MATLAB

Note Introduttive

Matlab

Cos'é MATLAB

- MATLAB (MATrix LABoratory) è un programma interattivo per il calcolo numerico, in cui l'elemento base è la matrice
- Viene oggi largamente utilizzato sia per la didattica che per la ricerca in svariati settori:
  - Calcolo matriciale
  - Matematica Applicata
  - Analisi di segnali
  - Grafica bi e tri-dimensionale
  - Controlli automatici
  - ...
- Esistono versioni per main-frame, PC, workstation
- Originariamente scritto in Fortran, ora in C.

Generalitá

- MATLAB è un interprete in grado di eseguire istruzioni native o contenute in files su disco
- Un M-file è un file ASCII che rispetta la sintassi MATLAB. Un utente può aggiungere delle nuove istruzioni aggiungendo i propri M-files, come $< nomefile >$.m. Essi verranno eseguiti semplicemente digitando $< nomefile >$. I *toolboxes* di MATLAB sono delle collezioni di M-files che risolvono particolari problemi (ad esempio il Control Toolbox).
- Elemento base di MATLAB è la matrice, che puo' essere costituita da elementi reali o complessi.
- Nota bene: Il punto e virgola al termine di un'istruzione MATLAB impedisce la visualizzazione del risultato dell'istruzione stessa. Il carattere % consente di introdurre commenti. Due o più punti ... consentono di estendere un'istruzione alla riga successiva.
- È disponibile un HELP in linea. Basta digitare help $< topic >$.

Introduzione di matrici

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

>>  [1 2 3; 4 5 6]
ans =
     1     2     3
     4     5     6

>> [1 2 3
    4 5 6]
ans =
     1     2     3
     4     5     6

>>  [1 2 3 ...
     4 5 6]
ans =
     1  2  3  4  5  6
```

Ancora sulle matrici

- Una matrice può essere generata mediante funzioni *built-in* di MATLAB. Ad esempio **zeros(n,m)**, **ones(n,m)**, **eye(n,m)** ...

  >> eye(2,2)

  ans =

  ```
  1     0
  0     1
  ```

- Gli elementi di una matrice sono indirizzabili mediante indice fra parentesi tonde

  >> A(2,1)

  ans =

  ```
      4
  ```

- Una matrice può essere costruita definendo le sue sottomatrici

  >> A=[1 2 3; 4 5 6];

  >> Y=[7 8 9];

  >> B=[A;Y]

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

Variabili ed espressioni

- Le istruzioni MATLAB sono spesso nella forma

  variabile = espressione

  o semplicemente

  espressione

- Le espressioni sono costituite da operatori, caratteri speciali, funzioni, variabili e numeri

  operatori:

  + * - / \ ^

- funzioni: nomi simbolici con argomenti fra parentesi: eye(2,2)

- numeri: reali, ad es. 5, e complessi, ad es. 5 + 2*i o indifferentemente 5 + 2*j

Fine della seduta di lavoro

- Per uscire da MATLAB si possono usare **quit** o **exit**. Uscendo da MATLAB tutte le variabili del workspace vengono perse. Per conservarle si può eseguire l'istruzione **save** e viene creato un file MATLAB.MAT. Rientrando in MATLAB si possono recuperare le variabili mediante l'istruzione **load**. **save** e **load** possono essere utilizzate anche specificando il nome del file in cui si vuole salvare le variabili.

Operazioni su matrici

- Trasposizione

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

  >> B=A'

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

- >> [1 2 3]'

  ans=

```
 1
 2
 3
```

- somma e sottrazione vengono fatte elemento per elemento

  >> [1 2 3; 4 5 6] + [3 3 3; 3 3 3]

```
ans =
4       5       6
7       8       9
```

- >> [1 0 2] - 5

```
ans =    -4      -5      -3
```

Operazioni su matrici

- il prodotto è definito come il prodotto matriciale righe per colonne

  >> [1 2; 3 4; 5 6]*[4 4; 5 5]

