Consegna Esame di Scientific Computing

Lorenzo Tecchia
N86004446
2023/03/27

Indice

Elenco delle figure

Listings

Capitolo 1

Primo Esercizio

1.1 Traccia

Trovare un approssimazione di $\sqrt[3]{25}$, usando il metodo di bisezione con un'accuratezza di 10^{-4}.

Determinare a priori il numero di suddivisioni necessarie per ottenere una tale approssimazione e confrontare il valore teorico con il valore sperimentale.

Dimostrare il teorema della convergenza.

1.2 Esecuzione

1.2.1 Input

Listing 1.1: Primo Esercizio

```
% funzione per la quale 5^(2/3) e' zero della funzione
f = @(x)x.^3 - 25;

% Valore di tolleranza
toll = 10^-4;

% valore teorico del numero di iterazioni necessarie
it_teorico = log2((5-0)/toll) - 1;

% approssimazione per eccesso
ceil(it_teorico)

% scelta dell'intervallo per l'algoritmo [0-5]
[x,it_vero] = myBisezione(f,0,5,toll);
```

```

% array conteneti i valori durante i passi di bisezione
disp(x)

% valore vero del numero di iterazioni necessarie per soddisfare
% la tolleranza
disp(it_vero)
```

1.2.2 Output

```
ans =

    15

  Columns 1 through 10

         0    5.0000    2.5000    3.7500    3.1250    2.8125    2.9688    2.8906↙
2.9297    2.9102

  Columns 11 through 18

    2.9199    2.9248    2.9224    2.9236    2.9242    2.9239    2.9240    2.9240

    16
```

Figura 1.1: OutPut Primo Esercizio

1.3 Commento

TEOREMA DELLA CONVERGENZA per $\alpha = \sqrt[3]{25}$

$$\lim_{\mu\to\infty} x_\mu = \alpha \quad \leftrightarrow \lim_{\mu\to\infty} |x_\mu - \alpha| = 0$$

$$|x_\mu - \alpha| < \frac{b_\mu - e_\mu}{2}$$
$$|x_\mu - \alpha| < \frac{b_0 - e_0}{2^{\mu+1}}$$

$$0 \leq |x_\mu - \alpha| \leq \frac{b_0 - e_0}{2^{\mu+1}}$$

$$\lim_{\mu\to\infty} x_\mu - \alpha = 0 \rightarrow \text{per il teorema dei carabinieri}$$

II teorema della convergenza è verificato poiché il numero di iterazioni che soddisfa la tolleranza è minore del numero di iterazioni teorico

Capitolo 2

Secondo Esercizio

2.1 Traccia

Costruire la spline cubica naturale per i dati seguenti:

$$x = (-1, -0.5, 0, 0.5), f = (0.86199480, 0.95802009, 1.0986123, 1.2943767)$$

Sapendo che i dati che sono ottenuti dalla tabulazione di $f(x) = \ln(e^x + 2)$, approssimare $f(0.25)$ e $f'(0.25)$, e calcolare l'errore.

Descrivere i passi principali dell'algoritmo.

2.2 Esecuzione

2.2.1 Input

Listing 2.1: Secondo Esercizio

```
% array di input
x = [-1 -0.5 0 0.5];
f = [0.86199480 0.95802009 1.0986123 1.2943767];

% vattore 'z' appoggio per la costruzione della spline [-1, 0.5]
z = linspace(-1,0.5);

% costituisco i valori per l'asse delle y
j = mySpline(x,f,z);

% funzione da traccia
f_1 = @(x)log(exp(x) + 2);

% derivata della funzione da traccia
```

```
f_2 = @(x)1 - 2./(2 + exp(x));

f_3 = f_2(x);

j_1 = mySpline(x,f_3,z);

% faccio il plotting dei punti assieme alla spline
plot(x, f, 'o', z, j);


% apporssimazione di f'(0.25) con errore 10^-11
for i =1:length(j_1)
    if f_2(0.25)- j_1(i) < 10^-11
        disp(i)
    end
end

disp(' ')

% approssimazione di f(0.25) con errore 10^-11
for i =1:length(j)
    if f_1(0.25)- j(i) < 10^-11
        disp(i)
    end
end
```

