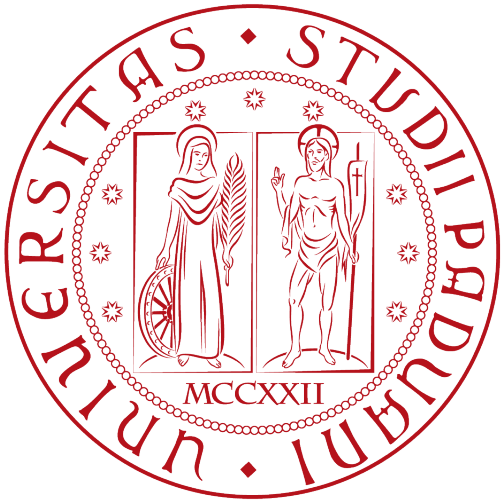
METODO MONTE CARLO E GENERAZIONE DI NUMERI CASUALI

RELATORE: Dott. Finesso

LAUREANDO: Francesco Fraccaroli

A.A. 2011-2012



UNIVERSITÀ DEGLI STUDI DI PADOVA
DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE
TESI DI LAUREA

METODO MONTE CARLO E GENERAZIONE DI NUMERI CASUALI

RELATORE: Dott. Finesso

LAUREANDO: *Francesco Fraccaroli*

Padova, 28 settembre 2012

Indice

Capitolo 1

Introduzione

La generazione di numeri casuali ha affascinato l'uomo sin da tempi antichissimi. Tuttavia solo recentemente il processo è stato velocizzato permettendone l'utilizzo su larga scala anche nella ricerca scientifica. Tali generatori sono sfruttati principalmente per simulazioni al calcolatore, tecniche di campionamento statistico o in ambito crittografico.

Nel prossimo capitolo verranno presentati alcuni dei generatori di numeri casuali in uso, o storicamente rilevanti, soffermando l'attenzione sui loro punti di forza e di debolezza. In questo modo sarà possibile riflettere meglio sulle proprietà desiderabili della classe di algoritmi presa in esame, in particolare:

- larga varietà dell'insieme delle uscite e minima distanza tra valori numericamente consecutivi;
- scarsa correlazione tra valori successivi, nella sequenza di generazione;
- facilità di implementazione a livello informatico;

Si sposterà dunque l'attenzione su una classe di generatori (deterministici), molto veloce, e di alcune sue modifiche, in grado di simulare in maniera soddisfacente, soprattutto per quanto riguarda la non correlazione, variabili aleatroie uniformi. Un semplice lemma permetterà di estendere le conclusioni raggiunte a qualsiasi funzione di densità (o distribuzione di probabilità), ad esempio la celeberrima gaussiana.

Verranno analizzate le caratteristiche generali alla base del funzionamento dei test statistici, mostrando qualche esempio in grado di mettere in evidenza il non soddisfacimento di alcune delle proprietà fondamentali sopra citate.

A conclusione di tutto si presenterà uno delle maggiori tecniche applicative, nel mondo scientifico contemporaneo, della generazione di numeri casuali, il metodo Monte Carlo, trattando in una breve panoramica la storia ed alcuni utilizzi pratici di interesse.

Capitolo 2

Generatori di numeri casuali

2.1 Panoramica di generatori

Lanciare un dado, tirare una moneta, mescolare un mazzo di carte sono tutte attività che consentono di introdurre dell'incertezza, che permettono di prendere decisioni in modo non deterministico, liberando dalla faziosità umana una scelta che non dovrebbe esserne viziata. Tuttavia i metodi tradizionali richiedono tempo ed offrono una scarsa gamma di risultati possibili, senza contare che anche un dado o una moneta possono risultare sbilanciati: per questa ragione nell'ultimo secolo due categorie di generatori hanno soppiantato quelli sopra citati: i generatori fisici e quelli algoritmici [8].

La prima modalità consta nella misurazione di fenomeni fisici ritenuti non affetti da distorsioni e nel successivo tentativo di correggere errori sistematici dovuti al processo di misura, la seconda invece si basa sulla creazione di algoritmi computazionali assolutamente deterministici, ma in grado di produrre lunghe sequenze di risultati apparentemente casuali. Questo "apparentemente" verrà chiarito in seguito, per ora basti sapere che, per il loro alto grado di somiglianza a generatori casuali, tali generatori sono definiti pseudo-casuali. Il loro risultato infatti, una volta conosciuti i parametri, sarà prevedibile in modo certo, tuttavia un'attenta scelta può produrre una sequenza di numeri compatibile con una generazione veramente aleatoria.

2.1.1 Generatori fisici

I metodi fisici si applicano a fenomeni atomici o subatomici (e.g. decadimento radioattivo), che, a causa della loro natura quantistica, assicurino imprevedibilità. Tuttavia gli stumenti usati per la raccolta dei dati tendono a rendere asimmetrica, e dunque non uniforme, la variabile aleatoria simulata. Risultano quindi necessarie delle correzioni sugli output generati, ma per operarle è necessario uno studio approfondito del materiale che introduce notevoli ritardi temporali all'effettivo processo di generazione.

Al giorno d'oggi è possibile generare numeri casuali in tempo reale durante un esperimento, grazie alla potenza di calcolo offerta dai moderni computer, ma non è stato sempre così. Curiosa è la storia di **"A Million Random Digits with 100.000 Normal Deviates"** [4], si tratta di un libro edito nel 1955 contenente appunto un milione di numeri generati casualmente da una simulazione elettronica di una roulette e salvati su schede perforate. Una sorgente di impulsi casuali ne forniva circa 100.000 al secondo [6]. Alcuni di essi attraversavano un cancello, aperto solo una volta al secondo, arrivando a dei circuiti di normalizzazione che li convertivano in contatori binari da 5 bit. Il dispositivo appena citato sostituiva una precedente roulette a 32 posti che, compiendo 3.000 giri a processo, forniva un valore al secondo. In entrambi i casi, 20 di questi valori (12 erano invece scartati) erano mappati in base 10, tramite un convertitore adeguato. Si procedeva, dunque, a prelevare la cifra delle unità ed a depositarla su schede perforate, con l'ausilio di una punzonatrice IBM. In questo modo era disponibile una quantità elevata di numeri generati casualmente, oltre ad alcune pagine di consigli riguardo all'effettivo metodo di scelta degli stessi [5], per far si che la sequenza utilizzata non fosse sempre la stessa. Infatti l'essere umano di per sé non garantisce ottime prestazioni in quanto a casualità, ma tende ad operare secondo uno schema abbastanza costante. Anche questo metodo era affetto da errore sistematico (prevalenza dei numeri dispari sui pari), come fu evidenziato da studi successivi su vasti campioni (125.000).

Ad ogni modo tutti i generatori di numeri casuali che sfruttano metodi fisici si dividono in due gruppi: quelli che sfruttano proprietà casuali-quantistiche e quelli che invece ne fanno a meno. Tra i primi si annovera ad esempio la generazione tramite **shot noise** (da Shottky, lo scienziato che per primo pubblicò qualcosa a riguardo) [14]. Si tratta di un rumore meccanico quantistico causato, per il

principio di indeterminazione, dall'arrivo di fotoni nel circuito. Tuttavia l'energia di questo rumore non risulta sempre ben distribuita all'interno della banda di interesse: l'uso di raddrizzatori immersi in campi magnetici trasversali consente di ottenere energia di rumore elevata al prezzo di distribuzioni molto aguzze, richiedendo quindi attenzione nella fase di progetto dei filtri per rendere piatta la distribuzione in un ampio spettro. Un fenomeno fisico, stavolta non quantistico, facilmente misurabile è il **clock drift** [14]; tutto si basa sull'utilizzo di più orologi a cristallo, cioè circuiti elettronici che utilizzano la frequenza di risonanza di un cristallo di materiale piezoelettrico per generare segnali elettrici a frequenza molto precisa. Sfruttando il fatto che minime variazioni di temperatura, caratteristiche del silicio o condizioni elettriche locali causano variazioni pressoché imprevedibili nel rapporto tra le frequenze di risonanza dei vari cristalli, si può considerare il risultato di tale rapporto come un numero casuale. Ad esempio, nel casi in cui il cristallo veloce oscilli un numero dispari di volte durante un periodo di quello più lento, si può associare alla misura il valore binario 1, 0 viceversa. Nel caso in cui ci si accorga di una particolare correlazione tra i valori ottenuti sono possibili metodi correttivi tramite software.

