(grafica di funzioni..., continua)

Una lista delle più comuni funzioni matematiche elementari in Matlab:

```
sin, cos, tan, asin, acos, atan
exp, log, log10, log2
abs, sqrt, sign, rem, mod, factorial
conj, real, imag, angle
round, floor, fix, ceil

Per una lista completa
>> help elfun

Provare anche il comando

>> fplot('fun',[a,b])

che produce il grafico 'accurato' di fun.m nell'intervallo [a,b].
```

Altri comandi grafici di utilità

```
>> xlabel('string')
>> ylabel('string')
>> grid    % grid off  per disattivarlo
>> subplot(m,n,i)   % crea il grafico i di una finestra con mxn grafici

Colori e stili:
y (yellow), m (magenta), c (cyan), r (red), g (green), b (blue), k (black)
. (point), o (circle), x (x-mark), + (plus), - (solid), * (star), : (dotted)
-. (dashdot), -- (dashed)
```

Esercizio. Scrivere un programma che confronti i grafici per punti di round, floor, fix e ceil in una stessa finestra, usando colori e simboli diversi.

Esercizio. Scrivere un programma che confronti i grafici per punti di round, floor, fix e ceil in quattro sottofinestre contigue mediante il comando subplot.

3D plots

Curve nello spazio:

Per ottenere il grafico di una curva parametrica (x(t),y(t),z(t)) nello spazio, al variare del parametro t in [a,b] basta create l'opportuno vettore dei punti in tale intervallo, e i tre corrispondenti vettori delle immagini. Quindi eseguire

```
>> t=[a:step:b]; x=funx(t); y=funy(t); z=funz(t);
>> plot3(x,y,z)
```

Esempio. Disegniamo un'elica attorno all'asse z:

```
>> t=0.01:0.01:20*pi; x=cos(t); y=sin(t); z=t.^3; plot3(x,y,z)
```

Superfici nello spazio:
Per ottenere il grafico di una funzione z=f(x,y) sul rettangolo [a,b]x[c,d] occorre prima costruire le matrici delle coordinate dei nodi mediante

```
>> [x,y]=meshgrid(a:stepx:b,c:stepy:d);
```

{costruisce la matrice x in cui ogni riga è uguale a [a:stepx:b], e ogni colonna ha la dimensione di [c:stepy:d], e la matrice y in cui ogni colonna è uguale a [c:stepy:d] e ogni riga ha la dimensione di [a:stepx:b]}
Poi si calcola la matrice delle altezze z_ij = f(x_i,y_j):

```
>> z=f(x,y);
>> mesh(x,y,z)
```

Il comando mesh crea il plot prospettico in 3D dei valori di z. Per ottenere un grafico a colori della superfice si può invece usare

```
>> surf(x,y,z)
```

Esercizio. Provare con l'esempio fornito da Matlab:
```
>> peaks
```
come cambia il grafico con i comandi:
```
>> shading flat
>> shading interp
>> shading faceted  % (default)
```

La gamma di colori si può variare attraverso il comando colormap (v. help).
Disegnare poi il grafico della funzione f(x,y)=1/(x^2+y^2+1)

Osservazione. Che succede se si usa solo mesh(z) o surf(z) ?

Curve di livello di una superficie:
Matlab fornisce un comando diretto per tale scopo:

```
>> contour(x,y,z)   % provare anche  contour(x,y,z,N) e contourf(x,y,z)
```

Esercizio. Creare le curve di livello della funzione peaks in [-3,3]x[-3,3]. Provare con diversi valori di N.

Esercizio. Cosa fanno i comandi seguenti subito dopo il comando contour?

```
>> fmax=max(max(z)); kmax=find(z==fmax); pos=[x(kmax),y(kmax)]
>> hold on; plot(x(kmax),y(kmax),'o')
>> text(x(kmax),y(kmax),'MAX')
>> hold off
```

Provare anche:

```
>> val=[a:step:b]; [C]=contour(x,y,z,val); clabel(C,val);
```

(Assegno a contour col vettore val i valori degli insiemi di livello di interesse, in C finiscono i dati delle relative curve, e con clabel aggiungo nel grafico le etichette corrispondenti)

```
>> meshc(x,y,z) % per avere nello stesso grafico superficie e insiemi di livello
```

Cicli e alternative

```
% numero di iterazioni prefissate
for i=1:n
   ... istruzioni ...
end

% numero di iterazioni condizionate
while condizione
   ... istruzioni ...
end

if condizione
   ... istruzioni ...
end

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

if condizione1
   ... istruzioni ...
elseif condizione2
   ... istruzioni ...
else
   ... istruzioni ...
end

switch (string)
case 'uno'
  ... istruzioni ...
case 'due'
  ... istruzioni ...
case 'tre'
  ... istruzioni ...
otherwise
  ... istruzioni ...
end
```

Operatori relazionali e logici

```
<, <=, >, >=
==  (uguale),  ~= (diverso)
& (and),  ~ (not), | (or), xor (or esclusivo)
any, all  (esempi...)
```

Le variabili booleane assumono solo i valori 0 (false) o 1 (true). Ogni altro numero diverso da zero è considerato true.

