Elaborato di

Calcolo Numerico

Anno Accademico 2018/2019

Alessio Falai - 6134275 alessio.falai@stud.unifi.it
Leonardo Calbi - 6155786 leonardo.calbi@stud.unifi.it

April 22, 2019

Capitoli

1 Capitolo 1

1.1 Esercizio 1

Verificare che, per h sufficientemente piccolo:

$$\frac{3}{2}f(x) - 2f(x-h) + \frac{1}{2}f(x-2h) = hf'(x) + O(h^3)$$

Usando gli sviluppi di Taylor fino al secondo ordine otteniamo:

$$f(x) = f(x_0) + f'(x_0)h + \frac{1}{2}f''(x_0)h^2 + O(h^3)$$

$$f(x-h) = f(x_0) + f'(x_0)(x-h-x_0) + \frac{1}{2}f''(x_0)(x-h-x_0)^2 + O(h^3)$$

$$f(x-2h) = f(x_0) + f'(x_0)(x-2h-x_0) + \frac{1}{2}f''(x_0)(x-2h-x_0)^2 + O(h^3)$$

Dato che $x - x_0 = h$, ponendo $x_0 = x - h$, si ha che:

$$f(x-h) = f(x_0) + O(h^3)$$

$$f(x-2h) = f(x_0) - f'(x_0)h + \frac{1}{2}f''(x_0)h^2 + O(h^3)$$

Effettuando le opportune sostituzioni, la relazione iniziale diventa:

$$\frac{3}{2}f(x_0) + \frac{3}{2}f'(x_0)h + \frac{3}{4}f''(x_0)h^2 - 2f(x_0) + \frac{1}{2}f(x_0) - \frac{1}{2}f'(x_0)h + \frac{1}{4}f''(x_0)h^2 + O(h^3) = hf'(x) + O(h^3)$$

Semplificando, si ottiene che:

$$f'(x_0)h + f''(x_0)h^2 + O(h^3) = hf'(x) + O(h^3)$$

A questo punto, scriviamo lo sviluppo di Taylor anche per $f'(x)$:

$$f'(x) = f'(x_0) + f''(x_0)h + O(h^2)$$

Infine, otteniamo che:

$$f'(x_0)h + f''(x_0)h^2 + O(h^3) = f'(x_0)h + f''(x_0)h^2 + O(h^3)$$

Dunque, l'uguaglianza iniziale è verificata.

1.2 Esercizio 2

Quanti sono i numeri di macchina normalizzati della doppia precisione IEEE? Argomentare la risposta.

Nella doppia precisione IEEE si utilizzano 64 bit per rappresentare un numero in virgola mobile in macchina. Di questi, il primo bit è riservato al segno $\pm$, i 52 bit successivi rappresentano la mantissa ρ e i restanti 11 l'esponente η. Un numero reale può essere rappresentato in macchina mediante la formula $r = \pm\rho\eta$, con $\rho = \sum_{i=1}^{m} \alpha_i b^{1-i}$ e $\eta = b^{e-\nu}$. Nel caso dei numeri normalizzati, abbiamo delle restrizioni sui numeri rappresentabili in macchina:

- La mantissa è assunta della forma $1.f$
- Lo shift ν è pari a 1023
- Il valore e deve essere compreso tra 0 e 2047, estremi esclusi

Dunque, per ricavare il numero di numeri normalizzati della doppia precisione IEEE, contiamo le possibili combinazioni di bit, date le condizioni sopra definite:

- Per il segno abbiamo solo due possibilità, $+$ oppure $-$
- Per la mantissa abbiamo esattamente 2^{52} opzioni
- Per il valore di e abbiamo esattamente 2046 opzioni

Dato che lo shift ν non cambia il numero di combinazioni possibili, moltiplicando i dati sopra ottenuti si ottiene:

$$2 \times 2^{52} \times 2046 = 2^{53} \times 2046 = 18\,428\,729\,675\,200\,069\,632$$

1.3 Esercizio 3

Eseguire il seguente *script* Matlab:

```
        format long e
        n=75;
        u=1e−300;for i=1:n,u=u*2;end,for i=1:n,u=u/2;end,u
        u=1e−300;for i=1:n,u=u/2;end,for i=1:n,u=u*2;end,u
```

Spiegare i risultati ottenuti.

