Calcolo di superfici con il metodo Monte Carlo

Soel Micheletti

Il seguente contributo di Soel Micheletti è una sintesi rielaborata del suo lavoro di maturità intitolato *Generazione di numeri casuali e applicazioni in informatica con il Metodo Monte Carlo* svolto sotto la guida delle docenti Enrica Cavalli-Petraglio e Karima Pabst presso il Liceo cantonale di Locarno nel 2016. Il lavoro, presentato nella primavera del 2017 al concorso Scienza e Gioventù, ha ottenuto il premio speciale *Odd Fellows*, che aveva permesso a Soel di partecipare, nel gennaio 2018, alla *Taiwan International Science Fair* di Taipei, manifestazione nella quale è pure stato premiato. Il testo integrale del lavoro di maturità di Soel può essere richiesto all'autore scrivendo a soel.micheletti@hotmail.it.

Questo articolo aspira a coinvolgere il lettore mostrandogli come matematica e informatica siano due discipline complementari che si sviluppano nella stessa direzione: da un lato infatti l'informatica poggia le sue basi sulla matematica e non esisterebbe senza alcune idee straordinarie che hanno permesso di arrivare ai potenti strumenti che oggi abbiamo a disposizione; dall'altro la matematica non può più prescindere dall'informatica, che permette di risolvere molto velocemente problemi che risultavano impossibili agli occhi di illustri matematici del passato.

Le varie parti di questo lavoro sono strettamente collegate e si è cercato di mettere in correlazione le varie idee esposte, rendendo così più chiaro e allo stesso tempo più coinvolgente l'intero discorso.

La prima sezione presenta un esempio molto intuitivo che vuole stuzzicare la fantasia del lettore mettendo sul piatto le questioni fondamentali del lavoro. In seguito tali punti vengono analizzati in maniera più precisa, per arrivare infine a un risultato che considero innovativo, ovverosia l'applicazione del Metodo Monte Carlo per approssimare le aree di laghi e ghiacciai partendo sia da coordinate cartesiane, sia da immagini satellitari.

1. Un esempio pratico e intuitivo

Nel 1928 il matematico austriaco Richard von Mises propose una nuova — per molti versi rivoluzionaria — concezione di *calcolo delle probabilità* e nel suo lavoro *Wahrscheinlichkeit, Statistik und Wahrheit* scrisse:

> *Von Wahrscheinlichkeit kann erst gesprochen werden, wenn ein wohlbestimmtes, genau umgrenztes Kollektiv vorliegt.*
> *Kollektiv ist eine Massenerscheinung oder ein Wiederholungsvorgang, bei Erfüllung von zwei Forderungen, nämlich: es müssen die relativen Häufigkeiten der einzelnen Merkmale bestimmte Grenzwerte besitzen und diese müssen umgeändert bleiben, wenn man durch willkürliche Stellenauswahl einen Teil der Elemente aus der Gesamtheit heraushebt.*[1]

von Mises proponeva dunque come soluzione ai problemi della *probabilità classica* una nuova definizione di probabilità basata sul concetto di limite. La sua idea si riassume con l'equazione (1), che indica come calcolare la probabilità di un evento E:

$$p(E) = \lim_{n\to\infty} \frac{n_E}{n} \tag{1}$$

dove n_E è il numero di casi in cui l'evento E si realizza e n è il numero totale di casi.

Questa teoria prese il nome di *probabilità frequentista* e fu unanimamente respinta dai grandi matematici contemporanei a von Mises in quanto da un lato entrava ed entra tuttora in contrasto con la definizione di limite e dall'altro il tempo impiegato per calcolare la probabilità di un evento era troppo alto. Infatti è assolutamente impensabile lanciare una moneta dieci miliardi di volte per controllare se la probabilità dell'evento *esce testa* è del 50%. Con un lancio al secondo ci vorrebbero infatti più di 317 anni. Con l'avvento di calcolatori estremamente efficienti, l'idea di probabilità frequentista può essere rivalutata in quanto ripetere un esperimento miliardi di volte non è più un'utopia. Questo non significa che si possa risolvere qualsiasi problema con questo approccio (ad esempio è impossibile disputare miliardi di campionati di hockey per poter calcolare la probabilità di vittoria di una squadra come l'Ambrì-Piotta), ma l'esempio esposto nelle righe seguenti — che ha come scopo principale quello di mostrare una prima applicazione pratica in cui si utilizzano i *numeri aleatori* — si basa proprio sull'idea di von Mises.

Esempio 1.1. Si consideri un quarto di cerchio di raggio r inscritto in un quadrato di lato r. La geometria elementare insegna che il rapporto ρ tra l'area del

[1] Tradotto liberamente in italiano:

> *Si può parlare di probabilità solo riferendosi a un insieme ben determinato e descritto in modo preciso, che può essere un fenomeno di massa o un processo ripetibile a piacere che soddisfa le due condizioni seguenti: in primo luogo, le frequenze relative delle singole osservazioni nell'insieme devono avere limiti definiti e, in secondo luogo, questi limiti devono rimanere invariati per ogni sottosuccessione di elementi scelta nell'insieme.*

Quanto citato corrisponde ai primi due punti della definizione di probabilità, che si può leggere per intero in [32] alle pagine 28–29.

quarto di cerchio e l'area del quadrato è descritto dall'equazione (1.1):

$$\rho = \frac{\text{area del quarto di cerchio}}{\text{area del quadrato}} = \frac{\frac{\pi}{4}r^2}{r^2} = \frac{\pi}{4}, \qquad \text{dove } \pi \approx 3.14159. \tag{2}$$

Osservazione 1.2. Siano N un insieme finito di punti che "coprono" il quadrato (bordo incluso) e sia $M \subset N$ il sottoinsieme dei punti appartenenti al quarto di cerchio (bordo incluso). Sulla base di considerazioni intuitive possiamo approssimare il rapporto descritto dall'equazione (1.1) nel modo seguente:

$$\frac{\operatorname{card} M}{\operatorname{card} N} \approx \rho = \frac{\pi}{4} \tag{3}$$

dove con card X indichiamo la cardinalità dell'insieme X.

In questo paragrafo si cerca di approssimare il valore di π con due esperienze che hanno un duplice scopo:

- da un lato vogliono alimentare nel lettore la curiosità di scoprire se esistano metodi efficaci per generare aleatoriamente un gran numero di punti da sostituire nell'equazione (3) in modo da ottenere un'*approssimazione probabilistica* del valore di π;
- dall'altro vogliono mostrare un primo esempio di come i numeri aleatori possano essere utilizzati per calcolare qualcosa di assolutamente non casuale, come appunto il valore di π. I metodi utilizzati in questo lavoro fanno parte di una grande famiglia che prende nome di *metodo Monte Carlo*.

Il primo punto sarà il filo conduttore della prima parte di questo articolo, in cui verranno dapprima presentati diversi modi per generare numeri casuali (paragrafo 2) che in seguito saranno valutati con dei test statistici in modo da capire quali generatori siano abbastanza performanti e affidabili da poter essere utilizzati nelle simulazioni (paragrafo 3).

Il secondo punto sarà sviluppato in seguito. Nel paragrafo 4, dopo brevi cenni storici, sono descritti — non senza alcuni esempi — le proprietà e gli algoritmi fondamentali del metodo Monte Carlo. Il paragrafo 5 è invece il fulcro di questo articolo, infatti vengono presentati due approcci su come utilizzare i numeri casuali e il metodo Monte Carlo per approssimare l'area di un lago (il lago Sascòla) e come generalizzare il risultato in modo da sviluppare un software in grado di approssimare il valore dell'area di qualunque superficie.

1.1. Due esperienze diverse per approssimare il valore di π

1.1.1. Il lancio della moneta

Per l'approssimazione di π si consideri un quadrato di lato 127 unità in cui è inscritto un quarto di cerchio di raggio 127. L'algoritmo 1.3 illustra come generare dei punti casuali all'interno del quadrato. Questi sono utili per ottenere il valore approssimato di π.

Algoritmo 1.3. Approssimazione di π.

(1) Si lanci 7 volte la moneta. Si scriva 0 se esce testa, 1 se esce croce.

(2) Si converta la sequenza di 0 e 1 in base decimale. Il valore trovato, che è compreso tra 0 e 127, coincide con l'ascissa x del punto.

(3) Si lanci 7 volte la moneta. Si scriva 0 se esce testa, 1 se esce croce.

(4) Si converta la sequenza di 0 e 1 in base decimale. Il valore trovato, che ancora una volta è compreso tra 0 e 127 coincide con l'ordinata y del punto.

(5) Si ripetano i passi (1)–(4) per generare altri punti.

È necessario ripetere l'esperimento molte volte in modo da avvicinarci all'idea di von Mises. La figura 1 mostra graficamente i risultati ottenuti nella generazione di 99 punti: 77 punti si trovano all'interno del cerchio, mentre 22 sono fuori dal bordo del cerchio.

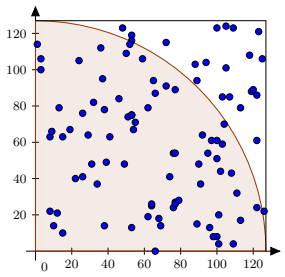

FIGURA 1. *Rappresentazione grafica dei punti ottenuti con il lancio della moneta.*

Inserendo i dati trovati nell'equazione (3) si trova la seguente *approssimazione probabilistica* di π:

$$\rho = \frac{77}{99} \cdot 4 = 3.\overline{1}\,. \tag{4}$$

L'errore rispetto al risultato teorico è dello 0.98%.

Attenzione però: ci si immagini di svolgere l'esperimento in un quadrato di lato 1 come quello della figura 2.

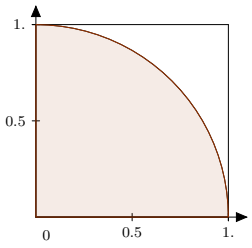

FIGURA 2. *Approssimazione di π in un quadrato di lato* 1.

Se non si considerano i numeri decimali (come nell'algoritmo 1.3), ipotizzando di avere un generatore di numeri casuali perfetto e di poter generare un numero infinito di punti, in questa situazione si potrebbero generare soltanto i punti $(0,0)$, $(0,1)$, $(1,0)$, $(1,1)$. L'approssimazione di π sarebbe dunque $\frac{3}{4} \cdot 4 = 3$.

Bisogna infatti tenere conto non solo del numero di punti generati, ma anche di come essi si distribuiscono all'interno del quadrato. Nel caso del lancio della monetina non si considerano i numeri decimali, dunque il rapporto ρ tra i punti all'interno del cerchio e quelli totali non è $\frac{\pi}{4}$, bensì un altro valore che si avvicina a π solo quando si utilizza un quadrato con lato molto lungo. Si osservi che utilizzare un quadrato di lato infinito porterebbe allo stesso risultato ottenuto considerando i numeri decimali.

1.1.2. Una simulazione virtuale

Per l'approssimazione di π con il computer si procede analogamente a come illustrato con il lancio della monetina. In questo caso, però, sarà un programma informatico (scritto in JAVA) a generare i punti (x, y) e a determinare se tali punti si trovano all'interno del cerchio, ossia se soddisfano

$$x^2 + y^2 \leq 127^2. \tag{5}$$

Algoritmo 1.4. Approssimazione di π.

```
1: VARIABILI
2:     dentro: numero di punti all'interno del cerchio;
3:     fuori: numero di punti fuori dal cerchio;
4:     x, y: numeri interi tra 0 e 127 generati casualmente;
5:     pigreco: valore approssimato di π;
6:     ripetizioni: numero di ripetizioni dell'esperimento;
7:     i: contatore;
8: i := 0;
9: repeat
10:    genera casualmente x e y;
11:    if x^2 + y^2 ≤ 127 then
12:        dentro = dentro + 1;
13:    else
14:        fuori = fuori + 1;
15:    i := i + 1;
16: until i = ripetizioni;
17: OUTPUT: pigreco = 4 · dentro/(dentro + fuori);
```

Implementando l'algoritmo 1.4 in JAVA si ottiene, per esempio, $\pi \approx 3.16$ per $N = 99$ (dunque con un errore dello 0.59%) e $\pi \approx 3.14158426$ per $N = 10^8$ (in questo caso l'errore si è ridotto allo 0.00027%).

1.2. Alcune considerazioni

Dopo questo semplice esempio si possono già trarre alcune considerazioni interessanti:

- esistono vari modi per generare numeri aleatori, come il lancio di una moneta o il generatore di libreria di JAVA.

