UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI INGEGNERIA INDUSTRIALE

DIPARTIMENTO MATEMATICA

DIPARTIMENTO DI MATEMATICA "TULLIO LEVI-CIVITA"

Laboratorio di Calcolo Numerico LAB 7

Comandi di input/output e vettorializzazione operazioni

Docenti: E. Bachini, L. Bruni

Email: elena.bachini@unipd.it Email: bruni@math.unipd.it

17 aprile 2024

Outline

Gestione dell'output su video

Visualizzare tabelle con disp I

Con il comando `disp` è possibile una veloce visualizzazione di tabelle. Ad esempio:

```
% tabellasemplice.m
% costruisce una tabella con tre colonne
%
n = input('Inserisci il numero di valori:  ');
x = linspace(0,pi,n);
xvs = sin(x);
xvc = cos(x);
% Attenzione: devono essere trasformati in vettori colonna!
% Altrimenti il risultato e' errato (provare!).
x = x(:); xvs = xvs(:); xvc = xvc(:);
% costruisco una matrice concatenando i vettori colonna
matrice = [x,xvs,xvc];

% definisco formato e visualizzo
format short e
disp('-------------------------------------------------------');
disp('Colonna 1 = angolo, Colonna 2 = seno, Colonna 3 = coseno');
disp('-------------------------------------------------------');
disp(matrice);
```

Visualizzare tabelle con disp II

Genera il seguente output su video:

```
>> tabellasemplice
Inserisci il numero di valori:  5
-------------------------------------------------------------
Colonna 1 = angolo, Colonna 2 = seno, Colonna 3 = coseno
-------------------------------------------------------------
             0              0   1.0000e+00
   7.8540e-01   7.0711e-01   7.0711e-01
   1.5708e+00   1.0000e+00   6.1232e-17
   2.3562e+00   7.0711e-01  -7.0711e-01
   3.1416e+00   1.2246e-16  -1.0000e+00
```

Si noti che in questo caso tutti i valori vengono rappresentati con lo stesso FORMATO (quello definito con il comando `format`). Se vogliamo utilizzare formati diversi per le colonne dobbiamo utilizzare il comando `fprintf`.

formato di **visualizzazione**

`format`

Il comando `format` controlla la configurazione numerica dei valori esposta da MATLAB; il comando regola solamente come i numeri sono **visualizzati** o **stampati**:

- `format short`
- `format short e`
- `format short g`
- `format long`
- `format long e`
- `format long g`

Attenzione: `format` non influisce su come MATLAB calcola o salva in memoria i numeri. MATLAB lavora sempre in doppia precisione.

Esempio

```
>> x = [4/3 1.2345e-6];
>> format short
>> x
    1.3333    0.0000
>> format short e
>> x
    1.3333e+00 1.2345e-06
>> format short g
>> x
    1.3333 1.2345e-06
>> format long
>> x
    1.33333333333333 0.00000123450000
>> format long e
>> x
    1.333333333333333e+00 1.234500000000000e-06
>> format long g
>> x
    1.33333333333333 1.2345e-06
```

Stampa a video I

Si può avere una gestione molto più personalizzata delle stampe a video:

`fprintf`

Per visualizzare in modo più elegante un insieme di dati di output con un certo *formato* si utilizzano i comandi `fprintf` e `sprintf`:

`fprintf` ('*formato*', *variabili*)

scrive su video il valore delle *variabili* indicate, utilizzando il *formato* definito (non è quello definito dal comando `format`!)

Stampa a video II

`sprintf`

$$str = \texttt{sprintf}\ (\text{'}formato\text{'},\ variabili)$$

indirizza su una stringa di testo di nome *str* i valori delle *variabili* indicate, con il *formato* definito. Per visualizzare il tutto, è sufficiente poi utilizzare in comando `disp(`*str*`)`.

'*formato*'

Il *formato* è una stringa che contiene:

- i caratteri che si vogliono visualizzare
- nelle posizioni in cui si vuole venga inserito il valore, deve essere indicato uno dei seguenti formati che iniziano con il carattere %.

Tali codici di formati sono abitualmente seguiti da due interi separati da un punto (ad esempio 6.3). *Il primo numero indica quante colonne si desiderano impegnare in uscita ed il secondo il numero di cifre della parte frazionaria.*

I formati speciali `\n` e `\t` servono per organizzare le linee dell'output.