Osservazione. Per inserire in uno script molte righe di testo si può digitare

```
if 0
... righe di testo
end
```

Esercitazione: la successione 3*x+1

Il seguente programma studia la successione per ricorrenza che partendo da un numero intero assegnato, ad ogni passo divide per due se il numero è pari, mentre moltiplica per 3 e somma 1 se il numero è dispari. La successione si arresta se viene raggiunto il numero 1 (*capite perché? o meglio, capite cosa succederebbe continuando?*)

Leggere con attezione il listato, cercando di capire prima di eseguirlo esattamente cosa fa ogni istruzione. Poi eseguirlo e tornare sulle istruzioni non capite.

trexpiuuno.m

```
% I PARTE
u0=input('Inserisci il numero intero iniziale n= ');
cont=0; c=1; x(1)=u0;
while ~cont
    if mod(u0,2)
        u0=3*u0+1;
    else
        u0=u0/2;
    end
    c=c+1; x(c)=u0;
    disp(num2str(u0));
    if (u0==1) cont=cont+1;
    end
end
disp(['Partendo da ',num2str(x(1)),' numero di termini  ',num2str(c)]);
M=max(x(1:c));
plot(1:c,x(1:c),'-')
axis([1 c 0 M])
title(['x0 = ',num2str(x(1)),', numero termini = ',num2str(c),...
    ', valmax = ',num2str(M)])

% II PARTE
N=input('Inserisci la dimensione del vettore N= ');
for i=2:N
    u0=i; x(1)=0;
cont=0; c=0;
while ~cont
    if mod(u0,2)
        u0=3*u0+1;
    else
        u0=u0/2;
    end
    c=c+1;
   % disp(num2str(u0));
    if (u0==1) cont=cont+1;
    end
end
disp([num2str(i),': numero di iterazioni  ',num2str(c)]);
x(i)=c;
end
[M,ind]=max(x(1:N));
figure
plot(1:N,x(1:N),'*')
title(['max#iteraz = ',num2str(M),' per  x0 =',num2str(ind)])
```

Esercizio: trovare il dato iniziale <= 1000 che genera la sequenza più lunga.

Ancora sugli M - files (.m)
Script files: Tutte le variabili usate sono globali. Ogni script può fare riferimento ad altri scripts, anche ricorsivamente.
Function files: Le variabili sono per default locali (se non espressamente dichiarate: v. >> help global)

Esempio (randint.m):

```
function c=randint(m,n,a,b)
% randint(m,n) crea una matrice mxn con valori random tra 0 e 9
% randint(m,n,a,b) crea una matrice mxn con valori random tra a e b
if nargin<3, a=0; b=9; end
c=floor((b-a+1)*rand(m,n))+a;
```

Scrivere ed eseguire in CW:

```
>> randint(4,5)
>> randint(4,5,1,20)
```

N.B. uso di nargin (# di argomenti input) e nargout (# di argomenti output).

Le prime righe commentate all'inizio di ogni M-file (script o function) vengono associate all'help corrispondente. Provate

```
>> help randint
```

Alcuni comandi di utilità negli scripts.

```
a=input('dammi a')      stampa il commento e sospende l'esecuzione finché non
                        viene inserito da tastiera un valore per a
pause       arresta l'esecuzione finché non si preme un tasto
pause(n)    arresta l'esecuzione per n secondi
break       arresta l'esecuzione del programma, o esce dal ciclo in cui si trova
            (Attenzione: un programma in esecuzione può sempre essere interrotto
            digitando nella CW uno o più volte il comando CTRL-C)
...         permette di spezzare una riga di comando per andare a capo
tic         inizia a contare il tempo
toc         finisce di contare il tempo e lo stampa a video
t=toc;      finisce di contare il tempo e lo memorizza nella variabile t
disp('commento')  stampa il commento contenuto tra apici
error('commento') stampa il commento contenuto tra apici e arresta il programma
```

Esercizio. Confrontiamo in termini di tempi di calcolo due modi di risolvere un sistema lineare Ax=b. Scrivere un programma che assegnata una matrice quadrata A di dimensione n (dato in input) e un vettore b di dimensione nx1 (per esempio generati con la function randint), calcola la soluzione del sistema una volta attraverso la funzione Matlab \ (x=A\b), una volta attraverso l'inversione diretta della matrice (x=inv(A)*b), calcolandone e confrontandone i tempi di esecuzione. Provare con n=10,100,1000,2000,3000.

****

Per finire una panoramica veloce di quel che si può trovare già fatto:

```
>> help elfun     % funzioni matematiche elementari
>> help specfun   % funzioni matematiche speciali
>> help matfun    % funzioni di utilità per le matrici
>> help funfun    % funzioni di utilità per le funzioni
>> help optimfun  % funzioni di ottimizzazione e ricerca degli zeri
```

E ora al lavoro !