2.1.2 Il metodo del *middle-square*

Anche per quanto riguarda gli algortimi computazionali più strade sono percorribili, tuttavia la maggior parte di esse non produce risultati soddisfacenti. Ne è un esempio il metodo del middle-square proposto da Von Neumann nel 1949 [7]. Un numero di 4 cifre (o di un qualunque altro numero pari), detto seme, viene elevato al quadrato in modo da ottenerne uno lungo il doppio, eventualmente con l'aggiunta di uno zero in posizione iniziale. Il passo successivo consiste nell'isolare la parte centrale del numero così ottenuto, da queste 4 cifre si farà ripartire il procedimento appena descritto. Inoltre esse costituiscono il successivo valore della sequenza.

A titolo di esempio, si prenda come seme il numero 2345, il cui elevamento al quadrato fornisce 5499025. Scartando le prime due cifre (05) e le ultime due (25) si isola 4990 che costituisce il secondo anello della catena. Un ulteriore passaggio porterebbe a 24900100 e dunque a 9001. Nel tentativo di dimostrare la limitatezza di tale metodo si può continuare per questa via. I valori successivamente ottenuti sono: (0)180; (0)324; 1049; 1004; (00)80; (00)64; (00)40; (00)16; (000)2;

(0000). Come si può notare da questo rapido esempio, la sequenza presa in esame non risulta affatto lunga, inoltre permette di evidenziare una proprietà di questo metodo: se la prima metà di un numero della sequenza è costituita da soli zeri, i suoi successori decresceranno fino a zero. Dunque, come sottolineò lo stesso Von Neumann, nemmeno questo generatore assicura ottime prestazioni, soprattutto per la ristretta dimensione del suo periodo, eventualmente preceduto da una parte transitoria (in questo caso 2345 4990 9001 180 324 1049 80 64 40 16 2).

Trattandosi di un algoritmo deterministico in cui ogni termine dipende solo dal precedente, una volta che un anello della catena dovesse ripetersi, si avrebbe come necessaria conseguenza la ripetizione di tutti gli anelli successivi. Studi approfonditi hanno dimostrato l'impossibilità di ottenere periodi maggiori di 8^n (invece che $10^n - 1$), con n numero di cifre del seme. Dunque un'ottima scelta dello stesso garantirebbe periodi non troppo elevati, mentre una pessima scelta addirittura ridicoli, basti pensare che nel caso $n = 4$, semi del tipo 0100, 2500, 3792, 7600 danno generazione costante, proprio come 0000.

2.1.3 Il metodo di Lehmer

Un altro algoritmo, nettamente meno acerbo, è quello di Lehmer [9]. Esso si basa sulla divisione modulo m, cioè sul prendere il resto della divisione per m, tecnica utilizzata anche dai moderni ed operativi linear congruential generators (analizzati nei paragrafi successivi). La catena segue la formula $x_{n+1} = a \cdot x_n \bmod m$, una volta definiti i parametri a, m ed il seme x_0. Scelte oculate dei parametri consentono di ottenere catene di periodo $m - 1$ (il valore zero non può apparire se non come unico valore della catena), eliminando anche l'imbarazzo di dover selezionare un seme adeguato. Il criterio qui enunciato soddisfa la maggior parte dei requisiti necessari per poter parlare di un buon generatore di numeri casuali, tuttavia risulta inadeguato per via della carenza di indipendenza tra i valori generati, i quali tendono a distribuirsi in zone ben precise (la cosa verrà chiarita meglio nel capitolo successivo, ma se ne darà una dimostrazione qui di seguito).

Nel caso di $m = 13$ e $a = 6$ la sequenza generata risulta:

$$\ldots\ 1, 6, 10, 8, 9, 2, 12, 7, 3, 5, 4, 11, 1, \ldots$$

Dunque indipendentemente dalla scelta del seme il periodo risulta massimo, tuttavia appare evidente la presenza di una certa correlazione tra i valori generati,

utilizzando il piano cartesiano. Si creino punti assegnando un numero all'ascissa ed il successivo all'ordinata: essi andranno a disporsi, lungo le rette

$$\begin{aligned} y &= -\frac{x}{2} + 13, \\ y &= -\frac{x}{2} + \frac{13}{2}, \end{aligned}$$

in corrispondenza a valori interi di entrambe le coordinate, dunque non in maniera *uniforme* su tutta l'area disponibile.

2.1.4 Definizione moderna

L'approccio moderno alla creazione di generatori di numeri (pseudo-)casuali è molto più teorico, per questa ragione si hanno ora a disposizione definizioni molto rigorose, quali ad esempio la seguente.

Con generatore di numeri (pseudo)-casuali si intende una struttura $\mathcal{G} = (S, s_0, T, U, G)$, dove S è un insieme finito di stati, s_0 è un stato (o seme), $T : S \to S$ è la funzione di transizione, U è un insieme finito di simboli di uscita e $G : S \to U$ è la funzione di uscita [11].

2.2 Proprietà desiderabili

Se è vero che solo con metodi fisici è possibile ottenere una sequenza di numeri veramente casuale, è anche vero che, per ragioni di tempo reale (ripetibilità e portabilità), sono utilizzati dei metodi deterministici nelle moderne simulazioni. Per supplire alla carenza della tanto agognata aleatorietà gli algoritmi computazionali devono soddisfare alcune proprietà. Esiste necessariamente un test, non per forza sempre il medesimo, in grado di far fallire qualsiasi generatore di numeri pseudo-casuali e decretarne l'inadeguatezza generale. Tuttavia non bisogna mai dimenticarsi dell'utilità pratica assicurata da questi apparecchi né del contesto in cui verranno impiegati ed è proprio rapportandosi ad esso che va dato il giudizio sull'attendibilità. Sostanzialmente, per essere di una qualche utilità, il test non deve avere né complessità temporale né presupporre tempi di osservazione della sequenza troppo elevati.

Sia $\{\mathcal{G}_k,\ k = 12, ...\}$ una famiglia di generatori dove k indica il numero di bit richiesti a rappresentarne lo stato e siano i tempi richiesti a calcolare le funzioni

T e G (nel peggiore dei casi) polinomiali in k (dunque non esponenziali). Ci si restringa inoltre al caso in cui anche i test statistici abbiano tempo di esecuzione polinomiale in k, precludendo la possibilità di operare sull'intera sequenza. $\{\mathcal{G}_k\}$ è *perfetto in tempo polinomiale* se, per ogni test statistico (a tempo polinomiale) che provi a distinguere la sequenza di uscita del generatore da una sequenza infinita di variabili aleatorie iid $U(0,1)$, la probabilità che il test operi la scelta giusta non superi $\frac{1}{2} + e^{-k\epsilon}$, dove ϵ è una costante positiva. Il concetto alla base è che ciò che non può essere calcolato in tempo polinomiale, è praticamente impossibile da calcolare per k sufficientemente grande.

2.2.1 Periodo e non correlazione

Ogni sequenza di numeri generati pseudo-casualmente è per sua natura periodica; qualunque algoritmo, non potendo fornire un insieme illimitato di stati, alla lunga riproporrà le condizioni iniziali, che porteranno ad un'ineluttabile ripetizione dei valori fino a quel momento ottenuti. Proprio qui risiede uno dei più grossi limiti di questo tipo di generazione: ogni replica o schema è per sua natura nemica della casualità e dunque va evitata. Appare evidente l'esigenza di disporre di un periodo, ρ, il più lungo possibile, non solo, infatti non tutte le sequenze di elementi di lunghezza ρ, appartenenti all'insieme U, sono equivalenti: un ulteriore fattore discriminante per la scelta del miglior algoritmo è la correlazione tra i vari anelli della catena [11]. Deve trattarsi di una distribuzione uniforme in modo da poter prendere in esame, in un'ipotetica simulazione, la totalità dei casi presentabili, non permettendo alle eccezioni dell'esperimento di nascondersi dietro alle zone d'ombra di un generatore mal progettato.

Selezionato a caso l'indice n dall'insieme $\{1,2,...,\rho\}$, sia $\boldsymbol{u}_n = (u_n, ..., u_{n+t-1})$ un vettore di uscita, con $t \ll \rho$. Volere $\boldsymbol{u}_n$ uniformemente distribuito nell'insieme U^t dei vettori di lunghezza t imporrebbe $\rho \geq |U|^t$, essendoci ρ differenti valori di $\boldsymbol{u}_n$, in caso contrario solo una parte di $|U|^t$ sarebbe coperta; è auspicabile che il sottoinsieme dei punti rappresentabili $\Psi = \{\boldsymbol{u}_n, 1 \leq n \leq \rho\}$ sia uniformemente distribuito nell'iper-cubo $[0,1]^t$.

Esempio 1 Sia U=$\{0,1/100,...,99/100\}$ (dunque $|U|$=100) e $\rho = 10^4$, quindi per vettori di dimensione t si ha $|U|^t = 10^{2t}$ e $|\Psi| = 10^4$. In buona sostanza solo per t=1,2 si ha distribuzione uniforme su tutto $|U|^2$, mentre per $t > 2$ vorrem-

mo che i punti generabili dalla sequenza siano il più uniformemente distribuiti possibile.