Codici di formato

Codice di formato	Significato
`%s`	formato stringa
`%d`	formato decimale
`%g`	seleziona il formato per numeri interi, fixed point o esponenziali
`%f`	formato fixed point del numero (esempio 1343.675432)
`%e`	formato esponenziale del numero (esempio 1.34376e+003)
`\n`	inserisce carattere di ritorno a capo
`\t`	inserisce carattere di tabulazione

Esempi di formato

Valore	%6.3f	%6.0f	%6.3e	%6.3g	%6.3d	%6.0d
2	2.000	2	2.000e+00	2	002	2
0.02	0.020	0	2.000e-02	0.02	2.000e-02	2e-02
200	200.000	200	2.000e+02	200	200	200
sqrt(2)	1.414	1	1.414e+00	1.41	1.414e+00	1e+00
sqrt(0.02)	0.141	0	1.414e-01	0.141	1.414e-01	1e-01
sqrt($2 \cdot 10^{-20}$)	0.000	0	1.414e-10	1.41e-10	1.414e-10	1e-10

Visualizzazione di tabelle I

```
% tabella.m
% costruisce la tabella dei valori che il seno e il coseno
% assumono in n punti equispaziati di (0,pi)
n = input('Inserisci il numero di valori:  ');
x = linspace(0,pi,n);
s = sin(x);
c = cos(x);
disp('-------------------------------------------------------');
fprintf('k\t x\t       sin x\t       cos x\n');
disp('-------------------------------------------------------');
for i=1:n
 fprintf('%d\t %3.2f\t %8.5f\t %8.5f \n',i,x(i),s(i),c(i));
end
```

```
>> tabella
Inserisci il numero di valori:  5
-------------------------------------------------------
k     x          sin x         cos x
-------------------------------------------------------
1     0.00       0.00000       1.00000
2     0.79       0.70711       0.70711
3     1.57       1.00000       0.00000
4     2.36       0.70711      -0.70711
5     3.14       0.00000      -1.00000
```

Visualizzazione di tabelle II

NB: si poteva fare anche con `sprintf` ...

```
% tabella.m
% costruisce la tabella dei valori che il seno e il coseno
% assumono in n punti equispaziati di (0,pi)
n = input('Inserisci il numero di valori:  ');
x = linspace(0,pi,n); % vettore riga
s = sin(x);           % vettore riga
c = cos(x);           % vettore riga
disp('-------------------------------------------------------');

stringa = sprintf('k\t x\t      sin x\t      cos x');
disp(stringa);

% SI NOTI LA CONCATENAZIONE DI VETTORI RIGA e nessun ciclo for
disp('-------------------------------------------------------');
stringa=sprintf('%d\t %3.2f\t %8.5f\t %8.5f\n',[1:n;x;s;c]);
disp(stringa)
```

Gestione dell'output su file

Aprire file

`fopen`

Con il comando

```
fid=fopen('nomefile.estensione','codicepermessi');
```

si ottengono 2 effetti:

- alla variabile `fid` (identificativo del file) viene assegnato un valore che consente (vedi seguito) di "puntare" univocamente al file esterno indicato nel parametro `'nomefile.estensione'` di tipo stringa;
- se alla stringa `'codicepermessi'` corrisponde la possibilità di scrittura, allora (qualora un tale file non esistesse nella current folder) il file `nomefile.estensione` viene creato ed aperto.

Permessi

Ci sono molte possibilità, noi useremo solo `'w'` e `'a'`,

'r'	Open file for reading
'w'	Open or create new file for writing. Discard existing contents, if any
'a'	Open or create new file for writing. Append data to the end of the file
'r+'	Open file for reading and writing
'w+'	Open or create new file for reading and writing. Discard existing contents, if any
'a+'	Open or create new file for reading and writing. Append data to the end of the file
'A'	Open file for appending without automatic flushing of the current output buffer
'W'	Open file for writing without automatic flushing of the current output buffer

Si noti che usando l'opzione 'w', se il file esiste tutti i contenuti precedenti vengono perduti e vengono sostituiti con quanto viene poi scritto.
Con l'opzione 'a', invece, le nuove scrittura vengono effettuate dopo quanto si trova già scritto nel file esterno.

