Metodi numerici con elementi di Programmazione

A.A. 2013-2014

Esercizi

Lezione del 16-12-2013

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

Si consideri la funzione $$F(x) = \sum_{n=1}^{+\infty} f_n x^n,$$

definita per $x \in \mathbf{R}$ e $f_n = \begin{cases} f_{n-2} + f_{n-1} & n \in \mathbf{N},\ n > 1 \\ 1 & n = 0 \ or \ n = 1 \end{cases}$

Scrivere la funzione Matlab esercizio1.m che riceva in input un numero reale y e un numero naturale $N \geq 10$ e restituisca in output:

- la variabile S contenente il valore di F(y) ottenuto usando i primi N termini della serie;
- il vettore C dei coefficienti f_n usati. Se N non viene dato in input, porre N=1000.

La funzione deve graficare le componenti del vettore C usando una linea continua rossa per i primi 10 elementi e un marcatore di punto a scelta di colore blu per i successivi elementi. Il grafico deve essere completo di titolo, etichette per gli assi e legenda.

```
function [S,C] = esercizio1(y,varargin)
%  function [S,C] = esercizio1(y,varargin)
% ..........................................................................
% ..........................................................................
if nargin==1
   N=10000;
elseif nargin==2
   N = varargin{1};
   if N<10 | (round(N) ~=N)
      error('il valore di N non e'' ammissibile');
   end
else
   error('controllare il numero delle variabili date in input!!!')
end
C(1:2) = 1;, S = 0;
for k=1:N
   S = S + C(k+1)*y^k;
   C(k+2) = C(k+1)+C(k);
end
C([1 length(C)]) = [];
figure, plot(1:10,C(1:10),'r'), hold on, plot(11:length(C), C(11:length(C)),'b*')
```

Esercizio 2

Scrivere la funzione Matlab esercizio2.m che:

- legga in input una matrice *A*;
- effettui, se possibile, un opportuno scambio di righe per cui il numero di elementi non nulli su ogni riga di *A* cresca con l'indice di riga;
- restituisca in output la matrice *A*, eventualmente modificata in accordo al punto precedente;
- determini il numero di elementi nulli di *A* e restituisca in output il coefficiente *C* dato dal rapporto tra il numero di elementi nulli di *A* e la dimensione di *A* definito;
- Se *C* =0 stampi il messaggio 'La matrice non è sparsa'

Richiamare la funzione scritta sulle seguenti matrici e salvare tutti gli input e gli output nel file risultaties2.mat.

```
function [A,C] = esercizio2(A)
%  function [A,C] = esercizio2(A)
% ..........................................................................
% ..........................................................................
% ..........................................................................
% ..........................................................................
[m,n] = size(A);
C = length(find(A(:)==0))/(m*n);

if C==0
   disp('la matrice non e'' sparsa');
else
   for i=1:m
        nnz(i) =  length(find(A(i,:)));
   end
   [S,P] = sort(nnz);
   A=A(P,:);
end
```

Esercizio 3

Scrivere la funzione Matlab esercizio3.m che:

- legga in input le funzioni f, df, df2, con df e df2 rispettivamente derivata prima e seconda di f, due numeri reali a e b (con a < b) e due interi positivi M e nmax, con nmax non superiore a 1000. Se M non viene dato in input, assegni ad M il valore 5; se nmax non viene dato in input, assegni ad nmax il valore massimo consentito;
- determini un intervallo di separazione $I = [a1, b1] \subset [a, b]$ della radice di f con ampiezza non superiore a 0.25;
- approssimi la radice di f in I con il metodo delle tangenti, scegliendo come approssimazione iniziale x0 l'estremo di Fourier e arrestando il procedimento quando la approssimazione prodotta xn ha M decimali esatti o non appena il numero di iterazioni eseguite diventa pari a nmax. Se l'estremo di Fourier non esiste, si ponga x0 uguale all'estremo positivo più piccolo di I;
- restituisca in output xn, la variabile fxn contenente il valore di f in xn e l'approssimazione iniziale x0.

