26/11/2013

Informatica per la Finanza
Informatica per le Scienze Economiche
Informatica e Programmazione per il Turismo

Funzioni

- **Funzioni**: nel senso della programmazione tradizionale;
  - hanno parametri in uscita ed in ingresso ed un proprio spazio di memoria
    - Un file può contenere più funzioni
      - solo la prima è accessibile dall'ambiente comune,
      - le altre possono essere richiamate solo dalle funzioni all'interno del file
    - Il nome del file deve essere uguale al nome della prima funzione in esso contenuta

Funzioni

- Una funzione in MatLab ha la seguente sintassi:
- **`function [ri1,..,ri2]= Nome (p1,..,pn)`**
  - **`ri1, ..., rin`** sono i valori restituiti
    - Con un solo valore si possono omettere le parentesi quadre
    - Se non restituisce nulla si può non mettere nulla o usare **`[]`**
  - **`p1, ...,pn`** sono i parametri in ingresso
  - **`end`** al termine del codice della funzione non è obbligatorio ma è raccomandato

Funzioni

```
function [a b] = provaFun (t, q)
% prova

t=t+4;
q=q*10;

a = t+q;
b = -t;
```

Funzioni

- In MatLab le funzioni possono essere richiamate
  - con un numero di parametri inferiore a quelli richiesti
  - restituire un numero di valori inferiore a quelli richiesti
- è possibile determinare attraverso le variabili di ambiente
  - **`nargin`**
    - con quanti argomenti in ingresso è stata richiamata la funzione
  - **`nargout`**
    - con quanti argomenti in uscita è stata richiamata la funzione.
    - se il valore della funzione non è assegnato a nessun vettore viene visualizzato solo il primo valore restituito

Funzioni

```
function [a b] = provaFun (t, q)
% prova

if(nargin == 1) q = 4; end
t=t+4;
q=q*10;
a = t+q;
b = -t;
if (nargout == 1) a = a * b; end
```

ESEMPIO

```
function [s] = somma(n, m)
%somma(n,m) calcola la somma dei numeri fra n ed m
%se c'è un solo parametro calcola la somma fra 1 ed n

 if nargin == 0
      error('dammi almeno un numero!')
  end
  if nargin == 1
      m=n;
      n=1;
  end

  if n > m
      sc = n;
      n = m;
      m = sc;
  end

  % vale sempre n <= m
  s = sum(n:m);
end
```

La funzione Matlab calcola la somma dei numeri compresi fra n e d m dove n ed m sono i parametri della funzione (n > m > 0).
Se m non è specificato calcola la somma fra 1 ed n

Esempio

- Scrivere la funzione che calcola la soluzione
  - di un'equazione di primo grado se ha 2 parametri in ingresso,
    - aX = b => aX -b=0
  - quelle di un'equazione di secondo grado se ha tre parametri in ingresso
    - $aX^2+bX+c=0$

Soluzione iterativa

```
function [ u1, u2, nrad ] = equazioni( a, b, c)
%UNTITLED4 calcola le radici di un'equazione di 1^ o 2^ grado
u2=1;
u1=1;
if (nargin == 2)  %equazione di primo grado
    u1 = -b/a;
    nrad=1;
elseif (nargin == 3) %equzione di secondo grado
    delta2 = b^2 - 4*a*c;
    if (delta2 > 0)
        u1 = (-b + sqrt(delta2))/(2*a);
        u2 = (-b - sqrt(delta2))/(2*a);
        nrad=2;
    elseif (delta2==0)
        u1=-b/(2*a);
        u2=u1;
        nrad=1;
    else
        nrad=0;
    end
else
    error('attenzione tratto solo equazioni di primo e secondo grado');
end
end
```


