Appunti di Octave

Ing.Giampiero Granatella
Basato sul manuale *“Gnu Octave di John W. Eaton”*

Introduzione.

Panoramica dell'utilizzo di Octave.

Che cos'è

Octave è un linguaggio interpretato di alto livello per eseguire calcoli matematici distribuito sotto licenza GNU. E' stato sviluppato dal dipartimento di Ingegneria Chimica dell'University of Wisconsin-Madison e dell'Univeristy of Texas.
E' liberamente scaricabile dal sito www.octave.org sia per piattaforma Linux, sia per Windows ed è compreso in gran parte delle distribuzioni Linux. Le sue funzionalità sono molto simili a quelle di Matlab® di Mathworks.

Volo di uccello sulle funzionalita di Octave

Per lanciare il programma Octave basta digitare `octave` nella propria shell. Per uscire dal programma digitare `quit` o `exit`. Lanciato il programma si entra in una sessione interattiva avente come prompt `octave>` o `>>`.
Al prompt di octave si possono effettuare operazioni matematiche quali somma, prodotto, elevazione a potenza, radice quadrata, ecc. Il risultato è memorizzato in una variabile di default chiamata ans.

```
>> (2*3)-4^3
ans = -58
```

Esempio di radice quadrata.

```
>> sqrt(4)
ans = 2
```

Octave non ha bisogno di definire i tipi delle variabili, queste vengono definite a run-time quando vengono utilizzate.

```
>> pi = 3.1415
>> s = 'Ciao Mondo'
```

Per visualizzare il contenuto della variabile basta digitare il nome della variabile stessa.

Essendo pensato per calcoli matematici strutture dati come vettori e matrici sono già predisposti come tipi primitivi. Il seguente esempio inizializza un vettore (3x1, 3 colonne ed 1 riga) memorizzato nella variabile `A`.

```
>> A = [1, 2, 12]
```

I valori possono essere separati da spazi o da virgole.
Per definire una matrice le righe devono essere separate da un punto e virgola. Nell'esempio seguente viene definita una matrice 3x3.

```
>> A = [2, 3, 4; 12, 13, 3; 4, 1, 2]
```

Operazioni su Matrici

Octave permette in maniera semplice e immediata di effettuare operazioni su matrici. Di seguito ne viene mostrata una carrellata.
Nei seguenti esempi utilizzeremo una matrice 'A' 3x3.

```
>> A = [1,2,12; 4,5,6; 7,8,9]
A =

        1        2       12
        4        5        6
        7        8        9
```

Moltiplicazione di una matrice per uno scalare
Il risultato di una moltiplicazione di una matrice per uno scalare è una matrice che ha le stesse dimensioni e ogni suo valore è uguale al valore originario moltiplicato per lo scalare.

```
>> B = A * 2
A =

         2         4        24
         8        10        12
        14        16        18
```

Matrice trasposta
Una matrice traposta è una matrice il valore originario a_{ij} diviene a_{ji}.

```
>> B= A'
A =

         1         4         7
         2         5         8
        12         6         9
```

Matrice inversa
L'inverso di una matrice 'A^{-1}' è una matrice tale che $A*A^{-1}=I$.
Dove I è una matrice di tutti 0 tranne i valori sulla diagonale che sono uguali ad 1.

```
>> inv(A)
ans =

   0.11111  -2.88889   1.77778
  -0.22222   2.77778  -1.55556
   0.11111  -0.22222   0.11111

>> A * inv(A)
ans =

   1.00000   0.00000   0.00000
  -0.00000   1.00000   0.00000
  -0.00000   0.00000   1.00000
```

Determinante di una Matrice

```
>> det(A)
ans = -27.000
```

Soluzione di un sistema lineare in n incognite

Un sistema lineare in n incognite è così fatto

$a_{11} x_1 + a_{12} x_2 + ... + a_{1n} x_n = b_1$
$a_{21} x_1 + a_{22} x_2 + ... + a_{2n} x_n = b_2$
...
$a_{n1} x_1 + a_{n2} x_2 + ... + a_{nn} x_n = b_n$

E può essere scritto come $A\underline{x}=B$
Così che $\underline{x}=A^{-1}B$

Risolvere il suddetto sistema in Octave è semplicissimo basta utilizzare l'operatore '\'.

```
>> A=[2,3,4;1,7,9;3,2,1]
A =
```

```
         2         3         4
         1         7         9
         3         2         1

>> B=[1;7;3]
B =

         1
         7
         3

>> A\B
ans =

   -1.05000
    4.30000
   -2.45000
```

Soluzione di equazioni differenziali

Octave ha funzioni per risolvere equazioni differenziali del tipo:

$$\frac{dx}{dt}=f(x,t), f(t_0)=x_0$$

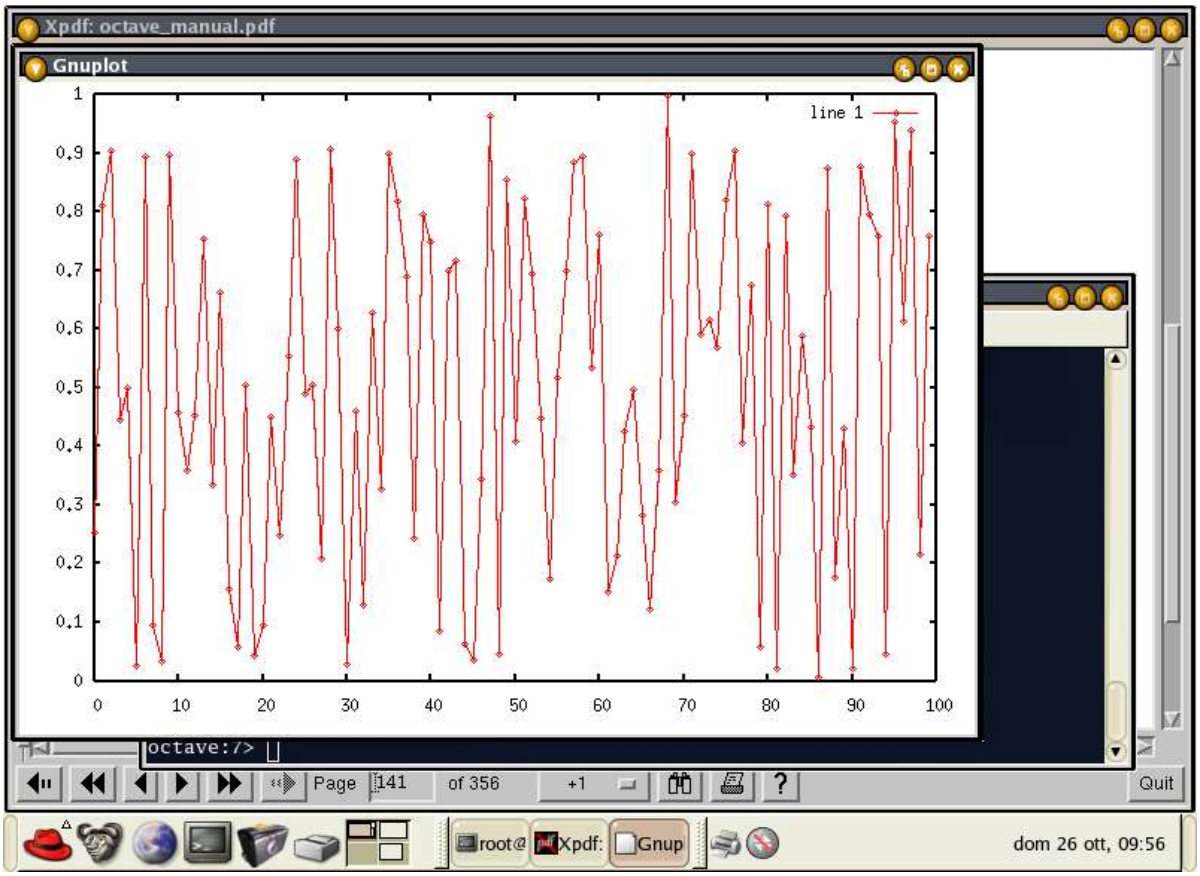
In questo caso è necessario, innanzitutto definire la funzione da integrare, come vedremo in seguito.