Scrivere su file e salvare

```
fprintf(fid,....);
```

Scrive l'output sul file puntato da `fid` l'output di `fprintf`.
Attenzione: se il file non è aperto con un permesso che consenta la scrittura il comando non ha effetto e matlab non restituisce errore!
Il comando

```
fclose(fid);
```

salva e chiude il file puntato da `fid`.

Comando type

Il comando

```
type nomefile.estensione
```

permette di visualizzare nella finestra comandi il contenuto di un file di tipo testo di nome `nomefile.estensione`, senza apici!(anche uno script o una function Matlab).

Esempi di scrittura su file I

```
>> % apro in scrittura (il file non esiste)
>> ff = fopen('file.txt','w');
>> x = 2; y = 3;
>> fprintf (ff,' x = %5.2f y= %5.2f \n',x,y);
>> fclose (ff);
>> type file.txt
  x =  2.00 y=  3.00

>> % riapro in scrittura (il file esiste gia')
>> ff1 = fopen('file.txt','w');
>> x = 20; y = 30;
>> fprintf (ff1,' x = %5.2f y= %5.2f \n',x,y);
>> fclose (ff1);
>> type file.txt
 x = 20.00 y= 30.00

>> % apro in modalita' aggiungi
>> ff2 = fopen('file.txt','a');
>> x = 200; y = 300;
>> fprintf (ff2,' x = %5.2f y= %5.2f \n',x,y);
>> fclose (ff2);
>> type file.txt
  x = 20.00 y= 30.00
  x = 200.00 y= 300.00
```

Esempi di scrittura su file II

```
>> % Scrivo tre valori con significato su una stessa riga sul file
>> n=4; x=1234.5467; err=1.345677e-4;
>> f1 = fopen('FILEOUT1.DAT','w');
>> fprintf(f1, ...
    'Iter = %5.0f \t Valore x = %10.7f \t Errore = %7.2e \n', ...
     n, x, err);
>> fclose(f1);
>> type FILEOUT1.DAT
Iter =     4      Valore x = 1234.5467000      Errore = 1.35e-04
```

```
>> % Creo una matrice e la scrivo sul file esterno
>> A = [1 2 3; 4 5 6; 7 8 9];
>> f2 = fopen('FILEOUT2.DAT','w');
>> fprintf(f2,'%5.0f', A);
>> fclose(f2);
>> type FILEOUT2.DAT

    1    4    7    2    5    8    3    6    9
>>
```

Cosa è accaduto? Gli elementi della matrice sono stati scritti sequenzialmente per colonne come se fosse un vettore!

Esempi di scrittura su file III

```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> f3 = fopen('FILEOUT3.DAT','w');
>> fprintf(f3,'%5.0f \n', A);
>> fclose(f3);
>> type FILEOUT3.DAT
     1
     4
     7
     2
     5
     8
     3
     6
     9
```

Cosa è accaduto? A nulla è servito usare nel `'formato'` il parametro `\n`. L'unica differenza è che li scrive in colonna.

Esempi di scrittura su file IV

Come fare allora, per scrivere una matrice? Posso usare un ciclo `for`

```
>> A = [1 2 3; 4 5 6; 7 8 9];
>> f4 = fopen('FILEOUT4.DAT','w');

>> for i=1:3
     fprintf(f4,'%5.0f %5.0f %5.0f \n', A(i,:))
   end

>> fclose(f4);
>> type FILEOUT4.DAT

    1     2     3
    4     5     6
    7     8     9
```

Esempi di scrittura su file V

```
>> % Creiamo un vettore riga x
>> x = 0:.1:1;
>> % Creiamo una matrice A che abbia per
>> %   prima riga il vettore x
>> %   seconda riga il vettore risultante exp(x)
>> A = [x; exp(x)];

>> f5 = fopen('exp.txt','w');
>> fprintf(f5,'%6.2f %12.8f\n', A);
>> fclose(f5);
>> type exp.txt

  0.00   1.00000000
  0.10   1.10517092
  0.20   1.22140276
  0.30   1.34985881
  0.40   1.49182470
  0.50   1.64872127
  0.60   1.82211880
  0.70   2.01375271
  0.80   2.22554093
  0.90   2.45960311
  1.00   2.71828183
```

Altri esempi per la scrittura di matrici su file I

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

     1     4     6     5     8
     6     7     4     3     1
     8     7     3     2     1
>> [n,m] = size(A);
>> f6 = fopen('matrice.out','w');
>> % USO DI DUE CICLI FOR
>> for i=1:n
>>     for j=1:m
>>         fprintf(f6,'%8.4f\t',A(i,j));
>>     end
>>     fprintf(f6,'\n');
>> end
>> fclose(f6);
>>
>> type matrice.out