2.2.2 Output

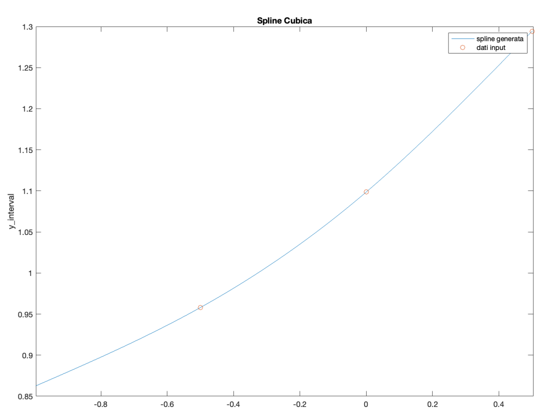

Figura 2.1: Output SplineCubica

2.3 Commento

Per l'approssimazione di $f(0.25)$ e $f'(0.25)$ si è fatto un ciclo sull'output della spline, per verificare scelto l'errore quali indici dell'array costituente la spline verificavano la tolleranza indicata sopra. Nel caso della prima approssimazione di funzione sono dall'indice 83 mentre per la seconda all'indice 84.

Capitolo 3

Terzo Esercizio

3.1 Traccia

Calcolare il valore approssimato dell'integrale

$$\int_0^\pi x^2 cos(x) dx$$

con un errore minore di 10^{-4}, usando sia il metodo dei trapezi che quello di Simpson.

Confrontare i risultati ottenuti in termini di accuratezza e costo computazionale.

Discutere sulle stime dell'errore utilizzate nei due casi.

3.2 Esecuzione

3.2.1 Input

Listing 3.1: Terzo Esercizio

```
% estremi dell'intervallo
a = pi;
b = 0;

% funzione da integrare
fun = @(x) x.^2.*cos(x);

% tolleranza
tol = 10.^-4;

```

```
% metodo di integrazione dei Trapezi
[I_t, err_t, x_t] = myTrapc(fun,a,b,tol)

% metodo di integrazione di Simpson
[I_s, err_s, x_s] = mySimpsonc(fun,a,b,tol)
```

3.2.2 Output

```
I_t =

    6.2832


err_t =

   2.9570e-05


x_t =

    3.1416
    3.1355
    3.1293
    3.1232
    3.1170
```

... Molte linee ...

```
    0.0491
    0.0430
    0.0368
    0.0307
    0.0245
    0.0184
    0.0123
    0.0061
         0
```

...

Figura 3.1: Output Terzo Esercizio

```
Warning: Tolleranza non raggiunta

I_s =

    6.6164


err_s =

    0.5676


x_s =

    3.1416
    2.3562
    1.5708
    0.7854
         0
```

Figura 3.2: Output Terzo Esercizio

3.3 Commento

Come si evince dalle figure soprastanti, il numero di nodi per la tecnica di integrazione composita adattiva è molto elevata. Più precisamente sono circa 500 nodi, quindi circa 500 valutazioni di funzione. Gli errori tra la formula di Simpson e quella dei trapezi non sono esattamente confrontabili, poiché la formula di Simpson non essendo né adattiva né composita non permette al codice di aggiungere nodi di integrazione con chiamate ricorsive come è stato fatto per la formula dei trapezi. Il costo computazionale è quindi molto elevato, lo sarebbe anche per la formula di Simpson se non si fosse arrestata per tolleranza non raggiunta.

Capitolo 4

Quarto Esercizio

4.1 Traccia

Usare le function implementate per calcolare $det(A)$, dove

$$A = \begin{pmatrix} 0 & 2 & 1 & 4 & 1 & 3 \\ 1 & 2 & -1 & 3 & 4 & 0 \\ 0 & 1 & 1 & -1 & 2 & -1 \\ 2 & 3 & -4 & 2 & 0 & 5 \\ -1 & -1 & 2 & -1 & 2 & 0 \end{pmatrix}$$

descrivere la procedura e contare il numero di operazioni.