Grafica

Octave permette, appoggiandosi al software gnuplot di avere una rappresentazione grafica dei propri calcoli.
Nell'esempio seguente vediamo la rappresentazione di numeri casuali in un un grafico 2D

```
>> gplot rand (100,1) with linespoints;
```

Il grafico prodotto è il seguente:


Xpdf: octave_manual.pdf
Gnuplot
line 1
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0
10
20
30
40
50
60
70
80
90
100
octave:7>
Page
141
of 356
+1
Quit
root@
Xpdf:
Gnup
dom 26 ott, 09:56

Iniziare ad usare Octave

Octave viene lanciato semplicemente eseguendo l'eseguibile Octave.
In ogni caso sono possibili alcuni opzioni da lanciare al momento dell'avvio.

```
>> octave nome_file
```

Esegue i comandi all'interno di nome file e esce al termine dell'esecuzione.

Ad esempio l'opzione –version riferisce il numero della versione.
L'opzione –exec-path *path* imposta la directory dove si trovano i programmi da eseguire.
L'opzione -h fornisce un breve help con l'elenco completo delle opzioni d'avvio.

Il comando "diary"

Diary tiene traccia di tutta la sessione (input ed output) fatta sul terminale.
Le opzioni sono *on* e *off* per attivarlo o spegnerlo e *nome_file* per salvare il diario su un file.

Errori in Octave

Octave segnala due tipe di errori quelli dovuti ad errori di interpretazione di comandi non riconosciuti (parsing errors) e quelli durante l'esecuzione quindi durante la valutazione dei comandi.

Programmi in Octave

Octave permette di scriversi propri programmi da eseguire.Questi programmi sono script che vengono interpretati da Octave. Il linguaggio con cui si scrivono programmi per Octave e simile al C.

Iniziamo con il buon vecchio Hello World.
Il file da scrivere è il seguente:

```
#! /usr/bin/octave -qf
# Esempio di programma in Octave
printf("Hello World\n");
```

Notiamo subito l'utilizzo del printf in maniera analoga al C.
Nella prima riga viene dato, dopo "#!" , il percorso completo al programma Octave. Viene in pratica specificato come in tutti gli script Linux quale interprete utilizzare. L'opzione qf serve a non far stampare i messaggi di benvenuto di Octave.
La seconda linea ci mostra un commento. Negli script di octave i commenti iniziano con il simbolo del cancelletto oppure dal simbolo di percentuale "%".

Quando si definisce una funzione e viene inserito un blocco di commenti dopo il nome questo blocco di commenti viene utilizzato come help della funzione.

```
octave:4> help gplot;

*** gplot:

Produce 2-D plots using gnuplot-like command syntax.
```

Personalizzare il prompt

Il prompt dei comandi in Octave è memorizzato in alcune variabili globali.
In PS1 abbiamo il prompt principali dei comandi:
`octave>`
in PS2 viene memorizzato il prompt di octave quando questo è in attesa di ulteriori parametri
>

Per personalizzare il prompt basta cambiare il valore di queste variabili.
Possono essere inseriti il nome utente \u o la data \d o il nome dell'host \h.
Consultare il manuale per un elenco completo.
Attenzione che per inserire questi valori in una stringa bisogna utilizzare \\ invece di \. Perchè \ è il simbolo di escape.

Tipi di Dati

Tutte le versioni di Octave hanno un insieme predefinito di tipi di dati, che includono anche vettori, matrici e numeri complessi. E' anche possibile creare nuovi tipi di dati attraverso le strutture, con definizioni del tutto simili ad istruzioni C.

Tipi di dati built-in

Numeri

Tutti i numeri in Octave sono trattati come numeri in virgola mobile a doppia precisione (double), questo include, naturamlmente, anche gli elementi per vettori e matrici.
Il massimo ed il minimo valore possibile sono memorizzati in **realmax** e **realmin.**

Numeri complessi

I numeri complessi vengono scritti come parte reale più parte immaginaria.
Ad esempio:

```
2+4i
```

La lettera i sta per la radice di meno uno.

Stringhe

Una stringa è una sequanza di simboli solitamente caratteri. Le stringhe sono gestite in octave come in C, racchiudendole fra doppi apici “.

Strutture

E' possibile definire strutture con sintassi simile al C.

Valori numerici

Dimensioni

```
octave> columns(a)
```

ritorna il numero di colonne

```
octave> rows(a)
```

ritorna il numero di colonne

```
octave> length(a)
```

ritorna la lunghezza di un vettore in caso di una matrice ritorna il maggiore fra righe e colonne

```
octave> size(a)
```

ritorna il numero di righe e colonne di una matrice

```
octave> isempty(a)
```

ritorna 1 se la matrice è nulla (numero di righe o colonne uguale a zero)

Matrice

E' facile definire una matrice in octave come abbiamo già visto. Le dimensioni dela matrice sono calcolate automaticamente dal programma.

```
> a = [1 2; 3 4]
```

E' poi possibile definire matrici a partire da espressioni.

```
> b = [a a]
> b = [a; a]
```

Intervalli

Un intervallo è un metodo utile per definire un vettore
Ad esempio

```
1:5
```
E' un vettore [1 2 3 4 5]. Si può aggiungere ancora un elemento che è il passo.
```
octave:2> 1:2:10
ans =

   1   3   5   7   9
```
Il passo può anche essere negativo.
```
octave:4> 1:-1:-10
ans =

    1    0   -1   -2   -3   -4   -5   -6   -7   -8   -9  -10
```

Valori logici
A differenza del C sono definiti dei dati primitivi in Octave per definire i valori logici booleani, questi valori sono *true* e *false.*
Predicati per valori numerici
- isnumeric(x) ritorna 1 in caso x sia un valore numerico
- isreal(x) ritorna 1 in caso x sia un valore reale
- is_complex(x), 1 in caso di numero complesso
- is_matrix(1) 1 in caso di matrice
- is_vector(1) 1 in caso di vettore
- is_scalar(1) 1 in caso di scalare
- is_square(x) 1 in caso di matrice quadrata
- is_symetric(x,tol) 1 in caso di matrice simmetrica usando la tollerenza specificata
- is_bool(x) 1 in caso di valore booleano.

Stringhe
Sono una sequenza di caratteri racchiusi da doppi apici oppure da apice singolo.
Alcuni caratteri, non rappresentabili, sono visualizzati tramite l'uso del back-slash \ o carattere di escape.
Esempio la nuova linea corrsisponde al simbolo \n

Le stringhe possono essere concatenate utilizzando la notazione delle matrici,
```
octave:5> ["Ciao", " a", " tutti" ]
ans = Ciao a tutti
```
Operazioni sulle stringhe
- blanks(n) resistuisce una stringa con n spazi
- int2str, num2str converte un numero in una stringa
- com2str(zz) converte un numero complesso in una stringa
- setstr(x) converte una matrice in una stringa, ogni valore è valutato con il corrispettivo ASCII
- strcat(s1, s2, ...) concatena più stringhe
- string_fill_char rende tutti le stringhe di uguale lunghezza aggiungendo il carattere immesso come parametro.

```
octave:14> string_fill_char = "_";
octave:15> ["aa"; "jjjj"; "q"]
ans =

aa__
jjjj
q___
```

- str2mat(s1,..) ritorna una matrice che ha per righe le stringhe passate

- istr(x) 1 se x è una matrice.

