Laboratorio di Calcolo Numerico
Introduzione a Matlab/Octave

Ángeles Martínez Calomardo
http://www.math.unipd.it/~acalomar/DIDATTICA/
angeles.martinez@unipd.it

Laurea in Informatica
A.A. 2018–2019

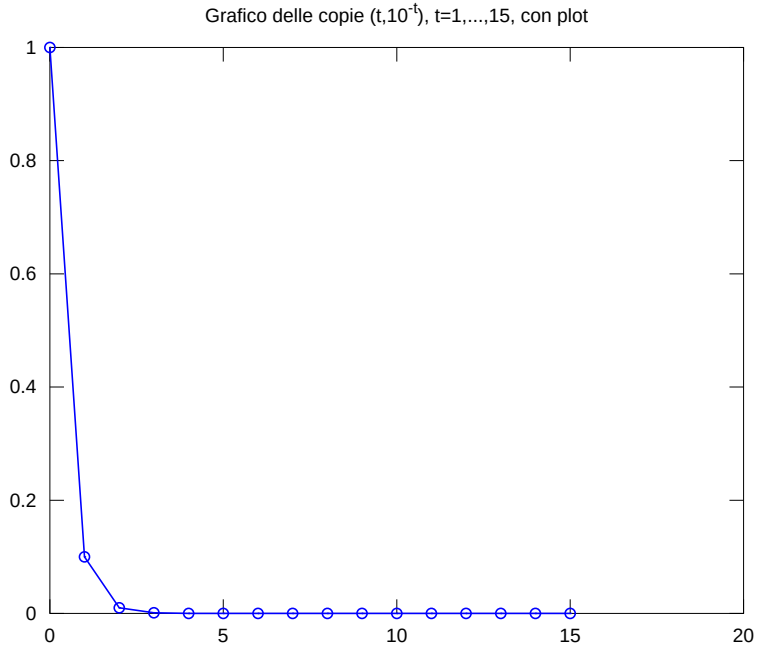
Grafici in scala semilogaritmica

- In molte aree scientifiche vengono usati grafici in scala semilogaritmica.
- Matlab/Octave fornisce tre comandi a tale proposito:
  - `semilogy`: equivalente a `plot` ma con l'asse delle ordinate in scala logaritmica
  - `semilogx`: idem con l'asse delle ascisse in scala logaritmica
  - `loglog`: entrambi gli assi in scala logaritmica
- Noi ricorreremo ai grafici in scala semilogaritmica, mediante il comando `semilogy`, nella realizzazione dei grafici che ci serviranno a studiare gli errori commessi dai metodi numerici.

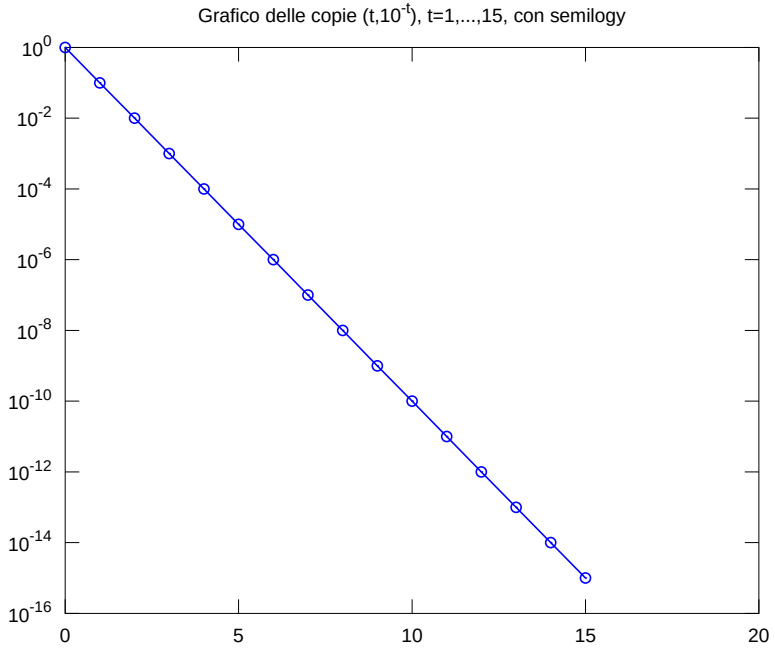
Grafici in scala semilogaritmica: comando `semilogy`

