Relazione 01:
Errore algoritmico e numerico

Cafasso Dario
De Maria Giuseppe
De Rosa Carmine
Pellecchia Pietro

June 14, 2018

Chapter 1

Tipi di errore

Una volta studiate le caratteristiche che distinguono i vari tipi di rappresentazioni dei numeri interi e reali, considerando il fatto che la memoria del computer è finita, abbiamo scoperto che fissato il numero di byte della nostra rappresentazione esistono delle limitazioni. Queste consistono in:

- Errori di Over-, Under-flow
- Precisione della rappresentazione dei numeri reali

1.1 Errori di Over-, Under-flow

1.1.1 Huge

Sia per le variabili di tipo integer che per quelle di tipo real esiste un numero massimo rappresentabile. La sua individuazione è possibile tramite l'esecuzione di un programma che, utilizzando il comando Huge, ne restituisce il valore su schermo. Una volta superato questo valore, il compilatore non segnala errore e le reazioni dei due tipi di variabili sono diverse. Queste consistono in:

- Per i numeri interi, a causa della ciclicità della rappresentazione, si ottengono numeri negativi. Più precisamente, il numero immediatamente successivo allo Huge (massimo della rappresentazione) è il numero negativo più piccolo rappresentabile;
- Per i numeri reali, a causa dell'esistenza di un massimo esponente raggiungibile dalla rappresentazione, il compilatore considera il suo superamento come "infinity".

Per verificare quanto detto, abbiamo scritto un programma che determina il valore dello Huge della rappresentazione in maniera approssimata e lo confronta con quello dato dal comando fortran prima citato e il cui risultato è riportato nella tabella 1.1. Per i reali l'avvicinamento avviene entro un fattore 10 e poi

2. Avvicinandosi tramite un fattore 2 ci aspettavamo che l'ultimo valore stampato fosse proprio lo huge, ma si è osservato che questo era circa uguale a più della metà dello huge a causa dell'errore di troncamento che si propaga sulle ultime cifre. Diversamente, avvicinandosi tramite un fattore 10 si è osservato che l'ultimo valore stampato era circa uguale allo huge (e non di un ordine di grandezza in meno come ci aspettavamo) a causa dello stesso errore. I valori ottenuti sono quelli in tabella 1.2. Invece, per gli interi, l'avvicinamento avviene sommando uno in maniera iterativa alla variabile intera considerata. Ovviamente il massimo valore corrispondeva a quello dello Huge e il successivo era il minimo dei negativi.

	Real	Integer
4 Byte	3.40282347E+38	2147483647
8 Byte	1.7976931348623157E+308	9223372036854775807

Table 1.1: Valori dello Huge rispettivamente per variabili reali e intere a 4 e 8 byte.

	Fattore 2	Fattore 10
4 Byte	1.70141183E+38	1.00000007E+38
8 Byte	8.9884656743115795E+307	9.9999999999999981E+307

Table 1.2: Valori dello Huge di una variabile real
ottenuti tramite approssimazione rispettivamente a 4 e 8 byte.

1.1.2 Tiny

Analogamente a quanto visto per lo Huge, per i numeri reali esiste un numero minimo rappresentabile in modulo. La sua individuazione è possibile tramite l'esecuzione di un programma che, utilizzando il comando Tiny, ne restituisce il suo valore su schermo. Una volta oltrepassato questo valore, il compilatore non segnala errore e le reazione è quella di restituire zero oppure di usare una rappresentazione detta subnormale. Quest'ultima restituisce valori al di sotto del vero valore di Tiny, perdendo l'errore relativo costante. Provando a stimarlo avvicinandoci tramite fattori 0,5 e 0,1 abbiamo ottenuto i dati riportati in tabella 1.3.

	Fattore 0,5	Fattore 0,1	Valore di Tiny
4 Byte	1.40129846E-45	1.40129846E-45	1.17549435E-38
8 Byte	4.9406564584124654E-324	9.8813129168249309E-324	2.2250738585072014E-308

Table 1.3: Valori del Tiny di una variabile real
ottenuti tramite approssimazione rispettivamente a 4 e 8 byte.