Soluzione vettoriale

```
function [s1, s2] = Prova(a, b, c)

if (nargin==2) %equazione di 1^ grado
    s1 = roots([a -b]);
elseif (nargin == 3) %equazione di 2^ grado
    if (nargout == 2)
        if (b^2 - 4*a*c < 0)
            disp('soluzioni complesse')
        elseif (b^2 - 4*a*c == 0)
            disp('soluzioni coincidenti')
        end
        sol = roots([a b c]);
        s1 = sol(1);
        s2 = sol(2);
    else
        error('devi appoggiare su due variabili');
    end
else
    error('occorrono almeno due argomenti in ingresso!')
end

end
```

Funzioni: parametri in ingresso

- La funzione Prova può esser richiamata con al massimo tre parametri (a, b, c)
  - se è richiamata con un parametro genera la matrice Identità a x a
  - Se è richiamata con due parametri genera la matrice di zeri axb
  - Se è richiamata con tre parametri, assume che questi siano le radici di un polinomio di terzo grado e genera il polinomio con queste radici

Funzioni

```
function x = Prova( a, b,c)
%Prova tre parametri in ingresso 1 in uscita
%   fa tante cose

if (nargin == 1)
    x=eye(a);
elseif(nargin == 2)
    x=zeros(a,b);
elseif (nargin == 3)
    x=poly([a,b,c]);
end

end
```

Funzione: parametri in uscita

- La funzione Prova può restituire al massimo tre valori:
  - quando viene richiamata con un solo valore da restituire restituisce 3,
  - quando viene richiamata con due valori da restituire restituisce 4 e 3
  - quando viene richiamata con tre valori da restituire restituisce la matrice identità 3x3, quindi 4 e quindi 3

Funzioni

```
function [ x, y, z ] = uscite(  )

 if nargout == 1
     x = 3;
 elseif nargout == 2
     x=4;
     y=3;
 elseif nargout == 3
     y=4;
     z=3;
     x = eye(3);
 end
end
```

Funzioni

- In Matlab è possibile creare funzioni con un numero di argomenti variabili
  - In ingresso usando **`varargin`**
    - Può essere usato da solo
    - Se è usato con altri parametri è l'ultimo parametro della funzione
    - È un vettore di celle
      - » le posizioni vanno racchiuse fra **`{}`** invece che fra **`()`**.
    - Il numero di parametri passati ad una funzione è contenuto nella variabile **`nargin`**.
  - In uscita usando **`varargout`**
    - Comportamento analogo a **`varargin`**
    - Il numero di parametri passati ad una funzione è contenuto nella variabile **`nargout`**.

Funzioni

- La funzione a lato stampa il valore di tutti i parametri in ingresso alla funzione
  - **`varargin`** contiene tutti i parametri (è un vettore di celle, si noti la sintassi per indicare l'argomento)

```
function testVarIn(varargin)
  if nargin > 0
   for k = 1:nargin
       varargin{k}
   end
  else
       disp('richiamata senza argomenti')
  end
end
```

Si noti l'uso delle parentesi graffe; **`varargin`** è un vettore di celle (in seguito si parlerà dei vettori di celle).

Funzioni

```
function prv(a, varargin)
 disp(nargin)
 for i=1:nargin
     if i == 1
         disp(a)
     else
         disp(varargin{i-1})
     end
 end
end
```

Il numero totale di parametri è indicato da **nargin** quindi anche il parametro **a** è conteggiato

Esempio

- Scrivere una funzione che ha come argomento dei vettori di numeri e restituisce il **vettore** con la somma degli elementi di ogni vettore. Restituisce anche la posizione e il vettore che ha somma maggiore.

Versione iterativa

```
function [vt,posMax,s]=maggiorSomma (varargin)
   s=[];
   if (length(varargin) == 0)
      s = [];  vt=[]; posMax=[];
      return;
   end
   posMax=1;
   s = [sum(varargin{1})];
   for i=2:length(varargin)
       if (s(posmax) < sum(varargin{i}))
          posMax=i;
       end
       s=[s, sum(varargin{i})]
   end

   vt=varargin{posMax};

end
```


