Automatica I (Laboratorio)

di Silvio Simani

Dipartimento di Ingegneria, Università di Ferrara

Versione 1.0, Novembre, 2001

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 1×1, 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 3×3 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.

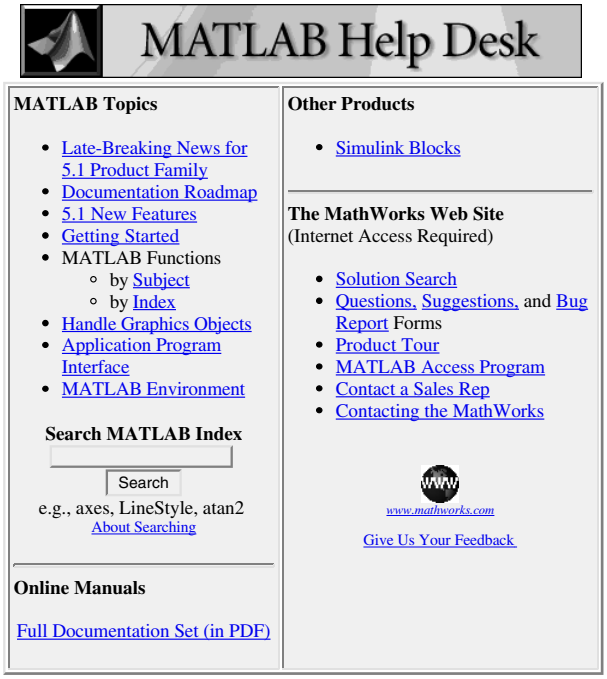


Figura 1.1: Helpdesk di *Matlab*

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.

Capitolo 2

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.

2.1 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.

2.2 Analisi di un circuito non lineare.

Si consideri il circuito rappresentato nella Figura 2.1 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.

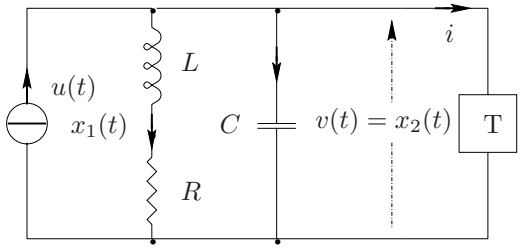


Figura 2.1: Circuito non lineare.

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

$$\begin{aligned} \dot{x}_1(t) &= -\frac{R}{L}\, x_1(t) + \frac{1}{L}\, x_2(t) \\ \dot{x}_2(t) &= -\frac{1}{C}\, x_1(t) + \frac{1}{C}\big(G_1 x_2(t) - G_2 x_2^3(t)\big) + \frac{1}{C} u(t) \end{aligned} \tag{2.1}$$

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

$$\bar{x}' = \begin{bmatrix} 0 \\ 0 \end{bmatrix},\ \bar{x}'' = \begin{bmatrix} \sqrt{\frac{G_1 R - 1}{G_2 R^3}} \\ \sqrt{\frac{G_1 R - 1}{G_2 R}} \end{bmatrix} \text{ e } \bar{x}''' = \begin{bmatrix} -\sqrt{\frac{G_1 R - 1}{G_2 R^3}} \\ -\sqrt{\frac{G_1 R - 1}{G_2 R}} \end{bmatrix}. \tag{2.2}$$

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 2.2 e 2.3. La Figura 2.3 è stata ottenuta partendo dallo stato iniziale $x_1(0) = 2$ e $x_2(0) = 2$.

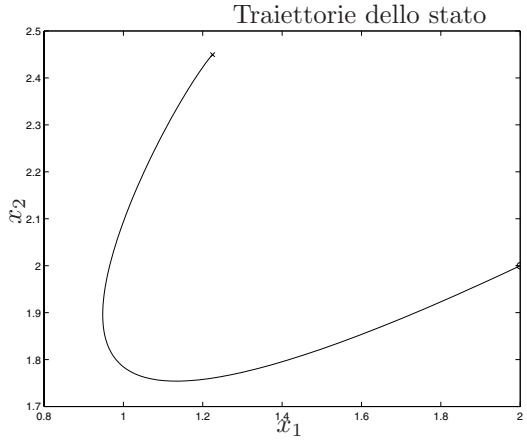


Figura 2.2: Traiettorie dello stato.

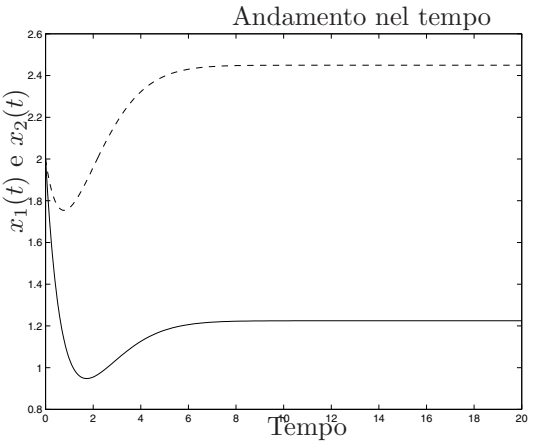


Figura 2.3: Andamento nel tempo delle variabili di stato.

L'istruzione di *Matlab* `ode45` consente di integrare sistemi di equazioni e verrà descritta nel paragrafo 2.3.

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 2.4 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.

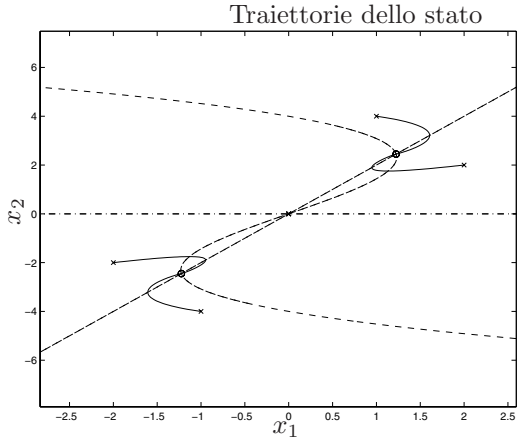


Figura 2.4: Traiettorie dello stato.

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 2.5 mostra l'andamento nel tempo delle traiettorie a partire da diversi stati iniziali, mentre in Figura 2.6 è riportato l'andamento smorzato delle due variabili di stato a partire dalla condizione $x_1(0) = 3$ e $x_2(0) = -3$.

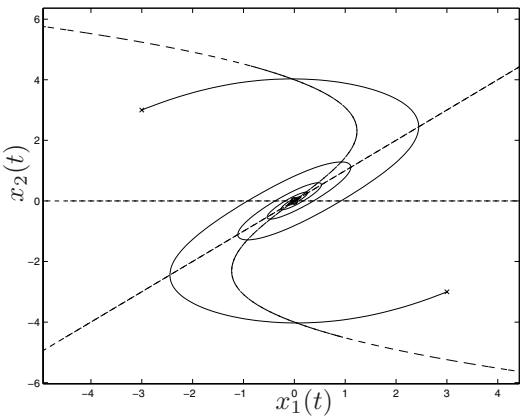


Figura 2.5: Traiettorie dello stato.

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 2.7

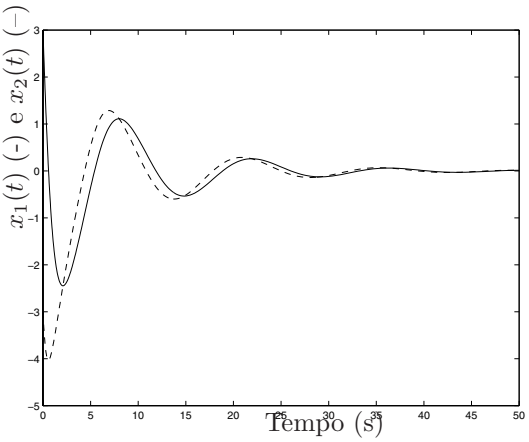


Figura 2.6: Andamento delle variabili di stato.

mostra l'andamento nel tempo delle traiettorie a partire da diversi stati iniziali, mentre in Figura 2.8, è riportato l'andamento oscillatorio delle due variabili di stato a partire dalla condizione $x_1(0) = 1$ e $x_2(0) = 1$.

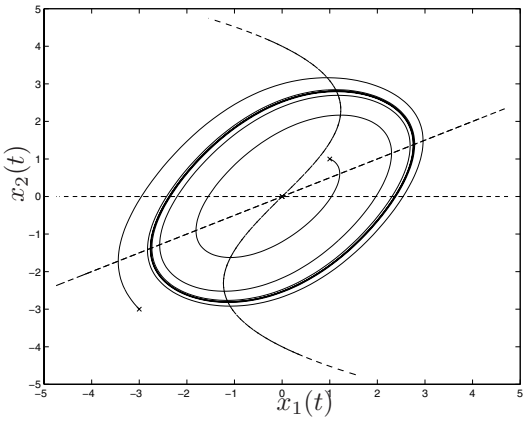


Figura 2.7: Traiettorie dello stato con un punto di equilibrio.

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)
%
```

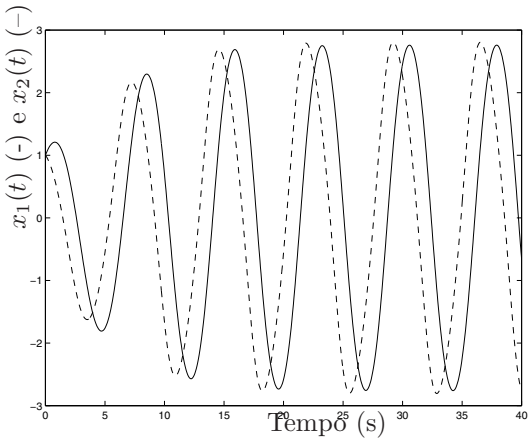


Figura 2.8: Andamento oscillatorio delle variabili di stato.

```
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 2.9 e 2.10.

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

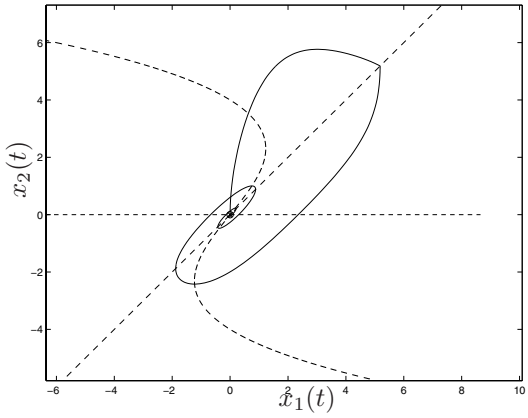


Figura 2.9: Traiettorie dello stato con un punto di equilibrio e impulso di corrente.

