Calcolo Numerico A/B
Esercitazione di Laboratorio 3

Gianna Del Corso <delcorso@di.unipi.it>

14 Novembre 2013

Quantità di esercizi: in questa dispensa ci sono *più esercizi* di quanti uno studente medio riesce a farne durante una lezione di laboratorio, specialmente tenendo conto anche degli esercizi facoltativi. Questo è perché sono pensate per "tenere impegnati" per tutta la lezione anche quegli studenti che già hanno un solido background.
Quindi fate gli esercizi che riuscite, partendo da quelli *non* segnati come facoltativi, e non preoccupatevi se non li finite tutti!

1 Vettori e matrici

Octave è pensato per lavorare con vettori e matrici; pertanto, ha una sintassi specifica e parecchi comandi dedicati, che rendono molto più semplice lavorare con i vettori rispetto a un linguaggio generico come il C.

1.1 Creare vettori e matrici

```
octave:1> A=[1 2 3; 4 5 6]
A =

   1 2 3
   4 5 6
```

```
 octave:1> zeros(3,2)
ans =

   0 0
   0 0
   0 0

octave:2> ones(3,2)
ans =
```

```
   1  1
   1  1
   1  1

octave:3> eye(3)
ans =

   1  0  0
   0  1  0
   0  0  1

octave:4> randn(2,3)
ans =

   0.567178 -0.126397 -0.090664
  -0.678601  0.504481  0.754911
```

1.2 Il *range operator* :

Con la sintassi `a:t:b` creiamo un vettore (riga) che contiene gli elementi `a`, `a+t`, `a+2t`... fino a `b` (o fino all'ultimo che sia minore o uguale a `b`). Se `t`=1, può essere omesso.

```
 octave:6> 1:0.5:4
ans =

   1.0000  1.5000  2.0000  2.5000  3.0000  3.5000  4.0000

octave:7> 1:10
ans =

   1  2  3  4  5  6  7  8  9  10

octave:8> 1:2:10
ans =

   1  3  5  7  9
```

Dove avete già usato l'operatore :?

1.3 Accedere agli elementi

```
octave:16> A=ones(2,3)
A =
```

```
   1 1 1
   1 1 1

octave:17> A(1,2)=2
A =

   1 2 1
   1 1 1

octave:18> A(1,2)
ans = 2
octave:19> A(5,10)
error: invalid row index = 5
error: invalid column index = 10
octave:19> A(5,10)=7
A =

   1 2 1 0 0 0 0 0 0 0
   1 1 1 0 0 0 0 0 0 0
   0 0 0 0 0 0 0 0 0 0
   0 0 0 0 0 0 0 0 0 0
   0 0 0 0 0 0 0 0 0 7
```

Se cerco di *leggere* un elemento che non esiste (perché la matrice è troppo piccola), ottengo un errore. Se cerco di *scrivere* un elemento che non esiste, la matrice viene automaticamente ingrandita.

1.4 Operazioni su vettori

```
octave:20> a=1:3
a =

  1 2 3

octave:21> b=4:6
b =

  4 5 6

octave:22> a+b
ans =

   5 7 9

octave:23> sin(a)
ans =
```

```
   0.84147 0.90930 0.14112

octave:24> 2*a+1
ans =

   3 5 7

octave:25> a.*b %operazioni elemento per elemento
ans =

    4 10 18

octave:26> c=a' %matrice trasposta
c =

   1
   2
   3

octave:27> C=a'*b %prodotto matrice-matrice
C =

    4 5 6
    8 10 12
   12 15 18

octave:28> length(a) %lunghezza di un vettore
ans = 3

octave:28> size(C) %dimensioni di una matrice - (righe, colonne)
ans =

   3 3
```

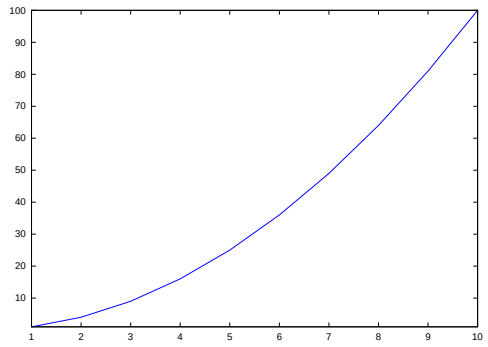
2 Grafici

Il comando `plot(x,y)` prende come argomenti due vettori della stessa lunghezza `x` e `y` e disegna sul piano cartesiano i punti `x(i),y(i)` collegandoli con una linea.

```
octave:28> r=1:10
r =

   1 2 3 4 5 6 7 8 9 10

octave:30> plot(r,r.^2)
```



Il seguente comando disegna un cerchio.

```
octave:23> t=0:.001:2*pi;
octave:24> plot(cos(t),sin(t))
```

Una funzione si può definire con il comando `@`. Ecco un esempio nel quale definisco la funzione $f(x) = x^2 + cos(x)$

```
octave:25>f=@(x)x.^2+cos(x)-1;
```

L'operatore . permette di valutare la funzione direttamente su un vettore. Quanto vale $f(\pi/2)$? Se $x = -6 : 0.5 : 6$ provate a fare $f(x)$. Per disegnare la funzione posso semplicemente fare

```
octave:26>x=-6:0.05:6;
octave:27>plot(x, f(x))
```

3 Metodo delle Tangenti

Esercizio 1. Scrivere una `function y=tangenti(f, d, x0, niter)` che prende una funzione f e la sua derivata d definite con il comando `f=@(x)....` e `d=@(x)....`, il punto iniziale x_0 e il numero di iterazioni niter e restituisce il valore y ottenuto applicando niter iterazioni del metodo delle tangenti applicato a f.