Avere una distribuzione (abbastanza) diluita sull'intero iper-cubo considerato, fissato t, è indice sia di uniformità (nel senso di variabile aleatoria) che di indipendenza, ed è quindi una caratteristica imprescindibile per un buon generatore.
In realtà Ψ risulta fin troppo ben distribuito su U (si parla di *superuniformità*), anzi una rapida occhiata è sufficiente a comprendere che l'eccessiva precisione del reticolo generabile non ha nulla a che fare con una variabile aleatoria vera e propria, per questa ragione si cerca di mantenere il numero di punti prelevati di parecchi ordini di grandezza rispetto al periodo in modo da mascherare dietro a numeri enormi un evidentissimo schema.

2.2.2 Discrepanza

Per riassumere i concetti sopra esposti può essere utile la definizione di una nuova quantità, detta discrepanza. Si considerino gli N punti $\boldsymbol{u}_n = (u_n, ..., u_{n+t-1})$, con n=0,...,N-1. Per ogni iper-rettangolo allineato con gli assi $R = \Pi_{j=1}^{t}[\alpha_j, \beta_j)$ con $0 \leq \alpha_j < \beta_j \leq 1$, siano $I(R)$ gli $\boldsymbol{u}_n$ interni ad R e $V(R) = \Pi_{j=1}^{t}(\beta_j - \alpha_j)$ il volume dell'iper-rettangolo. Detto $\mathcal{R}$ l'insieme delle regioni R si definisce *discrepanza (estrema) t-dimensionale* [11] dell'insieme di punti $\{\boldsymbol{u}_0,...,\boldsymbol{u}_{N-1}\}$ la quantità:

$$D_N^{(t)} = \max_{R \in \mathcal{R}} |V(R) - I(R)/N|.$$

Si parla di *discrepanza star* $D_N^{*(t)}$ nel caso in cui $\alpha_j = 0$ per ogni j, cioè $\mathcal{R}$ rappresenta gli iper-rettangoli con un vertice nell'origine.

Un basso valore della discrepanza corrisponde a buone distribuzioni dei punti nell'iper-cubo unitario, tuttavia può essere calcolata solo in casi molto particolari, ha infatti complessità $O(N^t)$ dove t rappresenta la dimensione dell'iper-spazio, quindi sebbene non esistano impedimenti teorici all'applicazione di questa formula, di fatto col crescere di t si presentano notevoli difficoltà.

Una sequenza di variabili aleatorie iid $U(0,1)$ presenta discrepanze, star ed estrema, dell'ordine di $O(N^{-1/2})$; Niederreiter, concentrandosi su MLCG (multiplicative linear congruential generators) a periodo pieno $N = \rho = m - 1$ (generatori simili a quello proposto da Lehmer), dimostrò facendo una media dei

risultati ottenuti con moltiplicatori a diversi, che le discrepanze sono di ordine $O(m^{-1}(\log m)^t \log\log(m+1))$ [11]. Per m grande tale approssimazione risulta molto maggiore di $O(m^{-1/2})$, determinando superuniformità. Anche per questa ragione è molto importante non usare mai più di una parte trascurabile della sequenza.

L'interesse per questo parametro è motivato dal fatto che fornisce limite all'errore deterministico dell'integrazione tramite Monte Carlo: minore la discrepanza, minore è l'errore numerico (anche se di fatto non assomiglia per nulla a variabili aleatorie iid $U(0,1)$). Sequenze per le quali la discrepanza dei primi N valori è molto ridotta sono dette *a bassa discrepanza* o *quasi casuali* è per quanto riguarda gli integrali con esse calcolati si parla di *integrazione quasi-Monte Carlo*.

Parecchie ragioni,elencate in precedenza, giustificano il bisogno di considerare $\rho \gg N$, inoltre Brian Ripley, esperto di statistica britannico, suggerisce la condizione $\rho \gg N^2$ per i diffusi ed attendibili linear congruential generator, in ogni caso ρ deve essere enorme, per dare un ordine di grandezza i generatori attuali utilizzano almeno $\rho = 2^{200}$ [11].

2.2.3 Efficienza, memoria, ripetibilità e portabilità

Altri fattori, meno matematici, ma assolutamente pratici contribuiscono a limitare la scelta dei generatori in uso; si tratta di condizioni che coinvolgono il mondo dell'informatica per l'implementazione dell'algoritmo. Alte velocità di calcolo garantiscono di ridurre il tempo di esecuzione della simulazione; sebbene generalmente esso dipenda principalmente dai calcoli operati sulla sequenza generata casualmente e non dalla generazione in sé, altre volte, soprattutto in esperimenti fisici, sono necessarie grandi moli di dati, che tendono a rallentare eccessivamente il suddetto tempo. Memorie più capienti invece assicurano la possibilità di gestire parallelamente più generatori contribuendo a creare algoritmi via via più elaborati ed affidabili. La ripetibilità, cioè la proprietà di poter ripetere esattamente la medesima sequenza di numeri casuali, è importante per la verifica del programma e risulta un punto di forza rispetto ai generatori veramente casuali, i quali potrebbero assicurare un simile comportamento solo ricorrendo a supporti di memoria, al prezzo di un ulteriore rallentamento. Codici portabili permettono di generare risultati esattamente identici in tutti i compilatori e computer "standard", garantendo opportunità di utilizzo più vaste.

Infine particolari accorgimenti durante la scrittura del codice, di alto livello, possono migliorare di gran lunga l'efficienza dell'algoritmo, se ne darà dimostrazione pratica nell'esempio che segue.

Esempio 2 (Metodo di Schrage) Si consideri un generatore di Lehmer con $a = 7^5 = 16807$ ed $m = 2^{31} - 1 = 2147483647$, per ragioni che verranno chiarite al capitolo successivo, si tratta di una funzione generatrice a periodo intero. Un algoritmo basato sulla formula $z_{n+1} = a \cdot z_n \bmod m$ è sicuramente funzionante, ma impone di utilizzare una rappresentazione a 46 bit nel calcolo di $a \cdot z_n$, per alcuni valori sufficientemente elevati di z_n. Risulta dunque inutilmente dispendioso, dal momento che tra gli output non compaiono mai valori cosi elevati [12].

In Pascal il codice di un algoritmo efficiente potrebbe apparire in questa maniera:

```
function Random : real; (*Integer Version*)
const
    a = 16807;
    m = 2147483647;
    q = 127773;          (* m div a *)
    r = 2836;            (* m mod a *)
var
    lo, hi, test : integer;
begin
    hi := seed div q;
    lo := seed mod q;
    test = a*lo - r*hi;
    if test > 0 then
        seed := test
    else
        seed := test + m;
    Random := seed / m;
end;
```

L'idea è quella di costruire un algoritmo in grado di calcolare $f(z) = az \, mod \, m$ in maniera che i risultati intermedi siano limitati da $m-1$. Condizione necessaria affinché il periodo sia massimo è che a ed m siano primi tra loro, dunque si può

esprimere m come $aq + r$. La cosa è da un certo punto di vista scoraggiante, dal momento che se fosse stato più semplicemente $m = aq$ si sarebbe potuto invertire l'ordine delle operazioni di $f(z) = az \bmod aq = a(z \bmod q)$. Ad ogni modo: $q = m \operatorname{div} a$ e $r = m \bmod a$. Ripensando alla definizione di resto della divisione ed aggiungendo e sottraendo la quantità $m(z \operatorname{div} q)$ si giunge a

$$\begin{aligned} f(z) &= az \bmod m \\ &= az - m(az \bmod m) \\ &= az - m(az \bmod m) + m(z \operatorname{div} q) - m(z \operatorname{div} q) \\ &= az - m(z \operatorname{div} q) + m(z \operatorname{div} q) - m(az \bmod m) \\ &= az - (aq + r)(z \operatorname{div} q) + m[(z \operatorname{div} q) - (az \bmod m)] \\ &= a[z - (qz \operatorname{div} q)] - r(z \operatorname{div} q) + m\delta(z) \\ &= a(z \bmod q) - r(z \operatorname{div} q) + m\delta(z) \\ &= \gamma(z) + m\delta(z), \end{aligned}$$

dove: $\gamma(z) = a(z \bmod q) - r(z \operatorname{div} q)$ e $\delta(z) = (z \operatorname{div} q) - (az \bmod m)$.
Può essere dimostrato che se $r < q$ allora per ogni z in 1,2,...,$m - 1$ valgono:

[*i*] $\delta(z)$ assume solo valori 0 o 1;

[*ii*] sia $a(z \bmod q)$ che $r(z \operatorname{div} q)$ assumono valori interi in $\{0, \ldots, m - 1\}$;

[*iii*] $|\gamma(z)| \leq m - 1$.