```
ans =
14      14
32      32
50      50
```

- il prodotto scalare (elemento per elemento) viene indicato con .*

  >> [1 2 3 4 5 6] .* [7 8 9 10 11 12]

  ans =

  7 16 27 40 55 72

- Se **A** è una matrice quadrata e **p** è uno scalare

  A^p=A*A*A

- L'elevamento a potenza scalare (elemento per elemento) si indica con

  .^

Operazioni su matrici

- La divisione viene effettuata mediante due simboli
  / e \
  Supponendo che A sia una matrice quadrata non singolare
  X=B/A
  è soluzione di $X * A = B$ o $X = B * inv(A)$
  X=A\B è soluzione di $A * X = B$ o $X = inv(A) * B$
  con X che ha le stesse dimensioni di B.
- La divisione scalare (elemento per elemento) viene indicata con
  ./
  >> [1 2 3 4 5 6] ./ [7 8 9 10 11 12]
  ans =

```
0.1429   0.2500   0.3333   0.4000   0.4545   0.5000
```

Operazioni su matrici

- In MATLAB le espressioni del tipo **exp(A)**, **sqrt(A)**, **log(A)** sono considerate *array operations*, ovvero definite su ciascun elemento della matrice A. Le corrispondenti funzioni trascendenti matriciali si ottengono aggiungendo un **m** al finale della funzione. Ad esempio

  expm(A) equivale a $e^A = \sum_{i=0}^{\infty} \frac{A^i}{i!}$

- É possibile utilizzare convenientemente anche degli operatori relazionali,

  < minore di > maggiore di

  >= maggiore uguale <= minore uguale

  == uguale a ~= diverso da

  e operatori logici

  & and | or ~not

  Ad esempio, mediante l'istruzione **find**, si possono selezionare sottoinsiemi di matrici usando gli operatori relazionali

  >> A=[4 1 8 0];

  >> find(A>2)

  ans = 1 3

- Sul manuale: **Funzioni logiche e relazionali**, **Funzioni trigonometriche**, **Funzioni matematiche elementari**

Operazioni su matrici

- Esistono varie funzioni per generare dei vettori

  >> x=1:5

```
x =
1     2     3     4     5
```

- >> x=1:0.5:2

```
x =
1.0000    1.5000    2.0000
```

- >> x=linspace(-1,1,5)

  x =

  -1.0000 -0.5000 0 0.5000 1.0000

  >> x=logspace(-1,1,5)

```
x =
0.1000    0.3162    1.0000    3.1623    10.0000
```

- Esistono inoltre varie funzione per la manipolazione di matrici, come ad esempio **diag(x), fliplr, reshape, ...**

Operazioni su matrici

- A(x,y) individua la matrice che si ottiene utilizzando gli elementi del vettore x come indici di riga e gli elementi del vettore y come indici di colonna.

  >> A =[1 2 3; 4 5 6];

  >> A([1 2],[2 3])

  ```
  ans =
  2     3
  5     6
  ```

- >> A(:,1)

  ans =

  1

  4

- MATLAB mette a disposizione anche alcune funzioni *column-oriented*, che agiscono sulle colonne delle matrici e restituiscono un vettore riga. Ad esempio la funzione **mean** che esegue la media per colonne

  >> A =[1 2 3; 4 5 6];

  >> mean(A)

  ```
  ans=
  2.500     3.500     4.500
  ```

Operazioni su matrici

- Autovalori e autovettori: data la matrice quadrata **A** di dimensioni $n \times n$ l'istruzione

  a=eig(A)

  restituisce il vettore colonna **a** $n \times 1$ costituito dagli autovalori della matrice **A**. L'istruzione

  [V,D]=eig(A)

  restituisce la matrice **V** $n \times n$ degli autovettori normalizzati e la matrice diagonale **D** $n \times n$, che presenta sulla diagonale gli autovalori della matrice **A**.

- Altre funzioni consentono di calcolare il determinante (**det(A)**), l'inversa (**inv(A)**), il rango (**rank(A)**) e così via.

Operazioni su matrici

- Fattorizzazione triangolare: data la matrice quadrata **A** di dimensioni $n \times n$ l'istruzione

  [L,U]=lu(A)

  restituisce i fattori della fattorizzazione A=LU, con **L** triangolare inferiore e **U** triangolare superiore.

- Singolar value decomposition: data la matrice **A** di dimensioni $n \times m$ l'istruzione

  [U,S,V]=svd(A)

  restituisce i fattori della fattorizzazione $A = USV^T$, con **U** $n \times n$ e **V** $m \times m$, ortogonali e **S** diagonale $n \times m$ con i valori singolari di **A** sulla diagonale principale.

Rappresentazione dei polinomi

- Un polinomio è rappresentato da un vettore riga che contiene i coeficienti del polinomio in ordine decrescente delle potenze del polinomio. Ad esempio

  p=[1 0 -6 3]

  permette di rappresentare $x^3 - 6x + 3$.

- **r=roots(p)** permette di trovare le radici di p.

- **p=poly(r)** permette di trovare i coefficienti del polinomio con radici nel vettore r.

  >> p=[1 0 -6 3];

  >> r=roots(p)

  r =

