Metodi numerici con elementi di Programmazione

A.A. 2013-2014

Esercizi svolti in Laboratorio

Lezione del 26-11-2013

Docente: Vittoria Bruni

Email: vittoria.bruni@sbai.uniroma1.it

Ufficio: Via A. Scarpa,
Pal. B, I piano, Stanza n. 16
Tel. 06 49766648

Ricevimento: Giovedì 14.00-15.00

Il materiale didattico è disponibile sul sito http://ingaero.uniroma1.it/ nella pagina dedicata al corso Metodi Numerici con elementi di Programmazione

Per consultazione: Getting Started with MatLab – The mathworks www.mathworks.com

Esercizio 1

- Scrivere la funzione Matlab num_cond.m che legga in input una matrice quadrata A e la variabile numerica tipo_norma che può essere uguale a 1, 2 o inf. La funzione restituisca in output il numero di condizionamento rispetto alla norma indicata nella variabile tipo_norma. Se quest' ultima è diversa da 1, 2 o inf, la funzione deve calcolare il numero di condizionamento rispetto alla norma infinito.

Si usi la funzione num_cond.m per calcolare il numero di condizionamento delle seguenti matrici

$$A = \begin{pmatrix} 1 & 1.01 \\ 0.99 & 1 \end{pmatrix} \qquad A = \begin{pmatrix} 1 & 101 \\ 99 & 1 \end{pmatrix} \qquad A = \begin{pmatrix} 2 & 3 \\ -1 & 8 \\ 0 & 4 \end{pmatrix}$$

Soluzione

- Determinare la lista delle variabili di input e di output

Variabili di input:

- matrice A di cui calcolare in numero di condizionamento
- tipo di norma (tipo_norma) rispetto al quale calcolare il condizionamento

Variabili di output:

- numero di condizionamento K

Controllo sugli input:

- la matrice deve essere quadrata
- il tipo di norma deve essere un numero pari a 1, 2, o inf

- Aprire la finestra di Editor
- Salvare il file come num_cond.m nella directory di lavoro
- Scrivere la seguente istruzione che contiene la lista degli output e degli input della funzione

```
function [K] = num_cond(A,tipo_norma)
```

- Scrivere lo help della funzione

```
%  function [K] = num_cond(A,tipo_norma)
%  calcola il numero di condizionamento della matrice quadrata
%  A rispetto alla norma indicata in tipo_norma.
%  Se tipo_norma è diverso da  '1', '2' oppure 'inf', il
%  condizionamento è calcolato rispetto alla norma infinito
```

- Scrivere la lista di istruzioni per il controllo degli input

```
dim = size(A);
if dim(1)~=dim(2)
   error('La matrice deve essere quadrata')
end

if (tipo_norma ~= 1) & (tipo_norma ~= 2) & (tipo_norma ~= inf)
   tipo_norma = inf;
end
```

Nota: Modificare includendo anche un controllo sul determinante della matrice

- Scrivere la lista di istruzioni per il calcolo del numero di condizionamento rispetto al tipo di norma scelto

```
IA = inv(A);                    % matrice inversa di A
NA = norm(A,tipo_norma);        % norma di A
NIA = norm(IA,tipo_norma);      % norma della matrice inversa
K = NA*NIA;                     % numero di condizionamento
```

- Salvare il file

Dalla finestra del Command Window

```
>> A = [1 1.01;0.99 1];
>> KA_1 = num_cond(A,1);
>> disp(KA_1)
   4.0401e+004
>> whos
      Name     Size          Bytes  Class     Attributes

      A        2x2              32  double
      KA_1     1x1               8  double

>> KA_2 = num_cond(A,2);
>> disp(KA_2)
   4.0002e+004

>> KA_inf = num_cond(A,inf);
>> disp(KA_inf)
   4.0401e+004
```

Dalla finestra del Command Window

```
>> KA = num_cond(A,-1);
>> disp(KA)
    4.0401e+004

>> A = [2  3; -1  8; 0  4];
>>KA_inf =num_cond(A,inf);
??? Error using ==> num_cond at 10
La matrice deve essere quadrata
```

Esercizio 2

Si consideri l'equazione di Keplero sulle orbite dei pianeti **M = E-e sin(E)**
in cui **E** rappresenta l'anomalia eccentrica, **M** l'anomalia media ed **e** l'eccentricità dell'orbita.