[*iii*] deriva banalmente da [*ii*], la quale a sua volta è facilmente dimostrabile ricorrendo alla scomposizione di m sopra proposta (ricordando che $z \bmod q < q$ e $a \cdot q < m$) ed al fatto che $r < q$ ($r(z \operatorname{div} q) < z$). [*i*] invece deriva dal fatto che, dovendo essere $1 \leq f(z) \leq m - 1$, δ può assumere unicamente i valori sopra enunciati, in modo da riportare il valore calcolato attraverso l'algoritmo all'interno del (co)dominio della funzione.

Si evince dunque che il punto di forza del metodo di Schrage sta nell'intrappolare l'operazione che causa l'overflow all'interno di $\delta(z)$ fornendo una tecnica per derivarne il valore, attraverso la conoscenza di $\gamma(z)$, evitando un dispendioso calcolo diretto.

2.3 Metodi lineari

2.3.1 Linear congruential generators (LRG) e multiple recursive generators (MRG)

I più famosi ed ancora largamente usati tipi di generatori sono i linear congruential generators (LCG), lo stato al passo n è un intero x_n e la funzione di transizione è definita dalla ricorsione:

$$x_n = (ax_{n-1} + c) \bmod m, \tag{2.1}$$

dove $m > 0$, $a > 0$ e c sono interi chiamati *modulo*, *moltiplicatore* e *costante additiva*. La funzione di output invece solitamente si occupa solamente di normalizzare lo stato all'intervallo [0,1] $u_n = G(x_n) = x_n/m$. Un'opportuna scelta dei parametri consente di disporre di periodi lunghi al più m [11].

Si consideri ora la ricorsione:

$$x_n = (a_1 x_{n-1} + ... + a_k x_{n-k}) \bmod m, \tag{2.2}$$

dove l'*ordine* k ed il *modulo* m sono interi positivi, mentre i *coefficienti* $a_1, ..., a_k$ sono interi nell'intervallo $[-(m-1), (m-1)]$. Detto $\mathbb{Z}_m$ l'insieme $\{0,1,...,m-1\}$ nel quale si eseguono le operazioni *modulo* m, lo stato al passo n del multiple recursive generator (MRG) è il vettore $s_n = (x_n, ..., x_{n+k-1}) \in \mathbb{Z}_m^k$. La funzione di uscita è definita da $u_n = G(s_n) = x_n/m$ [11].

Sia $P(z) = z^k - a_1 z^{k-1} - ... - a_k$ il polinomio caratteristico di (2.2). In questo caso il periodo massimo risulta $\rho = m^k - 1$, ottenuto se e sole se m è primo e P è un polinomio primo su $\mathbb{Z}_m$, identificato ora come un campo con m elementi.

Si supponga m primo e sia $r = (m^k - 1)/(m - 1)$. Il polinomio P è primitivo su $\mathbb{Z}_m$ se e solo se soddisfa le seguenti condizioni (dove ogni richiesta è supposta essere modulo m) [16]:

1. $[(-1)^{k+1} a_k]^{m-1/q} \neq 1$ per ogni fattore primo q di $m - 1$;
2. $z^r \bmod P(z) = (-1)^{k+1} a_k$;
3. $z^{r/q} \bmod P(z)$ ha grado > 0 per ogni fattore primo q di r, $1 \leq q \leq r$.

Nel caso $k = 1$ ci si riconduce agli LCG (con $c = 0$), dove le suddette condizioni si semplificano all'imporre $a \bmod m \neq 0$ e $a^{m-1/q} \bmod m \neq 1$ ([i]).

Per r più elevati trovare i fattori q che soddisfino la terza imposizione risulta sempre più difficile, dunque il trucco per avere polinomi primitivi sul campo, e conseguentemente sequenze a periodo massimo, sta nello scegliere m e k in modo che r sia primo.

In caso di m non primo la lunghezza del periodo risulta nettamente inferiore a $m^k - 1$. Ad esempio $k = 1$, $m = 2^e$, $e \geq 4$ il periodo massimo è 2^{e-2}, invece per $k \geq 1$, $m = p^e$, p primo, $e \geq 1$ il limite superiore raggiunge $(p^k - 1)p^{e-1}$.

Dunque imporre $p = 2$, sebbene assicuri un'agevole implementazione, condanna a periodi assai limitati. Per rimediare a questi inconvenienti è possibile considerare una costante additiva c come in (2.1). In queste circostanze Gli LCG hanno periodo di lunghezza m se e solo se [16]:

1. c ed m sono primi tra loro;
2. $a - 1$ è un multiplo di p per ogni fattore primo p di m (incluso m stesso se primo);
3. se m è multiplo di 4 allora lo è anche $a - 1$.

Una costante c può essere aggiunta anche al punto giusto della ricorsione (2.2) di ordine k, garantendo un comportamento del tutto equivalente ad una di ordine $k + 1$, cioè periodo limitato superiormente da $(p^{k+1} - 1)p^{e-1}$, nel caso di $m = p^e$, che risulta molto minore di m^k per e e k elevati, come si verifica nelle situazioni pratiche. La soluzione migliore risulta perciò quella di soddisfare le condizioni per la primitività del polinomio sul campo $\mathbb{Z}_m$ ed adattare ai vari casi il metodo proposto da Schrage del paragrafo precedente.

2.3.2 Salti in avanti

Alle volte può essere pratico saltare avanti, questo contribuisce a rimescolare la sequenza generata diminuendo il livello di correlazione tra i valori ottenuti, abbandonando la regolarissima struttura reticolare troppo uniforme. Nel caso di multiplicative liear congruential generators (MLCG), di fatto LCG con $c = 0$, per saltare avanti da x_n ad $x_{n+\nu}$ è sufficiente usare la relazione [11]:

$$x_{n+\nu} = a^\nu x_n \bmod m = (a^\nu \bmod m) x_n \bmod m,$$

dove in caso di salti di uguale ampiezza ν, la costante $a^\nu \bmod m$ può essere precalcolata riportando il problema al caso base.

Nel caso di LCG con $c \neq 0$ invece è possibile ricorrere alla formula di complessa implementazione:

$$x_{n+\nu} = \left(a^{\nu} x_n + \frac{c(a^{\nu} - 1)}{a - 1} \right) \bmod m.$$

Avendo a che fare con MRG, conviene considerare il problema da un altro punto di vista, quello matricale. Sia A una matrice $k \times k$ di coefficienti ed X_n un vettore colonna che rappresenti lo stato s_n. Di fatto si ha a che fare con un MLCG matriciale:

$$X_{n+\nu} = A^{\nu} X_n \bmod m = (A^{\nu} \bmod m) X_n \bmod m.$$

2.3.3 Struttura reticolare di LCG ed MRG

Come già presentato al paragrafo precedente, e ricordato pocanzi, LCG ed MRG presentano strutture assai definite che causano la superuniformità di cui si parlava, troppo regolare per fingersi aleatorietà. Un reticolo di dimensione t, nello spazio reale $t-$dimensionale $\mathbb{R}^t$, è un insieme della forma:

$$L = \left\{ V = \sum_{j=1}^{t} z_j V_j \mid z_j \in \mathbb{Z},\ \forall\, j \in \{1, 2, ..., t\} \right\},$$

dove $\mathbb{Z}$ è l'insieme di tutti gli interi e $\{V_1, ..., V_t\}$ è una base di $\mathbb{R}^t$. Di fatto L altro non è che un insieme di tutte le combinazioni lineari intere dei vettori $\{V_1, ..., V_t\}$, detti *base reticolare* di L. La base $\{W_1, ..., W_t\}$ di $\mathbb{R}^t$ che soddisfa $V_i^T W_j = \delta_{ij}\ \forall\, 1 \leq i, j \leq t$ (dove T indica la trasposizione e δ_{ij} la delta di Kroenecher) è detta *base duale* di $\{V_1, ..., V_t\}$ ed il reticolo generato da questa nuova base è detto reticolo duale [11].