  1.0000    4.0000    6.0000    5.0000    8.0000
  6.0000    7.0000    4.0000    3.0000    1.0000
  8.0000    7.0000    3.0000    2.0000    1.0000
```

Altri esempi per la scrittura di matrici su file II

```
>> A=rand(3,2);
>> [n,m] = size(A);
>> f7 = fopen('randmatrix.out','w');
>> fprintf(f7,'%s \n',' Scrive Matrice Random');
>> fprintf(f7,'\n');
>> for i=1:n
>>     for j=1:m
>>         fprintf(f7,'%18.16e\t',A(i,j));
>>     end
>>     fprintf(f7,'\n');
>> end
>> fclose(f7);
>> A
A =
    0.0503    0.8744
    0.4154    0.0150
    0.3050    0.7680
>> type randmatrix.out
  Scrive Matrice Random

  5.0268803746872939e-02   8.7436717158762622e-01
  4.1537486044322269e-01   1.5009498676616266e-02
  3.0499867700349187e-01   7.6795039001114140e-01
```

Gestione dell'input da file

Importare dati da file esterno I

In successione, bisogna

- Aprire il file
- Leggere dal file
- Chiudere il file

Aprire il file

fid = `fopen` (*'stringa'*);
dove *fid* è una variabile che identifica il file e *'stringa'* definisce il nome del file, apre il file.

Chiudere il file

`fclose` (*fid*);
dove *fid* è la variabile che identifica il file, chiude il file.

Importare dati da file esterno II

Importare dati da file ed assegnarli ad una variabile

$$var = \texttt{fscanf}\ (fid,\ 'formato',\ size);$$

dove *fid* è la variabile che identifica il file, legge tutti i dati contenuti nel file identificato da *fid*, convertendoli in base al *formato* (stringa) specificato e memorizzandoli nella variabile *var*.
size indica la dimensione della variabile *var* e può essere scritto come

nval	legge *nval* numeri memorizzandoli in un vettore colonna
[*nrighe*, *ncol*]	legge *nrighe* $\times$ *ncol* di numeri memorizzandoli in una matrice che ha tale dimensione

Il valore *ncol* può essere posto uguale alla costante `inf`. In tal caso legge per righe tutti i valori presenti nel file esterno.

Esempio 1 di lettura da file

```
>> x = 0.1:0.1:0.3;
>> y = 2:4;
>> f1 = fopen('dati.in','w');                % apre in scrittura
>> fprintf(f1,'%6.2f %12.8f\n', [x;y]);      % scrive nel file
>> fclose(f1);                               % chiude file
>> type dati.in                              % visualizza dati file
   0.10    2.00000000
   0.20    3.00000000
   0.30    4.00000000
>> f1 = fopen('dati.in');                    % apre file in lettura
>> vet = fscanf(f1,'%g %g',2)                % legge la prima riga
vet =
    0.1000
    2.0000
>> vet1 = fscanf(f1,'%g %g',2)               % legge la seconda riga
vet1 =
    0.2000
    3.0000
>> vet2 = fscanf(f1,'%g %g',2)               % legge la terza riga
vet2 =
    0.3000
    4.0000
>> fclose(f1);                               % chiude file
```

Esempio 2 di lettura da file I

```
>> x = 0:0.1:1;
>> y = exp(x);
>> f1 = fopen('exp.txt','w');              % apre file in scrittura
>> fprintf(f1,'%6.2f %12.8f\n', [x;y]);    % scrive nel file
>> fclose(f1);                             % chiude il file
>> type exp.txt
  0.00   1.00000000
  0.10   1.10517092
  0.20   1.22140276
  0.30   1.34985881
  0.40   1.49182470
  0.50   1.64872127
  0.60   1.82211880
  0.70   2.01375271
  0.80   2.22554093
  0.90   2.45960311
  1.00   2.71828183
```

Esempio 2 di lettura da file II

```
>> f1 = fopen('exp.txt');                    % apre in lettura
>> B = fscanf(f1,'%g %g',[2,inf]);           % il file ha 2 colonne
>> % legge per righe 2 elementi sino alla fine del file
>> % B sara' una matrice di dimensione 2 righe x 11 colonne
>> fclose(f1);                                % chiude il file