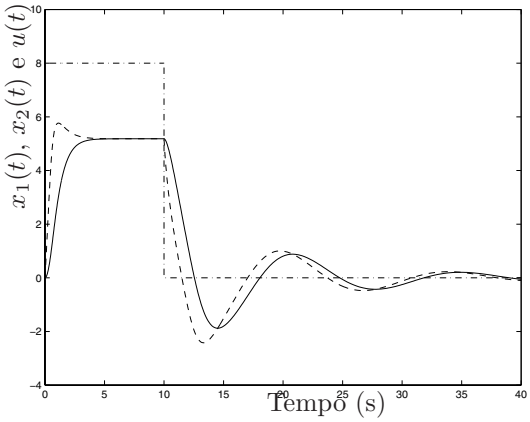


Figura 2.10: Andamento smorzato delle variabili di stato con impulso di corrente.

$$\begin{aligned} \delta\dot{x}(t) &= A\delta x(t) + B\delta u(t) \\ \delta y(t) &= C\delta x(t) \end{aligned} \tag{2.3}$$

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 2.11 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 2.12 riporta lo stesso grafico a partire da un nuovo stato iniziale ($x(0) = \begin{bmatrix} -2 \\ 2 \end{bmatrix}$), lontano dall'origine.
Le Figure 2.13 e 2.14 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.

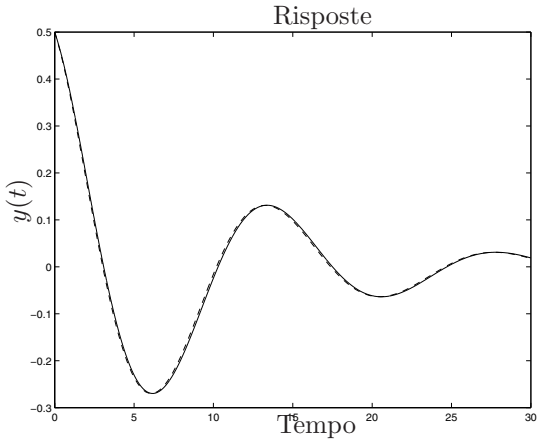


Figura 2.11: Risposta del sistema lineare e del sistema non lineare.

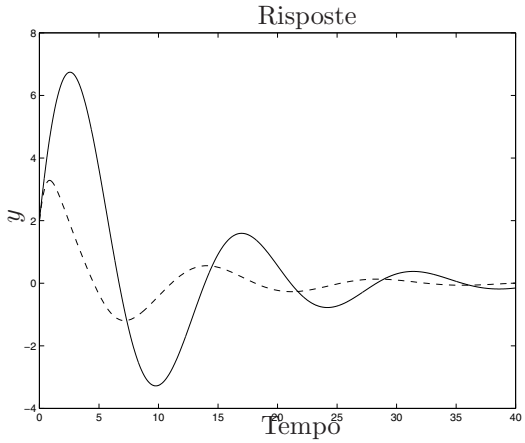


Figura 2.12: Risposta dei sistemi per una condizione iniziale distante dall'origine.

2.3 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.

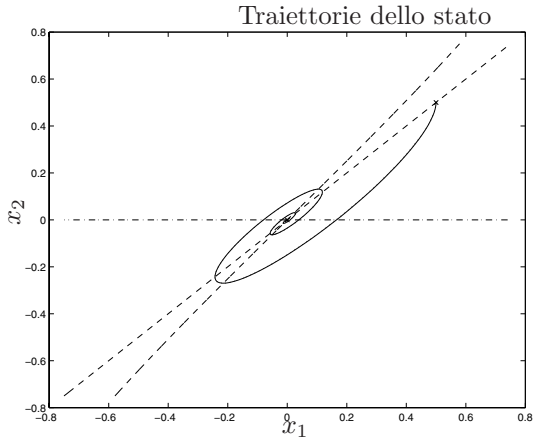


Figura 2.13: Traiettorie dello stato per condizioni iniziali vicino all'origine.

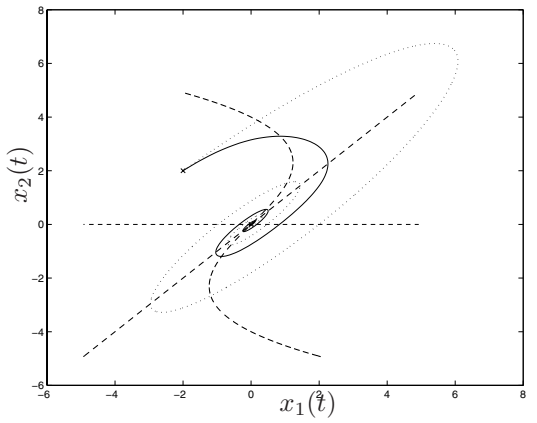


Figura 2.14: Traiettorie dello stato per condizioni iniziali lontane all'origine.

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 2.2, 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 ε_i sull'i-esima componente relativa alla stima della soluzione y_i dell'equazione differenziale, soddisfa la disequazione

$$|\varepsilon_i| \leq r|y_i| + a_i \tag{2.4}$$

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.

2.4 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 2.15. 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 [5].

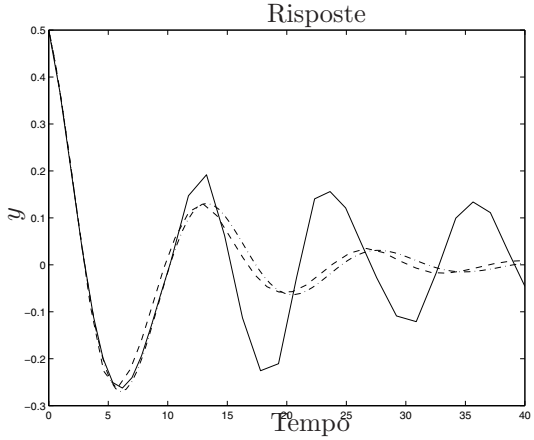


Figura 2.15: Risposta del sistema con diversi passi di integrazione.

2.5 Esercizi proposti in aula didattica.

1. Si consideri il modello matematico di Lotka-Volterra (2.5) che descrive la dinamica di due popolazioni interagenti

$$\begin{aligned} \dot{x}_1(t) &= a_1(1 - x_1(t)/k)x_1(t) - a_2 x_1(t)x_2(t) + u(t)(\ prede) \\ \dot{x}_2(t) &= -a_3 x_2(t) + a_4 x_1(t)x_2(t)(\ predatori) \end{aligned} \tag{2.5}$$

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 3

Introduzione a Simulink

Simulink, prodotto dalla *Mathworks Inc.* è un programma per la simulazione di sistemi dinamici [6]. 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*.

3.1 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 3.1) 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.

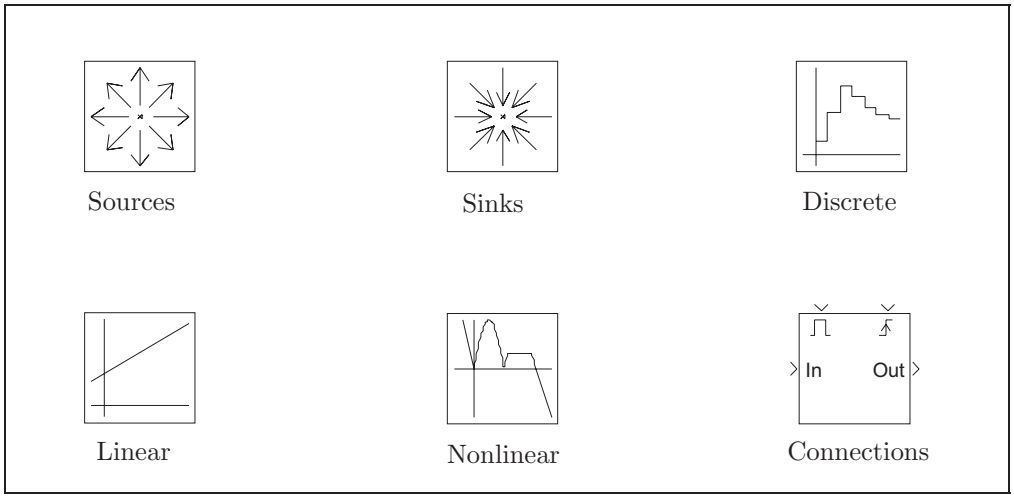


Figura 3.1: *Simulink* block library.

Ciascuna icona della Figura 3.1 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 3.2

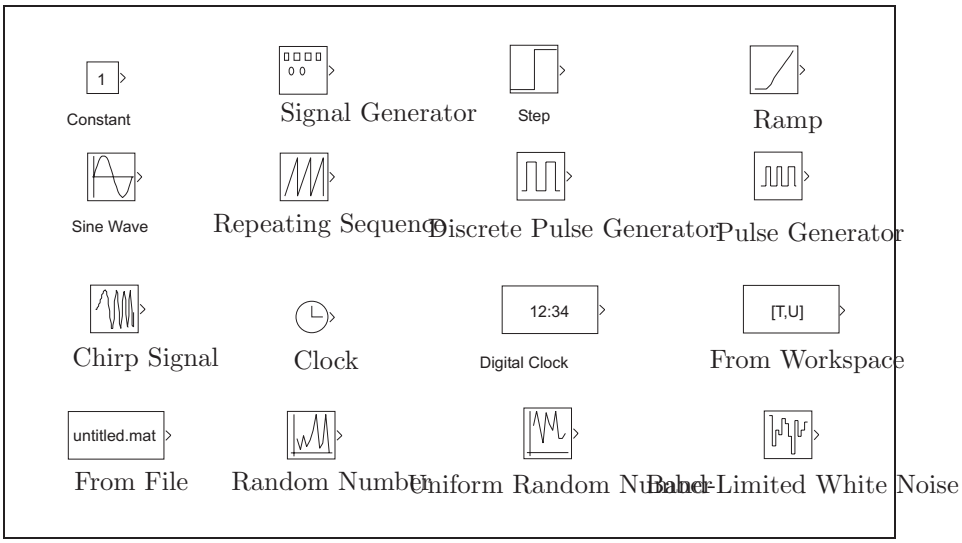


Figura 3.2: *Simulink* Signal Source library.

- **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 3.3

   - **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.

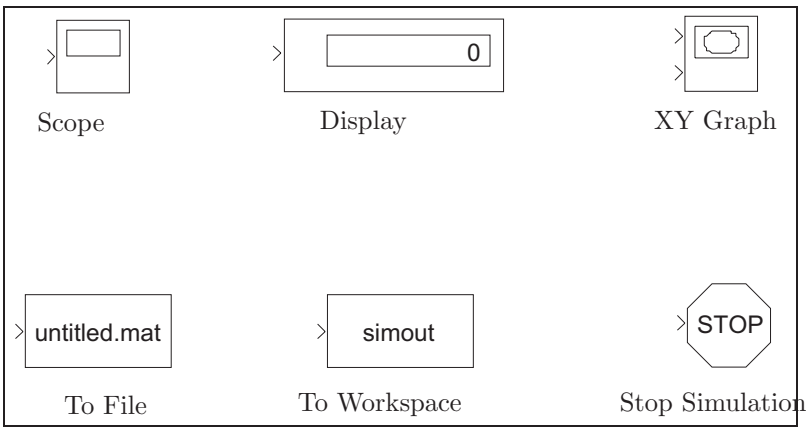


Figura 3.3: *Simulink* Signal Sinks library.

