

# Laboratorio di Calcolo per Fisici, Terza esercitazione

Canale Pb-Z, Docente: Lilia Boeri

Lo scopo della terza esercitazione di laboratorio è di fare pratica con le istruzioni di controllo di flusso `if...(then)...else`; `while...do`; `for...`, scrivendo un programma che localizza gli zeri di una funzione all'interno di un intervallo dato.

## ► Prima parte:

Scrivere un programma `sinxx.c` che, utilizzando un opportuno ciclo iterativo, calcoli il valore della funzione  $f(x) = \frac{\sin(x)}{x}$  in una serie di punti equispaziati lungo l'asse  $x$ , nell'intervallo  $x \in [-15, 15]$ . Si noti che:

1. Il numero di punti  $N_p$  deve essere sufficiente a dar vita ad un grafico ragionevolmente continuo delle funzione ( $N_p \geq 100$ );
2. Se la funzione viene calcolata nel punto  $x = 0$  il programma restituisce il valore NAN (*not a number*), dal momento che si tratta di valutare un'espressione del tipo  $\frac{0}{0}$ . Il limite per  $x \rightarrow 0$  di  $\frac{\sin(x)}{x}$  è noto e vale:  $\lim_{x \rightarrow 0} \frac{\sin(x)}{x} = 1$ . Inserire nel programma un'opportuna istruzione **if** che corregga il risultato numerico con il valore analitico corretto in  $x = 0$ .
3. Scrivere i risultati su un file `sinxx.dat` di  $N_p$  righe e 2 colonne, nella forma:  $x, y$ . Per compiere questa operazione potete o incollare manualmente l'output dello schermo su un file vuoto, o *ridirigere* l'output del vostro programma su un file con il comando:

```
./programma.x > file_out
```

4. Creare con python un grafico della funzione  $\frac{\sin(x)}{x}$  utilizzando i punti calcolati e salvarlo sul file `sinxx.png`. Quanti zeri ci sono nell'intervallo  $[-15, 15]$ ? Dove sono?

## ► Seconda parte:

Scrivere un programma `zero.c` che calcoli automaticamente il numero degli zeri della funzione  $f(x) = \frac{\sin(x)}{x}$  nell'intervallo  $[a, b] = [-15, 15]$  mediante il seguente algoritmo, basato sul teorema degli zeri di una funzione continua.

*(Attenzione, l'idea di base è simile a quella dell'algoritmo di bisezione, ma l'algoritmo è diverso!).*

1. L'intervallo di partenza viene diviso in  $N$  intervalli contigui, tutti uguali, di estremi  $x_L^i, x_R^i$ ,  $i = 0, \dots, (N - 1)$ .
2. Per ciascun intervallo l'algoritmo controlla la presenza di eventuali zeri calcolando il valore di  $s = f(x_L^i) \cdot f(x_R^i)$ . Se  $s > 0$ , non ci sono zeri; se  $s \leq 0$ , c'è uno zero.
3. Alla fine dell'esecuzione, il programma restituisce il numero totale di zeri trovati dall'algoritmo,  $N_{zero}$ .

---

► **Terza parte (facoltativa):**

1. Far girare il programma `zero.c` con un numero variabile di intervalli  $N \geq 1$  e creare un file che contenga  $N, N_{zero}$ .
2. Graficare l'andamento di  $N, N_{zero}$ : qual è il valore minimo di  $N$  per cui vengono trovati tutti gli zeri? Qual è la *risoluzione* corrispondente?
3. Se il valore  $N = 3, 4, 5, 6$  non fanno parte della vostra scelta iniziale, fate girare il programma anche per questi valori. Quanti zeri trovate? Perché? *Scrivete le risposte a queste ultime due domande su un file `risposte.txt` nella cartella EX3.*
4. Utilizzando diversi valori di  $N$  abbastanza grandi da trovare tutti gli zeri, fate stampare al vostro programma la vostra migliore stima del valore degli zeri con la relativa incertezza.

► **Attenzione!** A volte, per via di un errore di programmazione, può succedere che un programma entri in un *loop infinito*. Se questo succede, ci sono due modi di interrompere l'esecuzione:

- Se il programma è in esecuzione nella *shell*: premere la combinazione di tasti CTRL+C.
- Se il programma è in esecuzione in modalità *background*: aprire una nuova shell. Lanciare il programma `top` (non in modalità *background*). Individuare il PID (*Process Identifier*) del programma bloccato. Premere q per uscire da `top`. Digitare nella shell il comando: `kill -9 numeroPID` (Enter).