Si consideri l'insieme

$$T_t = \{U_n = (u_n, ..., u_{n+t-1}) \,|\, n \geq 0,\ s_0 = (x_0, ..., x_{k-1}) \in \mathbb{Z}_m^t\}$$

di tutte le $t-$tuple di successivi valori prodotti da (2.2), con $u_n = x_n/m$. Fondamentalmente si tratta dell'intersezione tra un reticolo L_t ed un iper-cubo unitario t-dimensionale. Per $t \leq k$ risulta chiaro dalla definizione di T_t che ogni vettore $(x_0, ..., x_{t-1}) \in \mathbb{Z}_m^t$ può essere preso come s_0, dunque $T_t = \mathbb{Z}_m^t/m = (\mathbb{Z}_m^t/m) \cap I^t$; cioè L_t è l'insieme di tutti i punti le cui coordinate siano multiple di $1/m$ e T_t è l'insieme degli m^t punti in L_t le cui coordinate appartengano a

$\{0, 1/m, ..., (m-1)/m\}$ (ovvero interni all'iper-cubo unitario). Nel caso invece $t > k$, l'insieme T_t contiene solo m^k degli m^t punti di $\mathbb{Z}_m^t/m$. Dunque, detto $h = \min\{t, k\}$, T_t racchiude in sé solo m^t elementi. Per MRG a periodo completo, il generatore copre tutto T_t, eccetto lo stato zero, in un ciclo, negli altri casi solo una piccola parte di esso, che a sua volta potrebbe essere un *sottoreticolo* di L_t.

In generale, la struttura reticolare implica che i punti di T_t giacciano su famiglie di iper-piani paralleli. Tra tutte queste famigli si è particolarmente interessati a quella per cui l'iperpiano successivo sia il più lontano possibile. La distanza d_t tra questi iper-piani costituisce un parametro ottimale per misurare la bontà del reticolo scelto: tanto più sarà elevata tanto peggio saranno distribuiti i punti nell'iper-spazio, concentrandosi sugli iper-piani suddetti e lasciando spesse fette vuote di spazio tra essi. Essendo $d_t = 1/l_t$ dove l_t è la lunghezza del più corto vettore non nullo dello spazio duale ad L_t, si può ricondurre il problema del calcolo di d_t ad una minimizzazione di quadrati di variabili intere, per ragioni storiche tale metodo prende il nome di *test spettrale* [11].

Tale distanza non può essere piccola a piacere, ma presenta un limite teorico inferiore d_t^*, dovuto al fatto che il diminuire di una dimensione causa come ovvia conseguenza l'aumentare di un'altra, dovendo essere l'iper-cubo unitario divisibile in m^h iper-volumi equivalenti. Si ha dunque:

$$d_t \geq d_t^* = \frac{1}{\gamma_t m^{k/t}},$$

dove γ_t è una costante che dipende solamente da t ed il cui valore è al momento noto solo per bassi valori di t. $S_t = d_t^*/d_t$ è una figura di merito che indica la bontà della distanza massima e dunque della distribuzione delle componenti della sequenza (tanto più si avvicina ad 1). Un ulteriore limite è fornito da:

$$d_t \geq \left(1 + \sum_{j=1}^{k} a_j^2\right)^{-1/2}.$$

2.3.4 Indici lacunosi

Costruire vettori con valori non consecutivi nella sequenza ma separati da distanze fissate può contribuire a rendere meno correlato l'output generato [11]. Sia $I = \{i_1, i_2, ..., i_t\}$ un insieme dato di indici e si definisca un reticolo per un MRG:

$$T_t(I) = \{(u_{i_1+n}, \ldots, u_{i_t+n}) \,|\, n \geq 0,\, s_0 = (x_0, \ldots, x_{k-1}) \in \mathbb{Z}_m^k\}.$$

Si consideri il reticolo $L_t(I)$ esteso da $T_t(I)$ e $\mathbb{Z}^t$,e sia $d_t(I)$ la distanza massima tra due iper-piani nel reticolo. Possono essere dati dei limiti inferiori indicativi, "rapidi-e-sporchi":

1. se I contiene tutti gli indici i tali che $a_{k-i+1} \neq 0$, allora

$$d_t(I) \geq \left(1 + \sum_{j=1}^{k} a_i^2\right)^{-1/2}.$$

2. se m può essere scritto come $m = sum_{j=1}^{t} c_{i_j} a^{i_j}$, per alcuni interi c_{i_j}, allora

$$d_t(I) \geq \left(1 + \sum_{j=1}^{t} c_{i_j}^2\right)^{-1/2}.$$

2.3.5 LCG ed MRG combinati

Molti autori suggeriscono la possibilità di combinare in qualche maniera differenti tipi di generatori, sperando che questo possa rendere il risultato migliore delle sue componenti. Per generatori fisici, e dunque veramente casuali, la combinazione porta ad un effettivo miglioramento della sequenza, permettendo di annullare tra loro più errori sistematici. Per quanto riguarda invece algoritmi quali quelli esposti precedentemente, nulla è prevedibile senza un'analisi strutturale diretta del generatore combinato stesso. Trattandosi infatti di processi deterministici, deterministicamente collegati tra loro, la casualità non ha modo di aumentare, l'unico effetto apprezzabile è l'incremento del periodo il quale tipicamente porta con sé un miglioramento nel comportamento statistico [11].

I due approcci più noti e largamente diffusi sono:

1. mescolare una sequenza con un'altra o con se stessa.

2. sommare due o più sequenze intere modulo qualche intero m_0, o sequenze di numeri reali in [0,1] modulo 1, o sommare numeri binari bit a bit modulo 2.

Il primo metodo può essere implementato in questa maniera, mescolando due LCG: si costruisce una tabella che associ ai numeri da 1 a d (in genere multiplo di 2) i primi d valori della sequenza generata dal primo LCG. Considerando i $\log_2 d$ bit più significativi dell'output successivo della seconda sequenza si ottiene

un indice $I \in \{1, \ldots, d\}$, che verrà utilizzato per sostituire il valore successivo del primo generatore alla posizione indicate da I, dopo aver prelevato, e fornito come output, il valore precedentemente immagazzinato. Di fatto il primo LCG si occupa di generare la sequenza, mentre il secondo, di mescolarla. Ci possono essere parecchie varianti all'esempio presentato, tuttavia questo approccio presenta invariabilmente due svantaggi: (1) gli effetti del mescolamento non è stato compreso abbastanza bene dal punto di vista teorico, e (2) non è possibile saltare avanti rapidamente ad un punto preciso della sequenza.

Si ha invece maggior padronanza della seconda strategia e per di più saltare avanti nella sequenza combinata equivale a saltare avanti in tutte le sue componenti ed operare il ricongiungimento solo all'ultimo passo.

Si considerino J MRG in parallelo, il j-esimo è basato sulla ricorsione

$$x_{j,n} = (a_{j,1}x_{j,n-1} + \ldots + a_{j,k}x_{j,n-k}) \bmod m_j,$$

per $j = 1, \ldots, J$. Supponendo che i moduli m_j siano primi tra loro e che ogni ricorsione abbia periodo ρ_j puramente periodico (senza transitorio) e siano $\delta_1, \ldots, \delta_J$ interi arbitrari tali che, per ogni j, δ_j ed m_j non abbiano fattori comuni. Entrambe le seguenti combinazioni portano a risultati soddisfacenti [11]:

$$z_n = \left(\sum_{j=1}^{J} \delta_j x_{j,n}\right) \bmod m_1 \qquad u_n = z_n/m_1 \tag{2.3}$$

$$w_n = \left(\sum_{j=1}^{J} \delta_j \frac{x_{j,n}}{m_j}\right) \bmod 1 \tag{2.4}$$

Siano $k = \max(k_1, \ldots, k_J)$ ed $m = \Pi_{j=1}^{J} m_j$. Valgono i seguenti risultati:

1. le sequenze $\{u_n\}$ e $\{w_n\}$ hanno entrambe periodo $\rho = \mathrm{lcm}(\rho_1, \ldots, \rho_J)$ (minimo comune multiplo delle lunghezze dei periodi delle componenti);

2. w_n obbediscono alla ricorsione

   $$x_n = (a_1 x_{n-1} + \ldots + a_k x_{n-k}) \bmod m \qquad w_n = x_n/m, \tag{2.5}$$

   dove gli a_i possono essere calcolati con una formula che non li lega ai δ_j, ma solo ai generatori componenti.

3. u_n e w_n possono non coincidere, anzi la cosa è molto frequente, tuttavia sono legati dalla relazione $u_n = w_n + \epsilon_n$, con $\Delta^- \leq \epsilon_n \leq \Delta^+$, dove Δ^-

e Δ^+ sono generalmente estremamente piccoli quando i moduli m_j sono sufficientemente vicini tra loro.

(2.3) e (2.4) possono essere viste come una maniera efficiente di implementare un MRG con modulo m molto elevato. Le varie componenti possono essere scelte con due soli coefficienti a_{ij} diversi da zero, entrambi soddisfacenti $a_{ij}(m_j \bmod a_{ij}) < m_j$ per motivi pratici (di fatto la funzione è la medesima di imporre $r < q$ nel metodo di Schrage). Nel caso di m_j primo dispari ed in grado di fornire periodo massimo $\rho_j = m^{k_j} - 1$, ogni ρ_j è pari dunque $\rho \leq (m_1^{k_1} - 1) \cdots (m_J^{k_J} - 1)/2^{J-1}$ risulta il limite teorico superiore nel caso in cui i fattori $(m_j^{k_j} - 1)/2$ siano primi tra loro.

