1. Generazione di numeri casuali

1.1 Metodi di generazione casuale

Nel cap. 7 abbiamo dimostrato che, per eseguire una qualsiasi simulazione, dobbiamo disporre di un sistema che generi numeri casuali uniformemente distribuiti nell'intervallo $[0, 1]$, poiché con essi siamo in grado di riprodurre una qualsiasi altra distribuzione di probabilità, discreta o continua. Nel seguito abbiamo sempre dato per scontato il fatto di avere a nostra disposizione un sistema di generazione "perfetto", lasciando però in sospeso ulteriori approfondimenti.

Riprendiamo ora questo discorso, occupandoci brevemente dei metodi con cui vengono prodotti i numeri casuali (da ora in poi sottointenderemo sempre, per semplicità, l'aggettivo uniformi).

L'unico modo in cui si può ottenere una sequenza di numeri casuali che, per definizione, non è prevedibile e quindi neppure riproducibile, è quello di usare un processo stocastico quale il movimento di un disco rotante, il decadimento radioattivo di nuclei instabili o il rumore termico di un'apparecchiatura elettronica. Tavole di numeri così compilate, già in uso agli inizi del secolo, sono oggi facilmente reperibili. Anche l'impiego di un fenomeno fisico per generare direttamente i numeri casuali di volta in volta necessari per una simulazione non pone alcun problema di principio, una volta che si è sicuri del buon funzionamento degli strumenti impiegati. Entrambi questi metodi sono ormai però completamente abbandonati poiché risulta complicato e macchinoso il loro utilizzo all'interno dei programmi per calcolatore.

L'impiego di tabelle preventivamente costruite presenta infatti lo svantaggio di richiedere una grande area, nella memoria del calcolatore utilizzato, esclusivamente destinata a contenere una sequenza abbastanza lunga da permettere anche l'analisi di sistemi molto complessi.

L'altro metodo è stato invece scartato poiché nella pratica è impossibile costruire e collegare al calcolatore dei sistemi che siano abbastanza rapidi da non rallentare la velocità di esecuzione dei programmi (occorre generare almeno qualche centinaio di numeri al secondo), senza contare poi che è necessario verificare spesso anche la "qualità" dei numeri casuali per evitare che impreviste anomalie delle apparecchiature impiegate alterino le caratteristiche delle sequenze prodotte.

L'unico modo per eliminare tutte queste difficoltà è quello di far generare i numeri casuali direttamente al calcolatore mediante appositi algoritmi matematici: le sequenze così ricavate vengono però denominate "pseudocasuali", in quanto, al contrario di quelle "veramente" casuali, esse non solo sono esattamente prevedibili, ma possono anche essere riprodotte in maniera identica quante volte si vuole semplicemente ripetendo il procedimento di calcolo. Indubbiamente può sembrare una contraddizione quella di considerare come casuali dei numeri che sappiamo già in anticipo non essere tali. Il loro utilizzo viene però pienamente giustificato dal fatto che, se ci pensate bene, per il metodo M.C. non sono necessari numeri che soddisfino tutti i criteri "formali" di casualità[1], ma numeri che abbiano *le stesse proprietà* di quelli casuali.

Quello che viene adottato è quindi un punto di vista estremamente pragmatico: non importa il modo in cui i numeri "casuali" vengono generati; l'unico requisito essenziale è il fatto che essi, se sottoposti a verifica, risultino indistinguibili da quelli "veramente" casuali. I vantaggi di questo modo di agire sono evidenti, soprattutto negli attuali centri di calcolo dove solo occasionalmente si eseguono delle simulazioni:

- i numeri pseudocasuali sono prodotti, con una minima occupazione di memoria, ad una velocità che è sostanzialmente quella dello stesso calcolatore utilizzato per la simulazione;
- una volta verificata la loro "bontà", essi possono essere impiegati per una qualsiasi simulazione;
- la loro riproducibilità diventa molto utile quando occorre trovare degli errori all'interno di un programma, operazione in cui è molto spesso conveniente ripetere un procedimento utilizzando gli stessi numeri casuali già impiegati in precedenza.