- le risorse informatiche che abbiamo a disposizione permettono di generare numeri aleatori in modo molto più rapido rispetto a metodi più tradizionali come il lancio di una moneta.
- i numeri aleatori possono essere utilizzati per fini pratici, come approssimare il valore di π.
- i risultati ottenuti con simulazioni Monte Carlo possono essere molto vicini al risultato teorico.

Nel seguito dell'articolo si cercherà di dare una risposta a questioni aperte di grande importanza:

(1) *esistono metodi per generare numeri aleatori migliori di altri?*

(2) *quali sono le proprietà di un buon generatore di numeri aleatori?*

(3) *un generatore di numeri aleatori può funzionare bene anche se non è veramente aleatorio?*

(4) *esiste il generatore perfetto di numeri aleatori?*

(5) *come si prepara una buona simulazione con il metodo Monte Carlo?*

(6) *che cosa si può simulare con il metodo Monte Carlo?*

(7) *quali sono i limiti del metodo Monte Carlo? E quali invece le qualità che lo rendono uno strumento preziosissimo nella ricerca scientifica attuale?*

2. Alcuni generatori di numeri casuali

Questo paragrafo presenta un viaggio nel mondo dei generatori di numeri casuali in cui, sulla base di un numero consistente di esempi, si vuole dare una prima risposta alle domande (1)–(3) del paragrafo precedente. Per approfondimenti si indicano al lettore [16], [3], [9] e [15].

Ci sono molti modi per generare numeri in modo non deterministico: dal lancio di un dado[2] all'estrazione di una carta da un mazzo da poker, passando per le sequenze di diverse giocate alla roulette. Queste attività garantiscono casualità (sempre che tutto sia perfettamente bilanciato), ma per scopi più squisitamente matematici — come ottenere delle realizzazioni indipendenti di variabili aleatorie distribuite uniformemente nell'intervallo $[0, 1[$ — si preferiscono metodi più veloci e per molti versi più efficaci. I generatori utilizzati nel mondo scientifico si suddividono oggi in due categorie principali: da un lato ci sono i *generatori fisici*, che sfruttano il carattere aleatorio di fenomeni naturali, dall'altro quelli *algoritmici*, che si basano invece sulle prestazioni di algoritmi particolarmente efficienti.

[2]Per i latini lanciare un dado era l'attività casuale per eccellenza. Il termine *aleatorio* deriva proprio dal latino *alea*, con cui si indicava il gioco dei dadi.

2.1. Metodi fisici

Nel paragrafo 1 si è visto un primo, rudimentale, esempio di generatore fisico. Si può generalizzare l'algoritmo 1.3 per ottenere sequenze di numeri casuali di lunghezza k con l'algoritmo 2.1

Algoritmo 2.1. Lancio di una moneta k volte.

(1) Lanciare una moneta.

(2) Se esce testa porre $z_i = 0$; se esce croce porre $z_i = 1$.

(3) Ripetere k volte i passi (1)–(2).

(4) Normalizzare la sequenza per ottenere il risultato nella base desiderata.

L'algoritmo 2.1 rappresenta l'esempio più classico di generatore fisico, ma è evidente che non sia del tutto affidabile in quanto entrano in gioco fattori come la distribuzione del peso nella moneta e l'abilità del tiratore.

Si preferiscono allora generatori fisici che sfruttano il carattere casuale di fenomeni fisici molto complessi che garantiscono imprevedibilità. La meccanica quantistica offre, anche grazie all'approccio probabilistico utilizzato per studiare tali fenomeni, una vasta gamma di possibilità. Ecco alcuni esempi.

- Il generatore introdotto da Frigerio e Clark[3] nel 1975 permise all'Argonne National Laboratory[4] di generare una sequenza di 2.5 milioni di bit estremamente performante nei test statistici. Essi utilizzarono un generatore di particelle α e un contatore che misurava il numero di decadimenti avvenuti ogni 20 ms. Veniva memorizzato uno 0 se il numero di decadimenti in tale intervallo di tempo era pari, un 1 se era dispari. Per eliminare una possibile distorsione dovuta al fatto che le probabilità degli eventi *il numero di decadimenti è pari* e *il numero dei decadimenti è dispari* non siano per forza di cose equivalenti, si analizzavano i risultati a due a due: se i due numeri erano uguali (ad esempio due zeri di fila) il risultato veniva scartato, mentre se i due numeri erano diversi veniva considerato soltanto il secondo.

- Particolari tipi di interferometri come quello ideato da cinque ricercatori dell'Università di Vienna che nel 2008 hanno costruito, utilizzato, testato e discusso un interferometro che — grazie a dei separatori chiamati *beam splitter*[5] — manda un fotone in una direzione con probabilità $\frac{1}{2}$ e nell'altra con la stessa probabilità. La figura 3 schematizza l'esperienza: il fotone verde, per effetto del *beam splitter*, ha il 50% di probabilità di finire nel contenitore rosso e altrettante di finire in quello blu.

[3] Vedi: https://arxiv.org/pdf/1412.0171.pdf.

[4] L'Argonne National Laboratory, istituito ufficialmente nel 1946, è uno dei più grandi e antichi laboratori nazionali di ricerca degli Stati Uniti.

[5] Un *beam splitter*, divisore di fascio, è un dispositivo ottico che divide un raggio di luce (*beam*) in due parti.

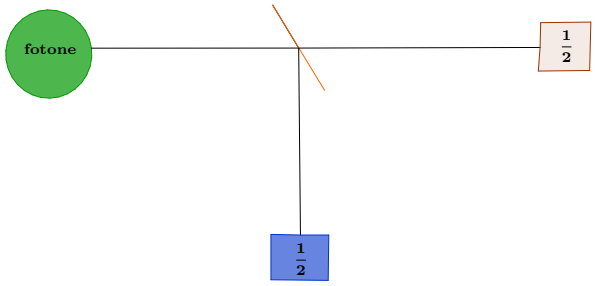

FIGURA 3. *Schema di interferometro con un beam splitter* 50 : 50.

- Il generatore proposto da alcuni dottorandi dell'Università di Ginevra sfrutta invece l'intervallo di tempo tra due manifestazioni del cosiddetto *shot noise*, un rumore quantistico causato dall'arrivo di fotoni in una fotocamera.
- Il tempo di decadimento radioattivo di nuclei di ^{85}Kr e ^{60}Co.
- Studio delle zone d'ombra nel caso di un raggio ionico attraverso una fenditura.

Un altro esempio, questa volta di natura non quantistica, è dato dalla desincronizzazione degli orologi (in inglese *clock drift*). Esso si basa sullo studio delle variazioni del rapporto tra le frequenze di risonanza di due orologi a cristallo. Tali variazioni sono dovute a variazioni di temperatura caratteristiche del materiale che compone l'orologio e sono pressoché imprevedibili. Nei casi in cui la differenza delle oscillazioni tra il cristallo più veloce e quello più lento sia dispari si può associare un uno, mentre se la differenza è pari si può associare uno zero.

2.2. Metodi deterministici

Per generare numeri aleatori si ricorre sempre più spesso ad algoritmi che generano una sequenza di numeri *apparentemente* senza alcuna correlazione tra loro. I metodi algoritmici si fanno preferire a quelli fisici per le seguenti ragioni:

- Sono facilmente computabili e occupano poca memoria. Computare un generatore fisico — andando così contro la definizione stessa di generatore fisico in quanto verrebbe meno la natura aleatoria del fenomeno — sarebbe estremamente dispendioso in termini di portabilità e memoria. La gestione delle variabili sarebbe infatti molto più complessa in quanto non basterebbero le operazioni elementari e di confronto necessarie ai metodi algoritmici.
- Le proprietà sono più facili da studiare. Dato un algoritmo, si possono effettuare test statistici (cfr. paragrafo 3) in grado di determinarne *pregi e difetti*. Il criterio base è *l'indistinguibilità statistica da un generatore veramente casuale*.
- Generare lunghe sequenze di numeri aleatori con un computer partendo da un algoritmo è un processo quasi istantaneo.

- Svolgere simulazioni con metodi fisici porrebbe problemi di ripetibilità. I metodi fisici presentati in questo paragrafo sono ripetibili, ma un leggero errore di misurazione potrebbe cambiare radicalmente il risultato di un esperimento. Si pensi ad esempio al generatore di Frigerio e Clarke: se avvenisse un decadimento in più in 20 ms, la sequenza finale cambierebbe in maniera considerevole.

In questo caso si parla tuttavia di *numeri pseudo-aleatori* poiché la sequenza è solo *apparentemente* casuale. In realtà infatti, una volta noto l'algoritmo generatore, si può determinare con esattezza ogni elemento della sequenza. Parlare di *casualità* vera e propria sarebbe tanto impreciso quanto controintuitivo.

2.2.1. Metodi ricorsivi

Per generare dei numeri pseudo-aleatori tra 0 e $m-1$ si considera l'insieme $\mathbb{Z}_m = \{0, 1, \dots, m-1\}$ e si scelgono $X_0 \in \mathbb{Z}_m$ e una funzione $f : \mathbb{Z}_m \longrightarrow \mathbb{Z}_m$. La successione (X_n), definita da

$$X_{i+1} = f(X_i), \qquad i \in \mathbb{N}, \tag{6}$$

sarà composta da numeri pseudo-aleatori.

Osservazione 2.2. Ad ogni sequenza generata con un metodo ricorsivo si può associare un *grafo rappresentativo* $G = (V, E)$.

Esempio 2.3. Siano $m = 10$ e

$$f(x) = 6x + 4 \pmod{10},$$

dove con $\pmod{10}$ si intende il resto della divisione per 10. Allora

$$\begin{cases} X_0 = 0 \\ X_{i+1} = f(X_i) = 6X_i + 4 \pmod{10}, \qquad i \in \mathbb{N}, \end{cases}$$

produce la sequenza $0, 4, 8, 2, 6, 0, 4, \dots$ Si può rappresentare graficamente tale sequenza come nella figura 4.

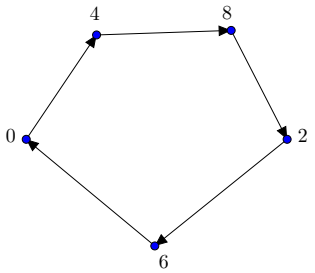

Figura 4. *Grafo rappresentativo della sequenza* 0, 4, 8, 2, 6, 0, 4, ...

È importante notare che qualunque sequenza generata con un metodo ricorsivo contiene, prima o poi, un ciclo di numeri che si ripetono. Per evitare di avere molte ripetizioni nella stessa sequenza esistono dei metodi per massimizzare la lunghezza del ciclo (detta *periodo*).

Algoritmo 2.4. Generazione di una sequenza priva di cicli.

(1) Scegliere $X_0 \in \mathbb{Z}_m$, $f : \mathbb{Z}_m \longrightarrow \mathbb{Z}_m$ e porre $i = 0$.

(2) Generare
$$\begin{aligned} X_{i+1} &= f(X_i) \\ X_{2i+1} &= f(X_{2i}) \\ X_{2i+2} &= f(X_{2i+1}) \end{aligned}$$

(3) Se $X_{i+1} = X_{2i+2}$ porre $L = i + 1$ e fermarsi, altrimenti tornare al passo (2) con i incrementato di 1.

Teorema 2.5. *La sequenza $X_0, \ldots, X_L$ generata con l'algoritmo 2.4 non contiene cicli.*

DIMOSTRAZIONE. Sia X_μ il primo elemento che si ripete nella sequenza.

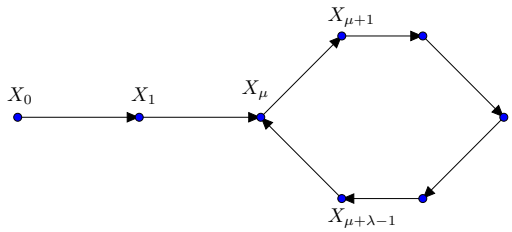

FIGURA 5.

