Laboratorio del 19 Marzo 2014

<u>Vettori in Matlab</u>

Consideriamo il seguente vettore riga **u** (tutto quanto diremo vale anche se il vettore fosse colonna).

	1	2	3
u	-2	-2	-2

In Matlab, gli indici dei vettori (e delle matrici) partono da 1. Ad esempio, per riferirsi al primo elemento del vettore **u** scriviamo **u**(1) e non **u**(0) o altro. Per riferirsi, invece, all'ultimo elemento del vettore **u** possiamo scrivere **u**(end); per riferirsi al penultimo, possiamo scrivere **u**(end-1) e così di seguito. Questo modo è comodo soprattutto per riferirsi agli elementi di un vettore che si trovano nelle ultime posizioni.

La lunghezza del vettore **u**, ossia quanti elementi contiene, è data da length(**u**). In questo modo, per riferirsi all'ultimo elemento di **u** possiamo anche scrivere **u**(length(**u**)).

In Matlab c'è il vettore che non contiene alcun elemento; è detto vettore vuoto ed è indicato con [] (parentesi quadra aperta "[" seguita da parentesi quadra chiusa"]"). Il vettore vuoto ha lunghezza nulla. Ad esempio, l'istruzione **v** = []; crea il vettore vuoto **v**.

<u>Operazioni con i vettori</u>

Per calcolare la <u>somma</u> e la <u>differenza</u> dei vettori (o matrici) **u** e **v** che hanno le *stesse dimensioni* (ossia, gli stessi numeri di righe e di colonne) si scrive **u + v** e **u - v**. Queste operazioni riguardano tutte le componenti dei vettori allo stesso tempo. Ad esempio, la somma **u** + **v** equivale ad eseguire la length(**u**) somme **u**(k)+**v**(k), k=1,…,length(**u**). In Matlab è permesso sommare o sottrarre un numero ad un vettore: il risultato che si ottiene è quello di sommare o sottrarre quel numero ad ogni componente del vettore. Ad esempio se **u** è un vettore, **u**+2 equivale a **u**(k)+2, k=1,…,length(**u**).

Per moltiplicare tutte le componente del vettore **u** per un numero, ad esempio 3, si scrive 3***u** (3 star **u**).

<u>Operazioni punto con i vettori</u>

Sono sostanzialmente tre:

1) **.*** (punto per)
2) **./** (punto diviso)
3) **.^** (punto elevato)

Permettono di eseguire le operazioni indicate (*, /, ^) componente per componente per due *vettori o matrici* delle *stesse dimensioni* (stessi numeri di righe e di colonne). Ad esempio, consideriamo i vettori riga **u** e **v**

	1	2	3
u	3	-2	5

	1	2	3
v	2	1	4

L'istruzione **w** = **u**.***v** (punto per)crea il vettore **w** di una riga e tre colonne con **w**(k)=**u**(k)***v**(k), k=1,…,length(**u**):

	1	2	3
w	3*2	-2*1	5*4

L'istruzione **w** = **u**./**v** (punto diviso)crea il vettore **w** di una riga e tre colonne con **w**(k)=**u**(k)/**v**(k), k=1,…,length(**u**):

	1	2	3
w	3/2	-2/1	5/4

L'istruzione **w** = **u**.^**v** (punto elevato) crea il vettore **w** di una riga e tre colonne con **w**(k)=**u**(k).^**v**(k), k=1,…,length(**u**):

	1	2	3
w	3^2	(-2)^1	5^4

Per elevare tutte le componenti del vettore **u** allo stesso numero, ad esempio 4, possiamo scrivere la forma abbreviata **u**.^4.
Le operazioni punto possono essere utilizzate combinate assieme; ad esempio, dato il vettore **u**, per calcolare il vettore **M** le cui componenti sono espresse dall'equazione

$$M(k-2) = \frac{|u(k) - u(k-1)|}{|u(k-1) - u(k-2)|^2}, \qquad k = 3, \dots, length(u)$$

