Lab 2 - Equazioni non lineari (traccia)

February 26, 2024

1 Lab 2 - Equazioni non lineari

Si consideri il problema della ricerca degli zeri dell'equazione non lineare $f(x) = 0$, dove f è definita da:

$$f(x) := e^x - x^2 - \sin(x) - 1, \qquad x \in [-2, 2] \, . \tag{1}$$

2 Rappresentazione grafica della funzione

Disegnare il grafico della funzione f e la retta $y = 0$ in modo da evidenziare le due soluzioni dell'equazione.

[1]:
```
# step 0 -> ricordarsi di importare i pacchetti numpy and matplotlib

import numpy as np
import matplotlib.pyplot as plt
```

Vediamo un nuovo costrutto utile per costruire funzioni matematiche "lambda function" in modo semplice e veloce.

```
f = lambda nome_variabili: esperessione della funzione
```

[2]:
```
# esempi di lambda function
```

Scrivere la funzione f dell'esercizio usando la lambda function

[3]:
```
# lambda function della funzione f
```

Rappresentazione grafica

[4]:
```
# rappresentazione grafica della funzione f
```

3 Bisezione

Domanda: Il metodo di bisezione è applicabile per calcolare entrambe le radici? Motivare la risposta.

Esercizio 1.1: implementazione bisezione

Scrivere una function `bisez` che implementi il metodo di bisezione. L'intestazione della funzione sarà ad esempio la seguente:

```
def bisez(f,a,b,toll):
#
# implementazione di bisezione
#
return xvect
```

Tale funzione riceve in input: * `f` → funzione di cui vogliamo calcolare gli zeri, * a → primo estremo * b → secondo estremo * `toll` → tolleranza richiesta

e in out * `xvect` → il vettore delle iterate.

[5]:
```
# definzione del metodo di bisezione
def bisez(f, a, b, toll):

  return
```

Esercizio 1.2 Quando è possibile, applicare il metodo di bisezione alla funzione f. Stampare il numero di iterazioni, lo zero trovato x^* e il valore della funzione $f(x^*)$.

[6]:
```
# scelta degli estremi a,b


# stampare iterazioni, valore di x* e valore di f(x*)
```

4 Metodo di Newton

Domanda: Discutere le proprietà di convergenza del metodo di Newton per entrambi gli zeri, valutando l'opportunità di applicare il metodo di Newton.

Esercizio 2.1: implementazione Newton Scrivere una function `newton` che implementi il metodo di Newton. L'intestazione della funzione sarà ad esempio la seguente:

```
def newton (f,df,x0,nmax,toll)
#
# implementazione del metodo di newton
#
return xvect
```

Tale funzione riceve in input: * `f` → funzione di cui vogliamo calcolare gli zeri, * `df` → la derivata della funzione f, * x_0 → punto di partenza * `nmax` → numero massimo di iterazione * `toll` → tolleranza richiesta

e in out * `xvect` → il vettore delle iterate.

Si utilizzi un criterio d'arresto basato sul modulo della differenza tra due iterate successive.

[7]:
```
# definizione del metodo di newton
def newton(f, df, x0, nmax, toll):
```

```
    return
```

Esercizio 2.2: modifica di Newton Si scriva, modificando opportunamente la function `newton`, il metodo di Newton modificato, passando come parametro in ingresso anche la molteplicità dello zero cercato. L'intestazione della funzione sarà la seguente:

```
def newton (f,df,x0,nmax,toll,m=1)
  #
  # modificare il passo iteratativo di newton
  #  x^{k+1} = x^k - m f(x^k)/f'(x^k)
  #
  return xvect
```

dove `m` è la molteplicità dello zero cercato.

Esercizio 2.3

Applicare il metodo di Newton e, quando è il caso, il metodo di Newton modificato (aggiungendo la specifica `m`) con tolleranza 10^{-6}, per la funzione

$$f(x) := e^x - x^2 - \sin(x) - 1, \qquad x \in [-2, 2]\ . \tag{2}$$

Riportare su un grafico in scala semilogaritmica l'andamento dell'errore in funzione del numero di iterazioni. Per il calcolo dell'errore si assuma come valore esatto dello zero non nullo, il valore $x = 1.279701331000996$. Cosa si osserva nell'ordine di convergenza?