  ```
  -2.6691
  2.1451
  0.5240
  ```

- >> poly(r)

  ans =

  ```
  1.0000      0.0000      -6.0000      3.0000
  ```

Operazione sui polinomi

- Il prodotto di due polinomi a(x) e b(x) si ottiene effettuando la convoluzione dei loro coefficienti (**conv(a,b)**). Ad esempio

  $a(x) = x^2 + 2x + 3$ e $b(x) = 4x^2 + 5x + 6$

  $a(x) * b(x) = 4x^4 + 13x^3 + 28x^2 + 27x + 18$

  si ottiene mediante

  >> a=[1 2 3]; b=[4 5 6];

  >> c=conv(a,b)

  ```
  c =
  4       13       28       27       18
  ```

- la divisione fra due polinomi c(x) e a(x) si ottiene effettuando la deconvoluzione dei loro coefficienti mediante l'istruzione **deconv(c,a)**

  >> [q r]=deconv(c,a)

  q =

  4 5 6 % polinomio quoziente

  r =

  0 0 0 0 0 % polinomio resto

- Il polinomio caratteristico della matrice quadrata A si ottiene con l'istruzione **poly(A)**. Per valutare il polinomio **p** in corrispondenza di un valore **k** si usa l'istruzione **polyval(p,k)**.

Operazioni sui polinomi

- Sviluppo in fratti semplici
  $\frac{n(s)}{d(s)} = \frac{r_1}{s-p_1} + \frac{r_2}{s-p_2} + \ldots + \frac{r_m}{s-p_m} + k(s)$
  con **[R,P,K]=residue(n,d)**, dove R è il vettore dei residui, P quello dei poli e K contiene i coefficienti del polinomio diretto.
- Ad esempio $\frac{s+1}{s^2+5s+6}$ si risolve con
  >> n=[1 1]; d=[1 5 6];
  >> [R P K]=residue(n,d)
  R = 2
  -1
  P = -3
  -2
  K = []
  e quindi lo sviluppo è $\frac{2}{s+3} - \frac{1}{s+2}$;
- MATLAB consente anche di calcolare il polinomio interpolante dati due vettori **x** e **y** di ordine m mediante il metodo dei minimi quadrati:
  p=polyfit(x,y,n)
  restituisce il polinomio **p** di grado **n** che interpola ottimamente i dati nel senso dei minimi quadrati

Grafici

- l'istruzione **plot** consente di effettuare grafici bidimensionali.

  plot(x) con x vettore $1 \times n$ produce un grafico con le ascisse costituite dagli indici 1:n e con x in ordinata

  plot(x,y) con x e y vettori $1 \times n$ produce un grafico con x in ascissa e y in ordinata

  plot(x,y,z,h) con x, y, z e h vettori $1 \times n$ produce due grafici, uno con x in ascissa e y in ordinata e l'altro con z in ascissa e h in ordinata

  plot(x,y,'- -') produce un grafico con linea tratteggiata

  ...

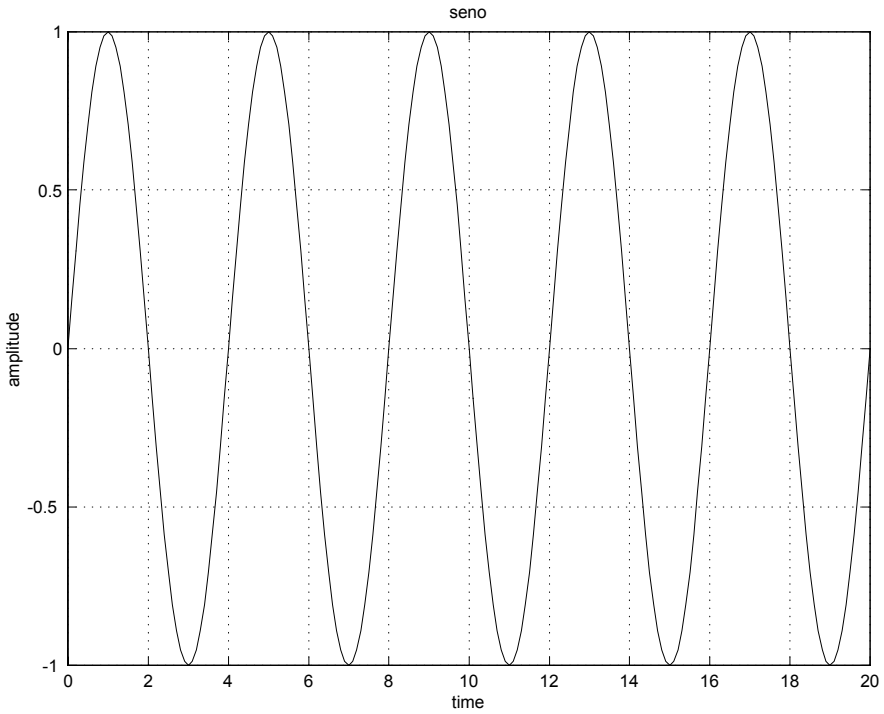