- **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 3.4

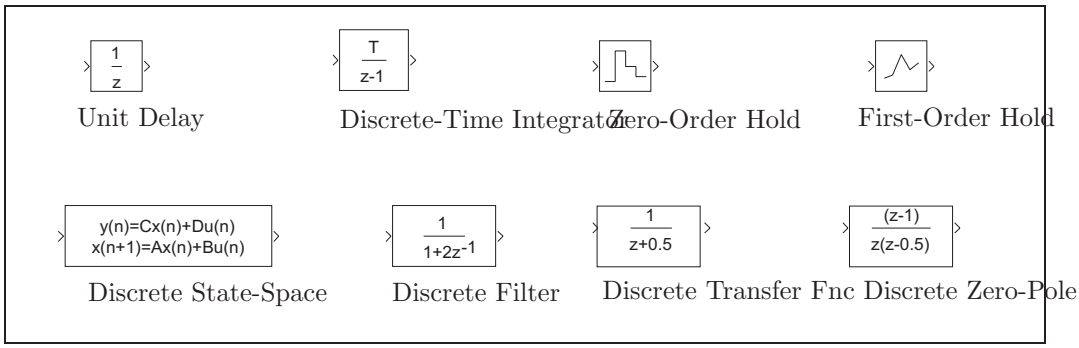


Figura 3.4: *Simulink* Discrete-Time library.

- **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 3.5

   - **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.

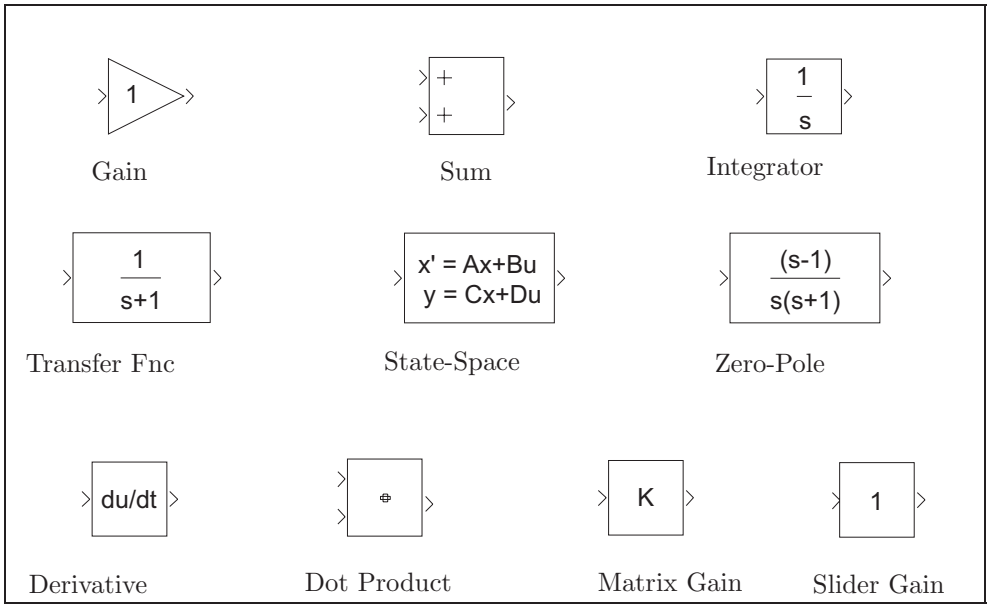


Figura 3.5: *Simulink* Linear library.

- **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 3.6

   - **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`.

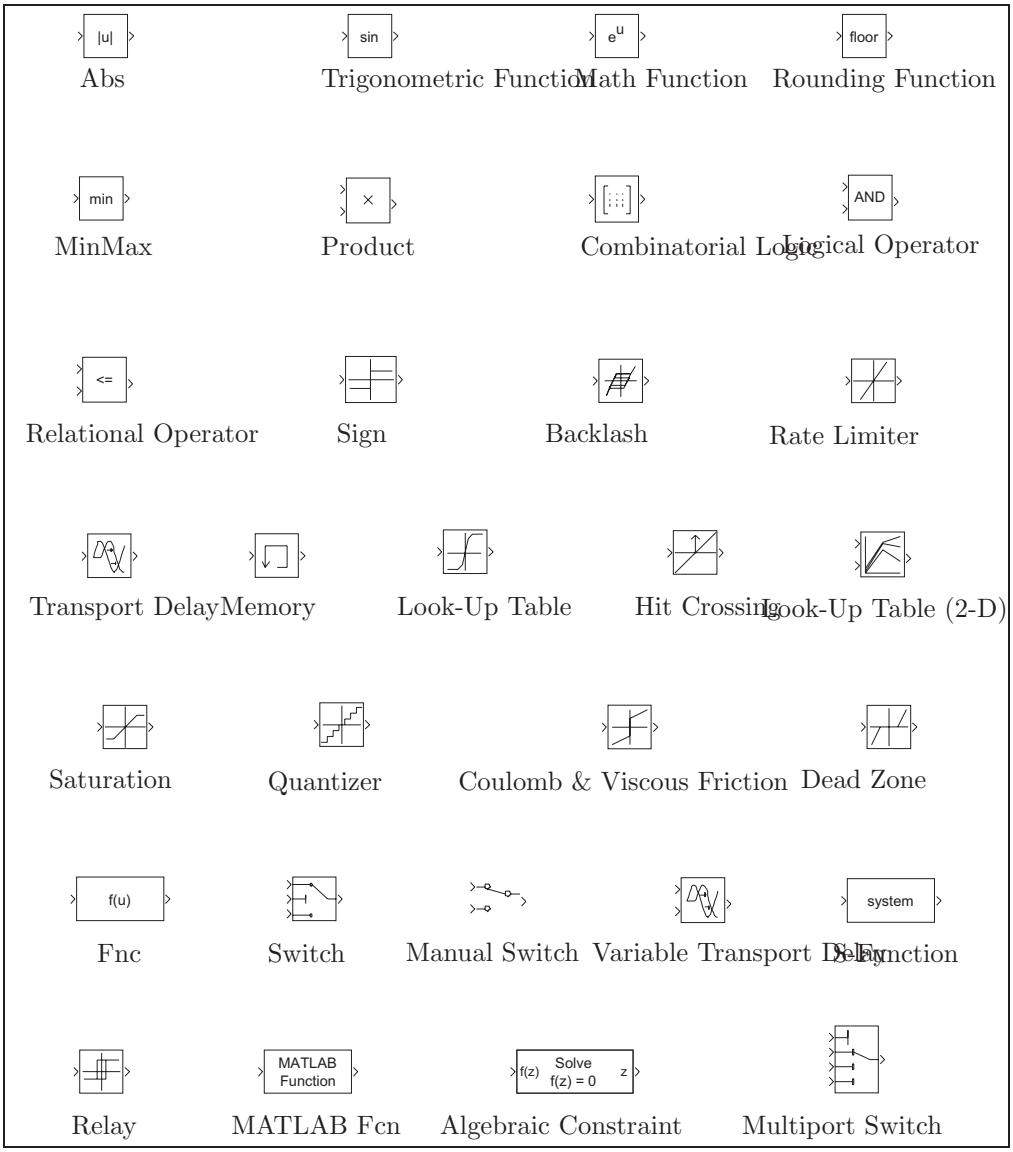


Figura 3.6: *Simulink* Nonlinear library.

- **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 3.7

   - **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.

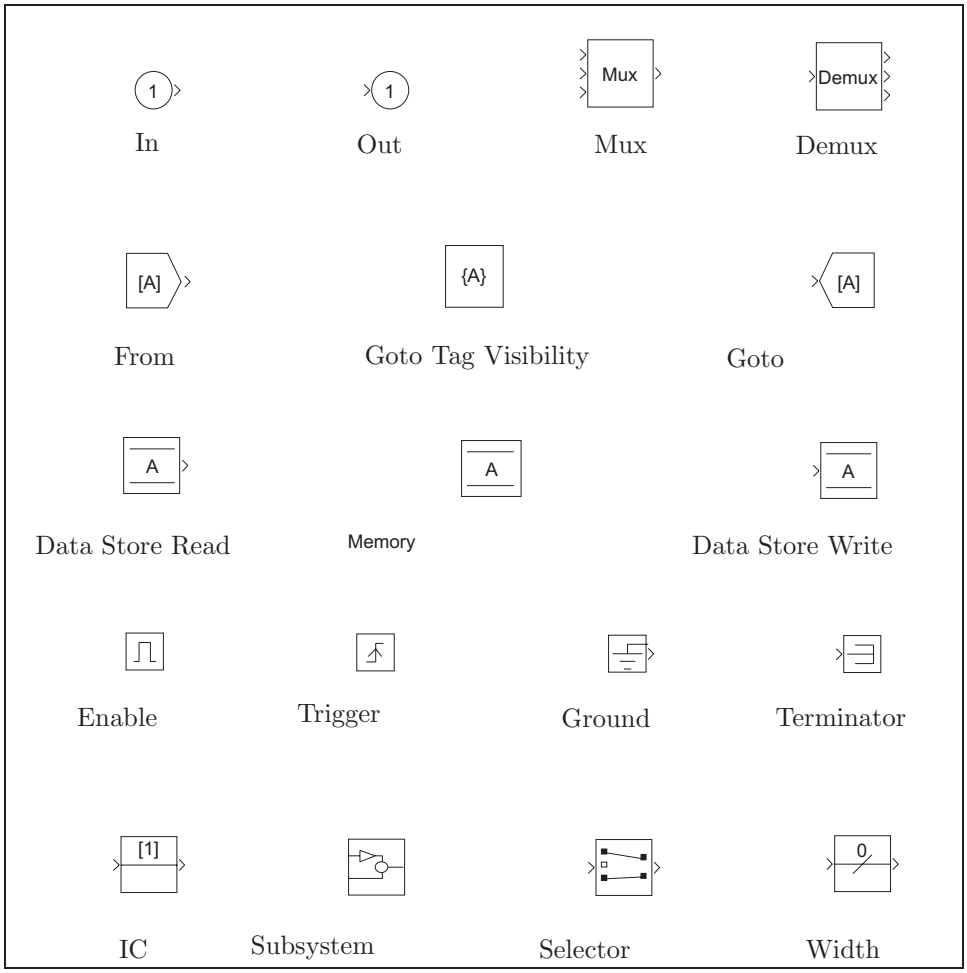


Figura 3.7: *Simulink* Connection library.

- **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 2 e in particolare nel Paragrafo 2.3:

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.

