Metodi numerici con elementi di Programmazione

A.A. 2013-2014

Esercizi svolti in Laboratorio

Lezione del 10-12-2013

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

- Modificare la funzione Matlab num_cond2.m e chiamarla num_cond3.m usando opportunamente l'istruzione switch/case/otherwise/end

A parte il nome, in questa prima parte non cambia nulla rispetto alla funzione num_cond2

```
function [K,varargout] = num_cond3(A,varargin)
%  function [K,varargout] = num_cond3(A,varargin)
%  calcola il numero di condizionamento K della matrice quadrata
%  A rispetto alla norma indicata in tipo_norma.
%  Se tipo_norma non è data in input, il  condizionamento è calcolato
%  rispetto alla norma infinito. tipo_norma non può assumere valori
%  diversi da  '1', '2' o 'inf'
%  Se si chiedono due output, allora il primo è K e il secondo è la
%  matrice inversa di A
%  Se si chiedono tre output, il primo è K, il secondo è la matrice
%  inversa di A, il terzo è la norma di A
```

- Controllo del numero delle variabili di input

L'istruzione condizionale annidata
if elseif elseif else end
usata in num_cond2 è sostituita con l'istruzione switch sulla variabile nargin

```
switch nargin
    case  0     % non sono assegnati input
          error('E'' necessario definire una matrice!!!')
    case  1     % è assegnato un solo input, cioè la matrice
          tipo_norma = inf;
    case  2     % sono assegnati due input, cioè la matrice e tipo_norma
          tipo_norma = varargin{1};
          if  (tipo_norma ~= 1) & (tipo_norma ~= 2) & (tipo_norma ~= inf)
              error('la norma deve essere 1 2 o inf')
          end
    otherwise
          error('troppe variabili di input!!!')
    end
```

non cambia nulla rispetto alla funzione num_cond2

- Lista di istruzioni per il controllo degli input

```
dim = size(A);
if dim(1)~=dim(2)
   error('La matrice deve essere quadrata')
end
if det(A)==0
   error('La matrice e'' singolare')
end
```

- 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
```

- L'istruzione condizionale annidata if elseif else end usata in num_cond2 è sostituita con l'istruzione switch sulla variabile nargout
- otherwise corrisponde sia ad un numero di output troppo grande rispetto a quello previsto, sia ad un numero troppo piccolo!!!

- Controllo e assegnazioni output

```
switch nargout
    case 2 % sono stati richiesti  due output
        varargout{1} = IA;
    case 3  %sono stati richiesti  tre output
        varargout{1} = IA;
        varargout{2} = NA;
    otherwise   % il numero delle variabili di output richieste è troppo
                % alto o troppo basso
        error('controllare il numero delle variabili di output!!!')
end
```

Dalla finestra del Command Window

non cambia nulla rispetto alla funzione num_cond2

```
>> A = [1 1.01;0.99 1];
>> KA_1 = num_cond3(A,1);
>> disp(KA_1)
    4.0401e+004

>> [KA,B] = num_cond3(A);
>> disp(KA)
    4.0401e+004
>> disp(B)
  1.0e+004 *
   1.0000   -1.0100
  -0.9900    1.0000


>> [KA,B,nA] = num_cond3(A);
>> disp(nA)
   2.0100
```

Esercizio 2

Scrivere la funzione Matlab successione.m che riceva in input un numero reale a e un intero positivo M.
Se |a| > 1 restituisca in output il vettore P contenente i primi M elementi della successione così definita:

$$\begin{cases} y_1 = 2a \\ y_2 = 4a^2 - 1 \\ y_k = 2a y_{k\_1} - y_{k\_2} \end{cases} \qquad k \in \mathbf{N}, k > 2$$

altrimenti interrompa l'esecuzione e richieda di inserire da tastiera un altro valore del parametro a stampando il messaggio 'il modulo di a deve essere maggiore di 1!'. Se dopo tre tentativi il valore di a dato in input non è accettabile, la funzione stampi il messaggio 'gli input inseriti non sono corretti!' e interrompa l'esecuzione. La funzione deve graficare il vettore P usando una linea solida nera e il vettore Q contenente gli elementi

$Z_k = \frac{1}{2\sqrt{a^2-1}}\left(|a+\sqrt{a^2-1}|^{k+1} - |a-\sqrt{a^2-1}|^{k+1}\right)$, k=1,…,M

usando una linea tratteggiata rossa.
Il grafico deve essere completo di titolo, etichette per gli assi e legenda.