- Supponiamo di voler plottare le coppie $(t, 10^{-t})$ per $t = 0, \ldots, 15$.
- Il commando `plot` non mostra adeguatamente la differenza tra i valori poiché hanno ordini di grandezza troppo diversi.
- Questa differenza è invece palese nel grafico ottenuto con `semilogy`.

`plot`



`semilogy`



Matlab/Octave come linguaggio di programmazione

- Matlab/Octave può essere considerato un linguaggio di programmazione alla stregua di Fortran, di C, ecc.
- Non viene compilato ma interpretato (poco efficiente per calcoli intensivi).
- Un programma Matlab/Octave deve essere salvato in un m-file (file avente estensione .m).
- I programmi Matlab/Octave possono essere di due tipi:
  - `script`
  - `function`
- Strutture di programmazione basilari in Matlab/Octave:
  - Istruzione condizionale (`if else end`)
  - Cicli (`for` e `while`)

Il costrutto `for`

- La sintassi del costrutto `for` è la seguente:

```
for k = vettore
        istruzioni
end
```

- I comandi che si trovano tra `for` e `end` sono eseguiti per tutti i valori di `k` che sono nell'`vettore`.
- Esempio: calcolare la somma dei primi 10 numeri interi positivi usando un ciclo `for`.

```
somm=0;
for n=1:10
    somm=somm+n;
end
```

- Possiamo usare più cicli `for` innestati.

Matlab/Octave come linguaggio di programmazione

Operatori logici e di relazione in Matlab/Octave

Operatori logici	
& &	AND
||	OR
~	NOT

Operatori di relazione	
==	uguale
~=	diverso
<	minore
>	maggiore
<=	minore o uguale
>=	maggiore o uguale

- Il valore restituito dagli operatori può essere vero o falso e MATLAB utilliza il numero 1 per indicare il valore vero e 0 per il valore falso.
- Se, ad esempio, poniamo `x=5;` e `y=1` e scriviamo la proposizione `x < y`, MATLAB risponde con `ans = 0` indicando che il confronto esprime una condizione falsa.

Il costrutto `while`

- Per il ciclo `while` la sintassi è data da:

```
while espressione logica
        istruzioni
end
```

- Questo ciclo è usato quando le istruzioni devono essere ripetute fino a quando rimane vera l'espressione logica (numero di volte indeterminato a priori).
- **Esempio**:

```
f=1; j=1;
while j < 10
      f=f*j;
      j=j+1;
end
```

- Il codice precedente calcola il fattoriale di 9!

Esercizio

Calcolare la somma dei primi n numeri interi positivi utilizzando un ciclo `while`.

Esempio di uso del costrutto `while`

Per trovare la soluzione dell'equazione $x = \cos x$, scriviamo le istruzioni:

```
x = 1;
dif = 100;
while dif > 1e-8
      xnew = cos(x);
      dif = abs(xnew-x);
      x = xnew;
end
xnew
```

che producono il risultato

```
xnew =  0.739085136646572
```

Il costrutto `if-else-end`

```
if espressione logica
        istruzioni
end
```

Esempio

```
if a > b
    maxval = a
end
```

```
if espressione logica
        istruzioni
else
        istruzioni
end
```

Esempio

```
if x > 0
    a = sqrt(x)
else
    a = 0
end
```

```
if espressione logica 1
        istruzioni
elseif espressione logica 2
        istruzioni
...
else
        istruzioni
end
```

Esempio di `if-else-end` e `for`

Il seguente codice costruisce una matrice

```
for m = 1:k
    for n = 1:k
        if m == n
            a(m,n) = 2;
        elseif abs(m-n) == 2
            a(m,n) = 1;
        else
            a(m,n) = 0;
        end
    end
end
```