1.2 Generatori di numeri pseudocasuali

Il primo algoritmo di generazione pseudocasuale, proposto da J. von Neumann alla fine degli anni '40 (vedere [20]), è noto con il nome di metodo del "medio quadrato" (*middle-square method*), per il fatto che ogni numero viene generato elevando al quadrato il predecessore e prendendo le cifre di mezzo del risultato. Se consideriamo, ad esempio, numeri di 4 cifre e partiamo con $x_0 = 5678$, otteniamo:

$$\begin{array}{lcll} x_0 & = & 5678 & x_0^2 = 32\underline{2396}84 \\ x_1 & = & 2396 & x_1^2 = 05\underline{7408}16 \\ x_2 & = & 7408 & x_2^2 = \ldots \end{array}$$

[1] Dare una definizione matematicamente rigorosa di "casualità" è, fra l'altro, un'operazione molto più difficile e complessa di quanto possa sembrare a prima vista: se non siete convinti di questo, consultate [3, 4, 14, 19].

È possibile dimostrare che, se x_0 viene opportunamente scelto, i numeri $x_1, x_2, \ldots$, ottenuti ripetendo in successione questa procedura (che vengono poi riportati nell'intervallo $[0, 1]$ attraverso la divisione per 10^4) hanno delle proprietà molto simili a quelli delle sequenze "veramente" casuali.

Da questo primo esempio siamo già in grado di notare due caratteristiche che sono comuni a tutti i generatori di numeri pseudocasuali:

- ogni numero viene ricavato in base a quello precedente (o, come vedremo, ad alcuni dei precedenti): per questo, se uno qualsiasi di essi riappare, l'intera sequenza viene identicamente ripetuta. Ogni algoritmo è perciò caratterizzato da un certo periodo, la cui lunghezza dipende anche dalla precisione del particolare calcolatore impiegato.
- sempre per il motivo sopra esposto, il grado di correlazione tra i vari numeri generati non è mai completamente nullo: se un numero ξ_i ha un valore "vicino" ad un altro numero ξ_j ricavato in precedenza, allora anche ξ_{i+1} sarà ugualmente "vicino" a ξ_{j+1}.

Il metodo del medio quadrato è stato però ben presto abbandonato poiché presenta diversi gravi difetti. La maggior parte dei generatori oggi impiegati si basa invece sul metodo "congruenziale" (*congruential method*). Dati tre numeri interi positivi m (che viene chiamato modulo), a (incremento) ed x_0 (valore iniziale) (a ed x_0 sono compresi tra 0 ed $m-1$), i numeri pseudocasuali sono prodotti mediante la formula:

$$\begin{cases} x_i = & ax_{i-1} \mod m \\ \xi_i = & x_i/m \ , \end{cases} \tag{1.1}$$

in cui x_i rappresenta il resto della divisione di (ax_{i-1}) per m.

Se, ad esempio, scegliamo: $a = 5$, $x_0 = 1$ ed $m = 64$, la sequenza pseudocasuale diventa:

$$1, 5, 25, 61, 49, 53, 9, 45, 33, 37, 57, 29, 17, 21, \ldots \tag{1.2}$$

Le proprietà di questo tipo di generatori, che dipendono quasi esclusivamente dal fattore a, sono ben conosciute ed in particolare è stato scoperto che il loro principale difetto risiede in una forte correlazione multidimensionale tra i vari numeri generati ([10]). Questo cattivo comportamento può essere tuttavia evitato combinando più generatori congruenziali all'interno dello stesso algoritmo ([15]).