1.2 Precisione della rappresentazione dei numeri reali

Il tipo di rappresentazione che utilizziamo per i numeri reali ha come conseguenza l'esistenza di un errore relativo che tiene conto della possibilità delle ultime cifre della mantissa di variare. Ciò comporta che la nostra rappresentazione possiede un numero finito di cifre significative e di conseguenza l'esistenza di un numero diverso da zero che sommato ad uno restituisce uno, in quanto minore dell'errore relativo del nostro numero. Il numero più grande che gode di questa proprietà può essere trovato tramite il comando fortran epsilon. Analogamente a quanto fatto in precedenza, abbiamo scritto un programma che, oltre a stampare il valore di epsilon, vi si avvicina stampando ogni volta la somma di 1 e di una variabile x dimezzata ad ogni iterazione. I valori ottenuti sono riportati in tabella 1.4. Volendo invece conoscere il numero di cifre significative della nostra rappresentazione, possiamo utilizzare il comando fortran precision. I valori restituiti da quest'ultimo per variabili a 4 e 8 Byte sono riportati in tabella 1.5.

	Fattore 0,5	Valore di epsilon
4 Byte	5.96046448E-08	1.19209290E-07
8 Byte	1.1102230246251565E-016	2.2204460492503131E-016

Table 1.4: Valori del epsilon di una variabile real
ottenuti tramite approssimazione rispettivamente a 4 e 8 byte.

	Valore di precision
4 Byte	6
8 Byte	15

Table 1.5: Valori del precision di una variabile real
ottenuti tramite approssimazione rispettivamente a 4 e 8 byte.

Chapter 2

Serie numeriche

2.1 Sommare una serie oscillante 1

Determinare il valore di e^{-x} per $x = 0.1,\ 1,\ 10$ utilizzando la serie $e^{-x} = \sum_{n=0}^{\infty} \frac{(-x)^n}{n!}$ con una precisione di 10^{-8} prendendo il valore fornito dal comando Fortran exp(-x) come valore vero. Per fare ciò utilizziamo tre metodi:

- Sommando i termini della serie così come sono;
- Sommando i termini ottenuti in maniera iterativa come $a_n = a_{n-1} \cdot (\frac{-x}{n})$ e inizializando $a_0 = 1$;
- Considerando $e^{-x} = \frac{1}{e^x}$ e calcolando prima e^x, utilizzando il metodo precedente con x al posto di -x, e poi facendone il reciproco.

Il primo metodo è fallimentare a causa della crescita fuori controllo del valore della variabile fattoriale incorrendo in un errore di Overflow. Il secondo metodo funziona con una precisione di 10^{-8}, ma è fallimentare con una precisione richiesta a 10^{-15} a causa della presenza di differenze catastrofiche. Il terzo metodo invece è preciso anche a 10^{-15}. I risultati ottenuti sono riportati in tabella 2.1.

valori di x	Primo metodo	Secondo metodo	Terzo metodo
0.1	0.90483741668764672	0.90483741668764672	0.90483741668764672
1	0.36787943434218784	0.36787944117144245	0.36787944117144228
10	-139538777053.55316	13.396865995695904	4.5472151391750421E-005

Table 2.1: Risultati dell'esercizio 1

2.2 Serie oscillante 2

Determinare il valore della serie $\sum_{n=1}^{2N} (-1)^n \frac{n}{n+1}$ in funzione di N (tra 100,000 e 1,000,000 con passo 100,000 per 4B, da determinare in funzione dell'analisi per 8B) utilizzando tre metodi:

- Il metodo 1 dell'esercizio precedente;
- Separando i contributi pari da quelli dispari;
- Ricombinandoli in un'unica serie non oscillante.