19

Versione vettoriale

```
function [vt,posMax,s]=maggiorSomma (varargin)
   s=[];
   if (isempty(varargin))
      vt=[]; posMax=[];
      return;
   end

   for i=1:length(varargin)
    s = [s sum(varargin{i})];
   end
   posMax = max(s);
   vt=varargin{posMax};
end
```


20

Esempio

- Scrivere la funzione `monica()` che ha come argomento un vettore che rappresenta un polinomio e lo restituisce in forma monica (coefficiente di grado maggiore uguale ad 1)

```
function [ polUs ] = monica( polIn )
  polUs = polIn./polIn(1);
end
```


Esercizio 1

- Scrivere la funzione `inversa()` che ha in ingresso una matrice di 2 dimensioni e che restituisce la sua inversa se non è singolare, altrimenti restituisce `NaN`

Esempio 1

```
function [inve] = inversa(m)
% calcola la matrice inversa di m se possibile

%verifica se ha due dimensioni
if(ndims(m) ~= 2)
    inve = NaN;
    return
end
% verifica se m e' quadrata
sz = size(m);
if(sz(1) ~= sz(2))
    inve = NaN;
    return
end

% verifica se è invertibile

if (det(m) ~= 0)
    inve = inv(m);
else
    inve = NaN;
end
end
```


Esempio 2

- Scrivere la funziona `sommaPoli()` che dati due polinomi li somma fra loro
  - Usare il semplice operatore + non va bene se i polinomi hanno dimensioni diverse
    - Si deve lavorare con polinomi della stessa dimensione
  - La funzione può avere un numero arbitrario di argomenti

Esempio 2

- Pol = [2 3 4]
- 4 + 3 x + 2 x^2
- Lavoro su coppie di polinomi

- P1 = [3 5 7] corto
- P2 = [9 5 4 7 10] lungo

Esempio 2

```
function [sommaPol] = sommaPoli(p1, p2)
   l = [length(p1)];
   l = [l length(p2)];
   lmax = max(l);

   p1=[zeros(1,lmax-length(p1)), p1];
   p2=[zeros(1,lmax-length(p2)), p2];

   sommaPol = p1 + p2;
end

function [ sommaP ] = sommaPoliMult( varargin )
  if isempty(varargin)
      sommaP=[];
  elseif length(varargin) == 1
      sommaP = varargin{1};
  else
    sommaP = sommaPoli(varargin{1}, varargin{2});
      for i=3:length(varargin)
          sommaP=sommaPoli(sommaP, varargin{i});
      end
  end
end
```

Esempio 2

```
function [s]=sommaPoli (varargin)

if (length(varargin)==0) %non ci sono argomenti
    s=[];
else
    if (length(varargin) == 1)
        s = varargin{1};
    else %ci sono due o piu' polinomi
        s = varargin{1};

        for i=2:length(varargin)
            s = somma2pol(s,varargin{i});
        end
    end
end

end

function [r]=somma2pol(p1, p2)
r = [];
lprimo = length(p1);
lsecondo= length(p2);
if (lprimo > lsecondo)
    p2 =[zeros(1,lprimo-lsecondo),p2];
elseif (lprimo < lsecondo)
    p1=[zeros(1,lsecondo-lprimo),p1];
end
r = p1 + p2;
end
```


27

Esempio

- Scrivere la funzione `estrai()` che ha come argomento una matrice M nXm, un valore V e restituisce il vettore con tutti i valori maggiori di V contenuti in M; se V non è specificato vale 0.

```
function [ vt ] = estrai( M, V )
  if nargin == 0
      error('pochi argomenti')
  end

  if nargin == 1
      V = 0;
  end

  vt = M(find(M>V));
end
```

Esempio

- Dato un polinomio sottoforma di vettore determinare gli zeri reali e quindi disegnare:
  - la funzione fra i primi due zeri reali con 5000 punti equispaziati;
  - altrimenti disegnare la funzione fra -1 ed 1 con 5000 punti equispaziati