- l'istruzione **subplot(mnp)** dove m e n sono gli interi 1 e 2 e p è compreso fra 1 e m*n, suddivide la pagina grafica in $m \times n$ finestre e seleziona la finestra p seguendo l'ordine $\begin{matrix} 1 & 2 \\ 3 & 4 \end{matrix}$

- l'istruzione **grid** visualizza una griglia sullo schermo; le istruzioni **title('titolo')**, **xlabel('label')** e **ylabel('label')** consentono di mettere il titolo e delle etichette sull'asse x e y, rispettivamente.

- l'istruzione **axis([xiniz xend yiniz yend])** consente di riscalare il grafico nel settore specificato.

Grafici

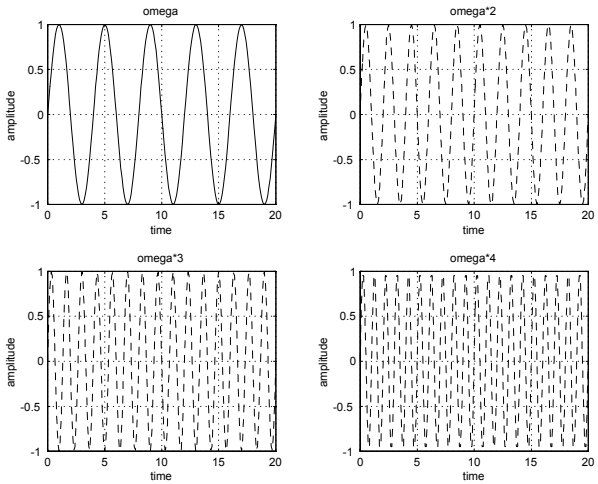
- >> time=[0:0.1:20];omega=pi/2;
  >> y=sin(omega*time);
  >> plot(time,y),grid,title('seno');
  >> xlabel('time'),ylabel('amplitude')



Grafici di esempio

- >> y1=sin(omega*time);y2=sin(2*omega*time);
  >> y3=sin(3*omega*time);y4=sin(4*omega*time);

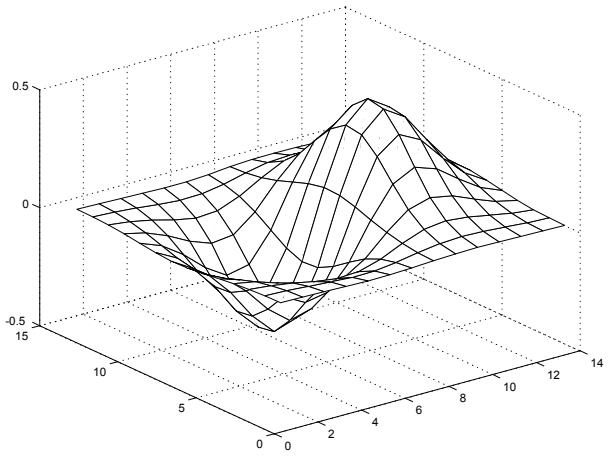
>> subplot(2,2,1), plot(time,y1),grid,title('omega'),
>> xlabel('time'),ylabel('amplitude')
>> subplot(2,2,2), plot(time,y2,'- -'),grid;
>> title('omega*2')
>> xlabel('time'),ylabel('amplitude')
>> subplot(2,2,3), plot(time,y3,'- .'),grid;
>> title('omega*3')
>> xlabel('time'),ylabel('amplitude')
>> subplot(2,2,4), plot(time,y4),grid,title('omega*4')
>> xlabel('time'),ylabel('amplitude')



Grafici tridimensionali

- l'istruzione **mesh(A)** crea una prospettiva di un grafico a tre dimensioni definito dalle ordinate corrispondenti agli elementi della matrice A su un piano x-y la cui griglia rettangolare è definita dagli indici della matrice stessa. Per cambiare la griglia si usa l'istruzione **meshdom**
- Ad esempio, volendo produrre il grafico

  $z = xe^{-x^2+y^2}$ in $-2 \leq x \leq 2, -2 \leq y \leq 2$
- >> x=-2:0.1:2; y=-2:0.1:2; [x,y]=meshgrid(x,y);
- >> z=x.*exp(-x.^2-y.^2);
- >> mesh(z)



M-files

- I files che contengono istruzioni esguibili da MATLAB sono detti m-files in quanto hanno come suffisso **.m**
- Se un m-file memorizza una sequenza di istruzioni è detto *script-file*. Se contiene una nuova funzione viene detto *function file*.
- Un **function file**:

  function y=linear(x,alfa,beta)

  y= alfa + beta *x;

  questo file restituisce una trasformazione lineare del vettore x;

  >> x=[1 4 5 9];

  >> linear(1,4,x)