```
function [P] = successione(a,M)
%[P] = successione(a,M) calcola i primi M elementi della successione così definita
%    P(1) = 2*a
%    P(2) = 4*a^2-1
%    P(k) = 2*a*P(k-1)-P(k-2),     k>2
%
% e la confronta con la successione così definita
%    Q(k) = 1/(2*sqrt(a^2-1) )* (abs(a+sqrt(a^2-1))^(k+1) - abs(a-sqrt(a^2-1))^(k+1));
%
%
% INPUT
%     a = numero reale tale che |a|>1. La funzione offre tre possibilità per
%         correggere il valore della variabile data in input, altrimenti
%         interrompe l'esecuzione
%     M = numero intero positivo

% OUTPUT
%     P = vettore contenente gli elementi della successione Pn
```

```
% controllo del numero delle variabili di input e di output
if (nargin<=1) | (nargin>2)
   error('il numero degli input non e'' corretto!!!')
end

if (nargout>1)
   error('il numero degli output non e'' corretto!!!')
end

% controllo del primo  input
tentativi = 0;
while abs(a)<=1 & tentativi <3
        a= input('il modulo di a deve essere maggiore di 1! a= ');
        tentativi = tentativi + 1;
end
```

Nota: All'uscita del ciclo while si possono avere le due situazioni seguenti:

- Il valore di a è in modulo maggiore di 1 e tentativi<=3
- Il numero di tentativi ha raggiunto il valore 3 e il valore di a è in modulo minore o uguale a 1

```
if  abs(a)>1
   % controllo del secondo input
   if (round(M)~=M) | (M<=0)
      error('il secondo input deve essere un numero intero positivo!!!')
   end
   % calcolo del vettore P
   P(1) = 2*a;
   P(2) = 4*a^2-1;
   for k=3:M
        P(k)=2*a*P(k-1)-P(k-2);
   end
   % calcolo del vettore Q
   Q(1:M) = 1/(2*sqrt(a^2-1)) * (abs(a+sqrt(a^2-1)).^(2:M+1) - abs(a-sqrt(a^2-1)).^(2:M+1));

% grafico dei risultati
   figure,  plot(P,'k-')
   hold on, plot(Q,'r--')
   xlabel('k'), ylabel('successioni'), title('confronto successioni')
   legend('vettore P','vettore Q')
else
  error('gli input inseriti non sono corretti!')
end
```

Dal Command Window

```
>> [P] = successione(2,-10);
??? Error using ==> successione at 38
il secondo input deve essere un numero intero positivo!!!

>> [P] = successione(2,1.5);
??? Error using ==> successione at 38
il secondo input deve essere un numero intero positivo!!!

>> [P] = successione(2,-1.6);
??? Error using ==> successione at 38
il secondo input deve essere un numero intero positivo!!!

>> [P] = successione(6);
??? Error using ==> successione at 22
il numero degli input non e' corretto!!!
```

Dal Command Window

```
>> [P] = successione(-.5,100);
il modulo di a deve essere maggiore di 1! a= .5
il modulo di a deve essere maggiore di 1! a= .2
il modulo di a deve essere maggiore di 1! a= .4
??? Error using ==> successione at 53
gli input inseriti non sono corretti!
```

```
>> [P] = successione(2,10);
>> [P] = successione(2,100);
>> [P] = successione(6,100);
>> [P] = successione(-6,100);
```

A queste chiamate la funzione non da messaggi di errore e grafica i risultati

Esercizio 3

Scrivere la function Matlab lebesgue.m che:
1.riceva in input i vettori xnodi e xpunti; se xpunti non viene dato in input, assegni a xpunti 1000 punti equispaziati nell' intervallo [xnodi(1); xnodi(length(xnodi)];
2.costruisca il vettore fleb contenente i valori della funzione di Lebesgue calcolata in xpunti;
3.se xpunti contiene più di un elemento, grafichi la funzione di Lebesgue calcolata in xpunti etichettando opportunamente gli assi;
4.restituisca in output xpunti, il massimo valore di fleb e il punto p_Mfleb in cui è realizzato.

Utilizzare la function per il seguente vettore di nodi [1.0; 1.5; 2.0; 2.5; 3.0] e salvare gli output nel file risultati_es3.mat

```
function [xpunti,Mfleb,p_Mfleb] = lebesgue(xnodi,varargin)
% [xpunti,Mfleb,p_Mfleb] = lebesgue(xnodi,varargin) valuta e grafica la funzione  di
% Lebesgue costruita sui nodi contenuti in xnodi in corrispondenza dei punti in
% xpunti, ne calcola il massimo e il punto in cui quest'ultimo è realizzato.
% Se xpunti non  è dato in input, l'intervallo [xnodi(1), xnodi(end)] è suddiviso in
% 1000 punti equispaziati
%
% INPUT
%    - xnodi =  vettore dei nodi
%
% OUTPUT
%   -  xpunti = vettore dei punti in cui è stata valutata la funzione di Lebesgue.
%                 Se non è dato in input, è costituito da 1000 punti equispaziati in
%                [xnodi(1), xnodi(end)]
%   -  Mfleb  = valore massimo assunto dalla funzione di Lebesgue sul vettore di
%                 punti in xpunti
%   -  p_Mfleb  =  punto di xpunti in cui la funzione di Lebesgue assume il valore
%                      massimo
```