Con il passare degli anni, sono state via via sviluppate diverse altre classi di generatori casuali ed una rassegna completa dei vari algoritmi proposti è riportata in [5, 11, 14, 18, 17]. In questa nostra breve carrellata citeremo solo i cosiddetti "generatori di Fibonacci" che rappresentano la più valida alternativa a quelli congruenziali. Essi, analogamente alla famosa serie in cui ogni elemento è la somma dei due precedenti, sono essenzialmente basati sulla formula:

$$x_i = x_{i-p} \odot x_{i-q} \qquad \mod m \tag{1.3}$$

in cui p e q sono numeri interi minori di i e $\odot$ rappresenta un'operazione aritmetica o binaria come, ad esempio, quella di "OR" esclusivo (addizione senza riporto). In particolare occorre segnalare una variante di questo tipo di generatori proposta in [16] e che è reperibile nella libreria di programmi del CERN con il nome di RANLUX ([12]). Mentre la bontà di un qualsiasi altro algoritmo è soprattutto basata empiricamente sull'assenza o meno di controesempi, in questo caso le buone proprietà casuali del generatore sono state dimostrate per la prima volta matematicamente "a priori".

La libreria SCILAB offre la routine `grand`, che è un affidabile generatore di numeri casuali. Vi invitiamo a leggere attentamente la documentazione in linea di questa routine ed a verificarne le prestazioni con qualcuno dei test descritti qui.

Vi dobbiamo infine segnalare che, soprattutto per il calcolo degli integrali, è abbastanza diffuso l'uso dei cosiddetti numeri "quasi-casuali". Essi vengono generati da procedimenti in cui si mantiene certo grado di correlazione nelle sequenze generate in cambio di una loro maggiore uniformità, proprietà che contribuisce ad aumentare la precisione sul risultato. Se siete interessati ad ulteriori approfondimenti di questo interessante argomento, consultate le referenze [1, 2, 6, 7, 8, 9, 10, 13, 18, 17].

1.3 Controlli statistici

Poiché, come abbiamo appena visto, è stato finora trovato solamente un algoritmo pseudocasuale sicuramente "buono", vi raccomandiamo vivamente di seguire il seguente consiglio di D.E. Knuth ([14]):

> *I numeri casuali non dovrebbero essere generati mediante un algoritmo scelto a caso.*

Come "regola numero uno" vi raccomandiamo naturalmente l'impiego di RANLUX, di `grand` o di uno degli altri generatori che sono già stati sufficientemente verificati ed utilizzati con successo. Tuttavia occorre notare che quando uno stesso algoritmo viene implementato su calcolatori differenti si ottengono quasi sempre delle sequenze differenti, soprattutto a causa della diversa precisione delle macchine utilizzate.

Una precauzione che vi consigliamo quindi di prendere quando dovete per la prima volta utilizzare un qualsiasi generatore (oltre ad essere sicuri che il suo periodo sia abbastanza lungo per la simulazione prevista) è quella di verificare empiricamente la casualità dei numeri prodotti per evitare, come è purtroppo già accaduto diverse volte, degli errori nei risultati finali causati da correlazioni troppo importanti tra i numeri generati. Un test molto semplice è, per esempio, quello di controllare che la media di N numeri pseudocasuali sia, entro gli errori statistici, sempre uguale a 0.5. Ma, poiché, come è facile intuire, vi è un numero infinito di funzioni utilizzabili, non si può mai in questo modo verificare con assoluta certezza che un generatore sia completamente

sicuro ed affidabile, anche se è possibile definire un numero minimo di test che tutti i "buoni" algoritmi devono superare. In [14] potete trovare un elenco completo di tutti i test che vengono di solito utilizzati per verificare la "bontà" dei generatori pseudocasuali. Tra quelli di uso più frequente vi segnaliamo:

a) Test di equidistribuzione (o di frequenza) per verificare la reale uniformità dei numeri generati. Essi vengono disposti in un istogramma di k canali e, con i metodi che vi abbiamo ampiamente spiegato nel cap. 6, si controlla se lo spettro così ottenuto proviene o meno da una distribuzione uniforme.
b) Test seriale: generalizza il test precedente verificando l'uniformità della distribuzione di coppie di numeri casuali. Esse vengono disposte in una matrice bidimensionale di $(k \times k)$ elementi e si controlla se in tutti gli elementi vi è lo stesso numero di coppie.
c) Test del poker; per scoprire eventuali correlazioni multidimensionali. In questo caso si considerano gruppi consecutivi di 5 numeri casuali e si determina quanti numeri uguali si trovano all'interno di ogni quintupla, secondo uno schema simile a quello del gioco del poker. La probabilità delle diverse configurazioni viene poi paragonata con quella prevista nel caso di una distribuzione di numeri puramente casuale, che viene facilmente ricavata partendo dalla distribuzione binomiale.
d) Test del *gap*. Data una sequenza di interi, si definisce *gap* la distanza che separa 2 numeri uguali. Nel caso di una successione casuale, la probabilità di avere un *gap* di lunghezza n è data da una distribuzione di tipo geometrico (vedere eq. 3.7). Anche in questo caso, una deviazione rispetto alle previsioni indica la presenza di una correlazione multidimensionale nei numeri generati.

Bibliografia

1. M. Berblinger and C. Schlier. Monte carlo integration with quasi-random numbers: some experience. *Computer Physics Communications*, 66:157–166, 1991.
2. M. Berblinger, Ch. Schlier, and T. Weiss. Monte carlo integration with quasi-random numbers: experience with discontinuous integrands. *Computer Physics Communications*, 99:151–162, 1997.
3. G.J. Chaitin. Casualità e dimostrazione matematica. *Le Scienze*, 85:30–35, settembre 1975.
4. A. Compagner. Definitions of randomness. *American Journal of Physics*, 59:700–705, 1991.
5. G.S. Fishman. *Monte Carlo Concepts, Algorithms, and Applications*. Springer-Verlag, New York, 1996.
6. J.M. Hammersley and D.C. Handscomb. *Monte Carlo Methods*. Chapman and Hall, London, 1964.
7. J.H. Hoogland and R. Kleiss. Discrepancy-based estimates for quasi-monte carlo i. general formalism. *Computer Physics Communications*, 98:111–127, 1996.
8. J.H. Hoogland and R. Kleiss. Discrepancy-based estimates for quasi-monte carlo ii. results in one dimension. *Computer Physics Communications*, 98:128–136, 1996.
9. J.H. Hoogland and R. Kleiss. Discrepancy-based estimates for quasi-monte carlo iii. error distributions and central limits. *Computer Physics Communications*, 101:21–30, 1997.
10. F. James. Monte carlo theory and practice. *Reports on Progress in Physics*, 43:1145–1189, 1980.
11. F. James. A review of pseudorandom number generators. *Computer Physics Communications*, 60:329–344, 1990.
12. F. James. Ranlux: A fortran implementation of the high-quality random number generator of lüscher. *Computer Physics Communications*, 79:111–114, 1994.
13. F. James, J.H. Hoogland, and R. Kleiss. Multidimensional sampling for simulation and integration: measures,discrepancies, and quasi-random sampling. *Computer Physics Communications*, 99:180–220, 1997.

14. D.E. Knuth. *The Art of Computer Programming*, volume 2: Seminumerical Algorithms, chapter 2. Addison-Wesley, Reading, second edition, 1981.
15. P. l'Ecuyer. Efficient and portable combined random number generators. *Communications of the ACM*, 31:742–774, 1988.
16. M. Luescher. A portable high-quality random number generator. *Computer Physics Communications*, 79:100–110, 1994.
17. W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Wetterling. *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge, second edition, 1992.
18. B. Ripley. *Stochastic Simulation*. J.Wiley and Sons, New York, 1986.
19. R. Rosa. *Lezioni sul Metodo di Monte Carlo*. Lo Scarabeo, Bologna, 1992.
20. J. von Neumann. Various techniques used in connection with random digits. In A.H. Taub, editor, *John von Neumann collected works*, volume V, pages 768–770. Pergamon Press, Oxford, 1963.