Valgono:

$$\begin{aligned} X_{\mu+\lambda} &= X_\mu \\ X_{k+\lambda} &= X_k, \quad \forall k \geq \mu \\ X_{k+l\lambda} &= X_k, \quad \forall k \geq \mu, l \in \mathbb{N}. \end{aligned}$$

Esiste un k^* tale che

$$\mu \leq k^* \leq \mu + \lambda \text{ e } k^* = l \cdot \lambda$$

e per questo k^* vale

$$X_{2k^*} = X_{k^*+l\lambda} = X_{k^*}.$$

Ciò dimostra che la scelta di L è corretta. □

Per scegliere un buon generatore non è tuttavia sufficiente basarsi unicamente sulla lunghezza del suo periodo, ma la scelta di f deve essere oculata. L'esempio 2.6 mostra una sequenza con un periodo potenzialmente infinito che però — sulla base di considerazioni intuitive — non sembra per nulla casuale.

Esempio 2.6. Sia $m \in \mathbb{N}$, $m > 1$. Allora

$$\begin{cases} X_0 = 0 \\ X_{i+1} = f(X_i) = X_i + 1 \pmod{m} \end{cases}$$

genera una sequenza che non sembra per nulla casuale:

$$0, 1, 2, \ldots, m-1, 0, 1, \ldots$$

Gli esempi trattati finora suggeriscono che un buon generatore di numeri casuali deve fornire una sequenza con

- un periodo considerevolmente lungo;
- che non sia facilmente prevedibile (ad esempio numeri successivi, numeri pari successivi, approssimazioni di numeri irrazionali quali π o e, valori di funzioni quali sin o cos in intervalli definiti, ...).

La domanda fondamentale ora è: *come scegliere la funzione f?*

2.2.2. Metodo *middle-square*

Nel 1949 John Von Neumann, cercando di rispondere al quesito precedente, propose il seguente metodo, noto come *metodo middle-square*, che credeva potesse garantire ottime prestazioni.

Sia X_0 un numero intero positivo di k cifre, con k pari (il metodo funziona, con opportuni adattamenti, anche per k dispari). I termini

$$X_{i+1} := \left\lfloor \frac{X_i^2}{10^{k/2}} \right\rfloor \pmod{10^k}, \quad i \in \mathbb{N}, \tag{7}$$

dove con $\lfloor x \rfloor$ si intende il più grande intero $\leq x$, formano una sequenza di numeri aleatori interi positivi di al massimo k cifre.

L'applicazione del metodo può essere formulata con l'algoritmo 2.7.

Algoritmo 2.7. Metodo *middle-square*.

(1) Si prende un numero X qualunque di k cifre, k pari.

(2) Si calcola X^2 ottenendo un numero di al massimo $2k$ cifre (se il risultato ha meno di $2k$ cifre sarà necessario premettere un numero sufficiente di zeri).

(3) Si isolano le k cifre centrali.

(4) Si riparte dal passo (1) con le k cifre ottenute.

Esempio 2.8. Partendo dal numero 2345 bisogna tralasciare le prime e le ultime due cifre di $2345^2 = 05499025$. Resta così il numero 4990. Applicando l'algoritmo 2.7 otteniamo la seguente sequenza:

$$2345, 4990, 9001, 180, 324, 1049, 1004, 80, 64, 40, 16, 2, 0.$$

Il metodo permette di generare sequenze molto lunghe,[6] tuttavia molte scelte portano a risultati insoddisfacenti. Da notare che:

- se i primi $\frac{k}{2}$ valori del numero aleatorio sono degli zeri, i suoi successori decresceranno fino a zero.
- se le $\frac{k}{2}$ cifre di destra sono zeri (ma non quella appena precedente), tutta la sequenza avrà la stessa caratteristica. Ad esempio con $k = 4$: 3100, 6100, 2100, 4100, ...
- se le $\frac{k}{2} + 1$ cifre di destra sono zeri, i suoi successori decresceranno fino a zero.
- non si è mai riusciti a superare la lunghezza della sequenza massima di 8^k numeri.
- esistono valori iniziali che generano sequenze costanti, per esempio con $k = 4$: 0000, 0100, 2500, 3792 e 7600.

Il metodo ha sicuramente un potenziale importante, tuttavia non si conoscono in maniera sufficientemente approfondita le sue proprietà teoriche (anche se l'esperienza di molti studiosi suggerisce che il metodo tende ad avere molti punti fissi e piccoli cicli) e risulta difficile controllarlo. I metodi utilizzati in larga scala sono affini al *metodo lineare congruente*.

2.2.3. Metodo lineare congruente

Il *metodo lineare congruente*[7] è un metodo tanto semplice quanto utilizzatissimo, poiché le sue proprietà teoriche ed empiriche sono facili da determinare. Nel concreto è formato da

- un modulo $m \in \mathbb{N}$;
- un moltiplicatore $a \in \mathbb{N}, a < m$;
- un incremento $c \in \mathbb{N}$;
- un valore iniziale $X_0 \in \mathbb{N}$

ed è definito dalla ricorsione

$$X_{i+1} = aX_i + c \pmod{m}. \tag{8}$$

Esempio 2.9. Con $m = 20$, $a = 6$, $c = 4$ e $X_0 = 0$ si ottiene la sequenza

$$0, 4, 8, 12, 16, 0, 4, \ldots$$

Essa ha periodo 5, molto minore di m, dunque poco utile.

[6] D. E. KNUTH riporta in [15] un esperimento di Metropolis in cui, partendo da un numero di 38 cifre, si è ottenuta una sequenza di 750000 numeri che superavano buona parte dei test statistici.
[7] Il metodo fu ideato da D.H. Lehmer, pertanto talvolta è anche chiamato *metodo di Lehmer*.

Il teorema 2.10 garantisce l'esistenza, sotto opportune ipotesi, di sequenze di periodo massimo m.

Teorema 2.10. *Una sequenza generata con il metodo lineare congruente ha periodo m se e solo se*

(1) c e m sono coprimi;

(2) $a - 1$ è un multiplo di ogni fattore primo p di m;

(3) se m è un multiplo di 4 allora lo è anche $a - 1$.

Esempio 2.11. Si possono modificare i parametri dell'esempio 2.9 in modo da soddisfare i criteri del teorema 2.10 ponendo $a = 21$ e $c = 17$. Con $X_0 = 0$ si ottiene la sequenza di periodo massimo $m = 20$:

$$0, 17, 14, 11, 8, 5, 2, 19, 16, 13, 10, 7, 4, 1, 18, 15, 12, 9, 6, 3, 0, 17, \ldots$$

Osservazione 2.12. La sequenza di numeri aleatori $X_0, X_1, \ldots, X_n$ sia stata generata con il metodo lineare congruente. I termini della sequenza $U_0, U_1, \ldots, U_n$, definita da

$$U_i := \frac{(aX_i + c) \pmod m}{m}, \quad i = 0, 1, \ldots, n, \tag{9}$$

sono numeri nell'intervallo $[0, 1[$. Applicando la formula (9) all'esempio 2.11 si ottiene

$$0,\ 0.85,\ 0.7,\ 0.55,\ 0.4,\ 0.25,\ 0.1,\ 0.95,\ 0.8,\ 0.65,\ 0.5,\ 0.35,\ 0.2,$$
$$0.05,\ 0.9,\ 0.75,\ 0.6,\ 0.45,\ 0.3,\ 0.15,\ 0,\ 0.85,\ \ldots$$

Sia $X_0 \in \mathbb{Z}_m$ e siano $U_0, U_1, \ldots, U_n$ definiti dalla (9). Per ogni $t \in \mathbb{N}$, $t > 0$, denotiamo con u_k la t–tupla di termini successivi

$$u_k = (U_k, U_{k+1}, \ldots, U_{k+t-1}), \quad k \in \mathbb{N}.$$

Per esempio, con $t = 2$:

$$u_0 = (U_0, U_1), \quad u_1 = (U_1, U_2), \quad u_2 = (U_2, U_3), \quad \ldots;$$

con $t = 3$:

$$u_0 = (U_0, U_1, U_2), \quad u_1 = (U_1, U_2, U_3), \quad u_2 = (U_2, U_3, U_4), \quad \ldots$$

Indichiamo con T_t l'insieme di tutte le t–tuple sovrapposte di valori successivi partendo da un dato valore iniziale $X_0 \in \mathbb{Z}_m$:

$$T_t = \{u_k \mid k \geq 0, X_0 \in \mathbb{Z}_m\}.$$

È chiaro che se la sequenza (X_i) ha periodo massimo m allora esisterà un solo insieme T_t, indipendente dal valore X_0.

Rappresentando gli elementi degli insiemi T_t nell'ipercubo a t–dimensioni $[0, 1[^t$ otteniamo una struttura reticolare. Si può dimostrare che tutti i punti di T_t giacciono su un piccolo numero di iperpiani paralleli ed equidistanti. Chiaramente: più iperpiani paralleli sono necessari per ricoprire i punti, migliore sarà la distribuzione dei punti.

Esempio 2.13. La figura 6 mostra la struttura reticolare della sequenza generata nell'osservazione 2.12 nel caso $t = 2$. Poiché il periodo della sequenza è massimo ($m = 20$) esiste solo un insieme T_2, indipendente dal valore iniziale X_0.

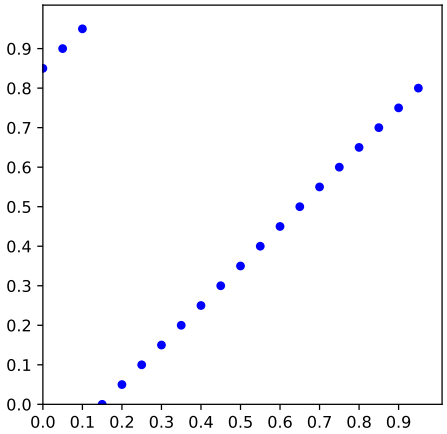

FIGURA 6. *Struttura reticolare della sequenza generata nell'osservazione* 2.12.

Esaminando la struttura reticolare rappresentata nella figura 6 appare ovvio che la sequenza abbia poco a che fare con la casualità poiché i punti sono disposti su due rette. Per ottenere una sequenza soddisfacente non basta prestare attenzione alle condizioni del teorema 2.10, ma la scelta dei parametri iniziali deve soddisfare anche altre condizioni.

Definizione 2.14. Siano $a \in \mathbb{Z}_m$, a e m coprimi. Si dice:

- *ordine moltiplicativo di a modulo m* il più piccolo numero intero positivo λ tale che $a^\lambda = 1 \pmod{m}$;
- *elemento primitivo modulo m* ogni a avente ordine massimo. L'ordine di un elemento primitivo si indica con $\lambda(m)$.

Esempio 2.15. Sia $m = 18$. Dalla tabella 1 segue che $\lambda(18) = 6$ e che 5 e 11 sono elementi primitivi modulo 18.

TABELLA 1. *Ordini moltiplicativi modulo* 18.

a	ordine	primitivo
1	1	no
5	**6**	sì
7	3	no
11	**6**	sì
13	3	no
17	2	no

Si può dimostrare che se la scomposizione in fattori primi di $m \in \mathbb{N}$, $m > 1$, è

$$m = p_1^{e_1} \cdot p_2^{e_2} \cdots p_k^{e_k},$$

dove $p_1, p_2, \ldots, p_k$ sono primi distinti ed $e_1, e_2, \ldots, e_k$ sono interi positivi, allora

$$\begin{aligned}
&\lambda(2) = 1; \qquad \lambda(4) = 2; \qquad \lambda(2^e) = 2^{e-2} \quad \text{se } e \geq 3; \\
&\lambda(p^e) = p^{e-1}(p-1) \quad \text{se } p > 2; \\
&\lambda(m) = \text{mcm}\{\lambda\left(p_1^{e_1}\right), \ldots, \lambda(p_k^{e_k})\}
\end{aligned}$$

(mcm denota il minimo comune multiplo).

I seguenti teoremi illustrano due risultati inerenti la lunghezza del periodo di una sequenza di numeri casuali generata con il metodo lineare congruente nel caso particolare $c = 0$.

Teorema 2.16. *La sequenza generata con il metodo lineare congruente definito da* (8) *con* $c = 0$ *ha il periodo massimo* $\lambda(m)$ *se e solo se*

(1) X_0 *e* m *sono coprimi.*

(2) a *è un elemento primitivo modulo* m*.*

Teorema 2.17. *Se* $m = 10^e$*,* $e \geq 5$*,* $c = 0$ *e* X_0 *non è multiplo di* 2 *o di* 5*, allora il periodo della sequenza generata con il metodo lineare congruente è* $5 \cdot 10^{e-2}$ *se e solo se* $a \pmod{200}$ *è uno dei 32 numeri seguenti:*

3, 11, 13, 19, 21, 27, 29, 37, 53, 59, 61, 67, 69, 77, 83, 91, 109, 117,
109 123, 131, 133, 139, 141, 147, 163, 171, 173, 179, 181, 187, 189, 197.