```
% controllo degli input
if nargin<1 | nargin>2
        error('il numero delle variabili di input non e'' corretto!');
elseif nargin==1
        if length(xnodi)>1
          xpunti = linspace(xnodi(1),xnodi(end),1000);
        else
          error('xnodi deve essere un vettore di almeno due elementi!');
        end
else
        if length(xnodi)>1
          xpunti = varargin{1};
        else
          error('xnodi deve essere un vettore di almeno due elementi!');
        end
end
```

NOTA: Modificare inserendo il seguente controllo su xpunti:

gli elementi di xpunti devono essere contenuti nell'intervallo [xnodi(1), xnodi(end)]

```
% Calcolo i polinomi di base di Lagrange e li valuto in ogni elemento di xpunti
n_xnodi = length(xnodi);
n_xpunti = length(xpunti);

for i=1:n_xpunti    % per ogni elemento di xpunti
   for n=1:n_xnodi % per ogni nodo in xnodi
      plag(i,n) = prod(xpunti(i)-xnodi(1:n-1))*prod(xpunti(i)-xnodi(n+1:n_xnodi))/...
                  (prod(xnodi(n)-xnodi(1:n-1))*prod(xnodi(n)-xnodi(n+1:n_xnodi)));
   end
end
```

NOTA: La variabile plag è una matrice con n_xpunti righe e n_xnodi colonne
Ogni elemento in posizione (i,n) contiene il valore che il polinomio della base di Lagrange relativo al nodo n-simo assume in corrispondenza dello i-simo elemento di xpunti

```
% calcolo il valore della funzione di Lebesgue in ogni elementi di xpunti
fleb = sum(abs(plag'));

% calcolo il massimo di fleb
[Mfleb,p_Mfleb] = max(fleb);

% grafico della funzione di Lebesgue
if n_xpunti>1
   figure, plot(xpunti,fleb)
   xlabel('xpunti')
   ylabel('funzione di Lebesgue')
end
```

Dal Command Window

```
>> xnodi=[1.0, 1.5, 2.0, 2.5, 3.0];
>> [xpunti,Mfleb,p_Mfleb] = lebesgue(xnodi);
>> save risultati_es3.mat   xpunti   Mfleb   p_Mfleb

>> [xpunti,Mfleb,p_Mfleb] = lebesgue(xnodi,linspace(xnodi(1),xnodi(end),1000));

>> [xpunti,Mfleb,p_Mfleb] = lebesgue(xnodi,2.2);
```

Sono equivalenti

Non produce grafici

```
>> [xpunti,Mfleb,p_Mfleb] = lebesgue(xnodi(1));
??? Error using ==> lebesgue at 24
xnodi deve essere un vettore di almeno due elementi!
```

Esercizio 4

Si consideri la successione

$$x_{n+1} = \begin{cases} 1 & \text{se} \quad x_n = 1 \quad (1) \\ \frac{x_n}{2} & \text{se} \quad x_n \text{ è } pari \quad (2) \\ 3x_n + 1 & \text{se} \quad x_n \text{ è } dispari \quad (3) \end{cases}$$

Scrivere la funzione Matlab successione2.m che riceva in input due interi positivi x0 e M e restituisca in output il vettore S contenente i primi M termini della successione x_n generata da x0.
Se la successione costruita raggiunge il valore 1 in un numero di passi N < M, la funzione deve stampare il valore di N dopo il seguente messaggio "Si raggiunge 1 in un numero di passi pari a ".
La funzione deve graficare gli elementi di S usando un pallino blu per gli elementi ottenuti secondo la regola (2), un asterisco nero per quelli ottenuti con la regola (3) e un quadrato rosso per l'eventuale elemento pari a 1.
Il grafico deve essere completo di titolo, etichette per gli assi e legenda.

Esercizio 5

Scrivere la function Matlab fun_esercizio5.m che:
1.riceva in input i vettori xnodi e ynodi, un numero reale z e un intero N; se N non viene dato in input, assegni ad N la lunghezza del vettore xnodi diminuita di 1;
2.detta f la funzione che assume i valori in ynodi in corrispondenza dei valori in xnodi, e detto p_N il polinomio di grado N che interpola f in un sottoinsieme opportuno di xnodi, valuti, se possibile, e restituisca in output l'errore che si commette approssimando f(z) con $p_N(z)$, altrimenti interrompa l'esecuzione e stampi un messaggio di errore.

Data la seguente tabella di dati

i	0	1	2	3	4
xi	-0.5	0.25	0.5	1.5	2.5
f(xi)	-0.750	-0.375	-0. 250	0.250	0.750

utilizzare la function per stimare l'errore commesso approssimando f(0) con $p_4(0)$ e salvare sia gli input che gli output nel file risultati_esercizio5.mat