Utilizzare la funzione scritta per approssimare la radice della equazione non lineare $x^3-\cos(x)+ 1/3 = 0$ nell'intervallo [0.25, 1] e salvare sia gli input che gli output nel file risultaties3.mat.

```
function [xn,fxn,x0] = esercizio3(f,df,df2,a,b,varargin)
% function [xn,fxn,x0] = esercizio3(f,df,df2,varargin)
% ....................................................................
% ....................................................................
% ....................................................................
% ....................................................................
switch nargin
   case 5
      M = 5;
      nmax = 1000;
   case 6
      M = varargin{1};
      nmax = 1000;
   case 7
      M = varargin{1};
      nmax = varargin{2};
   otherwise
      error('controllare il numero delle variabili di input!!!')
end
```

```
 if (M~=round(abs(M)))
    error('M deve essere intero positivo!!!')
end
if (nmax~=round(abs(nmax))) | nmax>1000
    error('nmax deve essere intero positivo e non superiore a 1000!!!')
end
if a>=b
    error('a deve essere strettamente minore di b!!!')
end

x = a:.2:b;
y=f(x);
pos = find(abs(diff(sign(y))));

if length(pos)~=1
    error('la radice non e'' unica')
else
    a1 = x(pos);
    b1 = x(pos+1);
end
```

```
if f(a1)*df2(a1)>0
   x0 = a1;
elseif  f(b1)*df2(b1)>0
   x0 = b1;
else
   if sum(sign([a1 b1]))<=0
     error('non e'' possibile fissare la condizione iniziale')
   elseif a1>0
      x0 = a1;
   else
      x0 =  b1;
   end
end
eps = 0.5*10^(-M);
err = eps+1;,  iter = 0;
while (err>eps) & (iter<nmax)
     xn = x0-f(x0)/df(x0);
     iter = iter + 1;
     err = abs(xn-x0);
     x0=xn;
end
fxn = f(xn);
```

```
function [c] = esercizio(y,z,m)
% .....
% .....
% .....
C = zeros(m);, v = zeros(m,1);, s = zeros(2*m-1,1);
for i = 1:length(y)
    temp = z(i);
    for j = 1:m
        v(j) = v(j) + temp;, temp = temp*y(i);
    end
    temp = 1;
    for j = 1:2*m-1
        s(j) = s(j) + temp;, temp = temp*y(i);
    end
end
for i = 1:m
    for j = 1:m
        C(i,j) = s(i+j-1);
    end
end
c = C\v;
```

Esercizio 4

Scrivere lo help della funzione descrivendo anche tutte le variabili di input e di output. Commentare le istruzioni.

Esercizio 5

Scrivere lo help della seguente funzione; commentare e completare le istruzioni.

```
function [V,M] = fun_esame(t,g)
%.....
%.....
%.....
%.....
%.....
J = length(g);
D = (t(end)-.....);
if .....
   V = (g(1)+2*sum(g(2:2:J-1))+2*sum(g(2:J-1))+g(end))*D/3;
   M = 0;
else
   V = D*sum(g) - 0.5*D*(g(1)+ .....);
   M = 1;
end
```

Esercizio 6
Completare e commentare le istruzioni della funzione e scriverne lo help

```
function [In,Nn,E] = my_fun(x1,x2,fun,varargin)
%.....................................................
%.....................................................
%.....................................................
if nargin<3
   error('input non sufficienti')
elseif ..............................
   tl=0.5*10^-5;
elseif nargin == 4
   tl = ..............................
else
   error('troppe variabili di input')
end
D = (x2-x1)/2;, I0 =D/3*(fun(x1)+4*fun(x1+D)+fun(x2));, E = tl+1;
while E >= tl
      D = D/2;
      Nn = ..............................
      xn = linspace(x1,x2,Nn);, fn = fun(xn);
      In = D/3*(fn(1)+4*sum(fn(2:2:Nn-1))+2*sum(fn(3:2:Nn-2))+fn(Nn));
      E = abs(In-I0)/15;
      I0 = ..............................
end
```

Esercizio 7

Scrivere la funzione Matlab esercizio7.m che

- riceva in input tre numeri a, b, eps e la funzione f; se eps non viene data in input, assegni il valore $0.5 \cdot 10^{-5}$

- approssimi l'integrale $I(f) = \int_a^b f(x)dx$ con la formula di Cavalieri-Simpson;
- approssimi I(f) con la formula delle parabole generalizzata dimezzando ogni volta il passo h e arrestandosi quando il modulo del resto Rh(f) diventa minore di eps;
- restituisca in output l' approssimazione finale dell' integrale, il numero totale di nodi utilizzati per calcolarla e la stima di Rh(f) ottenuta tramite il criterio di Runge.

Utilizzare la funzione realizzata per approssimare

$$I(f) = \frac{1}{\pi} \int_0^{\pi} cos(sin(x))dx$$

con 6 decimali esatti e salvare gli output nel file risultaties7.mat.