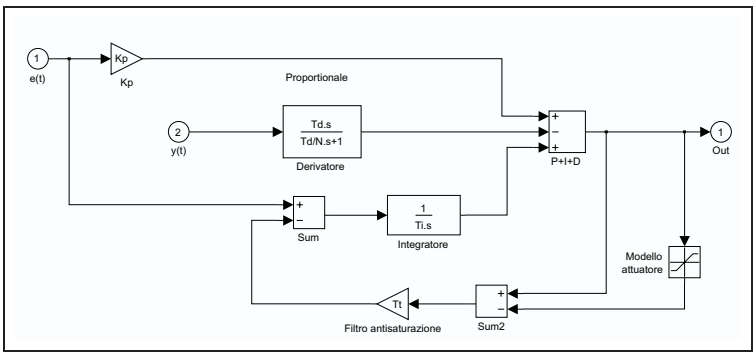



Figura 6.20: PID in *Simulink* a derivata limitata e anti-windup.

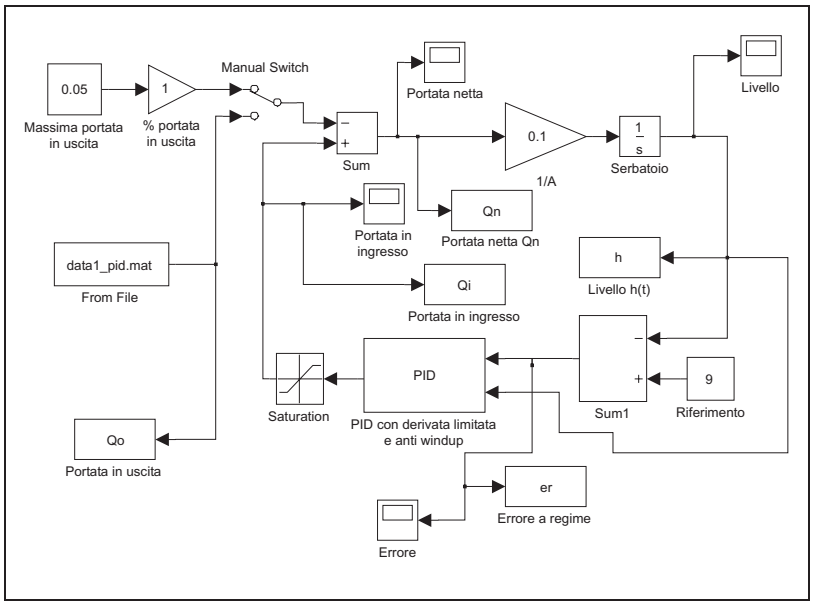
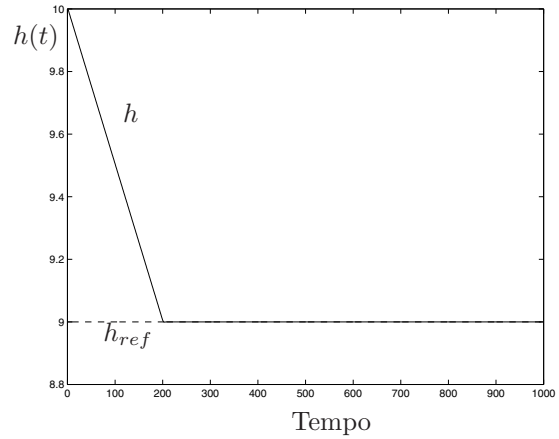
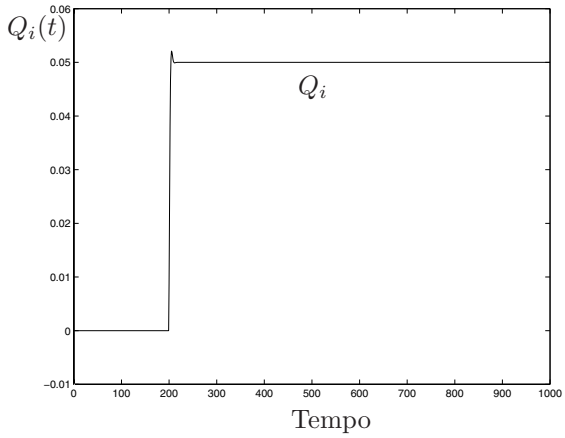



Figura 6.21: Schema *Simulink* di PID modificato e serbatoio.

struttura modificata rispetto a quella di un PID classico. Nel PID classico con saturazione dell'attuatore, infatti, a causa della saturazione del segnale di attuazione e della ridotta velocità della descrescita del segnale di errore, il termine integrale aumenta molto di più rispetto al caso privo di limitazione. Quando poi l'errore tende ad annullarsi, il segnale di controllo rimane alto a causa dell'elevato valore dell'integrale, causando quindi una elevata sovraelongazione e un tempo di assestamento più elevato.




(a)




(b)

Figura 6.22: Livello del serbatoio $h(t)$ con PID modificato con livello di riferimento (a) e portata in ingresso $Q_i$ (b).

## 6.7 Esercizi proposti in aula didattica.

1. Data la funzione di trasferimento

$$\frac{1}{(1+s)^3} \tag{6.8}$$

progettare un regolatore PID utilizzando le formule di Ziegler-Nichols (funzioni del valore del guadagno in retroazione $K_o$ che porta il sistema al limite della stabilità e del periodo delle oscillazioni ottenute $T_o$).

I parametri $K_o$ e $T_o$ possono essere ottenuti dal luogo delle radici del sistema chiuso in retroazione unitaria con un regolatore di tipo P secondo un guadagno proporzionale $K_o$. Con le formule di Ziegler-Nichols progettare quindi un PI, un PID ed un PID con limitazione della derivata (N=10) e confrontare i risultati ottenuti con i diversi regolatori.

2. Riprendendo il sistema descritto dall Equazione 6.8 realizzare un PID (con N=10) con limitazione dell'azione derivativa (cioè che effettui la derivata solo dell'uscita) e si confronti la risposta con quella ottenuta con PID in forma reale realizzato precedentemente.

3. Riprendendo lo schema che utilizza il PID con derivata reale della sola uscita, si introduca un disturbo sull'uscita stessa (rumore bianco con varianza 0.01) e si calcoli l'andamento nel tempo del segnale di controllo per $N = 10$ e $N = 30$. Fare prove con $N$ diverso e giustificare teoricamente i risultati ottenuti.

# Bibliografia


[1] K. Sigmon, *MATLAB Primer*. University of Florida, Florida, Second Edition ed., 1992. (Si scarica dalla rete).

[2] The MathWorks, Inc., *Matlab, The Language of Technical Computing. Getting Started with MATLAB.*, version 5.1 ed., May 1997. (In formato pdf su CD Matlab).

[3] The MathWorks Inc., *Matlab User's Guide*, 1993.

[4] G. F. Franklin, J. D. Powell, and M. Workman, *Digital Control of Dynamic Systems*. Addison-Wesley, Third Edition ed., 1998.

[5] The MathWorks Inc., *Simulink User's Guide*, 1995.

[6] L. F. Shampine and M. W. Reichel, "The Matlab Ode Suite," tech. rep., The MathWorks, Inc, 1997. (Disponibile anche come file in formato pdf).

[7] G. Marro, *TFI: insegnare e apprendere i controlli automatici di base con Matlab*. Bologna: Zanichelli, I ed., Ottobre 1998.

[8] C. Fantuzzi, *Controllori Standard PID*. Versione 1.2, Appunti del Corso, 1a ed., Maggio 1997.