Esempio 2.18. La sequenza

$$X_{i+1} = 211 \cdot X_0 \pmod{10^5} \qquad \text{con } X_0 = 3 \tag{10}$$

normalizzata con l'equazione (9) ha la struttura reticolare in due dimensioni come quella della figura 7.

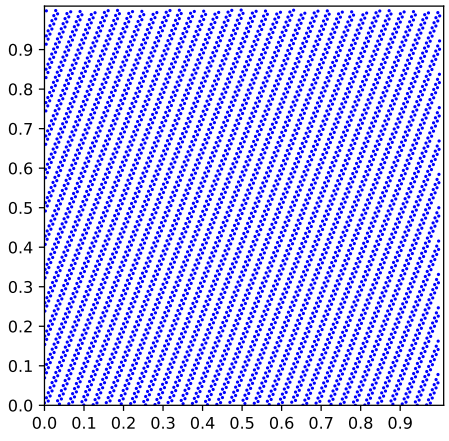

Figura 7. *Struttura reticolare della sequenza generata con l'equazione* (10).

Il reticolo rappresentato nella figura 7 è decisamente migliore rispetto a quello della figura 6. Tuttavia il metodo lineare congruente può essere ancora affinato per ottenere risultati ancora più performanti.

Definizione 2.19. Si chiama *potenza* di una sequenza con periodo massimo il più piccolo numero naturale s tale che

$$b^s = 0 \pmod{m} \tag{11}$$

dove $b := a - 1$ soddisfa le condizioni del teorema 2.10. Si può dimostrare che si può scrivere la sequenza in forma esplicita come

$$X_n = \frac{(a^n - 1)\, c}{b} \pmod{m} \tag{12}$$

che equivale a

$$X_n = c \left(\sum_{k=1}^{s} \binom{n}{k} b^{k-1} \right) \pmod{m}. \tag{13}$$

Donald Ervin Knuth[8] ha proposto, analizzando l'equazione (13), una scelta dei parametri in modo da ottenere risultati estremamente soddisfacenti. Per esempio nella figura 8 sono stati usati il valore iniziale $X_0 = 0$ e i parametri $m = 2^{31}$, $a = 314159265$ e $c = 453806245$. Si noti che nell'immagine di sinistra ci sono soltanto i primi 100 punti generati, mentre in quella di destra i primi 25000.

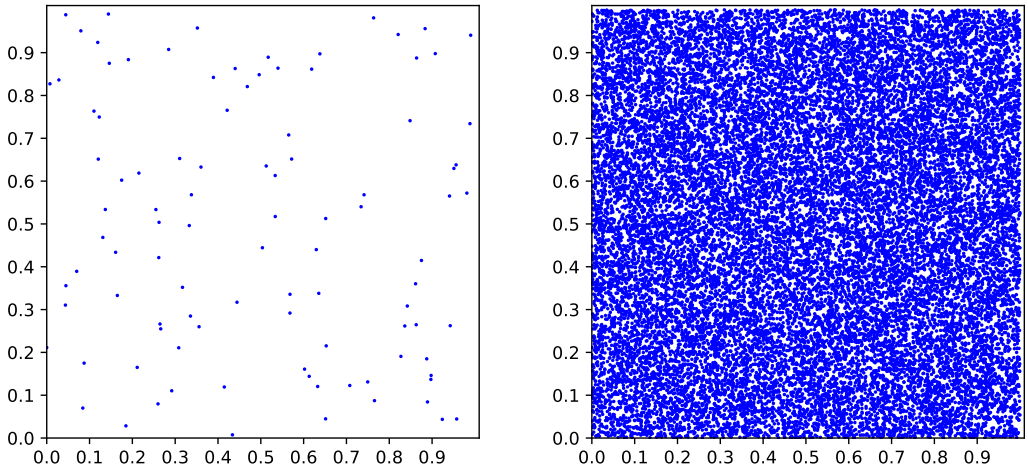

FIGURA 8. *Struttura reticolare della sequenza generata con i parametri di Knuth.*

Anche altri studiosi hanno trovato dei parametri interessanti per generare delle buone sequenze di numeri aleatori, per esempio:

metodo di	m	a	c
Goodman e Miller	$2^{31} - 1$	7^5	0
Gordon	2^{31}	5^{13}	0
Leormont e Lewis	2^{31}	$2^{16} + 3$	0

[8]D. E. KNUTH (1938) è un informatico statunitense che ha scritto una monumentale opera intitolata *The Art of Computer Programming*. Il secondo volume di tale opera, pubblicato per la prima volta nel 1981, si occupa proprio della generazione di numeri casuali e dei test statistici per valutarne le prestazioni. Nel 1974 è stato insignito del *Premio Turing*, il più prestigioso premio in informatica.

In questo paragrafo si è visto che la qualità di un generatore dipende da due grossi fattori: la *praticità* e il *comportamento statistico*. Infatti sono stati rigettati da un lato i generatori fisici poiché troppo laboriosi, e dall'altro metodi deterministici promettenti come il metodo *middle-square* perché non offrono la sicurezza statistica necessaria. Esistono dunque generatori di numeri aleatori migliori di altri; inoltre il fatto che un generatore sia solamente pseudo-aleatorio non pregiudica la sua utilità nelle simulazioni. Come si vedrà nel paragrafo 3 i generatori di numeri pseudo-casuali, vista la loro grande praticità e il loro ottimo comportamento statistico, sono proprio i generatori ideali per le simulazioni Monte Carlo.

3. Test di casualità

In questo paragrafo si vogliono completare le risposte ai quesiti (1)–(3) che sono stati già stati trattati sotto un altro punto di vista nel paragrafo 2.

Come visto precedentemente i metodi algoritmici si fanno preferire a quelli fisici ma, per far fronte alla mancata aleatorietà di tali metodi, è necessario che essi soddisfino alcune proprietà. Le quattro principali sono le seguenti:

- il periodo ρ deve essere sufficientemente lungo da permettere alla sequenza $(u_n, \ldots, u_{n+t-1})$, con $t < \rho$, di essere distribuita in maniera uniforme nell'intervallo $]0, 1[$.

- la sequenza deve interrompersi prima di entrare in un ciclo (la ripetizione di parti di una sequenza entra in contrasto con il concetto di aleatorietà).

- la struttura reticolare deve essere uniforme. Per valutare la qualità della struttura reticolare esistono test come il *test dell'analisi spettrale* o l'*analisi della trasformata di Fourier* che però non sono trattati in questo lavoro.

- deve esserci indipendenza tra valori vicini nella sequenza.

Ci sono inoltre altre proprietà di natura più strettamente pratica che è bene considerare:

- facilità di implementazione in un linguaggio di programmazione. Alte velocità di calcolo permettono di generare molti numeri in poco tempo.

- utilizzo di poco spazio in memoria affinché sia possibile intrecciare più generatori per ottenere algoritmi sempre più elaborati e sempre più vicini ad un generatore casuale.

- ripetibilità dell'algoritmo. Come già accennato nel paragrafo 2 e come verrà approfondito nel paragrafo 4, poter ripetere una simulazione porta a vantaggi in termine di affidabilità e trasparenza.

Per valutare le prestazioni dei generatori deterministici si possono effettuare dei *test statistici*, che indicano *pregi e difetti* di ciascun generatore. I test statistici si suddividono in due categorie: da un lato ci sono i *test teorici* che studiano le caratteriche teoriche della sequenza come la media, la varianza, la correlazione e la struttura reticolare. Dall'altro ci sono i *test empirici* che controllano se l'ipotesi

H_0 = *la sequenza generata non è distinguibile da una sequenza veramente casuale*

possa essere considerata vera. In realtà H_0 — visto che i test statistici si applicano solo a sequenze generate con il computer — è sempre falsa, tuttavia se il test non individua le caratteristiche deterministiche dell'algoritmo può essere considerata vera. Se l'ipotesi H_0 è considerata vera in una vasta gamma di test statistici per un dato generatore, significa che tale generatore è affidabile e può essere utilizzato nelle simulazioni Monte Carlo.

È tuttavia fondamentale tenere sempre presente alcuni aspetti:

- non esiste alcun generatore di numeri casuali in grado di ottenere risultati positivi in tutti i test.
- non esiste alcun test statistico in grado di dare una risposta netta e univoca alla domanda: *la sequenza generata è indistinguibile da una sequenza veramente casuale?*

Di seguito sono riportate alcune considerazioni riguardanti i risultati che vari metodi tendono ad ottenere nei test statistici:

- generatori con periodo inferiore a 2^{32} falliscono la maggior parte dei test.
- generatori che intrecciano varianti di metodo lineare congruente con periodo sufficientemente grande ottengono punteggi elevati nei test.
- generatori con una buona struttura reticolare ottengono punteggi elevati nei test.
- il punteggio ottenuto nei test è tanto migliore quanto più frequentemente vengono utilizzati artifici che complicano le ricorsioni o che fanno fare *salti* in alcuni punti della sequenza.

Tra i principali test statistici utilizzati per valutare la qualità di un generatore di numeri casuali si ricordano il *test del* χ^2, il *test di Kolmogorov–Smirnov*, il *test seriale*, il *test della curva di Gauss* (o *test del limite centrale*) e il *test di autocorrelazione*.

Un buon generatore di numeri casuali supera facilmente i test statistici citati, anche se i numeri generati sono solo pseudo-casuali. Il generatore perfetto di numeri aleatori purtroppo non esiste, ma i generatori che abbiamo a disposizione garantiscono buone prestazioni che permettono di ottenere risultati assai accurati con il metodo Monte Carlo; in particolare il generatore di numeri aleatori della libreria di JAVA dà buoni risultati.

4. Descrizione del metodo Monte Carlo

In questo paragrafo viene descritto il metodo Monte Carlo sia da un punto di vista prettamente teorico, sia con la presenza di algoritmi e di simulazioni preliminari. Lo scopo è quello di rispondere alle domande (5)–(7) del paragrafo 1, lasciando al lettore il piacere di completare la risposta alla domanda

> *Quali sono le qualità che rendono il metodo Monte Carlo uno strumento preziosissimo, nella ricerca scientifica attuale?*

e di verificarla sulle innovative applicazioni pratiche proposte nel paragrafo 5.

4.1. Cenni storici

Nel 1777 Georges-Louis Leclerc, conte di Buffon, si chiese come calcolare la probabilità P che un ago di lunghezza L, fatto cadere casualmente su un pavimento avente assi di larghezza d, si trovi su una linea tra due assi. Con molta fatica giunse alla soluzione descritta dalla seguente equazione:

$$P = \frac{2L}{\pi d}. \tag{14}$$

Qualche anno più tardi Pierre-Simon de Laplace fece notare che ripetere molte volte l'esperimento sarebbe stato un metodo per determinare un'approssimazione di π. Tuttavia, a quel tempo, lo studio dei numeri aleatori non aveva ancora raggiunto risultati importanti e approcciarsi al problema con lo studio del calcolo integrale appariva più veloce e più preciso. L'idea di generare numeri casuali per affrontare problemi di questo tipo si deve a William Thomson — conosciuto come Lord Kelvin — che generò molti numeri casuali estraendo carte da un mazzo per risolvere un problema di termodinamica. Nello stesso periodo William Sealy Gosset — conosciuto con lo pseudonimo di Mr. Student — ebbe la stessa idea per i suoi studi di statistica.

La nascita del metodo Monte Carlo risale però agli anni '40 del Novecento quando a Los Alamos, una piccola cittadina nello Stato del New Mexico, venne avviato il *Progetto Manhattan*, in cui una squadra di scienziati e matematici (tra cui Enrico Fermi, John von Neumann e Stanisław Marcin Ulam) progettò le tristementi famose bombe atomiche di Hiroshima e Nagasaki. Enrico Fermi aveva già utilizzato il metodo negli anni '30 per il suo studio sul *trasporto dei neutroni*, ma esso venne affinato insieme ai suoi colleghi proprio nell'ambito del *Progetto Manhattan*. La leggenda vuole che Ulam, appassionato di solitari, si pose la domanda sulla probabilità di vincere una partita ma, scoraggiato dalla complessità dei calcoli, cercò di combinare le riflessioni del *Progetto Manhattan* con la potenza di calcolo dell'ENIAC — un calcolatore rivoluzionario costruito nel 1946 — e diede così una sistemata quasi definitiva alla materia. Il nome *Monte Carlo* venne utilizzato per la prima volta da Nicholas Constantine Metropolis — e lo racconta nel suo articolo *The Beginning of the Monte Carlo Method*, facilmente reperibile in rete — ispirandosi ai casinò della nota città del Principato di Monaco. Il metodo

ebbe via via sempre maggiore fortuna e oggi viene applicato in diverse discipline quali la statistica, la fisica, la chimica, la biologia, l'economia e l'ingegneria.