è sufficiente scrivere

M = abs(**u**(3:end) **-** **u**(2:end-1))./(abs(**u**(2:end-1) **-** **u**(1:end-2))).^2

dove la funzione abs() calcola il valore assoluto di ciascuna componente del vettore (o matrice) che ha come argomento. Vediamone il perché con l'esempio di un vettore di 5 elementi. In base alla formula, dobbiamo calcolare

$$M(3-2) = \frac{|u(3) - u(3-1)|}{|u(3-1) - u(3-2)|^2} = \frac{|u(3) - u(2)|}{|u(2) - u(1)|^2}$$

$$M(4-2) = \frac{|u(4) - u(4-1)|}{|u(4-1) - u(4-2)|^2} = \frac{|u(4) - u(3)|}{|u(3) - u(2)|^2}$$

$$M(5-2) = \frac{|u(5) - u(5-1)|}{|u(5-1) - u(5-2)|^2} = \frac{|u(5) - u(4)|}{|u(4) - u(3)|^2}$$

Sia **u** il seguente vettore:

	1	2	3	4	5
u	**u** (1)	**u** (2)	**u** (3)	**u** (4)	**u** (5)

Senza considerare il valore assoluto, il vettore **u**(3:end) **-** **u**(2:end-1) contiene tutti i numeratori delle precedenti equazioni; infatti, risulta

1	2	3		1	2	3		1	2	3
u (3)	**u** (4)	**u** (5)	**-**	**u** (2)	**u** (3)	**u** (4)	**=**	**u** (3)-**u** (2)	**u** (4)-**u** (3)	**u** (5)-**u** (4)
	u(3:end)				**u**(2:end-1)				**u**(3:end)- **u**(2:end-1)	

Notiamo, ad esempio, come l'istruzione **u**(3:end) crei un vettore di 3=length(**u**)-2 componenti.

L'istruzione abs(**u**(3:end) **-** **u**(2:end-1) **)** calcola il valore assoluto di ognuna delle componenti del vettore **u**(3:end) **-** **u**(2:end-1):

1	2	3
|**u** (3)-**u** (2)|	|**u** (4)-**u** (3)|	|**u** (5)-**u** (4)|

abs(**u**(3:end) - **u**(2:end-1))

In modo perfettamente analogo possiamo costruire il vettore che contiene tutti denominatori:

1	2	3
u (2)	**u** (3)	**u** (4)

u(2:end-1)

-

1	2	3
u (1)	**u** (2)	**u** (3)

u(1:end-2)

=

1	2	3
u (2)-**u** (1)	**u** (3)-**u** (2)	**u** (4)-**u** (3)

u(2:end-1) - **u**(1:end-2)

Prendo il valore assoluto delle componenti di **u**(2:end-1) **-** **u**(1:end-2) e poi le elevo al quadrato:

1	2	3
|**u** (2)-**u** (1)|	|**u** (3)-**u** (2)|	|**u** (4)-**u** (3)|

abs(**u**(2:end-1) - **u**(1:end-2))

1	2	3
|**u** (2)-**u** (1)|^2	|**u** (3)-**u** (2)|^2	|**u** (4)-**u** (3)|^2

(**abs(** **u**(2:end-1) - **u**(1:end-2))).^2

Applicando l'operazione ./ ai vettori **abs(** **u**(3:end) **-** **u**(2:end-1)) e (**abs(** **u**(2:end-1) **-** **u**(1:end-2))).^2 si ottiene il risultato.

1	2	3
|**u** (3)-**u** (2)|	|**u** (4)-**u** (3)|	|**u** (5)-**u** (4)|

abs(**u**(2:end-1) - **u**(1:end-2))

./

1	2	3
|**u** (2)-**u** (1)|^2	|**u** (3)-**u** (2)|^2	|**u** (4)-**u** (3)|^2

(**abs(** **u**(2:end-1) - **u**(1:end-2))).^2

=

1	2	3
$\frac{\lvert u(3)-u(2)\rvert}{\lvert u(2)-u(1)\rvert^2}$	$\frac{\lvert u(4)-u(3)\rvert}{\lvert u(3)-u(2)\rvert^2}$	$\frac{\lvert u(5)-u(4)\rvert}{\lvert u(4)-u(3)\rvert^2}$

abs(**u**(3:end) **-** **u**(2:end-1)).(abs(**u**(2:end-1) **-** **u**(1:end-2))).^2