>> % bisogna trasporre la matrice per avere 11 righe e 2 colonne
>> format long g
>> B = B'
B =
                         0                         1
                       0.1                1.10517092
                       0.2                1.22140276
                       0.3                1.34985881
                       0.4                 1.4918247
                       0.5                1.64872127
                       0.6                 1.8221188
                       0.7                2.01375271
                       0.8                2.22554093
                       0.9                2.45960311
                         1                2.71828183
```

Esempio 3 di lettura da file I

```
>> type FILEOUT4.DAT              % contiene una matrice scritta per righe
     1     2     3
     4     5     6
     7     8     9
>> f4 = fopen('FILEOUT4.DAT');          % apre il file in lettura
>> A = fscanf(f4,'%g %g %g',[3,inf]);   % il file ha tre colonne
>>  % legge per righe 3 elementi sino alla fine del file
>>  % A sara' una matrice di dimensione 3 righe x 3 colonne
>> fclose(f4);                          % chiude il file
>>  % bisogna trasporre la matrice per avere le componenti
>>  % corrispondenti a quelle della matrice del file
>> A = A'
A =
     1     2     3
     4     5     6
     7     8     9
```

Il comando `fscanf` può anche essere utilizzato nella forma

$$[var, numval] = \texttt{fscanf}\ (fid, formato, size);$$

dove il valore restituito *numval* indica il numero di dati letti sull'intero file.

Esempio 3 di lettura da file (continua) I

```
>> type exp.txt
   0.00    1.00000000
   0.10    1.10517092
   0.20    1.22140276
   0.30    1.34985881
   0.40    1.49182470
   0.50    1.64872127
   0.60    1.82211880
   0.70    2.01375271
   0.80    2.22554093
   0.90    2.45960311
   1.00    2.71828183
>> f1 = fopen('exp.txt');                % apre il file in lettura
>> [B, nn] = fscanf(f1,'%g %g',[2,inf]); % contiene due colonne
>> fclose(f1);                           % chiude il file
>> nn
nn =
     22
>>  % ha letto 22 dati!
```

Vettorializzazione delle operazioni

Morale:

Matlab nasce per il calcolo con array

Ai fini di una programmazione efficiente, in matlab è necessario vettorializzare quanto possibile i programmi, evitando quanto possibile i cicli (for o while) per la creazione di vettori e matrici

A tal fine si possono sfruttare (con estrema attenzione) i vari (e differenti) usi degli operatori (non commutativi!!) `.*`, `./` e `.^`.

Si raccomanda un ripasso delle slides LAB 4

Esempio semplice vettorializzazione I

Si calcolino, per $N = 1, 2, \ldots, 10^2$, i valori della somme parziali S_N della serie

$$\sum_{j=1}^{+\infty} \frac{(-1)^j}{j}.$$

Evitiamo i cicli:

```
s = cumsum((-1).^(1:10^2).*(1:10^2).^(-1))
```

Si dia il comando `doc cumsum` per vedere il significato e come funziona tale comando.

Attenzione: questo algoritmo soffre di instabilità per cancellazione numerica per N grande.

Esempio vettorializzazione più complicata I

Assegnati i vettori $u \in \mathbb{R}^m$, $v \in \mathbb{R}^n$, vogliamo creare la matrice $(A_{i,j})_{i=1,2,\ldots m; j=1,2,\ldots,n}$

$$A_{i,j} = \begin{cases} \frac{1}{u_i - v_j}, & |i-j| \leq 3 \\ u_i - v_j, & |i-j| > 3 \end{cases}.$$

Una possibile soluzione potrebbe essere

```
m = length(u);
n = length(v);
for i = 1:m
    for j=1:n
        if abs(i-j) <= 3
            A(i,j) = 1/(u(i)-v(j));
        else
            A(i,j) = u(i)-v(j);
        end
    end
end
```

Esempio vettorializzazione più complicata II

ATTENZIONE: nonostante questo programma produca il corretto output è da considerarsi sbagliato a causa della presenza di inutili cicli innestati estremamente inefficienti.
Una corretta soluzione in Matlab ottimizzata in un unico comando di assegnazione è invece la seguente:

```
A=(u(:)-v(:)').^(-2*(abs((1:length(u))'-(1:length(v)))<=3)+1);
```