4.2. Definizione

Il seguente brano di Ulam (vedi [31], p. 35) è significativo:

> *L'idea era di sperimentare migliaia di tali possibilità e, ad ogni stadio, selezionare a caso; in altre parole, facendo ricorso ad un numero casuale con una opportuna probabilità, si potrebbe indagare sulla sorte di certi tipi di eventi, in modo da seguire, per così dire, una successione lineare, anziché considerare tutte le ramificazioni. Dopo aver esaminato i possibili andamenti soltanto in qualche migliaio di casi, si avrebbe a disposizione un buon campione ed una risposta approssimativa al problema. Tutto ciò che occorre conoscere è la media degli andamenti dei campioni prodotti. Un tale procedimento si rivelò particolarmente adatto per essere eseguito automaticamente e la nascita dei calcolatori elettronici fu proprio una conseguenza di tale fatto.*

Il termine *metodo Monte Carlo* indica un insieme di algoritmi computazionali non parametrici che possono essere applicati su una vasta gamma di problemi in cui un approccio numerico-analitico risulta impreciso, troppo laborioso o impossibile. L'idea base è applicare metodi probabilistici a problemi deterministici, solitamente raccogliendo un gran numero di campioni casuali che seguono la distribuzione di probabilità del fenomeno da indagare. Se il metodo è stato scelto e applicato correttamente, la rielaborazione dei campioni ottenuti permette la risoluzione del problema con ottima approssimazione.

La legge dei grandi numeri assicura infatti che, se il numero di campioni è sufficientemente grande, il risultato della simulazione sarà molto vicino al risultato teorico, mentre il teorema del limite centrale suggerisce in linea di massima quanti campioni sono necessari per ottenere una certa accuratezza.

TABELLA 2. *Confronto tra metodo Monte Carlo e metodo sperimentale.*

Metodo Monte Carlo	Metodo sperimentale
Domanda iniziale	Domanda iniziale Ipotesi
Progettazione della simulazione	Ideazione di una o più esperienze
Definizione di un possibile dominio dei dati in input	Scelta delle grandezze da misurare
Generazione dei campioni casuali	Esperimento per verificare l'ipotesi
Analisi dei risultati	Analisi dei risultati
Risultato	Formulazione di una legge

Nella tabella 2 è proposta un'analogia tra il metodo Monte Carlo, che come già visto nel paragrafo 1 è diventato uno strumento molto potente soltanto con l'avvento di calcolatori efficienti, e il metodo sperimentale che è stato imprescindibile per la costruzione del pensiero scientifico occidentale.

I due metodi hanno molto in comune: in entrambi i casi bisogna porsi delle domande su un fenomeno non del tutto conosciuto e cercare di fare chiarezza con un esperimento (che ha più valore di qualsiasi formulazione teorica). La

differenza fondamentale sta nel fatto che per utilizzare Monte Carlo è richiesta la conoscenza della distribuzione del fenomeno in esame, mentre il metodo sperimentale si basa soltanto sull'osservazione e sulla misurazione di grandezze definite. Inoltre uno dei punti fermi del metodo sperimentale è la possibilità di ripetere l'esperimento ottenendo sempre lo stesso risultato.

Sarebbe auspicabile ottenere risultati identici anche negli esperimenti computazionali, tuttavia uno studio molto autorevole ha elencato alcune problematiche che riguardano la riproducibilità delle simulazioni virtuali:

- i ricercatori talvolta non riescono a riprodurre un esperimento eseguito da loro stessi;
- molti studiosi non riescono a riprodurre i risultati delle esperienze virtuali provenienti da studi eseguiti da altri;
- gli studenti non riescono a sottoporre alcuni loro problemi ai professori.

Le cause di questi problemi sono varie: computer diversi, software diversi, versioni diverse di un software, sistemi operativi diversi, virus, *bug*, mancanza di attenzione nell'archiviazione dei dati di ricerche già pubblicate, ricercatori che non vogliono mettere a disposizione i codici con cui hanno lavorato dopo la pubblicazione di un articolo, ... Le soluzioni sono assai difficili da trovare e dunque, quando possibile, è meglio utilizzare altri metodi. Tuttavia il metodo Monte Carlo può essere considerato un *metodo sperimentale potenziato* che sfrutta le possibilità offerte dai calcolatori e dal calcolo delle probabilità per ottenere, *"nel limite dell'errore sperimentale"*, risultati impossibili o estremamente difficili da ottenere in altri modi. Non a caso la facoltà di ingegneria informatica al Politecnico Federale di Zurigo ha scelto come motto: *Informatica, la scienza del XXI secolo*.

4.3. Numero di campioni

La domanda centrale da porsi prima di affrontare una simulazione con il metodo Monte Carlo è: *quanti campioni sono necessari per avere un'approssimazione soddisfacente?* Infatti, da un lato un numero troppo elevato di campioni porrebbe problemi in quanto le risorse computazionali, anche se di ottimo livello, hanno dei limiti, dall'altro un numero esiguo di campioni sarebbe sinonimo di poca accuratezza.

Sia θ il valore reale da approssimare con il metodo Monte Carlo. Bisogna generare N variabili aleatorie $X_1, \ldots, X_N$ con valore atteso θ e varianza σ^2. Come stimatore di θ utilizziamo la *media campionaria* delle variabili aleatorie generate

$$\overline{X}_N = \frac{1}{N} \cdot \sum_{k=1}^{N} X_k,$$

che ha valore atteso

$$\left(\overline{X}_N\right) = E\left(\frac{1}{N} \cdot \sum_{k=1}^{N} X_k\right) = \frac{1}{N} \cdot \sum_{k=1}^{N} E(X_k) = \frac{1}{N} \cdot N \cdot \theta = \theta \tag{15}$$

e varianza

$$\mathrm{Var}\left(\overline{X}_N\right) = \mathrm{Var}\left(\frac{1}{N}\cdot\sum_{k=1}^{N} X_k\right) = \frac{1}{N^2}\cdot\mathrm{Var}\left(\sum_{k=1}^{N} X_k\right) = \frac{N\cdot\sigma^2}{N^2} = \frac{\sigma^2}{N}. \tag{16}$$

Si desidera un errore massimo $\varepsilon > 0$ con probabilità $1-\delta$, $\delta \in [0,1]$. Dalla legge debole dei grandi numeri segue che per ogni $\varepsilon > 0$ vale

$$P(|\overline{X}_N - \theta| \le \varepsilon) \ge 1-\delta,$$

da cui

$$P\left(\left|\frac{(\overline{X}_N-\theta)\sqrt{N}}{\sigma}\right| \le \varepsilon\frac{\sqrt{N}}{\sigma}\right) \ge 1-\delta. \tag{17}$$

La quantità

$$Z_N \coloneqq \left(\overline{X}_N - \theta\right)\frac{\sqrt{N}}{\sigma}$$

è la media campionaria standardizzata relativa a $\overline{X}_N$ e dunque la (17) può essere riscritta come

$$P\left(|Z_N| \le \varepsilon\tfrac{\sqrt{N}}{\sigma}\right) = P\left(-\varepsilon\tfrac{\sqrt{N}}{\sigma} \le Z_N \le \varepsilon\tfrac{\sqrt{N}}{\sigma}\right) \ge 1-\delta. \tag{18}$$

Il teorema limite centrale afferma che

$$\lim_{N\to+\infty} P(a \le Z_N \le b) = P(a \le Z \le b)$$

dove $Z = \mathcal{N}(0,1)$ è la distribuzione normale standardizzata di valore atteso 0 e varianza 1. Se con Φ indichiamo la funzione di ripartizione della distribuzione normale $\mathcal{N}(0,1)$ allora avremo:

$$\begin{aligned} 1-\delta \le P\Big(-\varepsilon\tfrac{\sqrt{N}}{\sigma} \le Z_N \le \varepsilon\tfrac{\sqrt{N}}{\sigma}\Big) &\approx P\Big(-\varepsilon\tfrac{\sqrt{N}}{\sigma} \le Z \le \varepsilon\tfrac{\sqrt{N}}{\sigma}\Big) \\ &= \Phi\Big(\varepsilon\tfrac{\sqrt{N}}{\sigma}\Big) - \Phi\Big(-\varepsilon\tfrac{\sqrt{N}}{\sigma}\Big) = \Phi\Big(\varepsilon\tfrac{\sqrt{N}}{\sigma}\Big) - \Big[1-\Phi\Big(\varepsilon\tfrac{\sqrt{N}}{\sigma}\Big)\Big] \\ &= 2\Phi\Big(\varepsilon\tfrac{\sqrt{N}}{\sigma}\Big) - 1, \end{aligned}$$

da cui segue che il numero di campioni N richiesto per soddisfare quanto desiderato è

$$N \ge \left[\frac{\sigma}{\varepsilon}\cdot\Phi^{-1}\left(1-\frac{\delta}{2}\right)\right]^2. \tag{19}$$

La disuguaglianza di Bienaymé–Čebyšev aiuta a trovare il numero di campioni che permettono un risultato accurato (il cosiddetto *worse-case scenario*):

$$P\Big(|\overline{X}_N - \theta| \ge \varepsilon\Big) \le \frac{\mathrm{Var}(\overline{X}_N)}{\varepsilon^2} = \frac{\sigma^2}{N\cdot\varepsilon^2} \le \delta,$$

da cui

$$N \ge \frac{\sigma^2}{\delta\varepsilon^2}.$$

Esempio 4.1. Sia $\sigma^2 = 0.25$. Per ottenere un errore massimo $\varepsilon = 0.001$ nel 95% dei casi (ossia: $\delta = 0.05$) occorrono, nel caso di *worse-case scenario*,

$$N \ge \frac{0.25}{0.05\cdot 0.001^2} = 5\cdot 10^6 \text{ campioni}. \tag{20}$$

4.4. Calcolo di integrali con il metodo Monte Carlo

Sia $f : [a, b] \to \mathbb{R}$ una funzione integrabile nell'intervallo $[a, b]$. Si supponga di voler calcolare l'integrale definito

$$I = \int_a^b f(x)\ \mathrm{d}x. \tag{21}$$

Per valutare tale integrale con il metodo Monte Carlo si considera la distribuzione di probabilità $g(x)$ uniforme su $[a, b]$ definita da

$$g(x) = \begin{cases} \frac{1}{b-a} & \text{per } a \le x \le b, \\ 0 & \text{altrimenti.} \end{cases} \tag{22}$$

L'integrale dell'equazione (21) può essere riscritto come

$$I = (b-a) \int_a^b f(x)g(x)\ \mathrm{d}x. \tag{23}$$

Il valore atteso di $f(x)$ è

$$E(f(x)) = \int_{-\infty}^{+\infty} f(x)g(x)\ \mathrm{d}x = \int_a^b f(x)g(x)\ \mathrm{d}x$$

e, per la *legge dei grandi numeri*, $E(f(x))$ può essere approssimato con lo stimatore

$$\overline{F} = \frac{1}{N} \sum_{i=1}^{N} f(x_i), \tag{24}$$

dove $x_1, \ldots, x_N$ sono numeri casuali appartenenti all'intervallo $[a, b]$. Segue così:

$$I \approx (b-a) \cdot \overline{F}. \tag{25}$$

Esempio 4.2. Applichiamo l'equazione (25) al calcolo di $\int_0^\pi \sin(x)\ \mathrm{d}x$.

