Elaborato di Fisica Computazionale

A.A 2024/2025

Andrea Rossi `N. 897139`

15 ottobre 2024

Indice

Introduzione 1

1.1 Struttura dell'elaborato

Nel seguente elaborato verrà illustrata l'analisi degli argomenti proposti nel corso di Fisica Computazionale.

Di seguito sono riportate le informazioni richieste per una fruizione completa del documento:

Codice sorgente e dati Il codice sorgente ed i dati analizzati nei vari esercizi sono reperibili al seguente repository Github nelle cartelle dei capitoli omonimi.
• La lingua utilizzata nel codice sorgente sar'a l'inglese per avere una maggiore coesione sintattica con i linguaggi di programmazione utiliz- zati. • Verranno ripetute, sopratutto nella parte introduttiva dei vari capi- toli, i punti chiave recuperati dalle risorse disponibili sull'e-learning del corso. Esse saranno riassuntive favorendo le precisazioni e lo studio degli esercizi per ottenere un quadro completo dei vari argomenti.
Many modern printed textbooks have adopted a layout with prominent margins where small figures, tables, remarks and just about everything else can be displayed. Arguably, this layout helps to organise the discussion by separating the main text from the ancillary material, which at the same time is very close to the point in the text where it is referenced.

Numeri 2

2.1 Rappresentazione

La rappresentazione numerica a cui il calcolo scientifico si riferisce principalmente è quella dei numeri reali; nell'ambito informatico tale rappresentazione utilizza il concetto di numeri a virgola mobile come standard: i numeri reali vengono rappresentati attraverso una notazione scientifica in base due tramite la seguente formula:

$$(-1)^S \left(1 + \sum_n M_n 2^{-n}\right) \cdot 2^E$$

Dove:

S è il valore booleano per il **segno**
M è la parte decimale detta **mantissa**
$E = e - d$ è l'**esponente** con d (offset), e (esponente dopo offset)

2.2 Esercizi

2.2.1 Precisione

Nozioni teoriche

Definizione La *precisione di macchina* (o *ϵ di macchina*) è la differenza tra 1 e il numero successivo rappresentabile dato il numero di bit richiesti, esso sarà dunque:

$$\epsilon = 2^{-M}$$

Nello standard dei numeri a virgola mobile (IEE 754) si studiano principalmente due sottoclassi di numeri i cui nominativi nei linguaggi C-like sono:

float numero a singola precisione (32 bit di memoria):

- M: 23 bit
- E: 8 bit
- Valore massimo: $3.40 \cdot 10^{38}$
- ϵ: $\sim 10^{-7}$

double numero a doppia precisione (64 bit di memoria):

- M: 52 bit
- E: 11 bit
- Valore massimo: $1.8 \cdot 10^{308}$
- ϵ: $\sim 10^{-16}$

Implementazione e osservazioni

File necessari sorgente: `number_precision.c`, dati: `number_precision.dat`

Il codice sorgente scritto utilizza funzionalità base del linguaggio C. [1]

1: Per formattare il codice secondo la richiesta del problema si usi la definizione `EXERCISE_FORMAT`, altrimenti verrà utilizzata una formattazione più compatta per leggere in maniera più diretta i dati, si consiglia di utilizzare quest'ultima per comprendere l'analisi sottostante

Analisi e conclusioni Dai dati ottenuti si possono notare in maniera esaustiva varie proprietà dei numeri a virgola mobile:

1. Esiste un *valore massimo* sia per singola ($\sim 3 \cdot 10^{38}$) sia per doppia precisione ($\sim 2 \cdot 10^{308}$), superato esso viene mostrato un valore esatto *inf* definito dallo standard descritto in precedenza;
2. I numeri hanno un *errore macchina* dettato dalla capienza di memoria della mantissa;
3. Come mostrerà più precisamente la prossima sezione, l'errore viene *propagato* nella somma:
   - $1 + f_{mult}$ perde completamente l'informazione su f_{mult}
   - $1 + d_{mult}$ la conserva soltanto per le prime iterazioni;

2.2.2 Propagazione degli errori

Nozioni teoriche E' immediato notare come i numeri a virgola mobile possano essere rappresentati come variaili casuali con errore associato, derivante dalla precisione di macchina.