Search e Replace

- deblank(s) rimuove gli spazi
- findstr(s, pattern) cerca il pattern nella stringa

```
octave:17> findstr("bla bla bla ... bla","la")
ans =

   2   6  10  18
```

- index(s, pattern) ritorna la prima posizione
- split (s,t) divide la stringa s utilizzando il pattern t

```
octave:18> split("bla bla bla ... bla"," ")
ans =

bla
bla
bla
...
bla
```

- strcmp(s1,s2) confronta s1, s2
- strrep(s1,s2,s3) rimpiazza in s1, s2 con s3

```
octave:20> strrep("Ciao Mondo", "Mondo", "a tutti")
ans = Ciao a tutti
```

- substr(s, inizio, fine) Preleva una sottostringa

Altri conversioni

- bin2dec,
- dec2bin,
- dec2hex
- hex2dec
- str2num
- toascii
- tolower
- toupper
- undo_string_escapes

Funzioni sui caratteri

- Isalpha(s) controlla se i caratteri nella stringa sono alfabetici

```
octave:23> isalpha("q6/bxc)(°")
ans =

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

- isalnum, alfabetci o numerici
- isacii
- isdigit
- islower
- isspace
- isupper

Strutture dati

Le strutture dati sono simili a quelle del C, anche in questo caso si definiscono durante il loro utilizzo.

```
octave:28> citta.nome="Genova";
octave:29> citta.temp=6;
octave:30> citta.regione="Liguria";
octave:31> citta
citta =
```

```
{
  regione = Liguria
  temp = 6
  nome = Genova
}
```

A differenza del C le strutture possono essere copiate. Le strutture possono essere ritornate da funzioni. Vediamo un esempio.

```
octave:33> function y=f(x)
> y.re=real(x);
> y.img =imag(x);
> endfunction

octave:34> f(5+4i)
ans =
{
  img = 4
  re = 5
}
```

Funzioni legate alle strutture

- is_struct
- struct_elements

```
octave:42> struct_elements(citta)
ans =

regione
temp
nome
```

Contenitori

Liste

```
octave:43> lista=list("a", "b", "c")
lista =
(
  [1] = a
  [2] = b
  [3] = c
)
```

Una lista è un insieme ordinato di elementi

Per accedere ad un elemento basta richiamare il nome della lista con la posizione ricercata.

```
octave:45> lista(1)
ans =
(
  [1] = a
)
```

Funzioni sulle liste

- append, aggiunge elementi alla lista
- reverse, inverte la lista
- is_list ritorna 1 se l'argomento è una lista.

Variabili

Una variabile è un nome associato ad un valore per un successivo riutilizzo. Le regole per definire i nomi di variabili in Octave sono identiche a quelle del C. Alcune regole.

- Il nome di una variabile in Octave non deve superare i 35 caratteri.
- I nomi di variabile che iniziano con un doppio underscore sono per variabili di sistema.
- Octave è case sensitive.
- Una variabile di qualsiasi tipo, può in caso di assegnazione successiva diventare un altro tipo di variabile.

Variabili globali

Una variabile dichiarata globale è una variabile che può essere utilizzata in diversi ambiti, ad esempio anche all'interno di una funzione senza dove essere passata.
Esempi di variabili globali.

```
octave:2> global x
octave:3> global test="Prova"
octave:4> global b, c=1
```

Passare una variabile globale ad una funzione crea una copia di tale variabile alla funzione senza toccare la variabile globale.

```
octave:9> global x=1;
octave:10> function f(x)
> x=2;
> endfunction
octave:11> f(3)
octave:12> x
x = 1
```

Alcune funzioni per le variabili globali

- is_global(x)

Stato delle variabile

Una variabile può essere pulita (scollegata dal valore a cui apparteneva) tramite la funzione *clear nome_variabili*. Se non si passano parametri vengono cancellate tutte le variabili.

Il comando *who -all* mostra tutte le variabili (built-in e definite dall'utente), e le funzioni all'interno del sistema.
Il comando type mostra il tipo di una variabile o di una funzione

```
octave:15> type f
f is a user-defined function:

function f (x)
  x = 2;
endfunction
octave:16> type test
test is a user-defined variable
"Prova"
```

Espressioni

Un espressione è un istruzione od un comando passato all'interprete.

Estrazioni da matrici

```
octave:19> a=[1,2,3;4,5,6;7,8,9]
a =

  1  2  3
  4  5  6
  7  8  9

octave:20> a(1, :)
ans =

  1  2  3

octave:21> a(2, :)
ans =

  4  5  6

octave:22> a(1:2, :)
ans =

  1  2  3
  4  5  6

octave:23> a([1,3], :)
ans =

  1  2  3
  7  8  9
```

Funzioni

Una funzione è un insieme di espressioni, a cui viene dato un nome per un successivo uso. I parametri possono essere passati alla funzione tramite un elenco di variabili separate da virgola. Se non ci sono argomenti le parentesi possono essere omesse.

Chiamate per valore

Tutte le chiamate fatte alle funzioni in Octave sono per valore e quindi non modificano i parametri passati. Non esistono i puntatori come in C. Per modificare variabili all'interno di una funzione bisogna utilizzare variabili globali.

```
octave:28> function f (x)
> while (x-- >0)
>  disp(x);
> endwhile
> endfunction
octave:29> a=5
a = 5
octave:30> f(a)
4
3
2
1
0
octave:31> a
a = 5
```

Ricorsione
La ricorsione è possibile e la vediamo attraverso la funzione fattoriale

```
octave:33> function result=fatt(x)
> if x > 0
> result = x * fatt(x-1)
> else
> result = 1
> endif
> endfunction
octave:34> fatt(4)
result = 1
result = 1
result = 2
result = 6
result = 24
ans = 24
```

Operazioni aritmetiche

- \+ - * / operazioni elementari
- ++, incremento
- --, decremento
- x\y, uguale al dire (inverse(x)*y)
- x^y, x alla y
- ' , trasposto

Operatori di confronto
==, >, <, >=, <=, !=
Operatori Logici
& |
per i confronti vanno usati && e ||
Valutare le espressioni

Valutazione delle espressioni

Le espressioni sono valutate quando vengono eseguite come comandi nell'interprete. Può essere utile eseguire comandi memorizzati in stringhe. Questo può essere fatto tramite il comando *eval*.

```
octave:2> eval ("f=12");
f = 12
```

Ugualmente si usa *feval* per lanciare una funzione a partire da una stringa.

```
octave:3> feval ("acos",-1)
ans = 3.1416
```

Istruzioni

Istruzioni condizionali

If

if (condizione)
    blocco di istruzioni
else
    blocco di istruzioni
endif

ESEMPIO

```
octave:10> function result=pari(x)
> if (rem(x,2)==0)
> result=1;
> else
> result=0
> endif
> endfunction
octave:11> pari(4)
ans = 1
octave:12> pari(3)
result = 0
ans = 0
```

Switch

switch ESPRESSIONE
  case CASO1
    BLOCCO
  case CASO2
    BLOCCO
  ...

  otherwise
    BLOCCO
endswitch

ESEMPIO

```
octave:15> x=3
x = 3
octave:16> switch(x)
> case(1)
>  test="pippo"
> case(2)
>  test="pluto"
> case(3)
>  test="paperino"
> otherwise
>  test="sconosciuto"
> endswitch
test = paperino
```

While

while (CONDIZIONE)

```
  blocco di istruzioni
endwhile
```

Do

```
do
  blocco di istruzioni
until (CONDIZIONE)
```

for

```
For var=expr
  BLOCCO
endfor
```

```
octave:18> fib = ones (1,10);
octave:19> for i = 3:10
> fib (i)= fib (i-1) + fib (i-2);
> endfor
octave:20> fib
fib =

   1   1   2   3   5   8  13  21  34  55
```

Break

L'istruzione break permette l'interruzione di un ciclo indipendentemente dalla condizione.

Continue

L'istruzione continue interrompe il ciclo concludendo, a differenza di break, aspetta la fine del blocco di istruzioni.

Try ... Catch

Octave supporta la gestione delle eccezioni. La gestione delle eccezioni permette di valutare un blocco, in caso di anomalie queste vengono gestite tramite il lancio e la ricezione dell'eccezione stessa.

Come in linguaggi evoluti quali Java o C# le eccezioni vengono gestite tramite try e catch.

