Laboratorio di Automatica

di Sergio Beghelli, Silvio Simani e Cesare Fantuzzi

Dipartimento di Ingegneria, Università di Ferrara

Versione 1.0, Marzo, 2000

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]. 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
```

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
```

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

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:

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

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

[3] i due formati sono completamante equivalenti

[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).

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

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

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

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

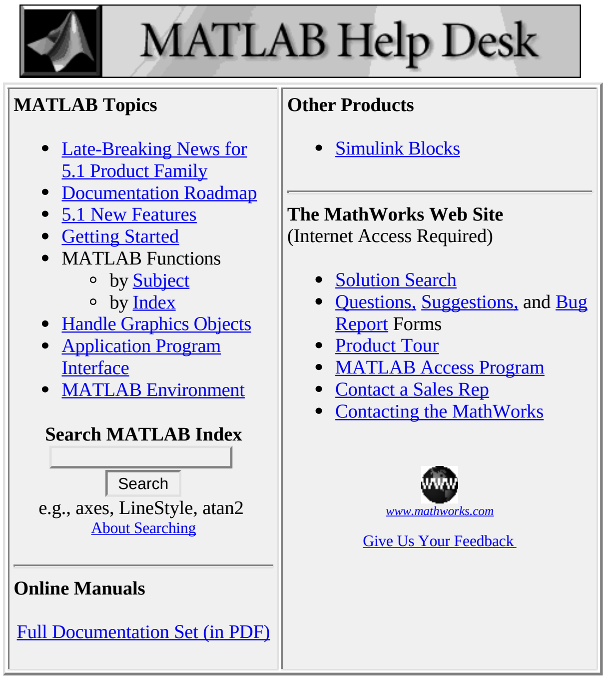

Figura 1.1: Helpdesk di *Matlab*

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

3. Data la terna $(\mathrm{A}_{n\times n}, \mathrm{B}_{n\times 1}, \mathrm{C}_{1\times n})$, calcolare la matrice $P = [B,\ A*B,\ \dots,\ A^{n-1}*B]$. 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 $Q = [C;\ (C*A);\ \dots;\ C*A^{n-1}]$ e effettuare la trasformazione `T` ricavata, come in precedenza, dall'immagine di `Q'` e dal suo complemento ortogonale. Si estraggano quindi le matrici $(\mathrm{A}_o, \mathrm{B}_o, \mathrm{C}_o)$.

   Si esegua l'esercizio con le matrici seguenti:

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

Risoluzione esercizi

1. La risoluzione di questo problema è assegnata come esercizio per l'esame finale.

2. Poiché vale la relazione $Q = [C;\ \ C * A;\ \ \ldots;\ \ C * A^{n-1}]$, la matrice `Q` può essere calcolata a linea di comando mediante le istruzioni

```
>>Q = [C];

>>Q = [Q;  C*A];

>>Q = [Q;  C * A^2];

   .         .
   .         .
   .         .

>>Q = [Q;  C * A^(n-1)];
```

   Il test del rango può essere effettuato ad ogni passo oppure alla fine attraverso la funzione `rank(Q)`

3. La matrice `Q` può essere anche calcolata in maniera automatica attraverso la funzione `obsv(A,C)`:

```
function Q = obsv(A,C)

n = size(A,1);
Q = C;
for i=1:n-1
    Q = [C; Q*A];
end
```

   In maniera analoga al caso precedente, il test del rango può essere effettuato attraverso la funzione `rank(Q)`.

4. Viene data la terna $(\mathtt{A}_{n\times n}, \mathtt{B}_{n\times 1}, \mathtt{C}_{1\times n})$.

```
A =

     2     4   -20
     0     3     6
     0     0     4


B =


     4
     6
     1
```

```
C =

     0     1     6
```

si costruisce la matrice Q

```
>>Q = obsv(A,C)

Q =

     0     1     6
     0     3    30
     0     9   138


>>rank(Q)

ans =

     2
```

Il rango di Q non è pieno, infatti la prima colonna è nulla. La base cercata per lo spazio generato dalle due colonne non nulle della matrice Q è calcolabile attraverso la funzione `orth(.)`, che fornisce un procedimento di ortonormalizzazione.

```
>>orth(Q')

ans =

    0.0000    0.0000
    0.0668   -0.9978
    0.9978    0.0668
```

Per ottenere il complemento ortogonale della matrice Q' si affianca alla matrice ottenuta una matrice identità di opportune dimensioni in modo da ottenere

```
>>orth([Q',eye(3)])

ans =

    0.0000    0.0000   -1.0000
    0.0668   -0.9978    0.0000
    0.9978    0.0668    0.0000
```

La matrice attraverso cui effettuare la trasformazione di base sarà T = [T1, T2] con

```
>>T1 = orth(Q')
```

```
T1 =

    0.0000    0.0000
    0.0668   -0.9978
    0.9978    0.0668

>>T = orth([P,eye(3)]);


>>T2 = T(:,3)

T2 =

   -1.0000
    0.0000
    0.0000
```

La nuova tripla di matrici (`Ao`,`Bo`,`Co`) ottenute dalla trasformazione lineare risulta

```
>>Ao =inv(T)*A*T

Ao =

    4.3956    0.0935    0.0000
   -5.9065    2.6044    0.0000
   19.6880    5.3275    2.0000


>>Bo = inv(T)*B

Bo =

    1.3987
   -5.9198
   -4.0000


>> Co = C * T

Co =

    6.0534   -0.5968    0.0000
```

da cui

```
>>Ao1 = Ao(1:2,1:2)

Ao1 =

    4.3956    0.0935
```

```
   -5.9065    2.6044


>> Bo1 = Bo(1:2)

Bo1 =

    1.3987
   -5.9198


>> Co1 = Co(1:2)

Co1 =

    6.0534   -0.5968
```

Bibliografia

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

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

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

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

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

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