

UD4 - MATLAB

M-file. Efficienza degli algoritmi.
Formati d'uscita

M-files

- In MatLab è possibile eseguire istruzioni contenute in file esterni;
- Tali file sono chiamati “M-file” perché devono avere estensione “.m”;
- `what` è il comando per elencare gli M-file e i Mat-file nella directory corrente;
- Gli M-file possono contenere riferimenti ad altri M-file o ricorsivamente a se stessi;
- Esistono due tipi di M-file:
 - Script file;
 - Function file.

Script files

- Contengono sequenze di istruzioni MatLab;
- Se lo script ha nome `myfile.m` la sua invocazione dalla linea dei comandi causa l'esecuzione delle istruzioni in esso contenute;
- Le variabili contenute sono globali e causano la modifica dei valori delle variabili contenute nella sessione corrente;
- Gli script file vengono usati generalmente per inserire dati di grosse dimensioni: usando un ambiente di editing è più semplice correggere eventuali errori;

Esempio d'uso di uno script file

- Mediante un qualunque editor ASCII (NotePad o l'editor ASCII di MatLab (edit)) creazione di un file `data.m`:

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

- Per acquisire i dati contenuti occorre invocare il nome del file senza estensione:
`>> data`
- Nella sessione corrente vengono caricati i dati contenuti nello script file.

Function Files

- Consentono l'estendibilità delle funzioni di MatLab;
- Le variabili definite nei Function File sono "locali" per default;
- Dalla versione 4.0 è possibile definire variabili globali in M-files;
- Devono essere memorizzati in file di estensione ".m".

Esempio d'uso di un Function File

```
function a = randint(m,n)
%RANDINT genera una matrice mxn di valori
% casuali interi compresi fra 0 e 9.
a = floor(10*rand(m,n));
```

- Tale funzione deve essere salvata in un file di estensione “.m” con lo stesso nome della funzione (randint.m);
- La prima linea rappresenta la dichiarazione della funzione (nome, parametri di ingresso e di uscita); senza di essa il file sarebbe uno “script file”;
- Una invocazione (es: randint(4,5)) determina l’assegnazione alla variabile “a” del risultato;
- Poiché la variabile è locale alla funzione, essa non entra in conflitto con una eventuale variabile “a” presente nella sessione corrente;
- Nella sessione corrente il risultato viene assegnato alla variabile *ans*;

Esempio d'uso di un Function File

```
function a = randint(m,n,a,b)
if nargin < 3
    a = 0; b = 9;
end
a = floor((b-a+1)*rand(m,n)) + a;
```

- nargin = “number of input arguments” (= numero degli argomenti in ingresso);

Funzioni con parametri d'uscita multipli

```
function [mean, stdev] = stat(x)
% STAT Media e deviazione standard per x
% Se x è un vettore colonna, stat(x) restituisce media e
    deviazione standard.
% Se x è una matrice, stat(x) restituisce due vettori riga
    contenenti rispettivamente
    media e varianza di ogni colonna.
[m n] = size(x);
mean = sum(x)/m;
stdev = sqrt(sum(x.^2)/m - mean.^2);
```

- Dalla linea dei comandi la funzione può essere richiamata:
>> [xm, xd] = stat(x)

Altro esempio di Function file

```
function a = gcd(a,b)
% GCD Greatest common divisor (massimo comun divisore)
% gcd(a,b) è il massimo comun divisore fra a e b
% entrambi diversi da 0
a = round(abs(a));
b = round(abs(b));
if a == 0 & b == 0
    error('The gcd is not defined when both numbers
    are zero')
else
    while b ~= 0
        r = rem(a,b);
        a = b; b = r;
    end
end
```

Stringhe e Messaggi

- In MatLab una stringa è una sequenza di caratteri racchiusa fra apici singoli:

```
s = 'This is a test';
```

- Le stringhe possono essere visualizzate tramite la funzione `disp`:

```
disp('Visualizzazione di questa stringa sullo standard output')
```

- I messaggi di errore possono essere visualizzati mediante la funzione `error`:

```
error('La matrice deve essere simmetrica')
```

Essi provocano la interruzione del programma se sono eseguiti dentro un M-file.

Input

- E' possibile visualizzare messaggi durante l'inserimento dei dati in modo interattivo:
- Es:

```
iter = input('Inserisci il numero delle iterazioni: ')
```

Efficienza degli algoritmi: cputime e etime

- Esistono due misure di efficienza degli algoritmi:
 - Tempo di cpu (funzione `cputime`);
 - Tempo di esecuzione (funzione `etime`);
- `t=cputime; your_operation; cputime-t`
Calcola il tempo di cpu necessario per eseguire “your_operation”;
- `etime(T1, T0)` restituisce il tempo in secondi trascorso fra T0 e T1
T è espresso nella forma [Year Month Day Hour Minute Second]
`t0 = clock; operation; etime(clock, t0)`
Calcola il tempo assoluto in secondi trascorso per eseguire “operation”
Es: `t = clock; x = A \ b; time = etime(clock, t)`

Formati dei valori delle variabili

- In Matlab tutte le operazioni sono effettuate in doppia precisione;
- I risultati possono essere visualizzati nei seguenti formati:

<code>format short</code>	virgola fissa e 4 valori decimali (default)
<code>format long</code>	virgola fissa e 14 valori decimali
<code>format short e</code>	notazione scientifica e 4 valori decimali
<code>format long e</code>	notazione scientifica e 15 valori decimali
<code>format compact</code>	visualizza i valori in forma più compatta

Copia dello standard output

- Il comando `diary` consente di copiare sul file omonimo tutto ciò che appare sullo standard output (linea comandi);
- `diary('nomefile')` copia lo standard output su `nomefile`;
- `diary off` e `diary on` consentono rispettivamente di interrompere e riprendere copia dello standard output

Es:

```
diary( 'prova' )  
diary on  
...  
diary off
```