[8]:
```
# funzione f e la sua derivata df
f = lambda t: np.exp(t) - t**2 - np.sin(t) - 1.0
df = lambda t: np.exp(t) - 2*t - np.cos(t)
```

[9]:
```
# ricerca dello zero xe=1.279701331000996
xe=1.279701331000996
# starting point
x0=1



# calcolo dell'errore

# rappresentazione grafica dell'errore usando plt.semilogy()
```

[10]:
```
# Ricerca dello zero xe=0
xe=0
# starting point
x0=0.1
```

5 Esercizi per casa

5.1 Metodo punto fisso

Si consideri il problema della ricerca degli zeri dell'equazione non lineare $g(x) = 0$, dove g è definita da:

$$g(x) := 4x - e^{x^2}, \qquad x \in \left[0, \frac{3}{2}\right] . \tag{3}$$

Esercizio 3.1 Disegnare il grafico della funzione g e la retta $y = 0$ in modo da evidenziare le due soluzioni dell'equazione. Inoltre, per ogni soluzione, determinare un opportuno intervallo che la contenga.

[11]:
```python
# plot della funzione g
```

Osservazione sulle soluzioni dell'esercizio 3.1:

Esercizio 3.2 Individuare la funzione di iterazione ϕ per il metodo di punto fisso e la sua derivata $\phi'(x)$. Plottare ϕ' nell'intervallo considerato. Il metodo di punto fisso è applicabile per calcolare entrambe le radici? Motivare la risposta.

[12]:
```python
# plot della derivata prima di phi
```

Motivazione esercizio 3.2:

Esercizio 3.3

Scrivere la function `puntofisso` che implementi il metodo di punto fisso. L'intestazione della funzione sarà ad esempio la seguente:

```
def puntofisso (phi,x0,nmax,toll)
  #
  # implementazione del metodo di newton
  #
  return xvect
```

Tale funzione riceve in input: * `phi` $\rightarrow$ funzione di iterazione, * x_0 $\rightarrow$ punto di partenza * `nmax` $\rightarrow$ numero massimo di iterazione * `toll` $\rightarrow$ tolleranza richiesta

e in out * `xvect` $\rightarrow$ il vettore delle iterate.

Si utilizzi un criterio d'arresto basato sul modulo della differenza tra due iterate successive.

[13]:
```python
def puntofisso(phi, x0, nmax, toll):


  return
```

Esercizio 3.3 Applicare il metodo del punto fisso, quando possibile, con numero massimo di iterazioni `nmax` $= 100$ e tolleranza pari a 10^{-5}, utilizzando la funzione `puntofisso` e dati iniziali $x0 = 0$, $x0 = 1.5$.

[14]:
```python
# verifica del metodo di punto fisso
```

5.2 Bisezione

Esercizio 4.1 Si consideri il problema della ricerca degli zeri dell'equazione non lineare $f(x) = 0$, dove f è definita da:

$$f(x) := e^x - x^2 - \sin(x) - 1, \qquad x \in [-2, 2] \, . \tag{4}$$

Provare ad implementare bisezione con il ciclo for. Ricordo che il numero massimo di iterazioni è

$$k > \log_2(\frac{|b-a|}{\text{toll}}) - 1 \tag{5}$$

con k un numero intero. Ricordo che nel caso in cui si ottenga un numero non intero nell'espressione a destra allora k va arrotondato al numero intero superiore. Un comando utile per il calcolo di k è `np.fix` (si veda `help(np.fix)`).

[15]:
```
def bisezfor(f, a, b, toll):



  return
```

Esercizio 4.2 Testare la function `bisezfor` nella ricerca degli zeri della fuzione f.

[16]:
```
f = lambda x: np.exp(x) - x**2 - np.sin(x) - 1.0
# scelta degli estremi a,b



# stampare iterazioni, valore di x* e valore di f(x*)
```

5.3 Bisezione e Newton

Esercizio 5 Si consideri il problema della ricerca degli zeri dell'equazione non lineare $f(x) = 0$, dove f è definita da:

$$f(x) := e^x - x^2 - \sin(x) - 1, \qquad x \in [-2, 2] \, . \tag{6}$$

Utilizzare in sequenza il metodo di bisezione e il metodo di Newton per la ricerca dello zero $1 < \alpha_2 < 1.5$; in particolare si adotti il metodo di bisezione per l'avvicinamento allo zero e successivamente il metodo di Newton per la convergenza "veloce" ad α_2, assumendo come punto di innesco lo zero approssimato con il metodo di bisezione. Nello specifico si considerino 5 iterazioni del metodo di bisezione sull'intervallo $[1, 1.5]$ e si assuma come tolleranza per il metodo di Newton il valore 10^{-10}.

[17]:
```
# estremi dell'intervallo
a=1
b=1.5
f = lambda x: np.exp(x) - x**2 - np.sin(x) - 1.0
df = lambda x: np.exp(x) - 2*x - np.cos(x)
```