```
octave:22> try
> [1,2,3]*[1;2];
> catch
> s="errore"
> end_try_catch
s = errore
```

Funzioni e File Script

Le funzioni sono un insieme di comandi che possono essere raggruppati per facilitare le elaborazioni dell'utente. In Octave possono essere definite in maniera iterattiva o in file esterni.

Una funzione è definita come:
function nome_funzione
blocco istruzioni
endfunction

ESEMPIO

```
octave:1> function wakeup
> printf("\a");
> endfunction
octave:2> wakeup
```

Ad una funzione è poi possibile passare parametri
function nome_funzione (parametro1, ...)
blocco istruzioni
endfunction

ESEMPIO

```
octave:3> function wakeup(message)
> printf("\a%s\n", message);
> endfunction
octave:4> wakeup("Alzati e cammina")
Alzati e cammina
```

Per utilizzare un parametro di ritorno, invece:

```
octave:3> function ritorno=wakeup(message)
> printf("\a%s\n", message);
> ritorno=”Sveglia Effettuata”;
> endfunction
octave:4> wakeup("Alzati e cammina")
Alzati e cammina
ans = Sveglia effettuata
```

ESEMPIO. Calcolare la media di un vettore

```
octave:8> function retval = avg(v)
> retval = 0;
> if (nargin != 1)
> usage ("avg (vector)");
> endif
> if (is_vector (v))
> retval=sum(v)/length(v);
> else
> error("Si devono passare dei vettori");
> endif
> endfunction
octave:9> vett=[1 2 3 4 5];
octave:10> avg(vett)
ans = 3
octave:11> avg(vett,"pippo")
usage: avg (vector)
error: evaluating if command near line 3, column 1
```

```
error: called from `avg'
octave:11> avg()
usage: avg (vector)
error: evaluating if command near line 3, column 1
error: called from `avg'
octave:11>
```

La funzione nargin ritorna il numero di parametri passati alla funzione.

Ritornare più valori

function [rect-list] = name (arg-list)
blocco istruzioni;
endfunction

Vediamo un esempio di funzione dove vengono restituiti più valori. Si tratta di una funzione che restituisce il massimo di un vettore e l'indice in cui si trova.

```
octave:11> function [max, idx]= vmax(v)
> idx = 1;
> max = v(idx);
> for i = 2:length(v)
> if (v (i) > max)
> max = v(i);
> idx=i;
> endif
> endfor
> endfunction
octave:12> [max, in]=vmax(vett)
max = 5
in = 5
```

Parametri

- **nargin** è il numero di argomenti passati alla funzione
- **nargout** è il numero di parametri in uscita
- **v_arg()** ritorna il primo parametro nella lista dei parametri indefiniti nella lista e fa puntare il puntatore interno al successivo parametro
- **v_start** Punta all'indice del primo parametro non dichiarato nella funzione

Numero di parametri indefiniti

Octave supporta il passaggio indefinito di parametri, cioè una funzione può ricevere un numero indefinito di parametri non specificati nella definizione della funzione.
Vediamo un esempio.

```
Function test (heading, ...)
disp (heading);
va_start();
while(n—argin)
 disp(va_arg());
endwhile
endfunction
```

Il programma appena scritto mostra tutti i parametri passati alla funzione test.

Return

La funzione return fa uscire da una funzione in qualsiasi momento. A differenza del C non serve a ritornare i valori. Per ritornare i valori, come abbiamo già visto, basta fare assegnazioni alle variabili di ritorno specificati nelle funzioni.

File di funzioni

Ad eccezioni di piccoli esempi non ha senso scrivere funzioni direttamente nell'interprete , ma è più comodo salvare le funzioni in file per poterle utilizzare più comodamente, e nel caso, correggerle e modificarle.

I file vengono salvati con l'estensione .m, questo per mantenere una compatibilità con la nomenclatura usata da Matlab.
I file possono essere memorizzati in una qualsiasi directory che poi deve essere specificata all'interno di octave. Octave quando incontra una funzione cerca il suo nome in quelle predefinite, nel caso non vengano trovate va a cercarla in file esterni (.m) specificata nella lista di directory all'interno della variabile d'ambiente LOADPATH.

Ad esempio LOADPATH=":/root/octave" specifica che i file .m si troveranno nella diretory "/root/octave" . In caso di più directory queste vengono separate dai due punt (":").

E' comodo ricordarsi che l'help chiamato su una funzione ritorna il primo commento dopo la definizione della funzione.

File Script

In un file script possono essere inseriti tutti i comandi che possono essere passati ad un interprete. A differenza delle funzioni, un file di script può non iniziare con il comando function

Funzioni esterne

Octave permette di utilizzare funzioni scritte in altri linguaggi (in particolar modo il C++). Per fare tutto ciò bisogna utilizzare alcune librerie. La trattazione di quest'argomento è rimandata al manuale del prof.Eaton.

Gestione degli Errori

Octave ha diverse funzioni per stampare errori o warning (avvertimenti). Quando si scrive una funziona bisogna utilizzare queste funzioni per segnalare comportamenti anomali rispetto all'algoritmo della funzione stessa.

- **error(template, ...)** Stampa l'errore specificato nel testo, è possibile inserire nell'output anche il valore di alcune variabili con le stesse modalità di printf.

```
> function somma(a,b)
> if (!isnumeric(a), !isnumeric(b)
>    error ("La somma può essere effettuata solo fra due numeri")
>    return
> endif
> a+b
> endfunction
```

- **warning(msg)** Stampa l'avvertimento specificato nel messaggio. A differenza dell'errore il warning è solo un avvertimento, un consiglio, che non implica un'interruzione dell'algoritmo.
- **usage(msg)** Stampa l'indicazione su come deve essere utilizzata la funzione (solitamente serve a fare un controllo sul numero ed il tipo di parametri passati alla funzione).

Input e Output

Octave utilizza due modi per gestire l'Input e l'Output. Uno maggiormente legato a Matlab, l'altro al linguaggio C.

Output da Terminale

Octave valuta le espressioni ogni qualvolta che viene battuto l'invio. Il risultato di queste espressioni è stampato sullo schermo, che è lo standard output.

- **disp(x)** è una funzione che stampa il valore della variabile x

Input da Terminale

Per inserire valori direttamente dallo standard input (il terminale) si utilizza la funzione input

- **input(testo) input(testo,"s")** Input stampa un messaggio e aspetta l'output dell'utente. Dato che può ricevere un solo valore, per utilizzare le stringhe si utilizza l'opzione "s"
- **menu(testo, op1, op2)** la funzione menu offre una serie di opzioni predefinite, utile per realizzare menu per sessioni iterattive

```
octave:1> risposta=menu("Sei d'accordo?","Si", "No", "Forse")
Sei d'accordo?