2.3.6 Generatori di Tausworthe

Tausworthe mise a punto un generatore basato su suequenze di 0 e 1 generate dalla ricorsione [15]:

$$b_i = (a_p b_{i-p} + \ldots + a_1 b_{i-p+q}) \bmod 2$$

dove le variabili appartengono all'insieme $\{0, 1\}$. Per questioni di efficienza la maggior parte dei coefficienti a assume valore nullo. La ricorsione si presenta dunque generalmente nella forma (ponendo unitarie le a non nulle):

$$b_i = (b_{i-p} + b_{i-p+q}) \bmod 2. \tag{2.6}$$

A questo punto la sequenza generata viene divisa in $l-$uple, interpretabili come numeri in base 2, dunque anche in base 10, a seguito di un'opportuna conversione.

Siano ad esempio $p = 4$, $q = 3$ $l = 4$ e i primi valori della sequenza 1, 0, 1, 0. Applicando 2.6 si ottiene:

$$1, 1, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 1, 0, 1, \ldots$$

che corrisponde, per $l = 4$, a 12, 8, 15, 5, ...

Come per gli LCG differenti scelte dei parametri, ed in caso anche del seme, portano a risultati più o meno soddisfacenti.

2.4 Metodi non-lineari

Per eliminare la struttura regolare è possibile procedere in due maniere differenti:

1. mantenere una funzione di transizione (T) lineare, ma adoperarne una non-lineare di uscita (G);

2. utilizzare una funzione di transizione non-lineare.

I generatori non-lineari possiedono l'apprezzabile caratteristica di non avere una struttura reticolare, facendo sembrare la sequenza veramente casuale. In sostanza vi sono dei limiti inferiori e superiori alla discrepanza il cui ordine asintotico (al tendere all'infinito della lunghezza del periodo) è lo stesso di una sequenza di variabile aleatorie iid $U(0,1)$. Inoltre hanno ottenuto risultati soddisfacenti anche nei test fino ad ora eseguiti.

2.4.1 Inversive congruential generators (ICG)

Una prima idea per costruire un generatore non-lineare è quella di aggiungere una distorsione non-lineare alla funzione di uscita (tipologia 1) [11]. Si consideri ad esempio il caso di un MRG a periodo massimo $\rho = m^k - 1$ con modulo primo m; si rimpiazzi la nota $u_n = x_n/m$ con:

$$z_n = (\bar{x}_{n+1}\bar{x}_n^{-1}) \bmod m \qquad \text{e} \qquad u_n = z_n/m,$$

dove $\bar{x}_i$ rappresenta l'i-esimo valore nella sequenza $\{x_n\}$ e $\bar{x}_n^{-1}$ è l'inverso di $\bar{x}_n$ rispetto all'operazione mod$m$. Per $x_n \neq 0$, $x_n^{-1} = x_n^{m-2} \bmod m$. La sequenza $\{z_n\}$ presenta periodo m^{k-1}. Per $k = 2$ vale la ricorsione

$$\begin{aligned} z_n &= (a_1 + a_2 z_{n-1}^{-1}) \bmod m \quad \text{se } z_{n-1} \neq 0; \\ z_n &= a_1 \qquad\qquad\qquad\qquad\quad \text{se } z_{n-1} = 0, \end{aligned}$$

dove a_1 e a_2 sono i coefficienti dell'MRG.

Un approccio più diretto è garantito dal metodo *inversivo esplicito*: sia $x_n = an + c$ per $n \geq 0$, $c \in \mathbb{Z}_m$ ed m primo. Si definisca:

$$z_n = x_n^{-1} = (an + c)^{m-2} \bmod m \qquad \text{e} \qquad u_n = z_n/m,$$

questa sequenza ha periodo $\rho = m$.

Sono possibili anche ICG a modulo potenza-di-2, tuttavia, in questi casi, appare molto più evidente la presenza di una struttura regola re alla base. L'idea di combinare più generatori può essere applicata anche al caso non-lineare offrendo, tra le altre cose, vantaggi computazionali. Si prendano ad esempio J generatori inversivi con moduli distinti $m_1, dots, m_J$, tutti maggiori di 4, ed a periodo massimo $\rho_j = mJ$. La formula

$$u_n = (u_{1,n} + \ldots + u_{J,n}) \bmod 1$$

fornisce una sequenza $\{u_n\}$ equivalente all'uscita di un generatore inversivo avente modulo $m = m_1 \cdots m_J$ e lunghezza del periodo $\rho = m$. Concettualmente tutto ciò risulta molto simile a quanto ricavato nel caso della combinazione di generatori lineari.

2.4.2 Quadratic Congruential Generators (QCG)

Si supponga di avere a disposizione una funzione di transizione quadratica anziché lineare (tipologia 2), ad esempio la ricorsione:

$$x_n = (ax_{n-1}^2 + bx_{n-1} + c) \bmod m,$$

dove $a, b, c \in \mathbb{Z}_m$ e $x_n \in \mathbb{Z}_m \, \forall \, n$. Nel caso di m potenza-di-2 questo generatore ha periodo massimo se e solo se si verificano: a pari, $(b - a) \bmod 4 = 1$ e c dispari. Tuttavia i punti t-dimensionali generati in questo modo risultano giacere in un unione di griglie; inoltre la discrepanza tende ad assumere valori eccessivamente elevati: per queste ragioni conviene evitare moduli di questo tipo, nonostante le semplificazioni a livello implementativo [11].

2.5 Distribuzioni normali

Quasi tutti gli algoritmi per la generazioni di numeri secondo una variabile aleatoria normale sono basati sulla trasformazione di distribuzioni uniformi. La maniera logicamente più immediata prevede la somma dei valori ottenuti da p variabili aleatorie uniformi; il teorema centrale del limite infatti assicura, per p sufficientemente elevato, di ottenere una distribuzione normale di media $p/2$ e varianza $p/12$. Dunque la somma di 12 valori generati casualmente tramite distribuzioni uniformi conduce approssimativamente ad una $N(6, 1)$; a questo punto sottrarre

un valore costante 6 al numero casuale così generato permetterebbe di ottenere un $N(0,1)$. In realtà un approccio del genere appare davvero insoddisfacente per due motivi: innanzi tutto perché sono necessarie 12 uniformi per per generare una normale, denotando dunque una certa lentezza; in secondo luogo perche la limitatezza di p causa un pessimo comportamento nelle code della gaussiana [13].

Un'idea più elaborata e più valida è quella di convertire la distribuzione uniforme in una normale.

Lemma Sia $X \sim U(0,1)$ ed $F(\cdot)$, invertibile di inversa $F^{-1}(\cdot)$, una funzione di distribuzione, e sia $Y = F^{-1}(X)$, allora:

$$Y \sim F(\cdot) \text{ risulta una variabile aleatoria.}$$

risulta una variabile aleatoria.

Dimostrazione La dimostrazione consiste in un calcolo diretto di $F_Y(y)$.

$$\begin{aligned} F_Y(y) &= P[Y \leq y] = P[F^{-1}(X) \leq y] = P[F(F^{-1}(X)) \leq F(y)] \\ &= P[X \leq F(y)] = F(y), \end{aligned}$$

dove, nel primo passaggio della prima riga si è utilizzata l'invertibilità di $F(\cdot)$, e nell'ultimo della seconda riga il fatto che X sia una $U(0,1)$.

Capitolo 3

Test statistici

Testare statisticamente generatori di numeri casuali è un attività assolutamente empirica ed euristica. L'idea principale è quella di cercare delle situazioni in cui le uscite del generatore si comportino in maniera significativamente differente da quella prevista teoricamente. Testare unicamente l'uniformità o al più la correlazione tra coppie (terne, ecc...) di valori consecutivi è molto lontano dall'essere sufficiente; i test migliori ambiscono a smascherare proprietà di correlazione di ordine superiore o modelli geometrici di numeri successivi. Quali siano i test migliori non è noto a nessuno, tutto dipende dalle circostanze.