4.2 Esecuzione

4.2.1 Input

Listing 4.1: Quarto Esercizio

```
% matrice in input
A = [0 2 1 4 1 3
     1 2 -1 3 4 0
     0 1 1 -1 2 -1
     2 3 -4 2 0 5
     1 1 1 3 0 2
     -1 -1 2 -1 2 0];

% eseguo il
[L, U, P] = myLU(A);

% determinante calcolato a partire da U e L
disp(det(L)*det(U));
```

```
% determinante calcolato con la funzione classica di matlab
disp(det(A));
```

4.2.2 Output

```
L =

    1.0000         0         0         0         0         0
         0    1.0000         0         0         0         0
    0.5000   -0.2500    1.0000         0         0         0
         0    0.5000    0.1538    1.0000         0         0
    0.5000    0.2500    0.2308   -0.0889    1.0000         0
   -0.5000    0.2500   -0.0769    0.2222    0.3779    1.0000


U =

    2.0000    3.0000   -4.0000    2.0000         0    5.0000
         0    2.0000    1.0000    4.0000    1.0000    3.0000
         0         0    3.2500    3.0000    0.2500    0.2500
         0         0         0   -3.4615    1.4615   -2.5385
         0         0         0         0    3.8222   -3.5333
         0         0         0         0         0    3.6686


P =

     0     0     0     1     0     0
     1     0     0     0     0     0
     0     0     0     0     1     0
     0     0     1     0     0     0
     0     1     0     0     0     0
     0     0     0     0     0     1

  -631.0000

  -631.0000
```

Figura 4.1: Output Quarto Esercizio

4.3 Commento

È stato confrontato il metodo di ricavo del determinante della matrice in input tramite la scomposizione LU, moltiplicando appunto i determinanti dei due, con il metodo di default di MATLAB `det()`. Si è verificato che i due metodi coincidono. Inoltre il numero di operazioni inteso come swap nella matrice è $n^3/3$, dove n è il numero di elementi nella matrice, proprio perché viene utilizzata un variabile d'appoggio.

Capitolo 5

Quinto Esercizio

5.1 Traccia

Dato il sistema $Ax = b$ dove

$$A = \begin{pmatrix} 3 & 12 & 0 & -1 & 0 & 0 \\ 3 & 0 & 31 & 1 & 0 & 0 \\ 2 & 1 & 0 & 0 & 17 & -3 \\ 27 & 2 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 & 1 & 11 \\ 0 & 0 & 0 & 24 & -1 & 0 \end{pmatrix} b = \begin{pmatrix} 38 \\ 117 \\ 12 \\ 98 \\ 14 \\ 55 \end{pmatrix}$$

trasformare la matrice A in modo che la condizione sufficiente per la convergenza sia soddisfatta , analizzare la velocità asintotica di convergenza in entrambi i casi, risolvere il sistema dato utilizzando sia la funzione che implementa il metodo di Jacobi che quella che implementa il metodo di Gauss-Seidel. Verificare sperimentalmente i risultati teorici.

5.2 Esecuzione

5.2.1 Input

Listing 5.1: Quinto Esercizio

```
% matrice in input
A = [3 12 0 -1 0 0
     4 0 31 1 0 0
     2 1 0 0 17 -3
     27 2 0 0 0 1
     0 0 0 -1 1 11
     0 0 0 24 -1 0];

```

```
% trasformazione della matrice
A = sparse(A);

% vettore dei termini noti
b = [38 117 12 98 14 55];

% risoluzione con metodi di BackSub e FarwardSub
[x_1] = SolveLinearSystem(A,b);

% risoluzione metodi Jacobi e Gauss-Seidel
[xv,iter] = myJacobi(A,b',zeros(8,1),400,1e-4);
[xv_1,iter_1] = myGS(A,b',zeros(8,1), 100,1e-4);

% Jacobi
disp(xv);
disp(iter);

% GS
disp(xv_1);
disp(iter_1);

% solver
disp(x_1);
```

5.2.2 Output

```
   NaN
  -Inf
   NaN
  -Inf
   NaN
  -Inf
     0
     0

     1

   12.6667
       Inf
      -Inf
      -Inf
      -Inf
       NaN
         0
         0

     0

    3.3901
    2.5116
    3.2623
    2.3089
    0.4143
    1.4450
```

Figura 5.1:

5.3 Commento

Controlliamo se la matrice A soddisfa la condizione di convergenza per i metodi di Jacobi e Gauss-Seidel, cioè se è una matrice diagonalmente dominante. Una matrice si dice diagonalmente dominante se il valore assoluto dell'elemento diagonale è maggiore o uguale alla somma dei valori assoluti degli altri elementi della stessa riga:

$$|3| \geq |12| + |0| + |-1| + |0| + |0|$$

$$|31| \geq |3| + |1| + |0| + |0| + |0|$$

$$|17| \geq |2| + |1| + |0| + |0| + |-3|$$

$$|27| \geq |2| + |0| + |0| + |0| + |1|$$

$$|11| \geq |0| + |0| + |-1| + |1| + |0|$$

$$|24| \geq |0| + |0| + |-1| + |0| + |0|$$

È verificata la convergenza quindi si può procedere con i metodi di Gauss-Seidel e Jacobi.