```
ans =
5     8     9     13
```

- MATLAB possiede le usuali istruzioni di controllo che consentono di effettuare programmazione.

  for condizione ...istruzioni ... **end**

  while condizione ...istruzioni ... **end**

  if condizione ...istruzioni ... **else** ... istruzioni **end**

M-files

- La prima volta che una funzione viene utilizzata essa viene compilata e posta in memoria.
- Quando l'interprete di MATLAB trova un nome, ad esempio prova, segue, nell'ordine, i seguenti passi:

  1. cerca nel workspace la variabile di nome *prova*
  2. cerca una funzione *built-in* di nome *prova*
  3. cerca un m-file di nome *prova* nella directory da cui si é lanciato MATLAB
  4. cerca un m-file di nome *prova* nel path indicato in una speciale variabile di sistema detta MATLABPATH. La variabile MATLABPATH può essere aggiornata nel sistema operativo Unix definendo la variabile di shell $MATLABPATH nel file di configurazione .cshrc, che si trova in ogni directory corrispondente ad un account.

M-files

- script-files:
  - **echo-on** abilita la riproduzione sullo schermo delle istruzioni in esecuzione.

    echo-off disabilita la riproduzione sullo schermo delle istruzioni in esecuzione.

- function-files:
  - **echo** nome-funzione **on** abilita la riproduzione sullo schermo delle istruzioni in esecuzione dalla funzione.

    echo nome-funzione **off** disabilita la riproduzione sullo schermo delle istruzioni in esecuzione.

    echo on all abilita la riproduzione sullo schermo delle istruzioni di tutte le funzioni

    echo off all disabilita la riproduzione sullo schermo delle istruzioni di tutte le funzioni

- **pause** sospende l'esecuzione fino a che non viene premuto un tasto della tastiera. **pause(n)** sospende l'esecuzione per n secondi.
- **input** permette di acquisire dati da tastiera

Utilitá

- Per rimuovere variabili o funzioni dalla memoria, si utilizza la funzione **clear**.

  clear elimina tutte le variabili dal workspace **clear x** elimina la variabile (o la funzione) x dal workspace **clear functions** elimina tutte le funzioni dal workspace **clear all** elimina tutte le variabili, le funzioni ed i file eseguibili esterni (mex files) dalla memoria

- **diary (on-off)** abilita e disabilita la registrazione di tutta la sessione di lavoro in un file chiamato DIARY.

Utilitá

- L'istruzione **dir** elenca i files contenuti nella directory corrente.
- L'istruzione **type** nomefile lista il contenuto di nomefile.
- Per effettuare delle istruzioni del sistema operativo (ad esempio listare i files della directory corrente in Unix), bisogna far precedere il comando dal punto esclamativo (ad esempio **!ls**).
- Per sapere quali variabili sono state create e sono residenti in memoria, si usa l'istruzione **who**. Il comando **whos** elenca le variabili in uso con le dimensioni assegnate.
- il comando **pack** compatta le aree di memoria utilizzate registrando su disco le variabili in uso, pulendo la memoria e ricaricando da disco le variabili registrate.
- Per sveltire l'esecuzione è preferibile assegnare inizialmente le dimensioni dei vettori che si useranno invece che incrementarle ogni volta.

  >> x=[]; for i=1:10 x(i)=i^2; end % non efficiente

  >> x=zeros(1,10);

  >> for i=1:10 x(i)=i^2; end % efficiente
- l'istruzione **format** imposta il formato di visualizzazione (long, short, hex, ...)

Lettura da file

- In MATLAB è possibile leggere un file di dati formattato utilizzando l'istruzione **fscanf**.

  1. In primo luogo bisogna aprire il file con l'istruzione **fid=fopen (nomefile)** (dove fid è un puntatore al file aperto, che deve essere un numero maggiore di 0).
  2. Quindi si utilizza il comando
     matrice=fscanf(fid,'format',size),
     [matrice,count]=fscanf(fid,'format',size),
     dove matrice contiene i dati letti e count (opzionale) contiene il numero di dati letti con successo.
     I parametri di fscanf sono fid (il puntatore a file), il formato, che puo' essere:
     %d numero decimale
     %e,%f,%g numero in floating point
     %s stringa di caratteri
     ed infine la dimensione che può essere:
     - n numero di elementi (in questo caso matrice sarà un vettore di n elementi)
     - inf fino a fine file (in questo caso matrice sarà un vettore)
     - [n,m] numero di righe e colonne della matrice (in questo caso matrice sarà una matrice n per m)

Lettura da file: Esempio