Metodo di Newton

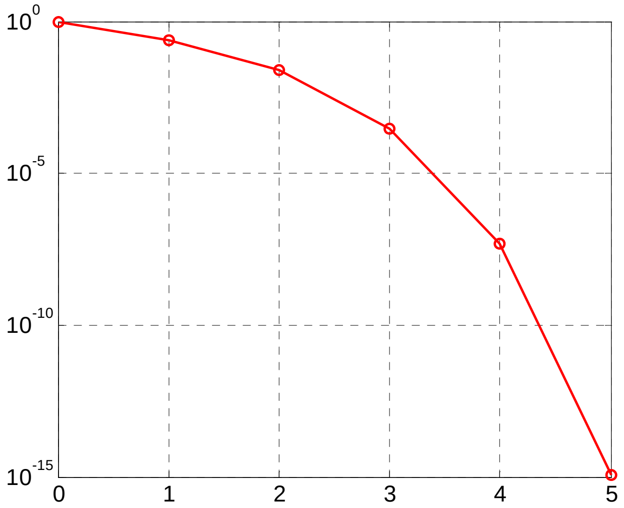
Il metodo è stato applicato per approssimare la radice x = 1 delle equazioni

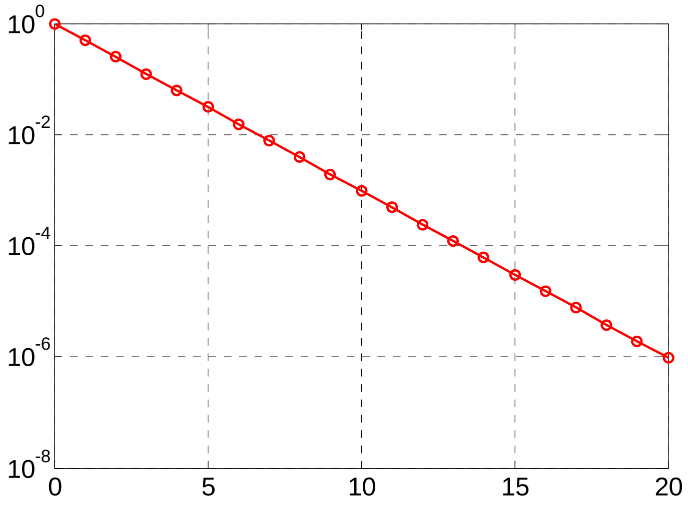
$$f(x) = x^2 - 1 \qquad e \qquad f(x) = (x-1)^2$$

La radice è semplice per la prima equazione perché f'(x)=2x e quindi f'(1)=2≠0. Dunque, il metodo ha una convergenza quadratica; come conseguenza, il grafico del logaritmo in base 10 dell'errore in funzione del numero di iterazioni ha la concavità verso il basso.

Per la seconda equazione, la radice ha molteplicità pari a 2 essendo f'(x)=2(x-1) e quindi f'(1)=0 mentre f''(x)=2 e quindi f''(1)=2≠0. Pertanto, il metodo di Newton ha ordine di convergenza p=1 e l'andamento del logaritmo in base 10 dell'errore in funzione del numero di iterazioni ha un andamento lineare.

Gli andamenti sono i seguenti





Codici

Ricordiamo che **tutti i file devono essere nella stessa cartella di lavoro del Matlab**. In pratica, questo si ottiene in due passi.

1) **Creare**, prima di aprire il Matlab, una **cartella** da qualche parte; ad esempio, lab_19_03_2014.
2) Aprire il Matlab; **spostarsi** nella cartella creata al passo precedente digitando, nella command Window di Matlab, il comando

   cd <percorso completo dove si trova la cartella>

   ad esempio,

   cd F:\materiale\corsi2014\calcoloVerona\materialeCorso\lezioni\laboratorio\lab_2\lab_19_03_2014