Dall'analisi si sa che il calcolo dell'integrale definito nell'intervallo $[0, \pi]$ della funzione $f(x) = \sin(x)$ equivale all'area sottesa al grafico. La primitiva di $\sin(x)$ è $\cos(x)$, da cui

$$I = \int_0^\pi \sin(x)\ \mathrm{d}x = -\cos(\pi) + \cos(0) = 2$$

L'approssimazione dell'integrale I con il metodo Monte Carlo è

$$I \approx \frac{\pi - 0}{N} \cdot \sum_{i=1}^{N} \sin x_i$$

e può essere calcolata implementando l'algoritmo 4.3. La media di I su 100 approssimazioni fatte con $N = 5 \cdot 10^6$ è stata 1.9999533972573689, con un errore relativo dello 0.0023%.

Algoritmo 4.3. Integrale di $\sin x$ tra 0 e π.

1: VARIABILI
2: `somma`: per calcolare la sommatoria;
3: $x_1, \ldots, x_N$: numeri casuali tra 0 e π;
4: **for** $i = 1, \ldots, N$ **do**
5: `somma` = `somma` + $\sin(x_i)$;
6: I = `somma` $\cdot (b-a)/N$;
7: OUTPUT: valore di I.

Osservazione 4.4. Questo primo metodo per approssimare gli integrali definiti funziona bene per "funzioni gentili" che non presentano troppe singolarità e che non variano molto nell'intervallo di integrazione. Per funzioni che ad esempio hanno piccole zone con un'area molto grande questo approccio risulta impreciso. Inoltre, per integrali unidimensionali semplici, i metodi numerici risultano molto più vantaggiosi in termini di precisione e velocità.

Nel resto del paragrafo sono presentati altri algoritmi di tipo Monte Carlo utili per approssimare integrali multipli del tipo

$$I = \int \cdots \int_D f(\vec{x}) \, \mathrm{d}x_1 \ldots \mathrm{d}x_n. \tag{26}$$

dove $D = [a_1, b_1] \times [a_2, b_2] \times \ldots \times [a_n, b_n] \subseteq \mathbb{R}^n$ è il dominio di integrazione della funzione $f : \mathbb{R}^n \to \mathbb{R}$ e $\vec{x} = (x_1, \ldots, x_n)$.

A titolo d'esempio confronteremo le approssimazioni con il valore dell'integrale multiplo

$$A := \int_0^3 \int_0^2 \int_0^6 \int_0^4 \int_0^5 \int_0^7 \int_0^3 x_1^2 x_2^6 x_3^4 x_4^3 x_5^5 x_6 x_7^2 \, \mathrm{d}x_1 \, \mathrm{d}x_2 \, \mathrm{d}x_3 \, \mathrm{d}x_4 \, \mathrm{d}x_5 \, \mathrm{d}x_6 \, \mathrm{d}x_7 \tag{27}$$

che ha il valore esatto 5928154283520000.

4.4.1. Metodo *media di un campione*

L'integrale (26) può essere riscritto come

$$I = (b_1 - a_1) \cdots (b_n - a_n) \cdot \int_{a_n}^{b_n} \ldots \int_{a_1}^{b_1} f(\vec{x}) \cdot g(\vec{x}) \, \mathrm{d}x_1 \ldots \mathrm{d}x_n \tag{28}$$

dove

$$g(\vec{x}) = \begin{cases} \dfrac{1}{(b_1 - a_1) \cdots (b_n - a_n)} & \text{se } \vec{x} \in D, \\ 0 & \text{altrimenti.} \end{cases}$$

Analogamente a quanto visto nel caso unidimensionale, il valore atteso di $f(\vec{x})$

$$E(f(\vec{x})) = \int_{a_n}^{b_n} \ldots \int_{a_1}^{b_1} f(\vec{x}) \cdot g(\vec{x}) \, \mathrm{d}x_1 \ldots \mathrm{d}x_n$$

può essere approssimato dallo stimatore

$$\overline{F} = \frac{1}{N} \sum_{i=1}^{N} f(\vec{x}_i), \tag{29}$$

dove $\vec{x}_1, \ldots, \vec{x}_N$ sono vettori casuali appartenenti al dominio D. Segue così:

$$I \approx (b_1 - a_1) \cdots (b_n - a_n) \cdot \overline{F}. \tag{30}$$

Esempio 4.5. L'algoritmo 4.3 è facilmente modificabile in modo da calcolare anche integrali multipli. La media dell'integrale A dell'esempio (27) su 100 approssimazioni fatte con $N = 5 \cdot 10^6$ è stata di 5929272317330399, con un errore relativo del -0.0189%.

4.4.2. Metodo *hit or miss*

Il *metodo hit or miss* è una generalizzazione del metodo utilizzato nel primo paragrafo per il calcolo del valore di π. Qui viene applicato nel caso particolare del calcolo dell'integrale definito

$$I = \int_a^b f(x)\, \mathrm{d}x,$$

dove f è integrabile e $f(x) \geq 0$ per ogni $x \in [a, b]$ (nota: questa condizione può sempre essere ottenuta traslando in verticale il grafico di f di una quantità opportuna). Il grafico di f nell'intervallo $[a, b]$ può essere racchiuso completamente nel rettangolo $R = [a, b] \times [0, c]$ dove $c \geq f_{\max} = \max\{f(x) \mid x \in [a, b]\}$. Per $i = 1, \ldots, N$ si generano i punti casuali (x_i, y_i) con $x_i \in [a, b]$ e $y_i \in [0, c]$ e si determina il numero N_I dei punti generati per i quali vale $y_i \leq f(x_i)$ (nota: questi punti sono "*hit*", gli altri sono "*miss*", vedi figura 9). Seguendo ragionamenti analoghi fatti in precedenza si avrà:

$$\frac{I}{\text{area}(R)} = \frac{I}{(b-a) \cdot c} \approx \frac{N_I}{N},$$

da cui segue un'approssimazione per I:

$$I \approx (b - a) \cdot c \cdot \frac{N_I}{N}. \tag{31}$$

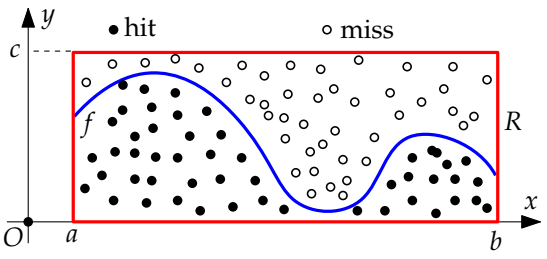

Figura 9. *Integrazione con il metodo hit or miss.*

Se possibile conviene scegliere $c = f_{\max}$; inoltre potrebbe essere ragionevole ridurre il numero di punti sbagliati (*miss*) delimitando il grafico di f con una figura diversa dal rettangolo R e che meglio approssima I.

Algoritmo 4.6. Integrazione mediante il metodo *hit or miss*.

(1) Grazie alla sequenza di numeri casuali u_i uniformemente distribuiti su $[0, 1]$ si costruisce la sequenza di numeri casuali $x_1, \ldots, x_N$ uniformemente distribuiti sull'intervallo $[a, b]$:

$$x_i = (b - a)\, u_i + a. \tag{32}$$

(2) Si costruisce la sequenza di numeri casuali $y_1, \ldots, y_N$ uniformemente distribuiti sull'intervallo $[0, c]$:

$$y_i = c \cdot u_i, \qquad \text{dove } c \geq f(x) \quad \forall x \in [a, b]. \tag{33}$$

(3) Si accettano ("*hit*") gli x_i soltanto se risulta

$$y_i < f\,(x_i)\,. \tag{34}$$

(4) Sarà

$$I \approx (b - a) \cdot c \cdot \frac{N_I}{N},$$

dove N_I è il numero dei punti di tipo "*hit*".

Il metodo *hit or miss* può essere generalizzato al calcolo dell'integrale multiplo I di dimensione n definito dalla (26). Si generano uniformemente N punti $(x_1, \ldots, x_n)$ nel dominio d'integrazione e, per ognuno di questi punti, si genera uniformemente sull'intervallo $[0, c]$ il valore y corrispondente (anche qui abbiamo supposto, per semplicità, che $f(x_1, \ldots, x_n)$ è non negativa nel dominio d'integrazione e c è maggiore di o uguale al massimo valore che può assumere f). Si ottiene poi la seguente approssimazione:

$$I \approx (b_1 - a_1) \cdots (b_n - a_n) \frac{N_I}{N}.$$

dove N_I è il numero dei punti generati per i quali vale $y \leq f(x_1, \ldots, x_n)$.

Esempio 4.7. L'algoritmo 4.6 è facilmente modificabile in modo da calcolare anche integrali multipli. La media dell'integrale A dell'esempio (27) su 100 approssimazioni fatte con $N = 5 \cdot 10^6$ e $c = 5928154283520000$ è stata di 5929806578681903, con un errore relativo del -0.0279%.

4.5. Come riportare i risultati di un'esperienza Monte Carlo

Come per il metodo sperimentale, anche quando si utilizzano delle simulazioni con il metodo Monte Carlo bisogna redigere un rapporto finale che rispetti determinate linee guida, in particolare:

(1) informazioni sulla simulazione, come il generatore utilizzato. È importante che il generatore sia scelto adeguatamente per gli scopi della simulazione.

(2) spiegazione dettagliata di ogni misura utilizzata per ridurre la varianza.[9]

(3) una giustificazione sul numero di campioni impiegato.

(4) dettagli sul linguaggio di programmazione utilizzato.

(5) codice completo relativo alla simulazione.

Prima di passare alle applicazioni pratiche più innovative si presenta un'ulteriore simulazione di tipo Monte Carlo che permette di approssimare il valore della funzione di ripartizione della variabile aleatoria normale, impossibile da calcolare con metodi esatti.

4.6. Approssimazione di Φ

Il metodo Monte Carlo si presta molto bene per approssimare la funzione di ripartizione Φ della distribuzione normale $N(0,1)$

$$\Phi(x) = P(-\infty < X < x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{x} \mathrm{e}^{-\frac{t^2}{2}}\, \mathrm{d}t. \tag{35}$$

Tenendo conto del fatto che

$$\int_{-\infty}^{0} \mathrm{e}^{\frac{-t^2}{2}}\, \mathrm{d}t = \frac{1}{2}\sqrt{\int_{-\infty}^{\infty} \mathrm{e}^{-\frac{t^2}{2}}\, \mathrm{d}t \int_{-\infty}^{\infty} \mathrm{e}^{-\frac{s^2}{2}}\, \mathrm{d}s} = \frac{1}{2}\sqrt{\int_{-\infty}^{\infty} \mathrm{e}^{-\frac{1}{2}\left(t^2+s^2\right)}\, \mathrm{d}t\, \mathrm{d}s}$$

$$= \frac{1}{2}\sqrt{\int_{0}^{\infty} \int_{0}^{2\pi} \mathrm{e}^{-\frac{1}{2}r^2} r\, \mathrm{d}\theta\, \mathrm{d}r} = \frac{1}{2}\sqrt{-2\pi \int_{0}^{\infty} \mathrm{e}^{-\frac{1}{2}r^2}(-r)\, \mathrm{d}r} = \sqrt{\frac{\pi}{2}}$$

si può applicare l'algoritmo 4.8 per approssimare l'integrale

$$\int_{0}^{x} \mathrm{e}^{-\frac{t^2}{2}}\, \mathrm{d}t, \quad x > 0. \tag{36}$$

Algoritmo 4.8. Funzione Φ.

1: VARIABILI
2: X: valore positivo in cui si vuole approssimare Φ;
3: $x_1, \ldots, x_N$: punti generati a caso con l'ascissa nel dominio di integrazione;
4: `dentro`: numero di punti che si trovano all'*interno* (ossia tra il grafico di Φ,
5: gli assi cartesiani e la retta verticale $x = X$;
6: `dentro` = 0;
7: **for** $i = 1, \ldots, N$ **do**
8: **if** x_i si trova *all'interno* **then** `dentro` = `dentro` + 1;

[9] Questo punto, vista la grande complessità matematica, è stato incluso solo implicitamente in questo lavoro. Intuitivamente appare però evidente che minore è la varianza, migliore è l'approssimazione ottenuta.

9: calcola $\Phi(X) = \left(\left(\frac{\texttt{dentro}}{N} \cdot X\right) + \sqrt{\frac{\pi}{2}}\right) \cdot \frac{1}{\sqrt{2\pi}}$;
10: OUTPUT: valore di $\Phi(X)$.