Esercizio 2 (facoltativo). Aggiungere la grafica, disegnando il grafico della funzione, e le varie iterazioni. (Suggerimento: ricordarsi che per disegnare la retta passante per i punti (x_0, y_0) e (x_1, y_1) posso usare il comando `plot([x_0, x_1], [y_0, y_1])`.)

Esercizio 3. Si consideri l'equazione $(x+3)(x-1)(x-4) = x^3+6x^2-11x+12 = 0$ e si applichi la funzione `tangenti` a partire dal punto $x_0 = 4.9$. Si osservi la convergenza alla radice 4, nella quale ad ogni iterazione l'errore tra la soluzione calcolata e la soluzione esatta raddoppia la precisione. Cosa succede all'equazione $(x+3)^2(x-1)(x-4) = 0$ quando si approssima la soluzione -3 a partire da -3.9? Sapete capire il motivo? Cosa osservate partendo dal punto $x_0 = 4.9$?

4 Cerchi di Gerschgorin

La seguente funzione disegna gli autovalori e i cerchi di Gerschgorin di una matrice quadrata `A`.

```
function gg(A)
  sz=size(A);
  n=sz(1); %cosa fanno queste due istruzioni?

  clearplot; %elimina i disegni preesistenti
  hold on; %fa si' che ogni disegno non cancelli il precedente
  axis('equal'); %forza la stessa scala su x e y

  autovalori=eig(A);
  plot(real(autovalori),imag(autovalori),'.*');
  % '.*': disegna solo i punti, non collegandoli con linee
  %"help plot" per altre stringhe magiche
  for k=1:n
    center=A(k,k);
    radius=0; %accumulatore
    for j=1:n
      if(j~=k)
        radius=radius+abs(A(k,j));
      endif
    endfor
    circle(center,radius);
  endfor
endfunction
```

Esercizio 4. Testare la funzione `gg` su alcune matrici test: `rand(10)`, `randn(10)`, `hilb(10)`,

```
octave:133> A=diag(1:10)+diag(ones(9,1),1)
A =

   1  1  0  0  0  0  0  0  0  0
   0  2  1  0  0  0  0  0  0  0
   0  0  3  1  0  0  0  0  0  0
   0  0  0  4  1  0  0  0  0  0
   0  0  0  0  5  1  0  0  0  0
   0  0  0  0  0  6  1  0  0  0
   0  0  0  0  0  0  7  1  0  0
   0  0  0  0  0  0  0  8  1  0
   0  0  0  0  0  0  0  0  9  1
   0  0  0  0  0  0  0  0  0 10
```

Esercizio 5. Scrivere una funzione `ggsecond(A)` che disegni i cerchi di Gerschgorin di A e mostri come variano gli autovalori quando gli elementi fuori dalla diagonale di `A` vengono ridotti pian piano fino a diventare zero, come nella dimostrazione del secondo teorema di Gerschgorin. Per farlo, generate tante matrici (diciamo $k = 100$) a intervalli uguali lungo il "segmento" che unisce A alla sua diagonale, e plottate i loro autovalori.

Esercizio 6. Nei grafici dell'esercizio precedente, noterete che per molte matrici (fate qualche esempio!) alcuni autovalori si "muovono" secondo questo schema: due autovalori reali si muovono nella stessa direzione e si avvicinano fino a coincidere, poi, una volta uniti, si staccano dall'asse reale e vanno in due direzioni opposte del piano complesso. Verificate questo fenomeno. Sapete spiegare perché questo accade?

Esercizio 7 (facoltativo). Scrivete una funzione `perturb(A,epsilon)` che, data una matrice A genera 50 matrici i cui elementi sono gli elementi di A modificati con un piccolo valore casuale di magnitudine circa `epsilon` (con le istruzioni `epsilon*randn(n) + 1I*epsilon*randn(n)`), e disegna i loro autovalori su un grafico. Di quanto si spostano gli autovalori per una perturbazione di grandezza `epsilon=1e-2`? Per una matrice casuale? Per l'identità?

Esercizio 8 (facoltativo). Poiché Octave è un linguaggio *interpretato*, eseguire ogni singola istruzione ha un "costo" non trascurabile (il computer deve leggere la riga, interpretarla e trasformarla in codice macchina). Per questo se si riesce a riscrivere le funzioni utilizzando delle operazioni sui vettori invece che dei cicli `for`, il programma gira molto più velocemente. Per esempio, è molto più veloce

```
s=sum(abs(v));
```

(una istruzione da interpretare) rispetto a

```
s=0;
for k=1:length(v)
 s=s+abs(v(k));
endfor
```

($O(n)$ istruzioni da interpretare, dove n è la lunghezza del vettore `v`).

L'operazione di sostituire tante istruzioni su scalari con una singola istruzione su un vettore si chiama *vectorization*. Riuscite a riscrivere i programmi della prima lezione (`fattoriale`, `pow`, `myexp`, `myexp2`) vettorizzando i cicli `for`, in modo che non siano più necessarie $O(n)$ istruzioni per calcolare un esponenziale? Potrebbe esservi utile il manuale di Octave alla sezione Sums and Products.