  [ 1] Si
  [ 2] No
  [ 3] Forse

pick a number, any number: 3
risposta = 3
```

I/O da File

Scrittura

E' possibile salvare variabili in un file utilizzando il comando save

```
save options file v1 v2 ...
```

Save salva le variabili v1, v2, ... vn nel file specificato. Le opzioni permettono di specificare il formato.

Alcune opzioni:

- -ascii salva in formato testo
- -binary salva in formato binario
- -mat-binary salva in formato binario compatibile con matlab

Esempio

```
octave:3> testo="Ciao Mondo"
testo = Ciao Mondo
octave:4> save -ascii /root/octavefun/testo.txt testo
octave:5> save -binary /root/octavefun/testo.bin testo
```

testo.txt

```
# Created by Octave 2.1.40, Sun Nov 16 19:54:29 2003 CET
<root@crcLiguria>
# name: testo
# type: string array
# elements: 1
# length: 10
```

```
Ciao Mondo
```
testo.bin
```
Octave-1-L#####testo##########
###Ciao Mondo
```

Lettura

In maniera del tutto simile il comando *load* serve per caricare variabili inserite in un file.

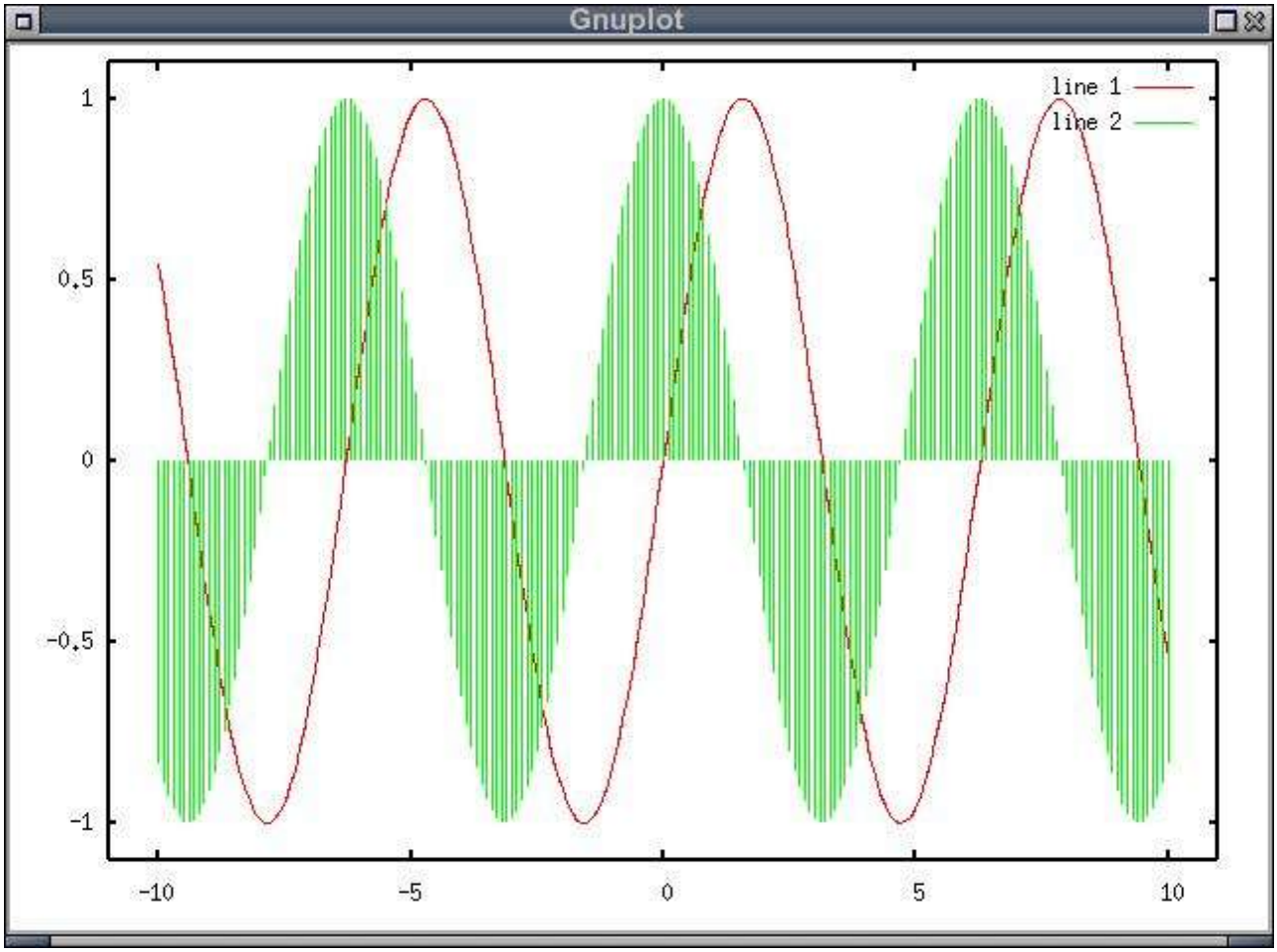
```
octave:6> clear testo
octave:7> testo
error: `testo' undefined near line 7 column 1
octave:7> load -ascii /root/octavefun/testo.txt testo
octave:8> testo
testo = Ciao Mondo
octave:9>
```

Grafica

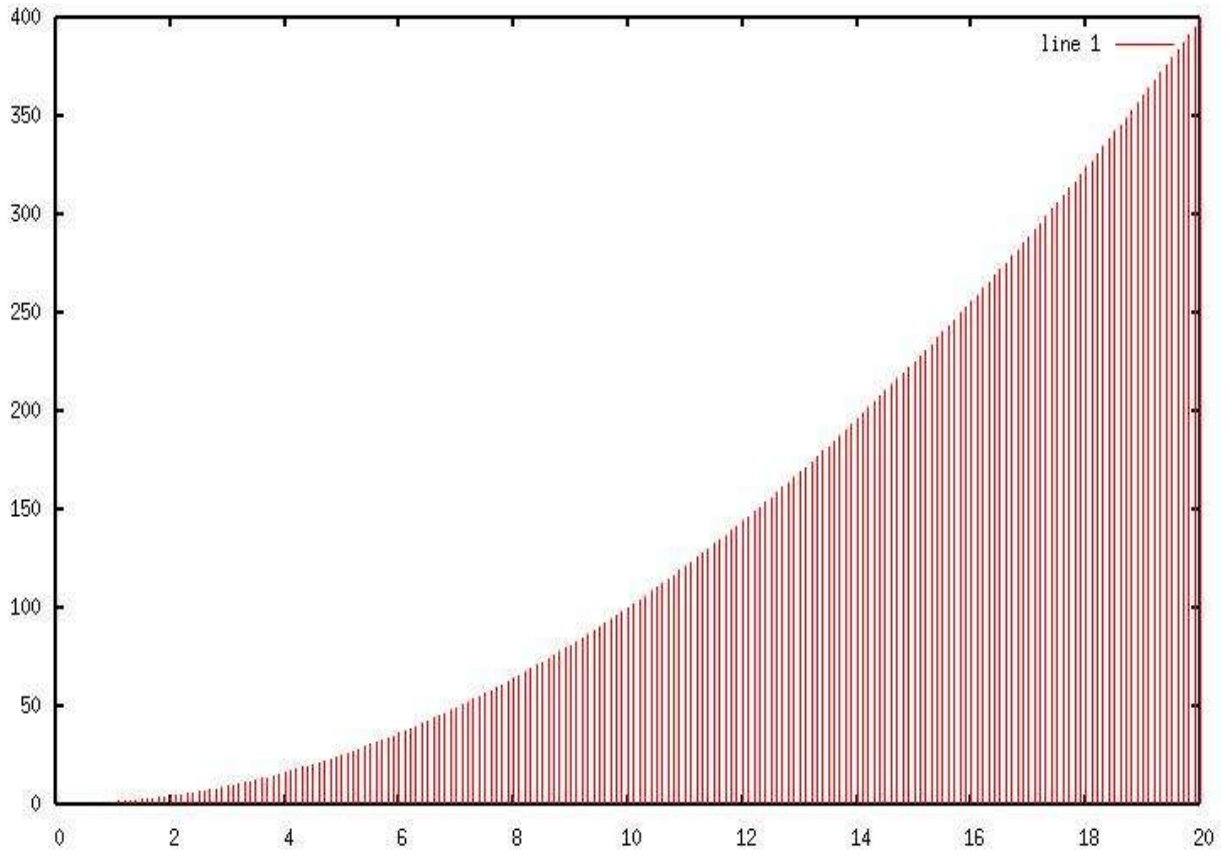
Tutte le funzioni di grafica utilizzate da Octave si basano sul programma 'gnuplot'.
gplot *intervallo using title style* genera un grafico in 2D.
Intervallo, using, title e style sono parametri opzionali.
L'intervallo è così definito *[x_min:x_max][y_min:y_max]*

