

# Laboratorio di Calcolo, Esercitazione 5, 14-15 novembre 2024

Canale Pet-Z, Docenti: Shahram Rahatlou, Sibilla Di Pace

Lo scopo di questa esercitazione è di implementare il calcolo della radice quadrata di un numero positivo con un metodo iterativo noto ai babilonesi e di utilizzare uno script di python per visualizzare la convergenza del metodo alla soluzione.

Esistono diversi algoritmi iterativi per il calcolo della radice quadrata di un numero positivo  $a$ . In questa esercitazione implementeremo due di questi metodi che usano successioni che convergono alla soluzione in base alla precisione  $\epsilon$  indicata dall'utente.

## ► Cartella di lavoro

Fare login sulla postazione utilizzando le credenziali user-id `studente` e password `informatica`. Creare una cartella `LCSR5` nella *home directory* con il comando `mkdir` in cui scriverete i programmi di oggi. Tutti i file di codice sorgente in C e in python dovranno trovarsi in questa cartella per essere visualizzati. **Le cartelle create sulla scrivania (Desktop) o in altre sotto-cartelle non verranno copiate né valutate.**

## ► Prima parte: Metodo Babilonese

La radice quadrata  $r = \sqrt{a}$  di un numero positivo  $a$  si può calcolare con un metodo iterativo come il limite della successione  $r_{n+1} = \frac{1}{2}(r_n + a/r_n)$ . Il valore del primo termine  $r_0$  è influente ai fini della convergenza e può essere scelto a piacere e potete verificare che cambiando tale valore iniziale il risultato non cambia! Il ciclo termina quando la differenza  $|r_{n+1} - r_n|$  tra due termini successivi è minore della precisione  $\epsilon$  desiderata.

Creare un programma `babilon-NNN.c`, dove `NNN` è il vostro numero di gruppo, ad esempio `098`, nella cartella `LCSR5` utilizzando l'editor di testo `emacs`, per eseguire le seguenti operazioni:

1. chiedere all'utente il valore della precisione  $\epsilon$  compreso nell'intervallo  $[10^{-6}, 0.1]$ ;
2. chiedere all'utente il valore del termine  $r_0$  ed assicurarsi che sia diverso da zero;
3. chiedere di inserire il valore di  $a$  positivo di cui si vuole calcolare la radice;
4. utilizzare un ciclo opportuno per implementare il metodo babilonese descritto sopra;
  - in ciascuna iterazione stampare sullo schermo il numero di iterazione  $n$  e la stima attuale  $r_n$  del risultato sulla stessa riga utilizzando come descrittore `"%3d \t %.10lf \n"`
5. Al termine del ciclo, scrivere sullo schermo il numero  $N_{\text{tot}}$  di iterazioni eseguite e la differenza  $|r_n - \sqrt{a}|$  tra il valore ottenuto da voi e il valore calcolato con la funzione `sqrt()` della libreria matematica, utilizzando il formato `%.5g` per apprezzare le piccole differenze;

Potete eseguire il programma per lo stesso valore di  $a$  ma variando la precisione  $\epsilon$  per vedere come cambia  $N_{\text{tot}}$  al variare di  $\epsilon$ .

Si ricorda che per creare l'eseguibile utilizzando la libreria matematica dovete usare il comando `gcc -Wall -o app.exe programma.c -lm` dalla riga di comando nella shell. **Si consiglia di scrivere il programma in modo incrementale, verificando la corretta compilazione e l'esecuzione almeno dopo ciascuno dei passi indicati nel testo.**

### ► Seconda parte: grafico con python

Vogliamo graficare l'andamento della successione  $r_n$  per la stima di  $\sqrt{a}$  in funzione del numero di iterazioni eseguite. Il codice python deve essere in uno file `convergenza-NNN.py` che potete creare utilizzando l'editor di testo `emacs` nella cartella `LCSR5`. A tal fine

1. creare un file `dati.txt` utilizzando l'editor `emacs` per copiare o scrivere a mano i valori di  $n$  e  $r_n$  stampati sullo schermo dal codice in `babelon-NNN.c`;
2. scrivere le seguenti sintassi di python nel file

```
1 # carica moduli pyplot, numpy e matematica di python
2 import matplotlib.pyplot as plt
3 import numpy as np
4 import math as m
5
6 plt.title("Calcolo della radice quadrata con il metodo babilonese")
7
8 # carica dati dal file
9 # iter e' un array con i dati nella prima colonna (theta)
10 # stima e' un array con i dati della seconda colonna (gittata)
11 iter, stima = np.loadtxt('dati.txt', unpack=True)
12
13 # crea grafica di stima in funzione di iter
14 plt.plot( iter, stima, 'x-', label='stima con il metodo babilonese')
15
16 # specifica limite inferiore superiore per asse x e y con numeri
    opportuni in base ai vostri dati
17 # per esempio se state stimando la radice di a = 15 con circa 5
    iterazioni
18 plt.xlim(0, 10)
19 plt.ylim(3., 5.)
20
21 # aggiungi legenda per gli assi
22 plt.xlabel('numero iterazione')
23 plt.ylabel('stima radice a')
24
25 # linea orizzontale rossa al valore di sqrt(a) usando la libreria
    matematica
26 plt.axhline( y=m.sqrt(15), color = 'red', linestyle='--')
27
28 # nome del file in cui salvare il grafico
29 plt.savefig("radice.png")
30
31 # mostra grafico
32 plt.show()
```

Listato 1: Programma convergenza.py

### ► Terza parte: Metodo Alternativo

L'inverso  $1/\sqrt{a}$  della radice quadrata di un numero positivo  $a$  si ottiene come il limite della successione  $y_{n+1} = \frac{3}{2}y_n - \frac{a}{2}y_n^3$ . La scelta del termine  $y_0$  è arbitraria, tuttavia questo metodo è molto più sensibile al valore scelto e può facilmente divergere (provare per credere!). Occorre quindi stimare il valore di  $y_0$  con il metodo babilonese della prima parte  $y_0 = 1/r_n$ .

Scrivere un programma `radice-NNN.c` per seguire le seguenti operazioni

1. chiedere all'utente il valore della precisione  $\epsilon$  compreso nell'intervallo  $[10^{-6}, 0.1]$ ;
2. chiedere di inserire il valore di  $a$  positivo di cui si vuole calcolare la radice;
3. usare un valore  $r_0$  a vostra scelta, senza chiederlo all'utente, e fare due iterazioni con il metodo babilonese per calcolare il termine  $r_2$  della successione;
4. implementare il nuovo metodo iterativo con la successione  $y_n$  con un opportuno ciclo che utilizzi come valore iniziale  $y_0 = 1/r_2$ 
  - in ciascuna iterazione stampare sullo schermo il numero di iterazione  $n$  e la stima di  $1/y_n$  sulla stessa riga utilizzando un'opportuna formattazione;
  - interrompere il ciclo quando  $|1/y_{n+1} - 1/y_n| < \epsilon$
5. Al termine del ciclo, scrivere sullo schermo il numero  $N_{\text{tot}}$  di iterazioni eseguite e la differenza  $|1/y_n - \sqrt{a}|$  tra il valore ottenuto da voi e il valore calcolato con la funzione `sqrt()` della libreria matematica, utilizzando un descrittore opportuno.

### ► opzionale:

1. grafica l'andamento di  $1/y_n$  in funzione del numero di iterazioni.