3.2 Funzioni e modelli usati nel capitolo

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

`modello tunnel.mdl`, modello *Simulink* di circuito non lineare.

`motorecc.mdl`, modello *Simulink* di motore in corrente continua.

`parametri tunnel.m`, file *Maltab* che contiene i parametri di inizializzazione per il modello di circuito con diodo tunnel.

`parametri motorecc.m`, file *Maltab* che contiene i parametri del modello del motore in corrente continua .

`ode_motorecc.mdl`, rappresentazione *Simulink* di motore in corrente continua con equazioni differenziali.

`param_ode_motorecc.m` file *Maltab* che contiene i parametri di inizializzazione per il modello del motore in corrente continua.

3.3 Analisi di un circuito non lineare.

Si riprenda l'esempio del circuito non lineare utilizzato nel Capitolo 2 Paragrafo 2.2. Utilizzando *Simulink*, il modello non lineare è rappresentato in Figura 3.8

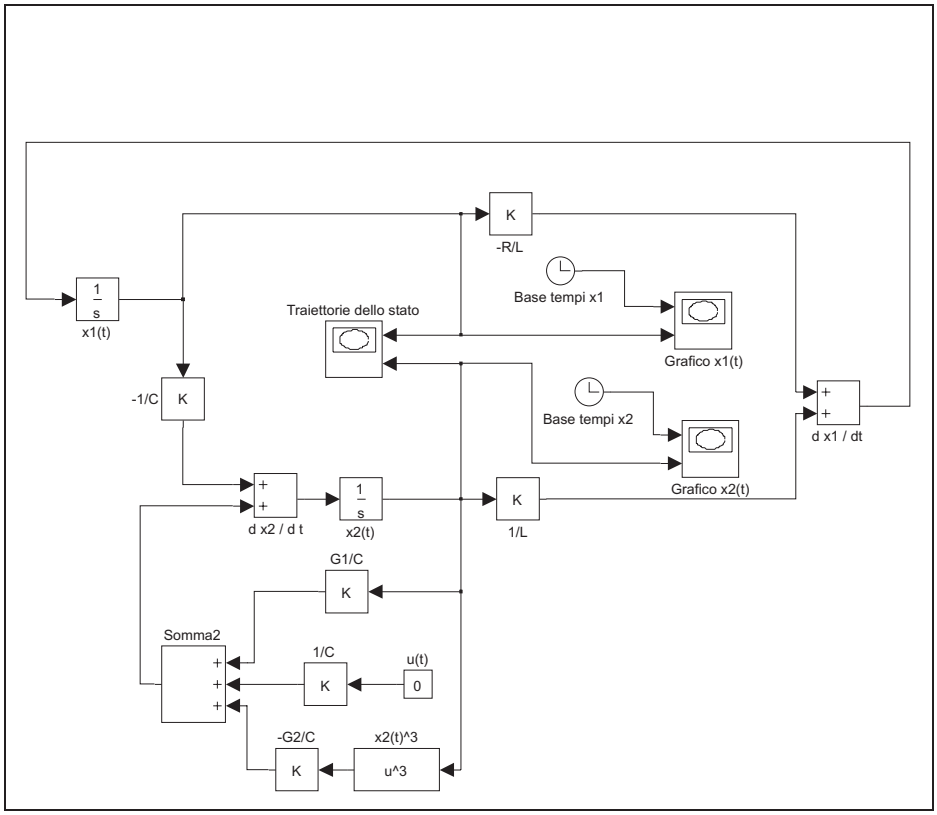


Figura 3.8: Circuito non lineare in *Simulink*.

da cui si ottengono i seguenti risultati delle simulazioni, già presentati nel Capitolo 2, 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 3.9

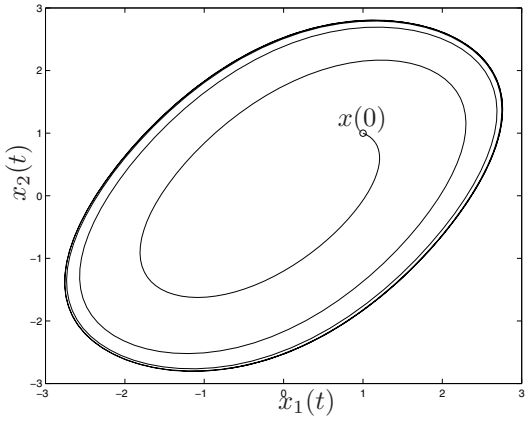


Figura 3.9: Traiettorie dello stato in *Simulink*.

e si confronti con la Figura 2.7.

Analogamente, l'andamento delle variabili di stato $x_1(t)$ e $x_2(t)$ nel tempo è graficato nelle Figure 3.10

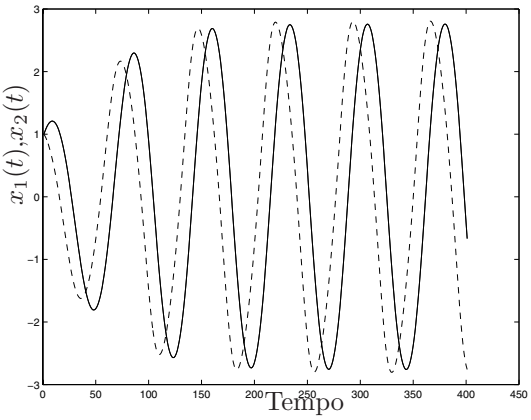


Figura 3.10: Andamento delle variabili di stato $x_1(t)$ e $x_2(t)$ nel tempo calcolata in *Simulink*.

Si confronti l'andamento con quello analogo mostrato nella Figura 2.8 del Capitolo 2.

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*.

3.4 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) 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 3.11.

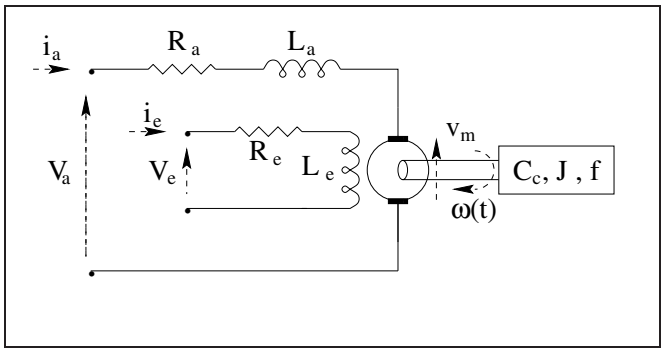


Figura 3.11: Motore in corrente continua.

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

$$\begin{aligned} \begin{bmatrix} \dot{i}_a(t) \\ \dot{\omega}(t) \end{bmatrix} &= \begin{bmatrix} -\frac{R_a}{L_a} & -\frac{k_m}{L_a} \\ \frac{k_m}{J} & -\frac{f}{J} \end{bmatrix} \begin{bmatrix} i_a(t) \\ \omega(t) \end{bmatrix} + \begin{bmatrix} \frac{1}{L_a} & 0 \\ 0 & -\frac{1}{J} \end{bmatrix} \begin{bmatrix} V_a(t) \\ C_c(t) \end{bmatrix} \\ \omega(t) &= \begin{bmatrix} 0 & 1 \end{bmatrix} \begin{bmatrix} i_a(t) \\ \omega(t) \end{bmatrix} \end{aligned} \tag{3.1}$$

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 3.12 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.

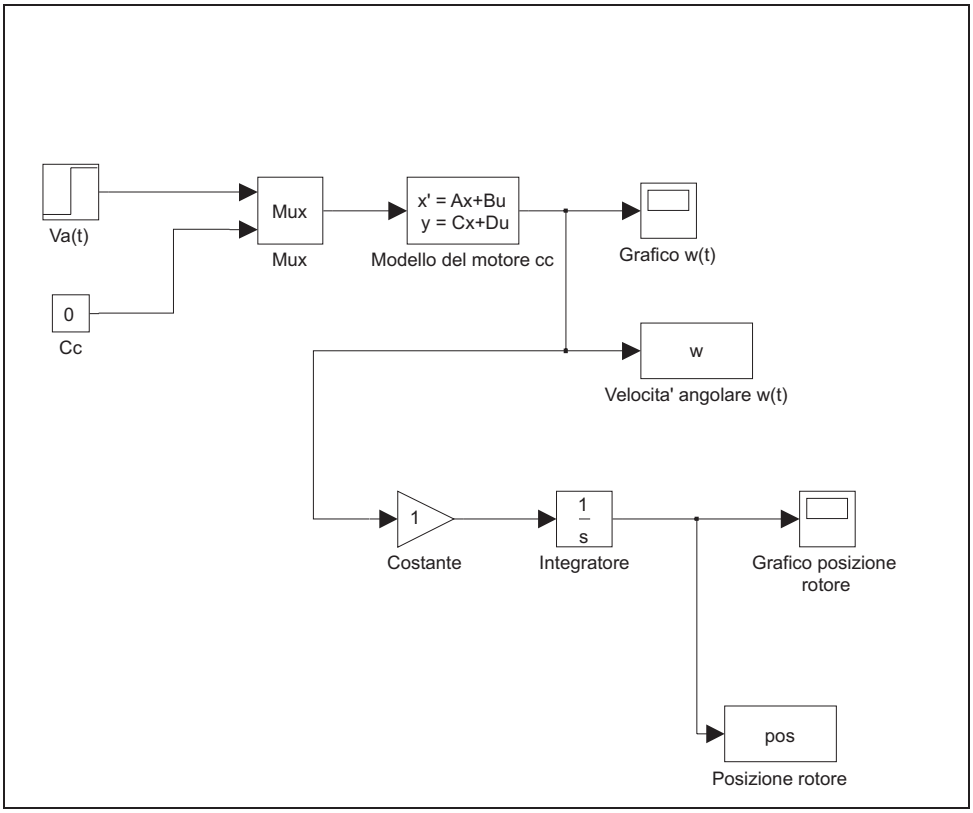


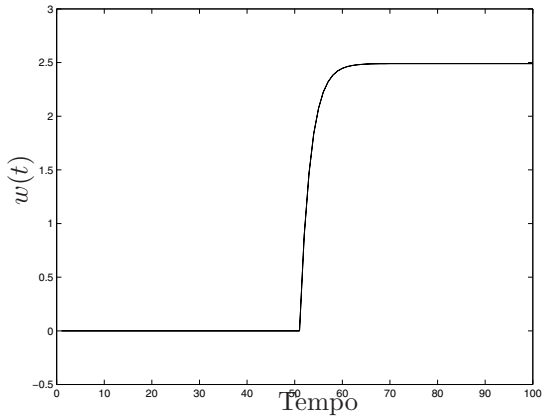
Figura 3.12: Modello *Simulink* del motore in corrente continua.