Esempio 1 Si supponga di voler generare n numeri casuali da un generatore che fornisca una sequenza di variabili aleatorie iid $U(0,1)$. Si ripeta il processo N volte, contando ognuna di esse la quantità T di casi in cui il valore estratto non supera $1/2$ [11]. Trattandosi di n binomiali di parametro $1/2$ (essendo la generazione uniforme), ci si aspetta che T segua il comportamento di una variabile aleatoria di Bernoulli di parametri $n/2$ ed $n/4$ (dunque deviazione standard $\sqrt{n}/2$). Nell'eventualità in cui: $N = 100$ ed $n = 10000$ si ottengono valori di media e deviazione standard rispettivamente di 5000 e 50. Il valore di n risulta sufficientemente elevato da giustificare di attendersi valori di T appiattiti verso la media in quasi tutti gli N esperimenti compiuti. In questo senso se 98 volte su 100 T dovesse risultare inferiore di 5000 si potrebbe pensare ad un errore sistematico che fa si che i numeri generati siano in prevalenza maggiori di $1/2$. Allo stesso modo se in 20 casi dovesse essere $T < 4800$ qualcosa dovrebbe metterci in guardia in quanto sarebbe in contrasto coi dati sulla deviazione standard. D'altra parte se T si comportasse come desiderato si potrebbe affermare, con relativa

tranquillità, di essere in possesso di un generatore in grado di superare i test a cui è stato sottoposto; Tuttavia nulla vieterebbe ad altri test statistici, relativi a qualche altro parametro di interesse, di condannarlo al fallimento.

3.1 Linee guida per la creazione di test significativi

Si definisce ipotesi nulla H_0, da verificare coi test, come: "L'uscita del generatore è una sequenza di variabili aleatorie iid $U(0,1)$". Di fatto tale ipotesi è falsa, trattandosi di algoritmi puramente deterministici (ad eccezione al più del seme) e periodici, tuttavia supponendo che queste caratteristiche non possano essere individuate in un tempo ragionevolmente basso, H_0 può essere assunta vera.

Un test statistico può essere definito da ogni funzione T, *statistica del test*, operante su un numero finito di variabili aleatorie $U(0,1)$, per cui la distribuzione sia nota o possa essere approssimata facilmente.

Una *procedura a singolo livello* calcola il valore di T in un caso particolare (poniamo che in questo risulti t_1), quindi calcola il p-value

$$\delta_1 = P[T > t_1 | H_0].$$

Sotto H_0, δ_1 dovrebbe risultare una variabile aleatoria $U(0,1)$, dunque, nel caso si operi un test bilaterale (unilaterale), valori di δ_1 troppo vicini a 0 e (o) 1 indicheranno l'inesattezza dell'ipotesi fatta. La decisione delle aree di rifiuto dipende da cosa precisamente voglia dimostrare il test [11].

Una *procedura a due livelli* richiede di essere in possesso di N copie di T, indicate con $T_1, \ldots, T_N$, dalle quali calcola la distribuzione *empirica* $\hat{F}_N$, che viene messa a confronto con la distribuzione *teorica* di T sotto H_0, detta F, tramite un test classico sulla bontà di adattamento, quale il Kolmogorov-Smirnov (KS) o l'Anderson-Darling (AD). Una versione del test di adattamento KS utilizza la statistica:

$$D_n = \sup_{-\infty<x<\infty} |\hat{F}_N(x) - F(x)|,$$

per il quale è disponibile un'approssimazione della distribuzione sotto H_0, posto F continua. Una volta noto il valore d_N della statistica D_N,di fatto una parametro

che indica di quanto si distacchi al massimo $\hat{F}_N$ da F, si calcola il p-value

$$\delta_2 = P[D_N > d_N | H_0],$$

il quale risulta una variabile aleatoria $U(0,1)$ sotto H_0. Questa volta l'ipotesi iniziale va respinta in caso δ_2 sia troppo vicino a 0 [11].

Scegliere $N = 1$ limita all'utilizzo di test a livello singolo. Non c'è un criterio che indichi quale sia la categorie di test vincente. Il più delle volte assumere $N = 1$, puntando su un campione sufficientemente elevato della statistica del test T, può essere un'ottima scommessa, altre volte, soprattutto se il calcolo di T cresce più che linearmente con la dimensione del campione, $N > 1$ permette un'analisi locale oltre che globale delle sequenze generabili.

Ogniqualvolta il valore-p calcolato risulti sospetto è buona norma aumentare il campione preso in esame o ripetere il test con altri segmenti della sequenza. Nel caso H_0 non venga respinto aumenta l'affidamento riposto sul generatore, tuttavia non è da escludere che il test successivo possa evidenziarne i limiti.

3.2 Due esempi di test empirici

Si considerino n vettori, con componenti non sovrapposte tra loro, nell'iper-cubo $[0,1)^t$

$$P_t = \{\bar{U}_i = (U_{t(i-1)}, \ldots, Uti - 1),\ i = 1, \ldots, n\},$$

dove $U_0, U_1, \ldots$ è l'uscita del generatore. Supposta vera H_0, P_t contiene n vettori aleatori iid uniformemente distribuiti sull'iper-cubo unitario.

Esempio 2 (*Serial Test*) [11] Si costruisca una (t,l)-equidissezione, in base 2, dell'iper-cubo (lo si suddivida in $k = 2^{tl}$ celle di uguali dimensioni). Sia X_j il numero di punti $\bar{U}_i$ che cadono all'interno della cella j, per $j = 1, \ldots, k$ e si definisca la statistica chi-quadro

$$X^2 = \sum_{j=1}^{k} \frac{(X_j - n/k)^2}{n/k}.$$

Sotto H_0 la media e la varianza teoriche di X^2 risultano rispettivamente $\mu = k-1$ e $\sigma^2 = 2(k-1)(n-1)/n$. Inoltre, per $n \to \infty$ e k fissato, X^2 converge in distribuzione ad una variabile aleatoria chi-quadro con $k-1$ gradi di libertà,

mentre per $n \to \infty$ e $k \to \infty$ simultaneamente, in modo che $n/k \to \gamma$, per qualche costante γ, $(X^2 - \mu)/\sigma$ converge in distribuzione ad una $N(0,1)$, giustificando l'approssimazione normale nel caso $k \gg n$.

p-value molto vicini ad 1 indicano che gli n punti sono troppo distribuiti tra le k celle rispetto a quanto ci si potrebbe aspettare da punti generati casualmente (X^2 è troppo piccolo). d'altra parte valori troppo prossimi a 0 indicano che alcune celle sono più frequenti di altre (X^2 troppo piccolo).

Esempio 3 (*Close-pairs Test*) [11] Sia $D_{n,i,j}$ la distanza euclidea tra i punti $\bar{U}_i$ e $\bar{U}_j$ nel toro unitario (parlando di toro, punti che giacciono nelle vicinanze di facce opposte dell'iper-cubo devono essere considerati vicini). Per $s \geq 0$, sia $Y_n(s)$ il numero di coppie di punti $i < j$ tali che $D^t_{n,i,j} V_t n(n-1)/2 \leq s$, dove V_t rappresenta l'iper-volume dell'iper-sfera di raggio unitario nello spazio reale t-dimensionale. Sotto H_0, per ogni costante $s_1 > 0$, per $n \to \infty$, il processo $\{Y_n(s),\, 0 \leq s \leq s_1\}$ converge debolmente ad un processo di Poisson con tasso unitario. Siano $0 = T_{n,0} \leq T_{n,1} \leq T_{n,2} \leq \cdots$ i tempi di arrivo del processo e $W_{n,i} = 1 - exp[-(T_{n,i} - T_{n,i-1})]$. Fissato un parametro m ed avendo a disposizione n sufficientemente elevato, le variabili aleatorie $W_{n,1}, \ldots, W_{n,m}$ sono approssimativamente iid $U(0,1)$ nell'ipotesi H_0. Per confrontare la loro distribuzione empirica con quella uniforme si può calcolare, per esempio, la statistica di Anderson-Darling

$$A^2_m = -m - \frac{1}{m} \sum_{i=1}^{m} \{(2i-1)\ln(W_{n,i}) + (2m+1-2i)\ln(1-W_{n,i})\}$$

e bocciare H_0 nel caso in cui il valore-p sia troppo piccolo (A^2_m troppo elevato).

p-value inferiori a 10^{-15} indicano che i salti del processo Y_n tendono ad risultare raggrumati, se non addirittura sovrapposti. La struttura reticolare infatti impone $D_{n,i,j}$ molto piccoli e frequentemente molto simili tra loro: questa è la causa della vicinanza dei salti. dalla definizione di $W_{n,i}$ si nota che parecchi di essi risultano molto vicini a 0 e la statistica A^2_m è molto sensibile a catturare problemi di questo tipo.