```
octave:1> gplot rand (100,1) with linespoints
```

```
octave:14> x=(-10:0.1:10)';
octave:15> data=[x,sin(x),cos(x)];
octave:16> gplot [-11:11][-1.1:1.1] data with lines, data using 1:3 with impulses
```



Di default vengono presi i primi due valori. Nell'esempio precedente vengono prese le serie 1 e 3.



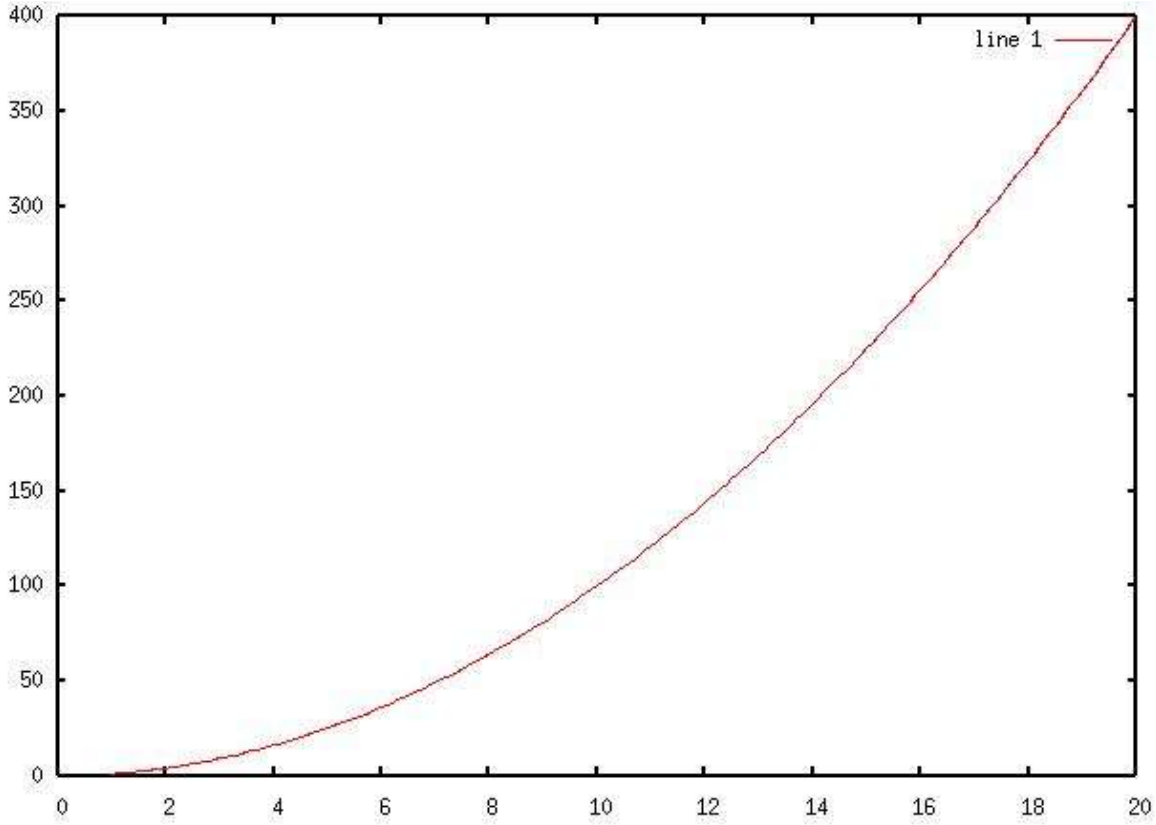
Per compatibilità con Matlab esistono anche altri comandi per la grafica.

Plot

Il comando *plot(x, y, fmt)* permette di disegnare grafici in due dimensioni.

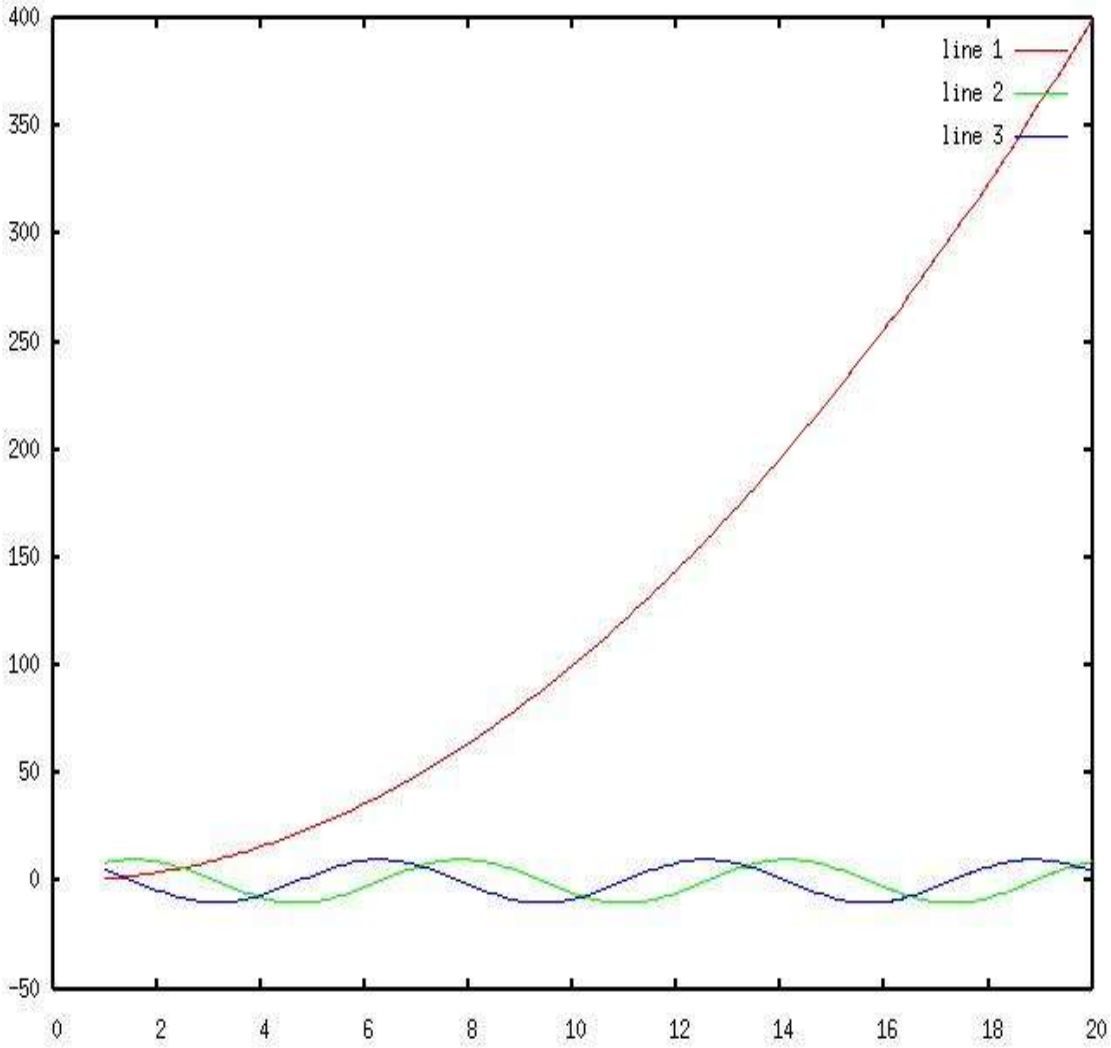
```
octave:24> x=(1:0.1:20);
octave:25> y=x.^2
octave:26> plot(x,y)
```

Fmt specifica il formato del grafico ad esempio '@' serve per avere i punti, 'L' per avere un grafico a gradini, '^' a impulsi.



Altri esempi.