```
% apro il file, che contiene 20 colonne ed un certo numero (non noto a priori) di dati
% in floating point
fid=fopen('testo');
% il formato sara' quindi %f, inoltre
% dato che la lettura del file avviene per riga,
% e il riempimento della matrice per colonne,
% devo trasporre il file che ho ottenuto per ricostruire
% il file come da originale
dati=fscanf(fid,'%f',[20 inf])';
```

Scrittura su file

- In MATLAB è possibile scrivere un file di dati formattato utilizzando l'istruzione **fprintf**.

  1. In primo luogo bisogna aprire il file con l'istruzione **fid=fopen (nomefile,'opzioni')** (dove fid è un puntatore al file aperto, che deve essere un numero maggiore di 0, mentre opzioni possono essere, 'r' (read), 'w' (write) and 'a' (append).
  2. Quindi si utilizza il comando **fprintf(fid,'format',matrice)**, dove matrice contiene i dati da scrivere ed il formato, che oltre al formato numerico (vedi lettura da files) può contenere anche altri caratteri speciali, come
     \n
     (il newline).
     Esempio:
     x=0:.1:1;
     y=[x; exp(x)];
     fid=fopen('exp.txt','w');
     fprintf(fid,'%f %f \n',y);
     flose(fid)
     Crea un file di due colonne con la variabile ed il suo esponenziale.