La Figura 3.13 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 $V_a(t) = 10\text{V}$ e durata $\tau = 40\text{s}$. I grafici della posizione del rotore $\alpha(t)$ in radianti e della relativa velocità angolare sono rappresentati nelle Figure 3.14 e 3.15.
Per ottenere l'ulteriore uscita $\alpha(t)$ a partire dal modello del secondo ordine descritto nell'Equazione 3.1 occorre notare che $\dot{\alpha}(t) = \omega(t)$ e, pertanto, ponendo

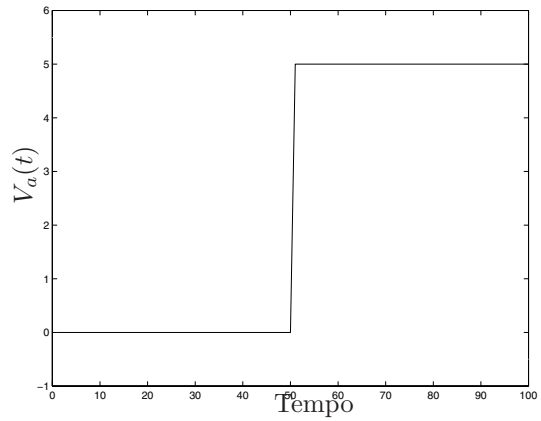
$$x(t) = \begin{bmatrix} i_a(t) \\ \omega(t) \\ \alpha(t) \end{bmatrix} \text{ e } y(t) = \begin{bmatrix} \omega(t) \\ \alpha(t) \end{bmatrix}$$

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

$$A = \begin{bmatrix} -\frac{R_a}{L_a} & -\frac{k_m}{L_a} & 0 \\ \frac{k_m}{J} & -\frac{f}{J} & 0 \\ 0 & 1 & 0 \end{bmatrix}, \; B = \begin{bmatrix} \frac{1}{L_a} & 0 \\ 0 & -\frac{1}{J} \\ 0 & 0 \end{bmatrix} \text{ e } C = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (3.2)$$



(a)



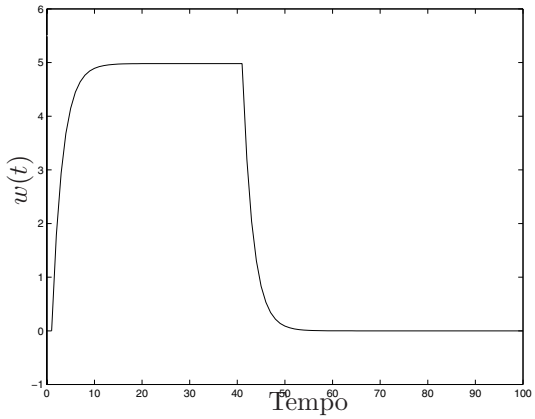
(b)

Figura 3.13: Velocità angolare del motore in cc (a) soggetto ad un gradino di tensione (b).

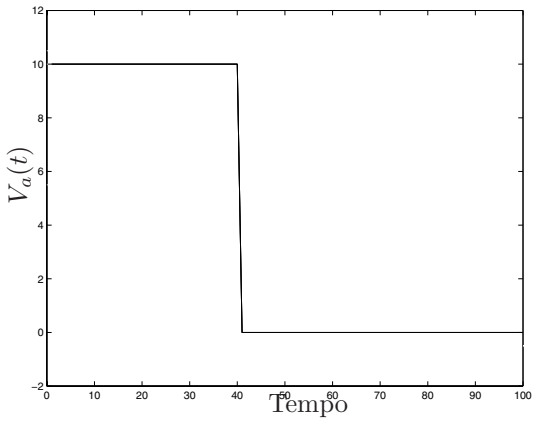
3.5 Esercizi proposti in aula didattica.

1. Si realizzi in ambiente *Simulink* il sistema di equazioni differenziali relative ai modello del motore in corrente continua (3.1) e (3.2). Se ne verifichi successivamente la correttezza confrontandolo con le realizzazioni equivalenti nello spazio degli stati (3.1) e (3.2).

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)

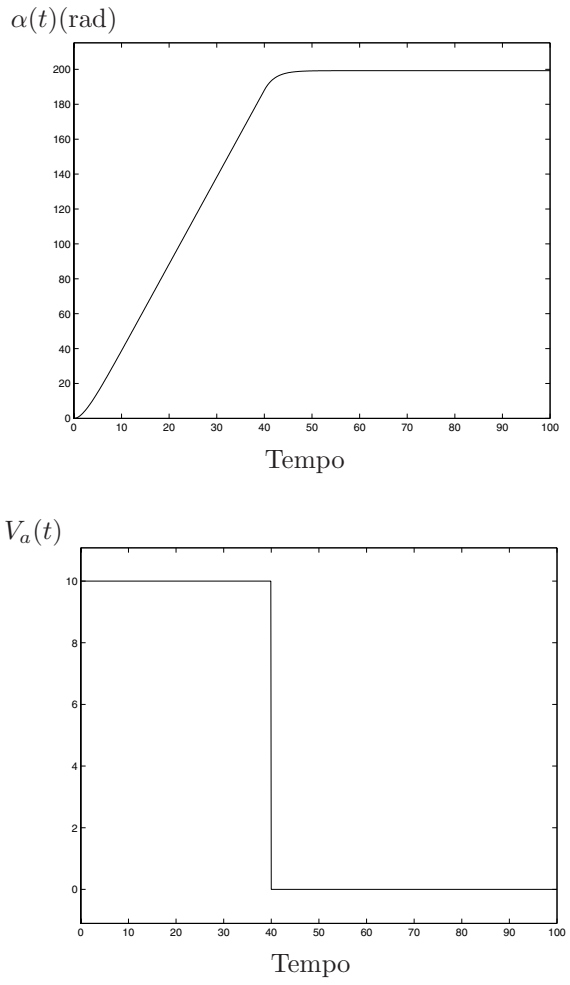


(b)

Figura 3.14: 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 τ in modo da raggiungere una posizione assegnata $\alpha = 20$rad., sempre nelle ipotesi di assenza di carico.

4. Fissato τ, 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 3.15: Posizione in radianti del rotore dell motore (a) soggetto ad un impulso di tensione (b).

Capitolo 4

Analisi di Sistemi a Dati Campionati

Un sistema di controllo digitale è costituito da elementi a tempo continuo (il processo da controllare, l'attuatore, il trasduttore analogico, il filtro anti-aliasing), almeno un dispositivo a tempo discreto (il regolatore digitale) e opportuni dispositivi di interfaccia (il convertitore A/D e D/A). Il progetto del regolatore richiede quindi di fare riferimento a metodologie riguardanti l'analisi dei sistemi ibridi [7] [4].

Ci si limiterà a considerare il caso di problemi di controllo SISO nei quali il processo è un sistema lineare e stazionario a tempo continuo, il regolatore è un sistema lineare e stazionario a tempo discreto ed i convertitori sono rispettivamente il campionatore ideale ed il mantenitore di ordine zero (ZOH).

La prima scelta da effettuare in fase di sintesi è quella relativa al periodo di campionamento T con cui operano i convertitori. Occorre impiegare valori di T tanto più piccoli quanto più è elevata la velocità di risposta che si desidera ottenere dal sistema di controllo. D'altra parte, la scelta di T deve tenere anche conto del costo dei dispositivi, dei problemi di natura numerica e del tempo di elaborazione dei segnali.

La determinazione della funzione di trasferimento del regolatore digitale avviene attraverso due possibili approcci.

Se si privilegia l'analisi a tempo continuo, si conduce il progetto del regolatore mediante tecniche classiche di sintesi a tempo continuo, in modo che la relativa funzione di trasferimento soddisfi tutte le specifiche di progetto. Successivamente si ricava la funzione di trasferimento del sistema a tempo discreto mediante le tecniche di discretizzazione. La funzione di trasferimento del regolatore digitale collegato in serie tra il campionatore e lo ZOH deve emulare il comportamento del regolatore analogico.

La sintesi del regolatore digitale può avvenire anche basandosi su una rappresentazione puramente a tempo discreto del sistema di controllo, ottenuta attraverso la descrizione del sistema a segnali campionati costituito dal mantenitore, dal processo e dal campionatore. La sintesi deve essere in questo caso effettuata utilizzando i metodi propri dell'analisi dei sistemi retroazionati a tempo discreto.

4.1 Discretizzazione di un regolatore a tempo continuo

Dato un regolatore analogico, si può determinare un'approssimante digitale utilizzando la semplice regola di sostituzione di variabile. In particolare, si ottiene la trasformazione di *Eulero in avanti* (EA) se

$$s = \frac{z-1}{T} \quad \text{(EA)} \tag{4.1}$$

la trasformazione di *Eulero all'indietro* (EI) se

$$s = \frac{z-1}{Tz} \quad \text{(EI)} \tag{4.2}$$

infine, quella di *Tustin*, quando

$$s = \frac{2}{T}\frac{z-1}{z+1} \quad \text{(TU).} \tag{4.3}$$

I tre metodi di discretizzazione sono basati sull'idea di utilizzare la trasformazione di campionamento $z = e^{sT}$. Infatti, tramite lo sviluppo dell'esponenziale nell'intorno di $s = 0$, si ricavano le formule alternative

$$z = e^{sT} \simeq 1 + sT \tag{4.4}$$

$$z = \frac{1}{e^{-sT}} \simeq \frac{1}{1-sT} \tag{4.5}$$

che corrispondono ripettivamente ad EA ed EI.

Inoltre, ricordando lo sviluppo di Padé, si ottiene

$$z = e^{sT} = \frac{1}{e^{-sT}} \simeq \frac{1+sT/2}{1-sT/2} \tag{4.6}$$

che coincide con TU.

Un'altra tecnica per determinare un regolatore digitale consiste nel sostituire al controllore analogico $R(s)$ la sua versione a segnali campionati. La funzione di trasferimento viene calcolata sulla base dello schema di Figura 4.1.

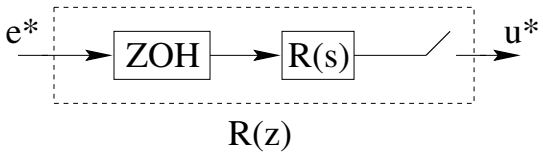


Figura 4.1: Regolatore analogico ai segnali campionati.

Tale metodo di discretizzazione può portare a soluzioni non soddisfacenti. Lo schema equivalente del regolatore risultante riportato in Figura 4.2 contiene una

doppia coppia di campionatore e mantenitore di ordine zero (ZOH). Ognuna di queste coppie introduce un ritardo che puø‘ deteriorare le prestazioni del sistema di controllo. Occorre perciò che il progetto preliminare di $R(s)$ garantisca un'eccedenza di margine di fase superiore a quella normalmente richiesta quando si utilizza l'approccio della discretizzazione. Il metodo presentato prende il nome di *Hold Equivalence* (HE) o di tenuta e campionamento.