Scrivere uno script Matlab (sol_keplero.m) per il calcolo dell'approssimazione numerica con 8 decimali esatti dell'anomalia eccentrica quando **M = 24.851090** ed **e=0.1** usando sia il metodo delle tangenti che quello delle secanti con estremi variabili.
Nel primo caso si usi come approssimazione iniziale **x0 = M**; nel secondo, **x0=M** e **x1** pari all'approssimazione ottenuta alla prima iterazione del metodo delle tangenti.

Lo script deve produrre anche l'approssimazione numerica dell'anomalia eccentrica, con la stessa precisione e punto iniziale richiesti al metodo delle tangenti, con i due procedimenti di punto fisso generati dalle seguenti funzioni di iterazione

g1(E) = M+e sin(E), g2(E) = arcsin((E-M)/e)

Lo script deve:

- Usare le funzioni newton_fun, secanti_fun, puntounito_fun così definite:

```
[x,n_iter,ERR] = newton_fun(f,df,x0,eps,max_iter)
                                % f= funzione di cui trovare lo zero
                                % df = derivata prima di f
                                % x0 = approssimazione iniziale
                                % eps = precisione richiesta alla approssimazione
                                % max_iter = no. massimo di iterazioni consentite

[x,n_iter,ERR] = secanti_fun(f,a,b,eps)
                                % f = funzione di cui trovare lo zero
                                % a = estremo inferiore dell'intervallo in cui è stato
                                %       isolato lo zero di f
                                % b = estremo superiore dell'intervallo in cui è stato
                                %       isolato lo zero di f
                                % eps = precisione richiesta alla approssimazione

[x,n_iter,ERR] = punto_unito_fun(f,x0,eps)
                                % f = funzione di iterazione
                                % x0 = approssimazione iniziale
                                % eps = precisione richiesta alla approssimazione
```

Per tutte e tre le funzioni

```
% x = approssimazione prodotta
% n_iter = numero di iterazioni eseguite
% ERR = vettore contenente l' errore commesso ad ogni iterazione, calcolato
    come differenza tra due approssimazioni successive
```

- stampare una tabella di quattro righe, ognuna riferita ad un metodo diverso, che riporti il numero di iterazioni eseguite, l'approssimazione prodotta e l'errore corrispondente
- stampare un messaggio relativo al metodo che ha raggiunto la soluzione più velocemente
- disegnare su uno stesso grafico il valore assoluto degli errori commessi ad ogni iterazione dai quattro procedimenti iterativi. Il grafico deve essere etichettato opportunamente e deve includere una legenda.

Soluzione

```
M = 24.851090 ;
e = 0.1;
f = @(E)[M – E+e*sin(E)];
df = @(E)[-1+e*cos(E)];
x0 = M;
eps = 0.5*10^(-8);
```

Esercizio: Modificare usando il comando input

```
[x_tan,n_iter_tan,ERR_tan] = newton_fun(f,df,x0,eps,10000);

[x_N,n_iter_N,ERR_N] = newton_fun(f,df,x0,eps,1);
[x_sec,n_iter_sec,ERR_sec] = secanti_fun(f,min(x0,x_N),max(x0,x_N),eps)

g1 = @(E)[M+e*sin(E)];
g2 = @(E)[asin((E-M)/e)];
[x_P1,n_iter_P1,ERR_P1] = punto_unito_fun(g1x0,eps)
[x_P2,n_iter_P2,ERR_P2] = punto_unito_fun(g2,x0,eps)

fprintf('%10d \t %15.10f \t %15.10f \n',[n_iter_tan x_tan ERR_tan(end);
n_iter_sec x_sec ERR_sec(end); n_iter_P1,x_P1, ERR_P1; n_iter_P2,x_P2, ERR_P2]')
```

Soluzione

Modificare usando opportunamente le istruzioni condizionali

```
[MI,pos_MI] = min([n_iter_tan, n_iter_sec, n_iter_P1,n_iter_P2]);

Metodi = ['tan';'sec';'g1 ';'g2 ']
fprintf('il metodo che converge piu'' velocemente e''  %15s ', Metodi(pos_MI,:)')


figure, plot(ERR_tan),
hold on, plot(ERR_sec,'r'), plot(ERR_P1,'g'), plot(ERR_P2,'m')
legend('tangenti','secanti','g1(E) = M+e sin(E)',' g2(E) = arcsin((E-M)/e)')
title('Confronto convergenza')
```

Prima di mandare in esecuzione lo script è necessario scrivere le funzioni

newton_fun.m
secanti_fun.m
punto_unito_fun.m

!!!!!