Esempio 4.9. Si vuole calcolare un'approssimazione di $\Phi(0.5)$ applicando l'algoritmo 4.8. Utilizzando:

(1) generatore utilizzato: generatore di libreria di JAVA;

(2) misure per ridurre la varianza: le ascisse x sono generate nell'intervallo $[0, 0.5]$, mentre le ordinate y si trovano nell'intervallo $[0, 1]$; inoltre si è sfruttato il fatto che
$$\Phi(0.5) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{0.5} e^{-\frac{t^2}{2}}\, dt = \sqrt{\frac{\pi}{2}} + \frac{1}{\sqrt{2\pi}} \int_{0}^{0.5} e^{-\frac{t^2}{2}}\, dt;$$

(3) numero di campioni utilizzato: *worse-case scenario* = 10^7;

(4) linguaggio di programmazione: JAVA.

La media di $\Phi(0.5)$ su 100 approssimazioni è stata di 0.6914628708042676, con un errore relativo del -0.00006%.

Il metodo Monte Carlo può essere utilizzato in molti ambiti diversi: dalla matematica pura alla biologia, passando dalla finanza e dalle simulazioni nel gioco d'azzardo, problemi di ottimizzazione, sviluppo di software per il calcolo di integrali e altro. In questo paragrafo si è così data una risposta ai quesiti (5)–(7) del paragrafo 1.

Affinché le simulazioni diano risultati soddisfacenti occorre conoscere bene il problema e saper scegliere l'algoritmo più adatto che porta alla soluzione. In realtà il metodo non presenta limiti invalicabili, ma per simulare alcuni processi ci sono metodi più veloci ed efficaci. In ogni caso il metodo Monte Carlo, *metodo sperimentale potenziato*, ha aperto numerose porte agli studiosi permettendo di risolvere problemi che in passato apparivano estremamente complessi o addirittura impossibili.

5. Applicazioni pratiche

Dopo aver risolto le questioni di partenza (domande (1)–(7) del paragrafo 1), è finalmente arrivato il momento di dedicarsi alla parte innovativa e, per certi versi, più entusiasmante dell'intero lavoro. Questo paragrafo è interamente dedicato all'applicazione di algoritmi Monte Carlo per approssimare aree di luoghi geografici della Svizzera. Il punto di partenza è il lago di Sascòla, ma i risultati saranno generalizzati in modo da poter calcolare l'area di qualunque superficie geografica (lago, ghiacciaio, eccetera). L'obiettivo più ambizioso è quello di sviluppare un software con interfaccia utente grafica (GUI) in grado

di applicare il metodo Monte Carlo a qualunque tipo di immagine, in modo da rendere particolarmente vasto il campo di applicazione.

I risultati saranno confrontati con quelli ottenuti dall'Ufficio federale di topografia Swisstopo, il quale si occupa — grazie a software che utilizzano coordinate cartesiane o supporti cartografici — di approssimare le aree di laghi e ghiacciai presenti sul territorio elvetico. I software impiegati da Swisstopo non utilizzano algoritmi Monte Carlo, bensì applicano metodi analitici a campioni che sono stati discretizzati. Il paragrafo è strutturato come segue:

- approssimazione del lago di Sascòla utilizzando coordinate cartesiane;
- presentazione di una congettura sulle coordinate cartesiane dei punti interni di un poligono;[10]
- approssimazione di diversi laghi utilizzando immagini satellitari;
- sviluppo di una GUI generale che permetta all'utente di inserire alcune informazioni riguardanti le immagini e di approssimare così l'area richiesta;
- discussione dei risultati.

5.1. Superficie lago di Sascòla

Qui il metodo Monte Carlo è applicato per simulare l'area del lago di Sascòla.

Il contorno del lago può essere approssimato da un poligono avente un numero adeguato di vertici dati in coordinate cartesiane, cosa che permette poi di trovare un'approssimazione dell'area del poligono con il metodo *hit or miss*. Per questa esperienza sono stati scelti, tramite apposite apparecchiature GPS, 600 punti sul contorno del lago.

Ecco quanto utilizzato nella simulazione:

(1) generatore utilizzato: generatore di libreria di Java;

(2) misure per ridurre la varianza: le coordinate x sono generate nell'intervallo $[687000, 688000]$, mentre le coordinate y si trovano nell'intervallo $[126000, 127000]$;

(3) numero di campioni utilizzato: *worse-case scenario* $= 10^7$;

(4) linguaggio di programmazione: Java.

[10] Ho utilizzato le condizioni indicate nella congettura 5.1 — che non ho trovato né su internet né in nessuna pubblicazione — per approssimare l'area del lago di Sascòla partendo da coordinate cartesiane e il risultato sembra essere corretto. Prima di applicare la congettura a questo caso, ho controllato il suo funzionamento in numerosi poligoni di vario tipo (concavi, convessi, ...) e non ho ancora trovato un caso in cui essa viene smentita. Tuttavia non sono ancora riuscito a trovarne una dimostrazione.

FIGURA 10. *Lago di Sascòla* (1740 m *s.l.m.*) *in Vallemaggia.*

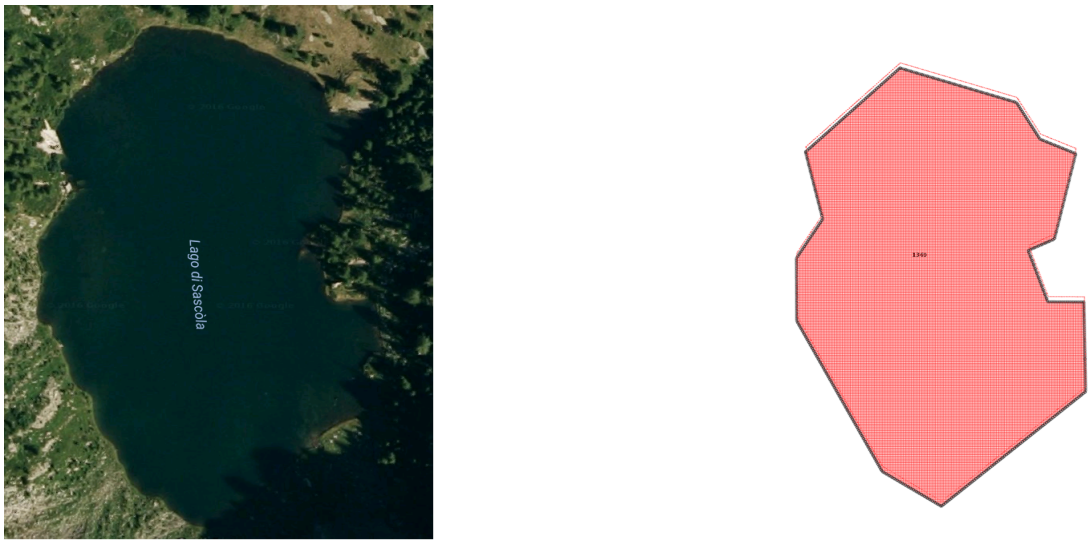

FIGURA 11. *Vista aerea del lago di Sascòla e sua approssimazione a un poligono di* 600 *lati.*

A questo punto è stato necessario risolvere un problema fondamentale: come determinare se un punto si trova o meno all'interno di un poligono? Per questo scopo esistono alcuni algoritmi (*ray-casting*, *winding number*, . . .), ma sono piuttosto difficili da programmare. Partendo da casi semplici e lavorando con casi limite, ho formulato la congettura 5.1 che sembra soddisfare lo scopo.

Congettura 5.1. *Un punto (x, y) si trova all'interno di un poligono se esiste un numero dispari di coppie di vertici consecutivi $i = (i_1, i_2)$, $j = (j_1, j_2)$ tale che*

(1) $i_2 < y < j_2$;

(2) $i_1 + \dfrac{y - i_2}{j_2 - i_2} \cdot (j_1 - i_1) < x.$

Anche se la congettura non è dimostrata, dopo averla applicata a numerosi casi e aver controllato che il risultato ottenuto fosse uguale a quello calcolato con la geometria euclidea, l'ho utilizzata nello sviluppare il software per calcolare l'area del lago di Sascòla.

Algoritmo 5.2. Lago Sascòla.

1: Variabili
2: `test`: punti casuali da testare;
3: `lagoSascòla`: coordinate dei *vertici* del lago;
4: `dentro`: numero di punti che si trovano *all'interno* del lago;
5: **for all** punti x_i nel dominio di integrazione **do**
6: **if** x_i si trova *all'interno* del lago **then** `dentro` = `dentro` + 1;
7: $\texttt{A} = \dfrac{\texttt{dentro}}{\texttt{punti totali}} \cdot 1000 \cdot 1000;$
8: Output: area `A`.

Si ottiene che l'area del lago di Sascòla vale 34446 m^2. Il valore Swisstopo calcolato nel 2012 vale 34176.9394 m^2, dunque l'approssimazione ottenuta qui con Monte Carlo si differenzia dello 0.78% da questo valore.

5.2. Superficie di aree partendo da immagini

In questa sezione il metodo Monte Carlo è applicato partendo da immagini satellitari.[11] I dati delle immagini sono riassunti nella tabella 3.

Tabella 3. *Dati concernenti le immagini* (*risoluzione*: 300 dpi).

Luogo	Dimensioni	Scala	Fonte	Superficie in m^2
Lago Sascòla	10×10	1:3000	Vector 25	35392.34
Lago Nero	14.8×21	1:4000	Vector 25	133378.72
Lago Tremorgio	5×5	1:20000	Vector 25	398507.00
Lago Retico	5×5	1:10000	Vector 25	87860.00
Lago di Ravina	5×5	1:5000	Vector 25	18829.00
Valdrecc Camara	14.8×21	1:4000	CN 1:25000	152716.88
Valdrecc di Sorda	5×5	1:25000	CN 1:25000	318169.00

[11]Tutte le immagini utilizzate in questa sezione e nel paragrafo 3 sono state fornite da C. Scapozza, IST–SUPSI. Base cartografica: Ufficio federale di topografia Swisstopo, Carta Nazionale 1:25000 versione raster o vettoriale 1:25000 (Vector 25) della stessa carta.

Il concetto che sta alla base di queste simulazioni è lo sfruttamento dei pixel dell'immagine. Ogni immagine è composta da un certo numero di pixel e la relazione che lega tale numero alla risoluzione è

$$\text{numero pixel} = \left(\frac{\text{risoluzione}}{2.54} \cdot \text{larghezza}\right) \cdot \left(\frac{\text{risoluzione}}{2.54} \cdot \text{altezza}\right). \tag{37}$$

Inoltre il colore di ogni pixel è dato dalla combinazione di tre numeri r, g, b che determinano il coefficiente di ognuno dei tre colori primari rosso (*red*, r), verde (*green*, g) e blu (*blue*, b). Uno dei punti chiave di questo programma è dunque il riconoscimento dell'area (*hit*) tramite lo spettro desiderato (quello del blu in quanto le immagini utilizzate presentano un'area azzurra).

È tempo di iniziare ad approssimare laghi e ghiacciai:

(1) generatore utilizzato: generatore di libreria di JAVA;

(2) misure per ridurre la varianza: implicite nell'algoritmo;

(3) numero di campioni utilizzato: *worse-case scenario* = 10^7;

(4) linguaggio di programmazione: JAVA.

Algoritmo 5.3. Monte Carlo su immagini.

1: VARIABILI
2: `rgb`: array contenente le informazioni sul colore di ciascun pixel;
3: `scala`, `larghezza`, `altezza`, `risoluzione`: informazioni sull'immagine;
4: `dentro`: numero di punti che si trovano *all'interno* del lago;
5: **for all** punti x_i nel dominio di integrazione **do**
6: **if** x_i si trova *all'interno* del lago (spettro del blu) **then** `dentro` = `dentro` + 1;
7: $\texttt{area} = \texttt{dentro} \cdot \left(\frac{\texttt{larghezza} \cdot \texttt{scala}}{100 \cdot \texttt{larghezza} \cdot \frac{\texttt{risoluzione}}{2.54}}\right) \cdot \left(\frac{\texttt{altezza} \cdot \texttt{scala}}{100 \cdot \texttt{altezza} \cdot \frac{\texttt{risoluzione}}{2.54}}\right)$;
8: OUTPUT: `area`.

I risultati delle simulazioni (le aree sono in m^2) sono riportati nella tabella 4.