```
octave:33> plot(x,y,x,sin(x),x,cos(x))
octave:34> plot(x,y,x,10*sin(x),x,10*cos(x))
```



Altri tipi di grafici a due dimensioni

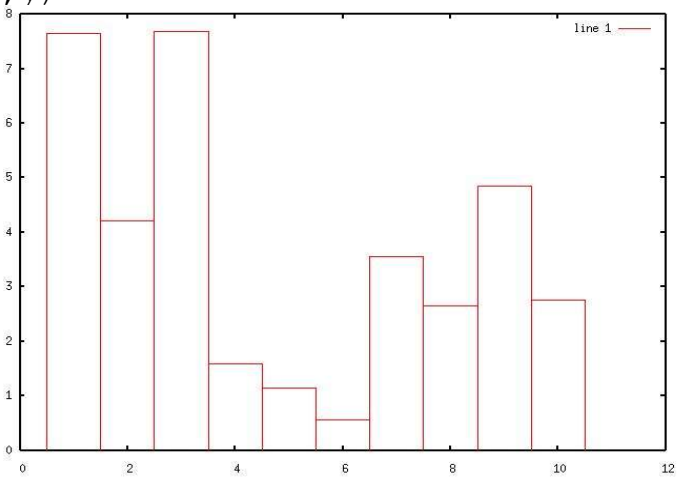
bar(x,y)

Realizza un grafico a barre a partire da due vettori.

```
octave:38> y=10 * rand(1,10)
octave:39> x=(1:10)
x =

   1   2   3   4   5   6   7   8   9  10

octave:40> bar(x,y)
```



Ci sono poi altri tipi di grafici. Consultare il manuale a seconda delle proprie esigenze.

Grafici a tre dimensioni

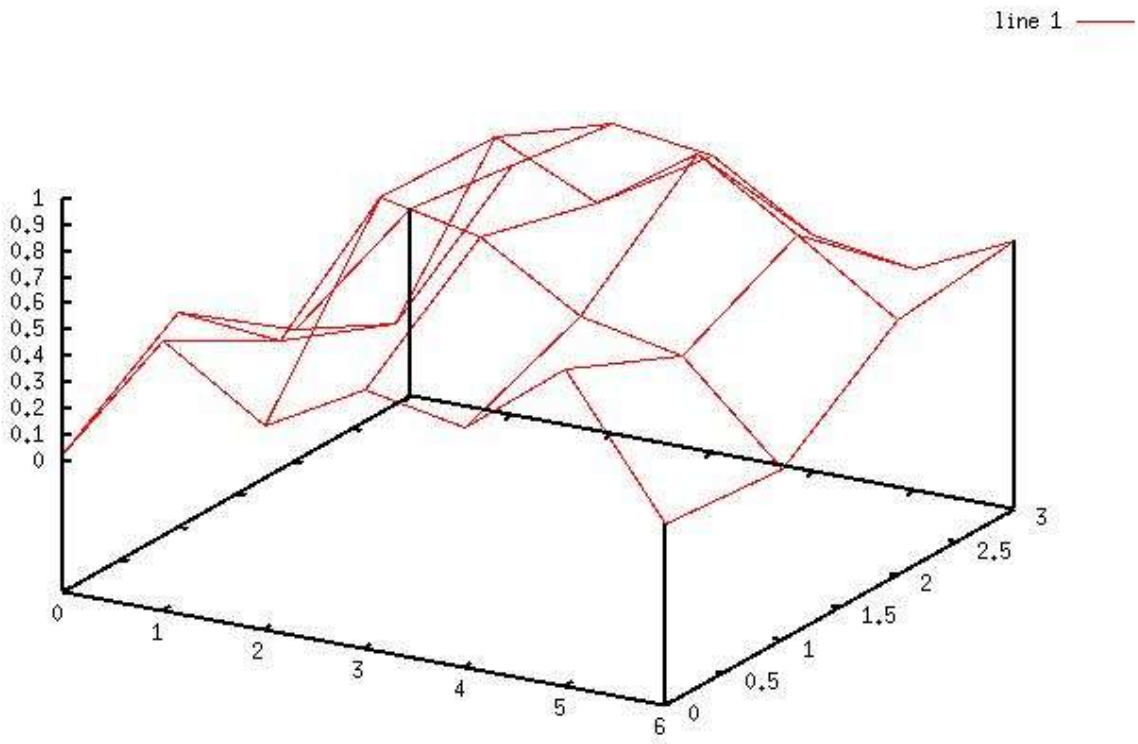
gsplot *intervallo using title style* genera un grafico in 3D.

L'intervallo è definito come quello in due dimensioni aggiungendo la coordinata z, [z_min:z_max]

Vediamo un esempio:

```
octave:44> gsplot rand(7,4)
```

In questo caso il grafico in 3D mostra su z il valore del rand e su x e y, l'indice rispettivamente delle righe e delle colonne.

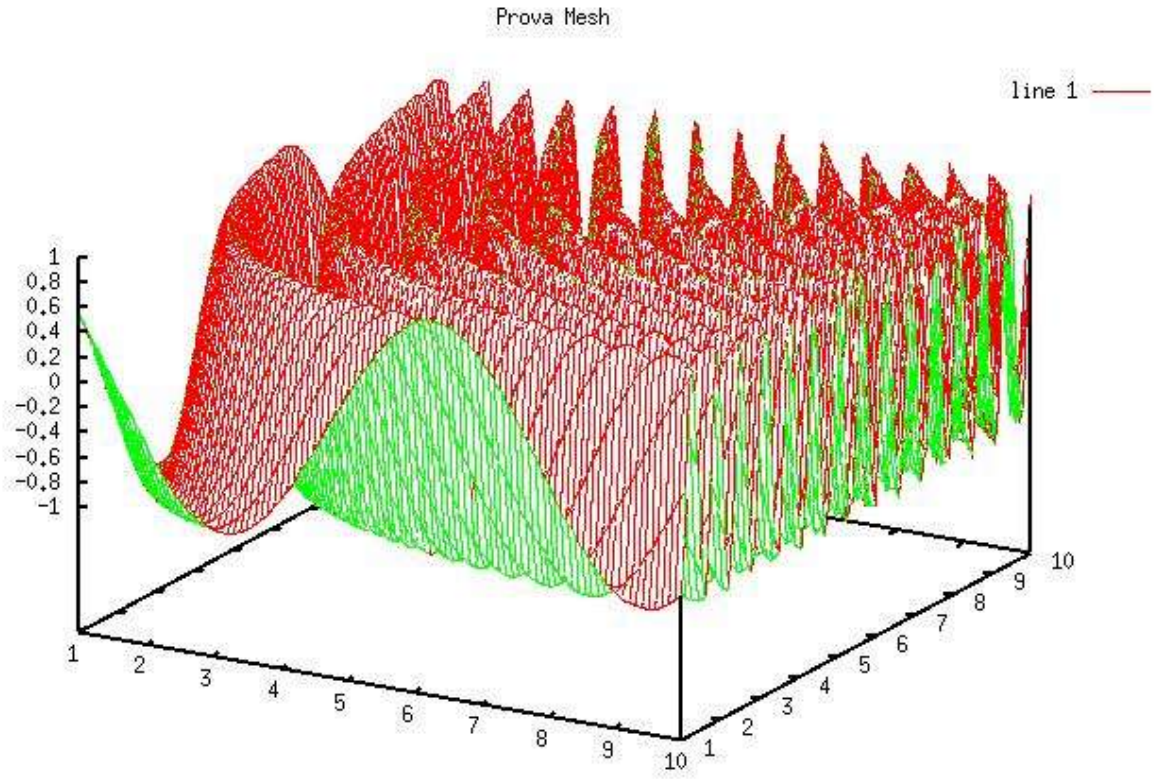


La funzione Matlab corrispondente è *mesh(x,y,z)*

Annotazioni sui grafici

- **title('titolo')** stampa un titolo sul grafico, in caso il grafico sia già mostrato a video utilizzare il comando **replot**.