I risultati ottenuti:

```
u =
        1.000000000000000e−300

u =
        1.119916342203863e−300
```

L'obiettivo del programma proposto è verificare, dopo una serie di operazioni, che il numero restituito sia uguale a quello di partenza e che quindi non si abbia perdita di informazione.
Prima di tutto viene impostato il formato di output a *longE*, formato long decimal (15 cifre dopo la virgola) con notazione scientifica.

Alcune premesse:
In caso di underflow, ovvero quando un calcolo produce un valore più piccolo di *realmin*, in accordo con l'opzione dello standard IEEE, in MATLAB i numeri vengono denormalizzati, perdono quindi il primo 1 sottinteso e sono definiti dalla loro rappresentazione binaria. Inoltre, il più piccolo numero positivo denormalizzato è $0{,}494 \times 10^{-323}$. Ogni risultato più piccolo di questo è posto a zero.

Alcune notazioni importanti:

- *realmin* è il minimo numero normalizzato rappresentabile in MATLAB
- *realmax* è il massimo numero normalizzato rappresentabile in MATLAB
- *eps* è la precisione di macchina

I primi 2 for effettuano prima 75 moltiplicazioni per 2 del numero u e poi lo stesso numero di divisioni per 2 di u, mantenendone il valore al di sopra di *realmin* e facendo sì che non ci sia perdita di informazione (underflow). I for successivi invece, a causa dello svolgersi prima delle 75 divisioni e poi delle 75 moltiplicazioni, portano alla denormalizzazione del numero e quindi a una sua approssimazione, che risulta nel valore finale di u.

1.4 Esercizio 4

Eseguire le seguenti istruzioni Matlab:

```
        format long e
        a=1.111111111111111
        b=1.11111111111111
        a+b
        a-b
```

Spiegare i risultati ottenuti.

I risultati ottenuti:

```
a =
        1.111111111111111e+00

b =
        1.111111111111110e+00

ans =
        2.222222222222221e+00

ans =
        8.881784197001252e-16
```

L'obiettivo del programma proposto è verificare se i risultati dati da operazioni di somma e sottrazione siano soggetti a cancellazione numerica o meno.

Innanzitutto, come nel precedente esercizio, viene impostato il formato di output a $longE$, formato long decimal (15 cifre dopo la virgola) con notazione scientifica. Dopodichè, osservando gli output ottenuti, l'operazione di somma risulta ben condizionata, mentre l'operazione di sottrazione di due numeri molto vicini tra loro risulta invece mal condizionata. Tale malcondizionamento deriva dal fatto che in macchina la semplice operazione $a - b$ viene effettuata come $fl(fl(a) - fl(b))$, dove con $fl(x)$ indichiamo il valore floating point del numero x.
Dunque, il numero $a - b = realmin + eps \times 4$, mentre invece dovrebbe essere 10^{-15}. Calcolando i vari valori floating, otteniamo:

$$fl(a) = 1{,}111\,111\,111\,111\,110\,938\,409\,751\,724\,98$$

$$fl(b) = 1{,}111\,111\,111\,111\,110\,050\,231\,332\,024\,85$$

$$fl(a) - fl(b) = 8{,}881\,784\,197\,001\,300 \times 10^{-16}$$

$$fl(fl(a) - fl(b)) = 8{,}881\,784\,197\,001\,252 \times 10^{-16}$$

Abbiamo così ricostruito l'output restituito da Matlab, che è frutto di una propagazione dell'errore relativa alla rappresentazione dei numeri decimali in macchina.