% script di prova per Newton (deve essere scritto in un m file, ad esempio prova Newton.m)

```
% definisco la funzione
%f  = inline('x.^2-1');
%df = inline('2*x');

f  = inline('(x-1).^2');
df = inline('2*(x-1)');

% definisco i parametri di arresto
nmax = 50;
toll = 1E-6;

% definisco il punto di partenza
x0 = 2.0;

% chiamo il metodo di Newton
[ xn ] = newton(f, df, x0, toll, nmax);

% disegno l'andamento degli errori
xv = 1.0;
absErrore = abs(xn-xv);

valoriAscisse = 0:(length(xn)-1);
semilogy(absErrore,'o-r','lineWidth',2)
grid on

% calcoliamo una stima della costante dell'errore
% che teoricamente vale |0.5*f''(1)/(f'(1)|=0.5
M = absErrore(2:end)./(absErrore(1:end-1)).^2

% ripetiamo la stessa cosa con gli scarti
%          xn(k+1)-xn(k)
%     M = ---------------------, k= 1, 2, ...
%          [ xn(k)-xn(k-1) ]^2

Ms = abs((xn(3:end)-xn(2:end-1)))./(xn(2:end-1)-xn(1:end-2)).^2
```

Questo è il codice della function di Newton; deve essere scritto in un m file che ha lo STESSO NOME della function, in questo caso newton.m.

```
function [xn] = newton(f, df, x0, toll, nmax)
%NEWTON metodo di Newton per equazioni
%  [XN] = NEWTON(F, DF, X0, TOLL)
%  dove
%       F: funzione f di f(x)=0
%      DF: derivata della funzione
%      x0: punto iniziale del metodo
%    TOLL: tolleranza sullo scarto
%      XN: vettore delle iterate prodotte dal metodo
%
%  La function risolve l'equazione f(x)=0
%  col metodo di Newton
%                      f( x(n) )
%      x(n+1) = x(n) -  ----------- , n=0,1,..
%                      f'( x(n) )
%
%  dove x0 e' il punto di partenza.
%  Il test di arresto e' basato sullo scarto
%  tra due iterate successive:
%         |  x(n+1) - x(n) | < toll
%
%

%  19-Mar-2014


%  Chiamo xold la x(n) e xnew la x(n+1)
xold = x0;  % xold <--x0
xnew = xold - f(xold)/df(xold); % xnew <-- x(1)

%  salvo le iterate create
xn = [xold; xnew];

n = 2;
while( abs( xnew-xold )>=toll & n<=nmax )
    % nuova iterazione di Newton
    xold = xnew;
    xnew = xold - f(xold)/df(xold); % calcolo x(n+1)
    xn   = [ xn; xnew ]; % accodo l'iterata

    n    = n+1;
end
```

File per lo studio delle operazioni con I vettori.

```
% 1: creare una cartella
%    nella propria home e
% 2: spostarsi in questa
%    cartella con il comando
%    cd del Matlab.
%    Ad esempio,
%       cd lab_19_03_2014
% La directory corrente si
% puo' vedere in alto della
% finestra Matlab.

% creo il vettore v
v = [1 3 5 8];

% per vedere quanti elementi
% ha v uso il comando
% length(v)
numElem = length(v);

% v(k) e' il k-esimo elemento
% di v. C'e' la convenzione
% v(end) trovo l'elemento
% in ultima posizione
vUltimo = v(end);

% voglio gli ultimi due
% elementi di v
vUltimiDue = v(end-1:end);

% vogliamo accodare un numero
% al vettore v
v = [v 13];

% e' una tecnica molto comoda
% per costruire vettori
% partendo, solitamente, dal
% vettore vuoto indicato con
% []:
% vettVuoto = []
% crea un vettore che non ha
% elementi
% Posso aggiungervi elementi
% accodandoli
vettVuoto = []
vettVuoto = [vettVuoto 1]
vettVuoto = [vettVuoto 2]

% ATTENZIONE: questo metodo
% di creare i vettori
```

```
% richiede continui accessi
% alla memoria per cui NON
% e' la strada corretta se
% ho esigenze di codici
% veloci!

% operazioni con vettori
w = [1 2 3 4 6 8]
% voglio calcolare
% ( w(k)-w(k-1) )^2, k=2,...,end
% Voglio il vettore
%
% w2=[ (2-1)^2  (3-2)^2  (4-3)^2  (6-4)^2  (8-6)^2]
%
% Posso usare un ciclo oppure...
% w2 = ([2 3 4 6 8] - [1 2 3 4 6]  ).^2
%    = (  w(2:end)  -  w(1:end-1)  ).^2

w2 = (  w(2:end)  -  w(1:end-1)  ).^2
```