Capitolo 6

Sesto Esercizio

6.1 Traccia

Si stima che l'area A della superficie di un essere umano dipende dal peso W e dall'altezza H. La seguente tabella riporta misure dell'area $A(m^2)$ effettuate su un numero di individui di altezza $H = 180cm$ e diversi pesi (kg).
Mostrare che i dati sono rappresentati ragionevolmente bene dalla legge $A = aW^b$.

W (kg)	70	75	77	80	82	84	87	90
A (m^2)	2.10	2.12	2.15	2.20	2.22	2.23	2.26	2.30

Determinare le costanti a e b, stabilire qual è l'area della superficie di una persona di 95kg. Descrivere la procedura utilizzata.

6.2 Esecuzione

6.2.1 Input

Listing 6.1: Sesto Esercizio

```
% Vettori di x e y su cui eseguire i minimi quadrati
x = [70 75 77 80 82 84 87 90]; % [70 - 90]
y = [2.10 2.12 2.15 2.20 2.22 2.23 2.26 2.30];% [2.10 - 2.30]

% plotto i dati ed estrapolo i coefficienti della retta
plot(x, y, 'o');
hold on

% coefficienti della retta
x_dat = myls(x, y, 2)
```

```
%retta
f = @(x)x_dat(2).*x +x_dat(1);

%plotting della retta
fplot(f, [70 90]);

% dati modificati in cui aggiungo 95 kg
x_1 = [70 75 77 80 82 84 87 90 95];% [70 - 95]
y_1 = [2.10 2.12 2.15 2.20 2.22 2.23 2.26 2.30 2.35];% [2.10 -
    2.35]

plot(x_1, y_1, 'o');
hold on

% plotto i nuovi dati e verifico la buona rappresentazione
x_dat = myls(x_1, y_1, 2);
f = @(x)x_dat(2).*x +x_dat(1);

% plotting della nuova retta
fplot(f, [70 95]);
```

6.2.2 Output

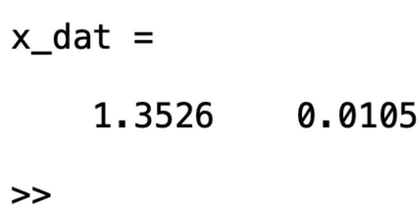

Figura 6.1: Output Sesto Esercizio

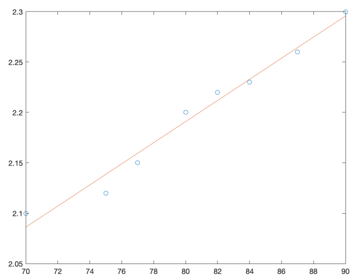

Figura 6.2: Secondo plot

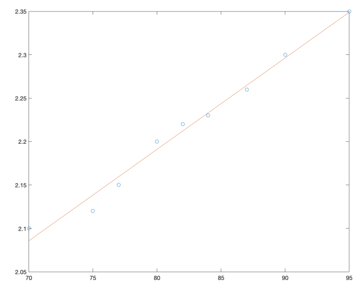

Figura 6.3: Primo plot

6.3 Commento

Viene usata la funzione dei minimi quadrati per generare un polinomio a due coefficienti che rappresenti i dati i input. (Due coefficienti per rappresentare una retta). I due coefficienti vengono estrapolati dopo aver linearizzato correttamente la formula in input e quindi verificando che i dati seguono la legge: $A = a * W^b$.
Linearizzazione di $A = aW^b$
$y = ax^b \Leftrightarrow \log y = \log e + b \log x \Leftrightarrow y = A + Bx$
dove $y = \log y \wedge x = \log x$
B è le pendenza della retta quindi:

$$a = 10^A$$

$$b = \frac{\log\left(\frac{y_2}{y_2}\right)}{\log\left(\frac{x_2}{x_1}\right)}$$