Esempio

```
function [  ] = disegna( poli )
  radici = roots(poli);

  radReal=[];
  %metto da parte tutte le radici reali
  for i=1:length(radici)
      if isreal(radici(i))
          radReal = [radReal radici(i)];
      end
  end

  radReal = sort(radReal);
  if length(radReal) > 1
      x=linspace(radReal(1),radReal(2),5000);
      y=polyval(poli,x);
  else
      x=linspace(-1,1,5000);
      y=polyval(poli,x);
  end
  plot(x,y);
end
```

Altri tipi di dati

- MATLAB ha anche altri tipi di dati
  - Array multidimensionali
    - con 3 o più dimensioni
  - Celle
    - Matrici multidimensionali i cui elementi sono copie di altre matrici (non hanno il vincolo di avere elementi dello stesso tipo e dimensione)
  - Stringhe
    - Vettori di caratteri
  - Strutture (Struct)
    - Matrici multidimensionali alle cui componenti si accede mediante il nome

Array

- Alcune funzioni che danno informazioni sugli array

`length(matrice)`: restituisce il numero di elementi della dimensione con più elementi

`ndims(matrice)`: restituisce il numero di dimensioni della matrice

`numel(matrice)`: restituisce il numero degli elementi della matrice

`size(matrice)`: restituisce il numero di elementi di ogni dimensione sottoforma di vettore

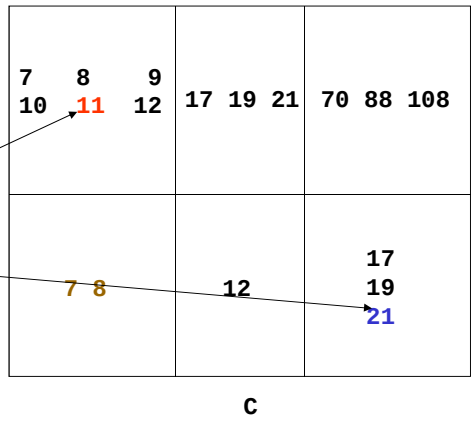
Per una trattazione sintetica sugli array si veda l'appendice Matlab

Celle

- Sono collezioni di array si definiscono con le parentesi graffe
  - Gli array sono copie degli array originali
  - L'accesso ai singoli elementi avviene in maniera posizionale tramite parentesi graffe

Celle

```
A=[7  8  9;
   10 11 12]
C = {
 A sum(A) prod(A);
 [7 8] 12 sum(A)'}
C{1,1}(2,2)
C{2,3}(3,1)
C{2,1}
```



Stringhe

- Le stringhe sono vettori di caratteri
  - Si definiscono mettendo fra apici la sequenza di caratteri

  Esempio

  S = 'Una stringa di caratteri'

  S(1)

  length(S)

Stringhe

- I singoli caratteri sono immagazzinati in formato numerico
  - Per trasformare la stringa in un vettore numerico si può usare

    `double(stringa)`
  - L'operazione inversa avviene con

    `char(vettore)`
  - Il seguente codice permette di visualizzare la tabella ASCII dei caratteri stampabili

```
F = reshape(32:127,16,6)';
char(F)
```

Stringhe

- Le stringhe possono essere concatenate come un normale vettore:

  s='uno';
  v='due'
  k=[s v 'tre']

  – Attenzione a quando si concatenano verticalmente perché le righe devono avere la stessa dimensione
    - In alternativa usare delle celle
      – **cellstr(stringa)**
      – **char(celle)**

Algoritmo di Cesare

```
function [ testoU] = Cesare( testoI, s )
%Cesare applica l'algoritmo di Cesare per criptare
%   testoI testo da criptare
%   s di quanto spostare (se non indicato 5)

   if (nargin == 0)
      error ('almeno il testo!')
   end

   if (nargin == 1)
        s = 5;
   end
   tN = double(testoI) + s;
   testoU = char(tN);

end
```

Strutture

- Le strutture sono dei vettori in cui l'accesso alle parti (campi) avviene dando il nome della parte:

```
s.uno=8
s.due='ciao'
```

In maniera equivalente si può scrivere:

```
S = struct ('uno',8,'due','ciao')
```