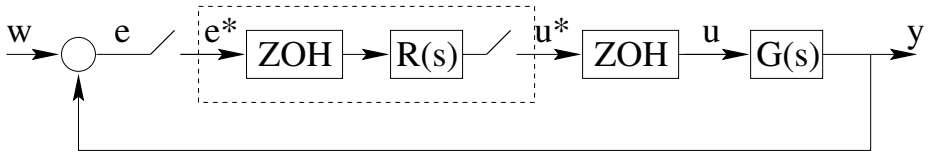


Figura 4.2: Schema equivalente del sistema di controllo digitale ottenuto con HE.

4.2 Scelta del periodo di campionamento

Una regola euristica impone che il tempo di campionamento T rispetti un vincolo del tipo

$$\frac{2\pi}{10\alpha\omega_c} \leq T \leq \frac{2\pi}{\alpha\omega_c} \tag{4.7}$$

con α di solito compreso tra 5 e 10. Legando il tempo di assestamento con la pulsazione critica ω_c (pulsazione per la quale il modulo della funzione di trasferimento è unitario), si può dimostrare che la (4.7) equivale a richiedere di avere da α a 10α campioni nell'intervallo del tempo di assestamento T_a stesso, quindi approssimativamente

$$\frac{T_a}{10\alpha} \leq T \leq \frac{T_a}{\alpha}. \tag{4.8}$$

4.3 Risposta frequenziale

Nelle funzioni di trasferimento a tempo discreto la pulsazione ω compare in forma non razionale

$$G(z) \qquad z = e^{j\omega T} \rightarrow G(e^{j\omega T}). \tag{4.9}$$

Per ovviare a questo inconveniente si usa la trasformazione

$$w = \frac{2}{T}\frac{z-1}{z+1} \qquad z = \frac{1+w\frac{T}{2}}{1-w\frac{T}{2}}. \tag{4.10}$$

Il fattore moltiplicativo $\frac{2}{T}$ garantisce che le funzioni di trasferimento nel piano w tendano a quelle nel piano s per $T \rightarrow 0$.

4.4 Progetto di Controllori Digitali

Per avere un quadro globale delle metodologie di progetto di un regolatore digitale si consideri il seguente schema in Figura 4.3.

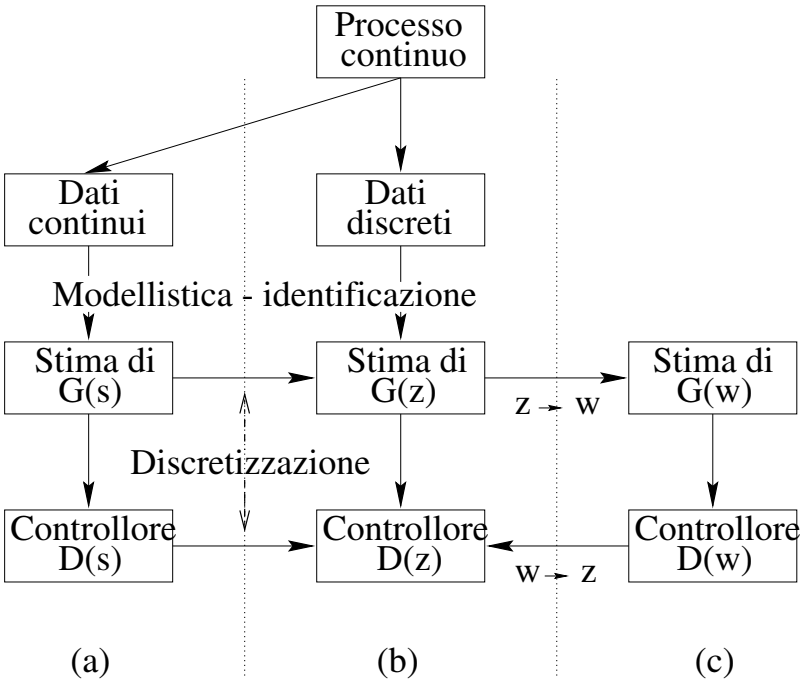


Figura 4.3: Quadro delle metodologie di progetto di un regolatore digitale.

La sezione (a) fa riferimento ad un progetto condotto cercando di soddisfare specifiche nel dominio dei tempi ed attraverso le formule di Ziegler-Nichols.

La sezione (b) presenta il procedimento di sintesi diretta del controllore digitale, ad esempio con il Controllo Dead-Beat, la Modifica di Dahlin e l'Algoritmo di Kalman.

Infine, la sezione (c) illustra il progetto basato sui diagrammi di Bode.

4.4.1 Sintesi di un regolatore mediante discretizzazione

Questo paragrafo illustra il procedimento di sintesi di un regolatore $D(s)$ nel piano s per il sistema controllato continuo $G(s)$ illustrato nella sezione (a) di Figura 4.3.

Per controllare il sistema descritto dalla funzione continua

$$G(s) = 0.2\frac{(1-2s)}{s(1+10s)(1+0.1s)} \tag{4.11}$$

si progetta un regolatore a tempo continuo rappresentato dalla funzione

$$R(s) = \frac{(1+10s)}{(1+0.1s)} \tag{4.12}$$

tale da garantire un margine di fase di circa 64^o a $\omega_c = 0.22\frac{rad}{s}$ ed un tempo di assestamento pari a 9.7s, eliminando la oscillazioni nella risposta al gradino. Il sistema non compensato era caratterizzato da una sovraelongazione pari al 54% e da un tempo di assestamento di 99s.

Quanto alla scelta del tempo di campionamento, in base alla regola (4.7), ponendo $\alpha = 5$, si può scegliere $T = 1$s.
Utilizzando quindi i metodi di discretizzazione visti precedentemente ed applicati al regolatore descritto dalla funzione di trasferimento (4.12) si ottengono rispettivamente i regolatori

$$R(z) = \frac{100z - 90}{z + 9} \quad \text{(EA)} \tag{4.13}$$

$$R(z) = \frac{10z - 9.091}{z - 0.09091} \quad \text{(EI)} \tag{4.14}$$

$$R(z) = \frac{17.5z - 15.8340}{z - 0.6667} \quad \text{(TU)} \tag{4.15}$$

$$R(z) = \frac{100z - 99}{z - 4.54 \times 10^{-5}} \quad \text{(HE)} \ . \tag{4.16}$$

I primi tre regolatori (EA, EI, TU) sono stati ricavati per sostituzione diretta, mentre l'ultimo (HE) attraverso il *TFI*. In particolare, la funzione `convert` di *TFI* può effettuare le seguenti conversioni, come si vede dall'help.

```
CONVERT  Conversione da tempo continuo a tempo discreto (TFI).
     CONVERT,gi,gj - da TFI [CONVERT('gi,'gj') - da Matlab] converte
     la funzione di trasferimento a tempo continuo gi alla funzione
     di trasferimento a tempo discreto gj. Possibili opzioni:
       1 - Z-trasformata della antitrasformata di Laplace campionata
       2 - dispositivo di tenuta di ordine zero
       3 - dispositivo di tenuta di ordine uno
     In ogni caso si puo' tener conto anche di un ritardo finito.
     Se CONVERT viene chiamato senza alcun argomento, le f.d.t. gi e gj
     possono essere specificate in interattivo.
```

Infatti, se `R` contiene la funzione di trasferimento del regolatore, la funzione ai dati campionati si ottiene

```
> convert,R,R3
   convert (conversione da tempo continuo a tempo discreto)

   1 - Z-trasformata dell'antitrasformata di Laplace campionata
   2 - conversione con tenuta di ordine zero
   3 - conversione con tenuta di ordine uno

   operare una scelta : 2
```

```
   specificare un ritardo finito (default zero) :

   conversione da tempo continuo a tempo discreto;
   il tempo di campionamento e' 1 sec
   il ritardo finito e' 0

     100 (z - 0.99)
R3 = --------------
     (z - 4.54e-005)

 >
```

Al fine di verificare se il sistema discreto in retroazione risulta stabile, occorre calcolare la funzione di trasferimento $G(z)$ ai dati campionati. La funzione `convert` fornisce il seguente risultato

$$G(z) = \frac{-93.39(-0.6018z + 1)(11.46z + 1)}{(-z + 1)(-1.105z + 1)(-2.203e + 004z + 1)}. \tag{4.17}$$

Utilizzando i regolatori ottenuto con (EA) si ottiene un sistema digitale in retroazione non stabile. I poli del sistema in retroazione determinabili in *TFI* attraverso la fattorizzazione del denominatore della funzione di trasferimento in retroazione sono

```
>>> Poli con gli ordini di molteplicita' rilevati <<<

     1  |  +8.7065e-002                      |   1
     2  |  +7.2025e-001                      |   1
     3  |  +8.9582e-001                      |   1
     4  |  -6.1528e+000                      |   1
```

Per il sistema con (EI), si ha

```
>>> Poli con gli ordini di molteplicita' rilevati <<<

     1  |  -6.4163e-002                      |   1
     2  |  +9.1124e-001                      |   1
     3  |  +7.0663e-001    +3.1160e-001 * j  |   1
     4  |  +7.0663e-001    -3.1160e-001 * j  |   1
```

che risulta stabile.

Analogamente, per il sistema con regolatore (HE), si ottengono i poli

```
>>> Poli con gli ordini di molteplicita' rilevati <<<

     1  |  -8.5680e-002                      |   1
     2  |  +9.9042e-001                      |   1
     3  |  +1.8228e+000    +1.0737e+000 * j  |   1
     4  |  +1.8228e+000    -1.0737e+000 * j  |   1
```

da cui un sistema instabile. Infine, per il regolatore con TU si ha

```
>>> Poli con gli ordini di molteplicita' rilevati <<<

    1  |  -2.2755e-001                          |   1
    2  |  +9.0478e-001                          |   1
    3  |  +5.1196e-001    +1.8184e-001 * j      |   1
    4  |  +5.1196e-001    -1.8184e-001 * j      |   1
```

da cui un sistema in retroazione stabile.

Si graficano quindi i diagrammi di Bode per le funzioni d'anello per i soli regolatori (EI) ed (TU) in Figura 4.4.

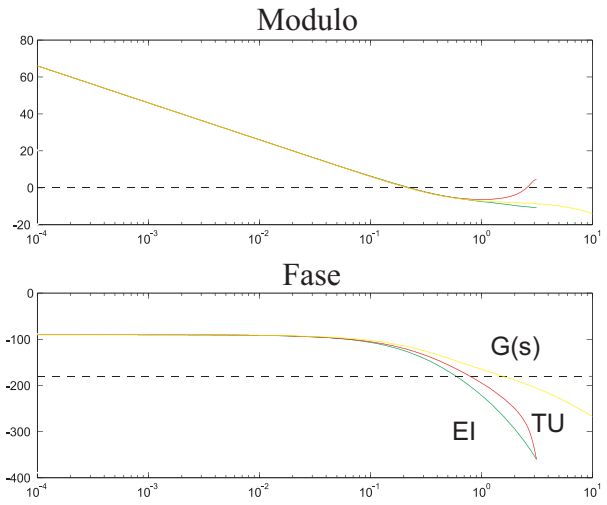


Figura 4.4: Diagrammi di bode delle funzioni d'anello.