Esercizio 3

«per casa»

- Scrivere la funzione Matlab esercizio_3.m che:
  - riceva in input un intero positivo n strettamente maggiore di 1
  - costruisca e restituisca in output la matrice di Hilbert di ordine n
  - ne calcoli il numero di condizionamento rispetto alla norme 1, 2 e infinito richiamando opportunamente la funzione num_cond.m
  - stampi un messaggio indicando il tipo e il valore delle norme che hanno prodotto il valore più alto e più basso
  - restituisca in output il tipo e la norma che ha prodotto il valore più basso.

Ricordando che

$$H = \begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{4} & \frac{1}{5} & \cdots & \frac{1}{n+1} \\ \frac{1}{3} & \frac{1}{5} & \frac{1}{6} & \cdots & \frac{1}{n+2} \\ \cdots & & & & \\ \frac{1}{n} & \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{2n-1} \end{pmatrix} \quad H = (h_{ij}) \;\; h_{ij} = \frac{1}{i+j-1} \;\; i,j = 1,\ldots,n$$

si considerino i seguenti valori per n

n = 2, n=3, n=5, n=-1, n = 10, n=30, n=100

Esercizio 4

«per casa»

Dopo aver letto lo help del comando Matlab cputime, scrivere lo script Matlab tempo_prodotto.m che crea una matrice quadrata di dimensione n e un vettore di lunghezza n e ne calcoli il prodotto matriciale. Gli elementi della matrice e del vettore sono numeri casuali; la dimensione n deve essere fatta variare da n=150 a n=9000 usando un passo di 150. Misurare e graficare opportunamente il tempo di esecuzione del prodotto A*v.

Si consiglia di svolgere l'esercizio prima di passare alle slide successive!!!!!

Durata di un programma

Per confrontare due programmi che risolvono lo stesso problema è utile misurare il tempo di CPU impiegato per eseguirli.
La funzione cputime misura il tempo di CPU usato da quando è stata aperta la sessione di Matlab

Per misurare il tempo richiesto dall' esecuzione di un programma, quindi, basta valutare la differenza tra il cputime valutato prima e dopo l'esecuzione del programma

In alternativa si possono usare in modo combinato i comandi tic e toc.
tic si scrive all' inizio del programma, o immediatamente prima della prima istruzione da cui si vuole iniziare a valutare il tempo di esecuzione

toc si scrive alla fine del programma, o immediatamente dopo l'ultima istruzione della parte di programma di cui si vuole valutare il tempo di esecuzione. toc contiene il tempo di esecuzione che viene stampato sulla finestra dei comandi

Soluzione Esercizio 4

```
% Lo script tempo_prodotto.m valuta il tempo di calcolo richiesto dalla
% valutazione del prodotto matrice per vettore al variare della dimensione
% della matrice
n= 9000;
p = 150;
A = rand(n);
V = rand(n,1);
T = [];
for k=150:p:n
      B = A(1:k,1:k);
      w = V(1:k);
      ti = cputime;
      P = B*w;
      tf = cputime-ti;
      T = [T tf];
end
figure,
plot(150:p:n,T), xlabel('dimensione della matrice'), ylabel('tempo')
```

In alternativa

```
% Lo script tempo_prodotto.m valuta il tempo di calcolo richiesto dalla
% valutazione del prodotto matrice per vettore al variare della dimensione
% della matrice
n= 9000;
p = 150;
A = rand(n);
V = rand(n,1);
T = [];
for k=150:p:n
      B = A(1:k,1:k);
      w = V(1:k);
      tic;
      P = B*w;
      toc;
      T = [T toc];
end
figure,
plot(150:p:n,T), xlabel('dimensione della matrice'), ylabel('tempo')
```

Esercizio 5

«per casa»

- Scrivere uno script Matlab che confronti il tempo richiesto per risolvere numericamente un sistema lineare usando il solutore di Matlab e il metodo di eliminazione di Gauss con pivoting scritto senza usare il solutore di Matlab