Per $k = 5$ si avrebbe:

```
a =
     2     0     1     0     0
     0     2     0     1     0
     1     0     2     0     1
     0     1     0     2     0
     0     0     1     0     2
```

Programmi in Matlab/Octave: script

- È una semplice raccolta di istruzioni o comandi Matlab/Octave senza interfaccia di input/output.
- Ad esempio, l'insieme di istruzioni

```
a=1; b=-3; c = 2;
delta = b^2-4*a*c;
if delta < 0
   disp('radici complesse')
else
   x1 = (-b-sqrt(delta))/(2*a)
   x2 = (-b+sqrt(delta))/(2*a)
end
```

una volta salvato in un m-file, di nome `eq2grado.m` diventa uno script.

- Per eseguirlo, è sufficiente scrivere dopo il prompt il nome senza estensione:

```
>> eq2grado
x1 =  1
x2 =  2
```

Programmi in Matlab/Octave: `function`

- Come lo script si definisce in un m-file, ad esempio `nomefun.m`
- La sua definizione inizia con la parola chiave `function`:

```
function [out1, ..., outn] = nomefun(in1, ..., inm)
```

- out1, $\cdots$, outn sono i parametri di output (opzionali);
- in1, $\cdots$, inm sono i parametri di input.

- Le variabili all'interno della `function` sono locali, il loro valore viene perduto al termine dell'esecuzione.
- Una funzione può esser invocata o da command window o da uno script.
- La `function` termina o all'ultima sua istruzione oppure quando si incontra per la prima volta il comando `return`.

Programmi in Matlab/Octave

Esempio

```
function [x1,x2,err] = radici(a,b,c)
err = 0;
delta = b^2-4*a*c;
if delta < 0
   err = 1; x1=0; x2=0;
   return
else
   x1 = (-b-sqrt(delta))/(2*a);
   x2 = (-b+sqrt(delta))/(2*a);
end
```

file `radici.m`

```
a=1; b=-3; c = 2;
[x1,x2,err] = radici(a,b,c)
```

file `scriptradici.m`

```
>> scriptradici
x1 =  1
x2 =  2
err = 0
```

```
>> delta
error: 'delta' undefined near line 99 column 1
```

Gestione dell'output su video

Comando `disp`

- Il comando `disp` serve per visualizzare una stringa di caratteri (testo racchiuso tra apici), o una variabile senza che ne venga visualizzato il nome.
-
```
>> x=1:2:19;
>> disp(x)
     1     3     5     7     9    11    13    15    17    19


>> disp('Questa e'' una stringa');
Questa e' una stringa
```

- Si possono visualizzare più dati in un unico comando `disp`:
  - Stringhe e variabili numeriche insieme

```
>> disp(['Convergenza in ',num2str(iter),' iterazioni']);
Convergenza in 23 iterazioni
```

- Il comando `num2str` converte un numero in una stringa.
  - Più variabili numeriche

```
>> disp([val,err,iter])
   2.1099e+00   1.0000e-10   2.3000e+01
```

- L'output del comando `disp` finisce sempre con un avanzamento di linea.

Gestione dell'output su video

Comando `fprintf`

- Per visualizzare un insieme di dati di output con un certo formato si usa il comando `fprintf` con i descrittori di formato:

Descrittore	Significato
%f	formato decimale (virgola fissa)
%e	notazione esponenziale
%i o %d	notazione per interi con segno
%g	la notazione piú compatta tra %f ed %e
%s	stringa di caratteri
\n	avanzamento di linea
\t	tabulazione
\b	backspace

- Tra % e il tipo di formattazione è possibile precisare il numero minimo di caratteri da stampare e il numero di cifre decimali dopo il punto.

Valore	%6.3f	%6.0f	%6.3e	%6.3g	%6.3d	%d
2	2.000	2	2.000e+000	2	002	2
0.02	0.020	2	2.000e−002	0.02	000	0
200	200.000	200	2.000e+002	200	200	200
`sqrt(2)`	1.414	1	1.414e+000	1.41	001	1