Il margine di fase associato al regolatore (EI) risulta di 51^o mentre quello relativo a (TU) di 58^o. Risultano quindi inferiori a quello associato alla funzione di anello del caso analogico.

La Figura 4.5 mette a confronto le risposte al gradino dei due regolatori digitali.
Guardando le caratteristiche delle risposte, per (EI) si ha che la massima sovraelongazione vale 16%, tempo di salita pari a 2s, e tempo di assestamento 13s. Per (TU), massima sovraelongazione 0.5%, tempo di salita 3s e tempo di assestamento 7s. Il sistema analogico di partenza non aveva invece nessuna sovraelongazione, tempo di salita pari a 6s e tempo di assestamento 10s.

Si osservi come in questo esempio il regolatore (TU) abbia caratteristiche migliori come tempo di assestamento di quello analogico. Questo risultato è dovuto alle caratteristiche del sistema sotto controllo che è a fase non minima e il regolatore è una rete anticipatrice.

4.4.2 Sintesi di un regolatore digitale nel dominio delle frequenze

La trasformazione nel piano w precedentemente introdotta consente di effettuare il progetto del regolatore discreto direttamente nel dominio delle frequenze. Una

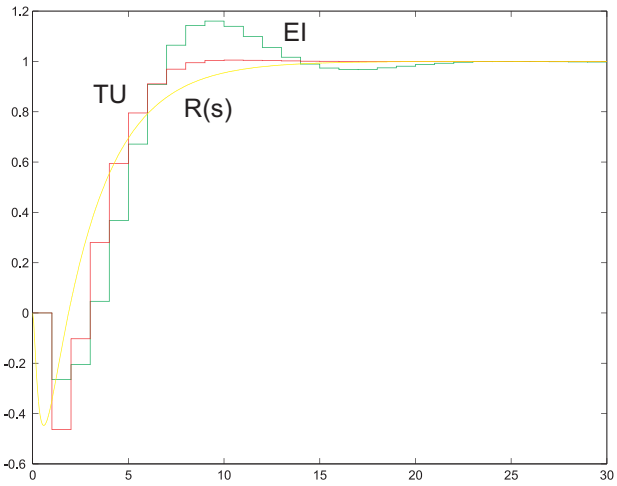


Figura 4.5: Risposta al gradino del sistema di controllo digitale.

volta che la funzione di trasferimento del processo controllato è stata trasformata nel dominio w, possono essere utilizzate nel discreto tutte quelle tecniche di progetto valide per i sistemi continui.

Si consideri il sistema descritto dalla seguente funzione di trasferimento

$$G_p(s) = \frac{2500}{s(s+25)}. \tag{4.18}$$

Se si assume che il sistema sia a dati campionati, si utilizza un dispositivo di campionamento e tenuta, con periodo $T = 0.01$ sec. Si progetti una rete anticipatrice ed una rete ritardatrice affinché il sistema compensato abbia un margine di fase di 55^o. Si confronti la risposta ad un gradino di ampiezza unitaria del sistema non compensato con le risposte dei sistemi compensati.
La funzione di trasferimento ai dati campionati $G_d(z)$ può essere ottenuta attraverso la funzione `convert` del *TFI*

```
convert

   funzione di trasferimento gi (ingresso) : Gp

   funzione di trasferimento gj (uscita) : Gd

   1 - Z-trasformata dell'antitrasformata di Laplace campionata
   2 - conversione con tenuta di ordine zero
   3 - conversione con tenuta di ordine uno

   operare una scelta : 2

   specificare un ritardo finito (default zero) :
```

```
   conversione da tempo continuo a tempo discreto;
   il tempo di campionamento e' 0.01 sec
   il ritardo finito e' 0

     0.1152 (z + 0.9201)
Gd = -------------------
     (z - 0.7788) (z - 1)

>
```

da cui

$$G_d(z) = 0.1152 \frac{z + 0.9201}{(z - 0.7788)(z - 1)}. \tag{4.19}$$

La risposta al gradino del sistema ai dati campionati non compensato presenta una sovraelongazione del 66% ed è rappresentata in Figura 4.6.

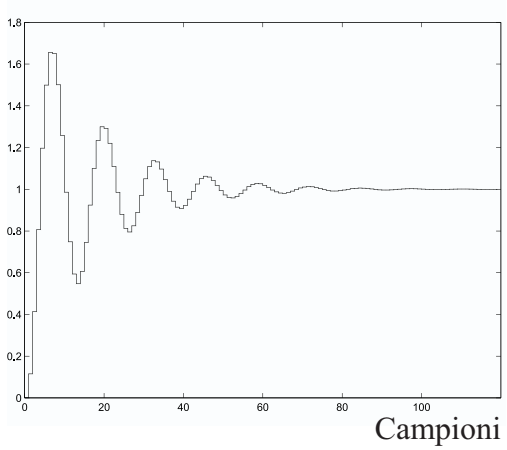


Figura 4.6: Risposta al gradino del sistema non compensato.

La funzione `tresp` del *TFI* fornisce le seguenti informazioni sulla risposta

```
RISPOSTA AL GRADINO :
   massima sovraelongazione: 65.62 percento per k=6 camp (t=0.06 sec)
   tempo di ritardo (al 50 percento): 3 camp (0.03 sec)
   tempo di salita (dal 10 al 90 percento): 3 camp (0.03 sec)
   tempo di assestamento (al piu'/meno 5 percento): 47 camp (0.47 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
```

Si effettui la trasformazione della funzione di trasferimento del sistema controllato nel piano w. Si può utilizzare direttamente la funzione `wplane` del *TFI*

```
 > wplane,Gd,Gw
   wplane (conversione da s a z o da z a s con l'equivalenza nel piano w)

   trasformazione dal discreto al continuo secondo la conversione
   al piano w diretta; il tempo di campionamento assunto e' 0.01 sec

     -0.002588 (s - 200) (s + 4805)
Gw = ------------------------------
     s (s + 24.87)

 >
```

ovvero

$$G_w(s) = 100\frac{(0.0002081s+1)(-0.005s+1)}{s(0.04021s+1)}. \tag{4.20}$$

I diagrammi di Bode dell'ampiezza e della fase del sistema in w sono rappresentati in Figura 4.7

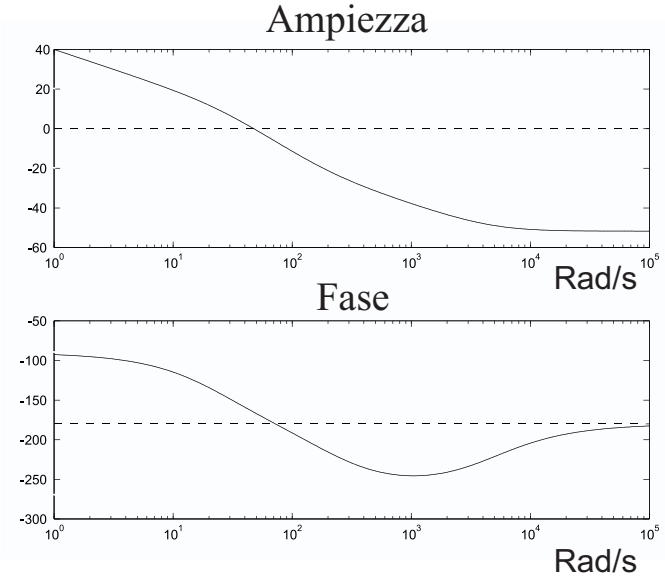


Figura 4.7: Diagrammi di Bode del sistema non compensato.

con le seguenti caratteristiche

```
RISPOSTA IN FREQUENZA AD ANELLO APERTO :
   margine di ampiezza:  2.086 (6.387 db) per omega= 72.24 rad/sec
   margine di fase: 14.76 gradi per omega= 47.61 rad/sec
   ascissa dell'asintoto verticale del diagramma polare: -4.5

   RISPOSTA IN FREQUENZA AD ANELLO CHIUSO :
   risonanza assoluta: 3.993 (12.03 db) per omega= 49.13 rad/sec
   guadagno statico: 1 (0 db); risonanza relativa: 3.993 (12.03 db)
   banda passante (-3db): 78.41 rad/sec
```

quindi un guadagno ed un margine di fase del sistema non compensato pari, rispettivamente, a 6.39dB e 14.76^o.

Si consideri il progetto di una *rete ritardatrice* utilizzando la funzione `lagc` del *TFI*. Si richiede che il sistema compensato abbia un margine di fase di 55^o. Si parte dal sistema non compensato descritto dalla funzione $G_w(s)$.

La rete ritardatrice ha le seguenti caratteristiche

```
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
```

Imponendo un margine di fase di 55^o si ottiene il seguente output

```
margine di fase senza correzione: 14.76 gradi
                    alla pulsazione: 47.62 rad/sec

   specificare il margine di fase voluto : 55

   valore massimo di alfa: 0.1731
   valore minimo di alfa : 0.01

   valore di primo tentativo: 0.08655

   **** premere un tasto per proseguire

   **** figura 1 ****

   **** figura 2 ****

   **** figura 3 ****

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

   400 passi di ricerca di tau nell'intervallo scelto:
```

```
   margine di fase richiesto trovato al passo: 171

   margine di fase senza rete correttrice: 14.75
   margine di fase con rete correttrice  : 54.91

   la rete correttrice ottenuta:
   alfa = 0.08655 , tau = 5.004 sec

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

   specificare alfa (min 0.01, max 0.1731), invio per uscire :

   LA RETE CORRETTRICE OTTENUTA :

   alfa = 0.08655 , tau = 5.004 sec

      0.08655 (s + 2.309)
Gcw = ------------------
      (s + 0.1998)
>
```

La funzione della rete ritardatrice nel piano w risulta

$$G_c(w) = 0.08655\frac{s+2.309}{s+0.1998} \tag{4.21}$$

mentre effettuando la conversione nel piano z attraverso la funzione `wplane` si ottiene

```
funzione di trasferimento gi (ingresso) : Gcw

   funzione di trasferimento gj (uscita) : Gcd

   trasformazione dal continuo al discreto secondo la conversione
   al piano w inversa; il tempo di campionamento assunto e' 0.01 sec

      0.08747 (z - 0.9772)
Gcd = --------------------
      (z - 0.998)
```

La funzione del regolatore risulta

$$G_c(z) = 0.08747\frac{z-0.9772}{z-0.998}. \tag{4.22}$$

In Figura 4.8 sono rappresentati i diagrammi di Bode del sistema senza compensazione e con rete ritardatrice
mentre in Figura 4.9 il confronto tra le risposte del sistema con e senza rete ritardatrice.