```
octave:100> x=(1:0.1:10);
octave:101> y=(1:0.1:10)';
octave:102> z= y*x;
octave:103> z=cos(z);
octave:104> mesh(x,y,z)
octave:105> title("Prova Mesh");
octave:106> replot
```



Altre funzioni sono:

- **xlabel(“testo”)**
- **ylabel(“testo”)**
- **zlabel(“testo”)**
- **bottom_title(“testo”)**

Appendice.
Panaoramica su funzioni avanzate

Integrazione

- quad(“nome_funzione”, x0, x1)

```
octave:6> function y=f(x)
> y=x.^2;
> endfunction
octave:7> quad("f",0,100)
ans =  3.3333e+05
octave:8> quad("f",0,10)
ans = 333.33
```

Funzioni non Lineari

Octave può risolvere qualsiasi funzione del tipo
f(x)=0
utilizzando la funzione fsolve.
La funzione fsolve richiede il nome della funzione ed un valore iniziale. Ritorna il risultato e una informazione info uguale a 1 se la soluzione converge.

ESEMPIO

$$-2x^2 + 3xy + 4\sin(y) - 6 = 0$$
$$3x^2 - 2xy^2 + 3\cos(x) + 4 = 0$$

```
octave:107> function y=f(x)
> y(1)=-2*x(1)^2 + 3*x(1)*x(2) + 4*sin(x(2))-6;
> y(2)=3*x(1)^2 -2*x(1)*x(2)^2 + 3*cos(x(1))+4;
> endfunction;

octave:108> [x, info]=fsolve("f",[1;2])
x =

  0.57983
  2.54621

info = 1
```

Funzioni differenziali

Ugualmente alla risoluzione dei sistemi non lineari si possono risolvere equazioni differenziali utilizzando le funzioni definite in Octave. Il meccanismo è lo stesso visto prima si definisce una funzione per gestire la funzione di f(x,t) e poi si definiscono le opportune condizioni iniziali. A seconda del tipo di equazione differenziale ci sono diverse funzioni.

Per una completa trattazione dell'argomento si consiglia di consultare il manuale.

ESEMPIO:

- **lsode** serve per risolvere equazioni del tipo
  dx/dt =f(x,t)
- **dassl** serve per risolvere equazioni del tipo
  0=f(x',x,t) data la condizione $x(t=0)=x_0$ e $x'(t=0)=x'_0$

Funzioni statistiche

Octave ha una serie di funzioni predefinite per gestire l'analisi statistiche. Per stessa ammissione del creatore di Octave questo è ancora un campo in pieno sviluppo.
E' comunque possibile effettuare analisi statistiche di base con un ampia gamma di funzioni di seguito elencate:

- **mean(x, opt)**, dato un vettore x mean ne calcola la media. L'argomento opzionale opt serve per determinare quale media calcolare, opt può assumere i seguenti valori:
  - **a** serie aritmetica
  - **g** serie geometrica
  - **h** serie armonica
- **median(x)** calcola la mediana
- **std(x)** deviazione standard
- **cov(x,y)** covarianza di x e y

```
octave:1> x=[1 1 2 3 4 3 5 5 2 2 1]
x =

   1   1   2   3   4   3   5   5   2   2   1

octave:2> mean(x)
ans = 2.6364
octave:3> median(x)
ans = 2
octave:4> std(x)
ans = 1.5015
```

- **corrcoef(x,y)** coefficente di correlazione

```
octave:8> corrcoef(x,y)
ans = 0.56974
```

Esistono poi ulteriori funzioni per calcolare i momenti, i curtosi e per fare analidi tipo ANOVA (Analisi della Varianza), test di regressione, ecc.
Sono poi già definite funzioni per le distribuzioni più comuni.

Altri ambiti di applicazione di Octave

- Ambito finanziario per analisi di tipo finanziario
- Ambito controllistico tramite la funzione DEMOcontrol
- Signal processing, che comprende ad esempio le trasformate di Fourier
- Manipolazioni di immagini, tramite il programma xloadimage
- Manipolazione di audio

Utilità di sistema

Octave mette a disposizione alcune utilities che possono essere utili per creare le proprie funzioni:

Funzioni per il tempo:

- time() Numero di secondi dal 1 Gennaio 1970
- ctime(time()) Traduzione di time in una stringa
- gtime(time()) Fornisce una struttura con il dettaglio sulla data
- clock() Fornisce un vettore con anno mese giorno ora min secondi
- date() Fornisce una stringa con la data
- pause(sec) Sospende l'esecuzione per i secondi specificati

```
octave:11> time
ans =  1.0702e+09
octave:12> ctime(time())
ans = Sun Nov 30 19:34:48 2003

octave:13> gmtime(time())
ans =
{
  usec = 847272
  year = 103
  mon = 10
  sec = 7
  mday = 30
  min = 38
  zone = GMT
  wday = 0
  hour = 18
  yday = 333
  isdst = 0
}

octave:14> clock
ans =

  2003.000    11.000    30.000    19.000    39.000    40.508

octave:15> date
ans = 30-Nov-2003
```

Funzioni per il file-system:

- rename(old, new) Riomina un file
- unlink(filename) Cancella un file
- readdir(dir) Visualizza i file in una directory
- mkdir(dir) Crea una directory
- rmdir(dir) Rimuove una directory

Funzioni per il controllo:

- system(string, return_output, type) Esegue un comando nella shell
- exec(programma, parametri)

```
octave:18> disp(system("echo $PATH"));
/usr/libexec/octave/2.1.40/site/exec/i386-redhat-linux-
gnu:/usr/libexec/octave/site/exec/i386-redhat-linux-
gnu:/usr/libexec/octave/2.1.40/exec/i386-redhat-linux-
gnu:/usr/bin:/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/
usr/bin:/usr/X11R6/bin:/opt/java/j2sdk1.4.1_02/bin/:/usr/local/apa
che/bin:PATH=/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/
```

```
usr/bin:/usr/X11R6/bin:/opt/java/j2sdk1.4.1_02/bin/:/usr/local/apa
che/bin:/usr/local/OpenOffice.org1.1.0/program/:/opt/jakarta-
tomcat-4.1.24/bin/
```

Directory:

- cd dir
- ls o dir Mostra l'elenco dei file nella directory
- pwd() Mostra la directory corrente

```
octave:19> pwd
/root
```

Informazioni sul sistema

- computer() Mostra informazioni sul sistema

```
octave:20> computer
i386-redhat-linux-gnu
```