Il primo metodo è impreciso a causa del carattere della serie, la quale, essendo oscillante, genera differenze catastrofiche. Il secondo metodo è efficace, dato che il risultato è la differenza tra due serie a termini positivi, ma meno del terzo, in cui la serie è tornata una sola semplificando alcuni contributi. I risultati ottenuti sono riportati nelle tabelle 2.2 e 2.3.

valori di N	Primo metodo	Secondo metodo	Terzo metodo
100,000	0.306859195	0.306859195	0.306800693
200,000	0.306860864	0.306860864	0.306800693
300,000	0.306861401	0.306861401	0.306800693
400,000	0.306861758	0.306861758	0.306800693
500,000	0.306861997	0.306861997	0.306800693
600,000	0.306862056	0.306862056	0.306800693
700,000	0.306862116	0.306862116	0.306800693
800,000	0.306862235	0.306862235	0.306800693
900,000	0.306862295	0.306862295	0.306800693
1,000,000	0.306862354	0.306862354	0.306800693

Table 2.2: Risultati dell'esercizio 2 a 4 Byte

valori di N	Primo metodo	Secondo metodo	Terzo metodo
100,000	0.30685031945876495	0.30685031945876495	0.30685031945880281
200,000	0.30685156944468650	0.30685156944468650	0.30685156944474207
300,000	0.30685198610873110	0.30685198610873110	0.30685198610880682
400,000	0.30685219444115475	0.30685219444115475	0.30685219444122647
500,000	0.30685231944074542	0.30685231944074542	0.30685231944080354
600,000	0.30685240277383152	0.30685240277383152	0.30685240277390713
700,000	0.30685246229750773	0.30685246229750773	0.30685246229757807
800,000	0.30685250694027266	0.30685250694027266	0.30685250694034594
900,000	0.30685254166239928	0.30685254166239928	0.30685254166250631
1,000,000	0.30685256944014638	0.30685256944014638	0.30685256944024097

Table 2.3: Risultati dell'esercizio 2 a 8 Byte

2.3 Serie 3

Determinare il valore della serie $\sum_{n=1}^{N} \frac{1}{n}$ in funzione di N (tra 100,000 e 1,000,000 con passo 100,000 per 4B, da determinare in funzione dell'analisi per 8B) sommando sia partendo da n = 1 per numeri crescenti sia partendo da n = N per numeri decrescenti. Effettuare il calcolo in entrambi le rappresentazioni reali a 4B e 8B. Le prime 6÷7 cifre del risultato nella rappresentazione a 8B può essere considerato il valore vero in rapporto al risultato nella rappresentazione a 4B. Il primo metodo risulta meno efficiente a causa del fatto che sommando dai termini più grandi ai più piccoli, gli ultimi potrebbero non contribuire alla somma una volta diventati minori di ε_m. Il secondo metodo si dimostra essere il migliore riuscendo a tenere conto anche di questi termini. I risultati dei 2 metodi in real 4 e 8 sono riportati nelle tabelle 2.4 e 2.5 .

valori di N	Primo metodo	Secondo metodo
100,000	12.0908508	12.0901527
200,000	12.7827568	12.7833014
300,000	13.1953249	13.1887655
400,000	13.4814272	13.4764490
500,000	13.6906919	13.6996069
600,000	13.8814268	13.8818817
700,000	14.0712557	14.0361919
800,000	14.1666231	14.1697330
900,000	14.2619905	14.2874250
1,000,000	14.3573580	14.3926516

Table 2.4: Risultati dell'esercizio 3 a 4 Byte

valori di N	Primo metodo	Secondo metodo
100,000	12.090146129863335	12.090146129863408
200,000	12.783290810429820	12.783290810429605
300,000	13.188755085205663	13.188755085205598
400,000	13.476436740991026	13.476436740991106
500,000	13.699580042305627	13.699580042305525
600,000	13.881901432432876	13.881901432432954
700,000	14.036051993212334	14.036051993212640
800,000	14.169583296550860	14.169583296551508
900,000	14.287366262762870	14.287366262763486
1,000,000	14.392726722864989	14.392726722865772

Table 2.5: Risultati dell'esercizio 3 a 8 Byte