Le caratteristiche della risposta del sistema non compensato sono

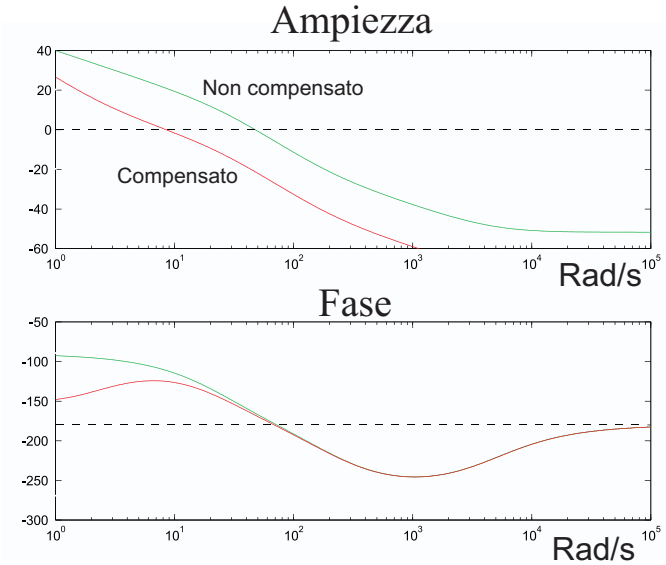


Figura 4.8: Diagrammi di Bode del sistema compensato e non compensato.

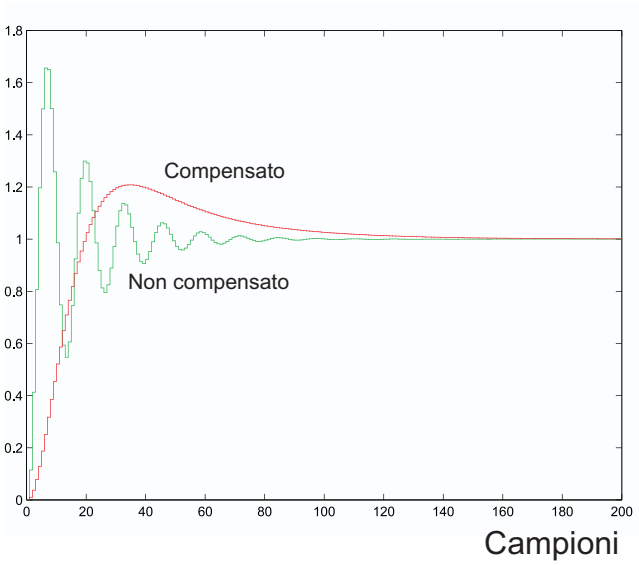


Figura 4.9: Risposta del sistema con e senza compensazione.

```
RISPOSTA AL GRADINO :
   massima sovraelongazione: 65.62 percento per k=6 camp (t=0.06 sec)
   tempo di ritardo (al 50 percento): 3 camp (0.03 sec)
   tempo di salita (dal 10 al 90 percento): 3 camp (0.03 sec)
   tempo di assestamento (al piu'/meno 5 percento): 47 camp (0.47 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
```

mentre quelle della risposta del sistema con rete ritardatrice diventano

```
RISPOSTA AL GRADINO :
   massima sovraelongazione: 20.82 percento per k=34 camp (t=0.34 sec)
   tempo di ritardo (al 50 percento): 10 camp (0.1 sec)
   tempo di salita (dal 10 al 90 percento): 13 camp (0.13 sec)
   tempo di assestamento (al piu'/meno 5 percento): 81 camp (0.81 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
```

Si progetti ora un *rete anticipatrice* tale da garantire lo stesso margine di fase di 55^o.

Si utilizza la funzione `leadc` del *TFI* che genera il seguente output

```
RETE ANTICIPATRICE

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

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

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

Al fine di ottenere un margine di fase richiesto occorre imporre durante l'interattivo un margine di fase di 65^o

```
margine di fase senza correzione: 14.76 gradi
                    alla pulsazione: 47.62 rad/sec

   specificare il margine di fase voluto : 65

   anticipo di fase necessario: 50.24 gradi
   valore di alfa di primo tentativo: 0.06536

   **** premere un tasto per proseguire

   **** figura 1 ****

   **** figura 2 ****

   **** figura 3 ****
```

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

   100 passi di ricerca di tau nell'intervallo scelto:
   margine di fase massimo trovato al passo: 47

   margine di fase senza rete correttrice: 14.75
   margine di fase con rete correttrice  : 54.77

   la rete correttrice ottenuta:
   alfa = 0.06536 , tau = 0.02732 sec

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

   specificare alfa (min .005, max 0.2615), invio per uscire :

   LA RETE CORRETTRICE OTTENUTA :

   alfa = 0.06536 , tau = 0.02732 sec

       15.3 (s + 36.61)
Gcw2 = ---------------
       (s + 560.1)
```

La funzione di trasferimeto della rete ritardatrice nel piano w risulta

$$G_{cw}(s) = 15.3\frac{s + 36.61}{s + 560.1} \tag{4.23}$$

mentre in quello z

$$G_c(z) = 4.763\frac{z - 0.6906}{z + 0.4737}. \tag{4.24}$$

I relativi diagrammi di Bode sono riportati in Figura 4.10.

Le caratteristiche del sistema compensato con una rete anticipatrice sono

```
RISPOSTA IN FREQUENZA AD ANELLO APERTO :
   margine di ampiezza:  2.997 (9.535 db) per omega= 350.1 rad/sec
   margine di fase: 54.77 gradi per omega= 76 rad/sec
   ascissa dell'asintoto verticale del diagramma polare: -1.947

   RISPOSTA IN FREQUENZA AD ANELLO CHIUSO :
   risonanza assoluta: 1.115 (0.9471 db) per omega= 52.55 rad/sec
   guadagno statico: 1 (0 db); risonanza relativa: 1.115 (0.9471 db)
   banda passante (-3db): 216.9 rad/sec
```

Le risposte dei sistemi ad un gradino unitario sono riportate in Figura 4.11 le cui caratteristiche sono

```
RISPOSTA AL GRADINO :
```

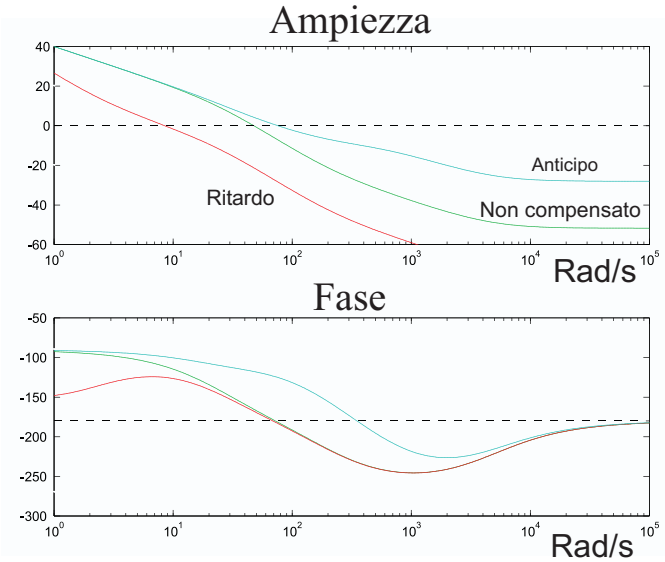


Figura 4.10: Diagrammi di Bode del sistema con rete anticipatrice, rete ritardatrice e senza compensazione.

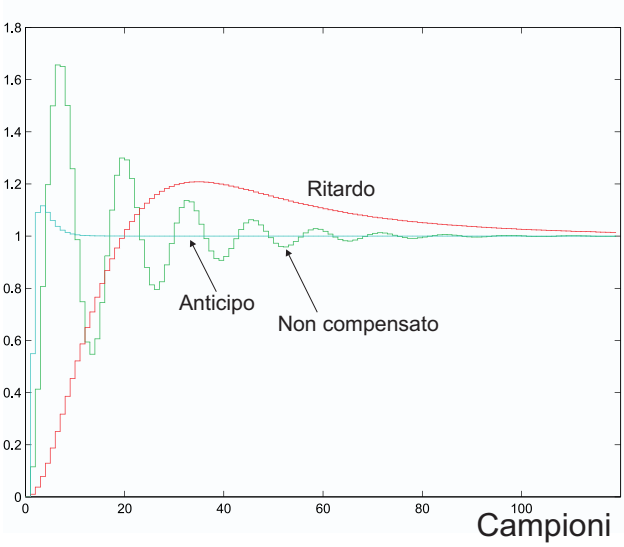


Figura 4.11: Risposta del sistema al gradino con rete anticipatrice, rete ritardatrice e senza compensazione.

```
massima sovraelongazione: 11.61 percento per k=3 camp (t=0.03 sec)
tempo di ritardo (al 50 percento): 1 camp (0.01 sec)
tempo di salita (dal 10 al 90 percento): 1 camp (0.01 sec)
tempo di assestamento (al piu'/meno 5 percento): 6 camp (0.06 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
```

4.5 Esercizi proposti in aula didattica

Si voglia progettare un regolatore digitale per il sistema con funzione di trasferimento

$$G(s) = \frac{0.2}{(1+s)(1+0.2s)} \tag{4.25}$$

adottando un tempo di campionamento di $T = 0.1$s e cercando di rispettare i seguenti requisiti relativi alla risposta ad un gradino del riferimento:

1. errore a regime nullo;
2. tempo di assestamento minore di 4s.

Si analizzi dapprima la possibilità di risolvere il problema di sintesi con un regolatore del tipo

$$G_{c1}(z) = \frac{K_1}{z - p_1} \tag{4.26}$$

e successivamente, come secondo tentativo, si ponga

$$G_{c2}(z) = K_2 \frac{z - z_2}{z - p_1} \tag{4.27}$$

con z_2 scelto in modo tale da cancellare il polo più "lento" di $G(z)$, ove $G(z)$ è la funzione di trasferimento ai dati campionati ottenuta da $G(s)$ attraverso un circuito di tenuta di ordine zero. Si determinino quindi K_1 e K_2 mediante l'analisi del luogo delle radici del sistema $G(z)G_{c1}(z)$ e $G(z)G_{c2}(z)$.

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

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

[7] C. Bonivento, C. Melchiorri, and R. Zanasi, *Sistemi di Controllo Digitale*. Bologna, Italy: Progetto Leonardo, Esculapio Ed., Marzo 1995.

[8] P. Bolzern, R. Scattolini, and N. Schiavoni, *Fondamenti di controlli automatici*. Milano: McGraw-Hill, I ed., Marzo 1998.

[9] B. C. Kuo, *Automatic Control Systems*. Englewood Cliffs, New Jersey: Prentice Hall, 7th ed., 1995.

[10] M. Tibaldi, *Note introduttive a Matlab e Control System Toolbox*. Bologna: Progetto Leonardo, 1993.

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

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