TABELLA 4. *Risultati ottenuti con la GUI.*

luogo	area simulata	area Swisstopo	differenza
Lago Sascòla	35493.03	35392.34	0.285%
Lago Nero	133426.32	133378.72	0.036%
Lago Tremorgio	400093.82	398507.00	0.398%
Lago Retico	88017.03	87860.00	0.179%
Lago di Ravina	18946.02	18829.00	0.621%
Valdrecc Camara	152650.36	152716.88	−0.044%
Valdrecc di Sorda	318467.11	318169.00	0.094%

5.2.1. Sviluppo di una GUI

Il software che è stato sviluppato permette di applicare il metodo Monte Carlo all'approssimazione dell'area di qualunque immagine (essa non deve necessariamente avere una risoluzione particolarmente alta).

La figura 12 mostra la schermata dopo una simulazione.

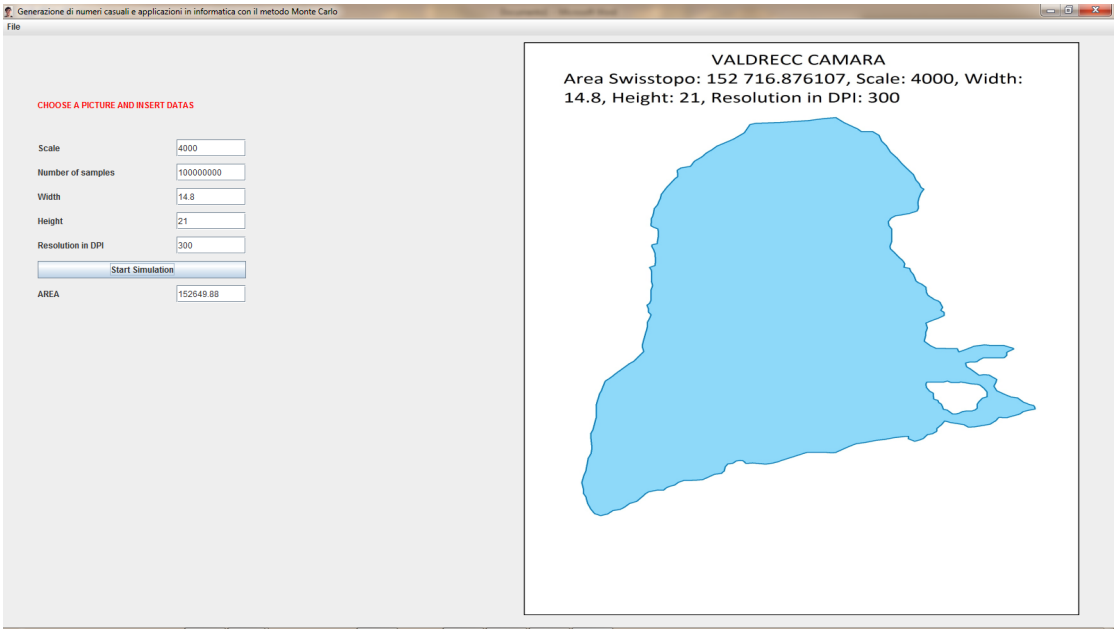

FIGURA 12. *Risultato di una simulazione.*

5.3. Discussione dei risultati

Utilizzare immagini è decisamente più conveniente che sfruttare le coordinate cartesiane. Le motivazioni sono molteplici:

- utilizzare immagini permette di considerare l'area da approssimare come insieme continuo.

- l'utilizzo di coordinate cartesiane presuppone che l'area in questione possa essere approssimata a quella di un poligono. In alcuni casi con forme particolarmente irregolari questo può condurre ad un errore maggiore, mentre in ogni caso si introduce una fonte di errore, dal momento che bisogna scegliere sul contorno un numero finito di punti.

- determinare le coordinate può essere molto dispendioso in termini di tempo, senza contare quanto si impiega a riportare i dati nel programma. Il codice del programma che usa le immagini non dev'essere modificato da un caso all'altro; questo non capita nel programma che si basa sulle coordinate.

- utilizzando le immagini non è necessario ricorrere alla congettura 5.1 che, pur sembrando funzionante, non è ancora dimostrata e dunque non è totalmente affidabile.
- il programma con le immagini presenta possibilità di sviluppo molto interessanti.

Tra le simulazioni che ho eseguito e i risultati trovati da Swisstopo ci sono delle differenze. Esse non possono essere considerate errori in quanto sono entrambe approssimazioni (non è possibile determinare con esattezza la superficie di un lago o di un ghiacciaio) e il valore dell'area non è costante (infatti entrano in gioco fattori quali la stagione, il momento della giornata, la temperatura, fenomeni di erosione e il livello dell'acqua).

Alla luce di questi elementi è possibile trarre alcune conclusioni. Il metodo Monte Carlo, utilizzando l'algoritmo *hit or miss*, si presta molto bene per approssimare aree di superfici geografiche. La GUI che ho sviluppato funziona molto bene in quanto non solo calcola in maniera esatta l'area dell'immagine desiderata, ma è anche molto maneggevole in quanto per utilizzarla non occorrono abilità particolari nell'uso del computer.

Il programma che ho sviluppato è sicuramente innovativo sia sul piano dell'approccio (infatti il metodo Monte Carlo solitamente non viene utilizzato in casi simili) sia sul piano dei contenuti (infatti permette di calcolare l'area di un lago, ma anche di un pezzo di stoffa o di un appartamento) e ben funzionante. Tuttavia le possibilità di sviluppo sono molteplici. Infatti la mia GUI *presuppone che si conosca la scala dell'immagine in esame*. Questo non è un problema quando si tratta di immagini satellitari (si può utilizzare ad esempio lo strumento di misura di Google Maps), ma è uno svantaggio considerevole in altri casi (infatti, nel calcolare l'area di qualcosa di piccolo, si impiegherebbe meno tempo a misurarla manualmente). Sarebbe molto interessante, ma ciò richiederebbe un'altra ricerca, sviluppare un'applicazione per *smartphone* che, sfruttando i sensori per far calcolare al dispositivo la scala dell'immagine, calcoli l'area di un'immagine appena scattata. Ci sono ancora molte possibilità.

Riferimenti bibliografici

[1] LUDOVICA ADACHER. *Generazione di numeri random*. Università degli studi Roma 3. URL: http://adacher.dia.uniroma3.it/automazione/GenerazioneRandom.pdf.

[2] ALAN ANDERSON. «Using Your Crystal Ball: Forecasting with Big Data». In: *Statistic for Big Data For Dummies*. Hoboken, New Jersey: Wiley, 2015.

[3] LEONARDO ANGELINI. *Tecniche Monte Carlo*. Dipartimento Interateneo di Fisica, Università di Bari, 2013. URL: www.ba.infn.it/~angelini/MMFA/dispensa.pdf.

[4] Giovanni Baiocchi. «Monte Carlo Methods in Environmental Economics». In: *Applications of Simulation Methods in Environmental and Resource Economics*. A cura di Riccardo Scarpa e Anna Alberini. Dordrecht (NL): Springer, 2005, pp. 317–340.

[5] Jonathan B. Buckheit e David D. Donoho. *WaveLab and reproducible results*. Università di Stanford, 1995. url: https://statweb.stanford.edu/~wavelab/Wavelab_850/wavelab.pdf.

[6] Marcello Chiodi. *Tecniche di simulazione in statistica*. Napoli: RCE, 2000.

[7] Georges-Louis Leclerc conte di Buffon. *Supplément à l'Histoire Naturelle*. Parigi: Imprimerie Royale, 1777.

[8] Laura Farinetti et al. *Sistemi di numerazione e algebra booleana*. Dipartimento di automatica e di informatica, Politecnico di Torino, 1992.

[9] Francesco Fraccaroli. «Metodo Monte Carlo e generazione di numeri casuali». Tesi di laurea. Facoltà di ingegneria dell'informazione, Università di Padova, 2012. url: http://tesi.cab.unipd.it/40844/1/tesi_default.pdf.

[10] W. S. Gossett. «Probable error of a correlation coefficient». In: *Biometrika* 6 (1908), pp. 302–310.

[11] Alessandra Iannuzzi. «Catene di Markov reversibili e applicazioni al Metodo Monte Carlo basato sulle catene di Markov». Tesi di laurea in calcolo delle probabilità e statistica. Università di Bologna, 2015. url: http://amslaurea.unibo.it/9010/1/iannuzzi_alessandra_tesi.pdf.

[12] F. James. «Monte Carlo Theory and Practice». In: *Experimental Techniques in High Energy Physics*. A cura di T. Ferbel. [La prima versione dello scritto, pubblicata originariamente su *Reports on Progress in Physics* nel 1980, può essere scaricata dall'indirizzo https://www.researchgate.net/publication/252622717_Monte_Carlo_theory_and_practice]. Menlo Park: Addison-Wesley, 1987, pp. 627–677.

[13] Thomas Jennewein et al. «A Fast and Compact Quantum Random Number Generator». In: *Review of Scientific Instruments* 71 (apr. 2000), pp. 1675–1680.

[14] Lord William Thomson Kelvin. «Nineteenth century clouds over the dynamical theory of heat and light». In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.7 (1901), pp. 1–40.

[15] Donald E. Knuth. «Seminumerical Algorithms». In: *The Art of Computer Programming*. Vol. 2. Reading: Addison-Wesley, 1981.

[16] Dirk P. Kroese. *Monte Carlo Methods*. Dipartimento di matematica, Università del Queensland, 2011. url: https://people.smp.uq.edu.au/DirkKroese/mccourse.pdf.

[17] Pierre L'Ecuyer. In: *Handbook of Computational Statistics*. Berlino: Springer, 2012. Cap. Random number generation, pp. 35–71.

[18] PIERRE SIMON LAPLACE. *Theorie analytique des probabilités*. Vol. 7. Parigi: Courcier, 1812.

[19] D. H. LEHMER. «Mathematical Methods in Large-scale Computing Units». In: *Proceedings of a Symposium on Large-Scale Digital Calculating Machinery*. Cambridge: Harvard University Press, 1951. URL: https://archive.org/details/proceedings_of_a_second_symposium_on_large-scale_/page/n177.

[20] THÉODORE LIEBLING. *Production de nombres pseudo-aleatoires*. Dipartimento di matematica, Università di Losanna, 1988.

[21] ALEXANDER. M. MOOD, FRANKLIN A. GRAYBILL e DUANE C. BOES. *Introduzione alla statistica*. Milano: McGraw-Hill, 1993.

[22] ATHANASIOS PAPOULIS. *Probabilità, variabili aleatorie e processi stocastici*. Torino: Boringhieri, 1977.

[23] DANIELA PICIN. *Studio dell'aleatorietà: proprietà di indipendenza ed uniformità*. Università degli Studi di Roma Tor Vergata, 2013.

[24] RANDOMNESS E INTEGRITY SERVICES LTD. *RANDOM.ORG*. URL: https://www.random.org/.

[25] FAUSTO RICCI. *Statistica ed elaborazione statistica delle informazioni*. Bologna: Zanichelli, 1975.

[26] RICCARDO RIGANTI. *Elementi di probabilità e statistica*. 2007.

[27] CHRISTIAN ROBERT E GEORGE CASELLA. *Monte Carlo Statistical Methods*. New York: Springer, 2004.

[28] COMMISSIONI ROMANDE DI MATEMATICA DI FISICA E DI CHIMICA. *Formulari e tavole*. Edition G d'Encre, 2013.

[29] BRUNO SANGUINETTI et al. «Quantum Random Number Generation on a Mobile Phone». In: *Physical Review X* 4 (mag. 2014). URL: https://journals.aps.org/prx/pdf/10.1103/PhysRevX.4.031056.

[30] DOVARI SUDHEERKIRAN. *The Java Swing Tutorial*.

[31] STANISLAV MARCIN ULAM. *Le avventure di un matematico*. Palermo: Sellerio editore, 1995.

[32] RICHARD VON MISES. *Wahrscheinlichkeit, Statistik und Wahrheit*. Berlino: Springer, 1928.

[33] JOHN VON NEUMANN. «Various techniques used in connection with random digits». In: *National Bureau of Standards Applied Math Series* 3 (1951). URL: https://mcnp.lanl.gov/pdf_files/nbs_vonneumann.pdf.

[34] MASAYUKI YANO et al. *Math, Numerics and Programming*. MIT Boston, 2013. URL: https://ocw.mit.edu/ans7870/2/2.086/S13/MIT2_086S13_Textbook.pdf.

[35] ZETCODE. *Java Swing*. URL: http://zetcode.com/tutorials/javaswingtutorial/.

SOEL MICHELETTI
soel.micheletti@hotmail.it