Prendiamo in esame una funzione $f(x, y)$ dove x, y sono variabili casuali indipendenti con rispettivo errore σ_x, σ_y, allora l'errore su f sarà:

$$\sigma_f^2 = \left(\frac{\partial f}{\partial x}\right)^2 \sigma_x^2 + \left(\frac{\partial f}{\partial y}\right)^2 \sigma_y^2$$

Assumendo ora $f = x + y$ otteniamo:

$$\sigma_f^2 = \sigma_x^2 + \sigma_y^2$$

Notiamo immediatamente quindi che se $x \gg y$ allora $\sigma_f \approx \sigma_x$ quindi si perde l'informazione su y nella somma.

Implementazione e osservazioni

File necessari sorgente: `error_propagation.c`

In base alle richieste l'output è il seguente:

1. $(0.7 + 0.1) + 0.3 =_? 0.7 + (0.1 + 0.3)$:

   ```
   Output: 1.1000000238418579, 1.1000000238418579
   ```

   La somma risulta associativa.

2. $[10^{20} + (10^{20})] + 1 =_? 10^{20} + [(10^{20}) + 1]$:

```
Output: 1.0000000000000000, 0.0000000000000000
```

La somma risulta non associativa.

Analisi Utilizzando le formule discusse si può studiare la propagazione dell'errore nella somma. In essa la propagazione dipende dall'errore assoluto dei singoli addendi. Assumendo numeri a singola precisione e ricordando che $\sigma_x \approx \epsilon \sim 10^7$, si ottengono i seguenti casi:

1. Per i valori 0.7, 0.1, 0.3 l'ordine di grandezza è lo stesso, quindi, tutti i valori possegono un errore assoluto $\Delta x \sim 10^{-8}$; propagando l'errore nella somma si ottiene dunque $\Delta_{output} \sim 3 \cdot \Delta x$ in accordo con i risultati.
2. Il risultato è describile come un caso limite nell'errore di propagazione rispetto alla singola precisione, infatti, 10^{20} avrà un errore assoluto di $\sim 10^{13}$ mentre 1 di 10^{-7}!
   La spiegazione dell'output ottenuto, dunque, si basa sulla differenza tra ordini di grandezza dei diversi addendi:
   - Nel termine a sinistra vengono sommati prima numeri con errore assoluto paragonabile. Si ottiene quindi ~ 0 che sarà poi sommato con un numero avente errore assoluto simile a 1.
   - Nel termine a destra, invece, si sommano due valori con venti ordini di grandezza di differenza: l'errore assoluto di 10^{20} prevale e si perde qualsiasi informazione nella somma per termini:
     $$x \ll 10^{20} \Rightarrow x + 10^{20} \sim 10^{20}$$
     Segue che *1* sarà ignorato nella somma a destra.

Conclusioni Nel manipolare numeri in un calcolatore l'operazione eseguita, la precisione e la differenza in ordine di grandezza dei numeri partecipanti devono essere tenuti sempre in considerazione specialmente nelle addizioni.

Approssimazioni 3

3.1 Prova

Matrici 4

4.1 Introduzione

Struttura del codice La parte principale del codice si basa su due header: `text`

4.2 Esercizi

4.2.1 Inversione di matrici triangolari

4.2.2 Eliminazione di Gauss

4.2.3 Decomposizione LU

Intepolazione 5

5.1 Interpolazione: esercizio 1

5.2 Funzione di Runge: esercizio 2

Greek Letters with Pronunciations

Character	Name	Character	Name
α	alpha *AL-fuh*	ν	nu *NEW*
β	beta *BAY-tuh*	ξ, Ξ	xi *KSIGH*
γ, Γ	gamma *GAM-muh*	o	omicron *OM-uh-CRON*
δ, Δ	delta *DEL-tuh*	π, Π	pi *PIE*
ϵ	epsilon *EP-suh-lon*	ρ	rho *ROW*
ζ	zeta *ZAY-tuh*	σ, Σ	sigma *SIG-muh*
η	eta *AY-tuh*	τ	tau *TOW (as in cow)*
θ, Θ	theta *THAY-tuh*	υ, Υ	upsilon *OOP-suh-LON*
ι	iota *eye-OH-tuh*	ϕ, Φ	phi *FEE, or FI (as in hi)*
κ	kappa *KAP-uh*	χ	chi *KI (as in hi)*
λ, Λ	lambda *LAM-duh*	ψ, Ψ	psi *SIGH, or PSIGH*
μ	mu *MEW*	ω, Ω	omega *oh-MAY-guh*

Capitals shown are the ones that differ from Roman capitals.