3.3 Test empirici: riassunto

Negli anni, l'esperienza di differenti tipi di test su differenti tipi di generatori ha portato alla stesura di alcune semplici regole:

1. generatori con periodo inferiori a 2^{32}, in particolar modo LCG, falliscono la maggior parte dei test e non dovrebbero essere utilizzati;

2. moduli potenza-di-2 rendono molto più facile il lavoro ad un software che cerchi di capire se la sequenza generate sia veramente casuale o solo pseudo-casuale;

3. generatori combinati con periodo elevato e buone proprietà strutturali ottengono elevati punteggi nei test;

4. in generale generatori con buone figure di merito (e.g. buona struttura reticolare, buon equidistribuzione sull'intero periodo se è selezionata solo un sottoinsieme di esso) ottengono elevati punteggi nei test;

5. generatori con ricorsioni più complicate e migliori proprietà teoriche si comportano meglio nei test.

Capitolo 4

Metodo Monte Carlo

4.1 Introduzione

Con metodo (di) Monte Carlo si intende una classe di algoritmi computazionali che, a partire da un insieme di numeri generati casualmente o pseudo-casualmente, permetta di ottenere un risultato medio molto vicino al comportamento teorico. Prevede dunque la risoluzione di complicati problemi deterministici con metodi probabilistici. Prende particolarmente piede, in campo scientifico, laddove l'osservazione diretta del fenomeno non sia più sufficiente per arrivare alla formulazione di un modello teorico (e.g. fisica subatomica). Prevale dunque l'idea di eseguire ripetutamente uno stesso esperimento, di volta in volta il più diverso possibile da se stesso, ricorrendo alla generazione (pseudo-)casuale di un certo numero di valori attribuiti alle condizioni iniziali e alle successive, eventuali, interazioni tra gli elementi considerati. Il passaggio successivo prevede l'unificazione dei risultati ottenuti attraverso un qualche tipo di media definito a priori. Di fatto il metodo è riconducibile, nella maggior parte delle circostanze, a 4 fasi successive [1]:

- definizione di un possibile dominio (anche multidimensionale) dei dati di input;
- generazione casuale dei dati di input tramite una variabile aleatoria definita sul domino;
- computazione deterministica dei vari output per ogni vettore di input;
- aggregazione dei risultati ottenuti.

Questa tecnica trova particolare impiego in vari ambiti scientifici, tra cui la simulazione di sistemi a molti di gradi di libertà (e.g. fluidi, strutture cellulari), la modellazione di fenomeni ad elevata incertezza nei dati (e.g. calcolo del rischio in affari) o la valutazione di integrali multidimensionali in un dominio complicato.

Appare evidente, dalla legge dei grandi numeri, che le conclusioni possano dirsi tanto più precise quanto maggiore ed eterogeneo risulti il campione di dati analizzato. Tuttavia l'incremento della mole degli input provoca un conseguente rallentamento delle operazioni di calcolo, motivo per cui il metodo non è stato adoperato su larga scala prima dell'avvento del computer elettronico.

4.2 Storia

Nel 1945 due importanti avvenimenti scossero il mondo scientifico: il successo nel test di Almagordo e la creazione del primo computer elettronico (ENIAC), gettando le basi per il clamoroso ritorno di una tecnica conosciuta ai matematici della vecchia guardia col nome di campionamento statistico. Se è vero che Enrico Fermi fosse già pratico con questo approccio, come dimostrano i suoi studi sulla diffusione di neutroni degli anni '30, è anche vero che a Los Alamos non vi si concentrò particolarmente. Furono invece principalmente altri suoi colleghi del Progetto Manhattan ad occuparsene con risultati notevoli, sebbene il suo aiuto non sia mancato in un secondo momento [2].

La leggenda vuole che nel 1946 Stanislaw Ulam, convalescente dopo una malattia, giocasse spesso ad un solitario [1]. Incuriosito dal sapere in quale percentuale dei casi il gioco si concludesse con esito positivo, il matematico polacco iniziò una lunga serie di calcoli per determinare la quantità desiderata. Sebbene avesse a disposizione tutti i dati possibili la lunghezza dei conti era scoraggiante, tanto da farlo propendere per un metodo più pratico: ripetere il solitario un numero elevato di volte, segnando i successi. Resosi conto della somiglianza delle difficoltà in cui era incorso con quelle del problema della diffusione di neutroni in materiali fissili e delle enormi prospettive offerte da ENIAC in quanto a potenza di calcolo, sottopose la sua trovata all'attenzione di John Von Neumann, il quale, intravedendo il potenziale dell'intuizione del collega, si adoperò per creare un progetto attorno all'idea in questione.

Fu Nicholas Metropolis a suggerire il nome Monte Carlo, giocando anche sul

fatto che uno zio di Ulam chiedesse denaro ai familiari per andare a giocarlo al Casinò omonimo nel Principato di Monaco [2]. Apparve subito chiaro che si trattava di una buona idea sia perché risultava essere il metodo più rapido di agire, sia perché quando falliva lo faceva in maniera spettacolare, non lasciando dubbi sull'esito. A questo proposito si notò una forte dipendenza dell'affidabilità dei risultati dalla scelta di un generatore di numeri casuali adeguato.

Il metodo Monte Carlo risultò centrale per le simulazioni richieste dal Progetto Manhattan nonostante la limitatezza degli strumenti computazionali di supporto. Dagli anni '50 furono utilizzati nei lavori di sviluppo della bomba a idrogeno e si diffusero nei campi della fisica, fisica chimica e ricerca operativa.

4.3 Applicazioni

I metodi Monte Carlo hanno trovato larga diffusione in vari settori, scientifici e non [1]. A fianco dei prevedibili utilizzi in fisica (metodi quantistici), astrofisica (modelli di evoluzione di galassie), biologia (sistemi biologici), finanza (calcolo del valore di una compagnia), ingegneria (variazioni in circuiti integrati), molti altri settori si sono aperti alla possibilità di sfruttare un approccio di tipo statistico; è il caso di videogiochi, meteorologia, progettazione grafica, ecc...

Ad ogni modo gli esempi più pratici ed efficaci per iniziare quantomeno a comprendere la potenza del metodo Monte Carlo si riferiscono alle scienze matematiche (geometria euclidea e cartesiana).

Esempio 1 (Stima di π) [3] Si voglia ottenere una stima di π usando il metodo Monte Carlo, ponendo di essere in grado di generare casualmente punti nel quadrato $\mathcal{Q} = [-1, 1] \times [-1, 1]$. Si supponga inoltre che la probabilità di ottenere un punto interno alla regione $\mathcal{R} \subset [-1, 1] \times [-1, 1]$ dipenda solamente dall'area di tale regione. Si tratta del caso in cui le coordinate del punto siano due variabile aleatorie i.i.d. uniformi nell'intervallo [-1,1].

Si calcoli la probabilità nel caso in cui $\mathcal{R}$ coincida col cerchio di raggio unitario centrato in (0,0), detto $\mathcal{C}$:

$$P_{\mathcal{C}} = \frac{area_{\mathcal{C}}}{area_{\mathcal{Q}}} = \frac{\iint_{\{x^2+y^2\leq 1\}} 1\, dxdy}{\iint_{\{-1\leq x,y\leq 1\}} 1\, dxdy} = \frac{\pi}{2 \cdot 2} = \frac{\pi}{4},$$

ovvero,

$$\pi = 4 \cdot P_{\mathcal{C}}.$$

Definita Z una variabile aleatoria binomiale di parametri n e p, ($Z \sim B(n,p)$), con n numero di lanci all'interno del quadrato e p probabilità che un lancio cada all'interno del cerchio, cioè $p = P_{\mathcal{C}}$. Ricorrendo ad una stima a massima verosimiglianza è possibile ottenere un valore indicativo di p, e dunque di π, tanto più precisi quanto più n elevato (Legge dei grandi numeri).

$$\hat{p} = \frac{Z}{n} \qquad\qquad \hat{\pi} = 4 \cdot \hat{p} = 4 \cdot \frac{Z}{n}.$$

Esempio 2 (Calcolo integrale) [3] Si voglia calcolare l'integrale (in una dimensione, per semplicità) della funzione $y = f(x)$ per $a \leq x \leq b$. Una tecnica utilizzabile consiste nel generare una sequenza di numeri casuali interna al dominio di integrazione $[a, b]$ e procedere calcolando il valore della funzione $f(x)$ nei punti estratti. La loro media porge un valore identificabile come l'altezza di un rettangolo di base b-a e di area approssimabile a quella sottostante $f(x)$ (per il teorema della media integrale).

$$\int_a^b f(x)\,dx = (b-a) \cdot \bar{f}_n(x),$$

dove con $\bar{f}_n(x)$ si intende la media degli n valori $f(x_i)$, generati casualmente rispettando $x_i \in [a, b]$, per $i = 1, ..., n$. Tale tecnica non risulta apprezzabilmente vantaggiosa, rispetto ad un approccio classico, nel caso trattato, tuttavia con l'incremento della dimensione del dominio di integrazione acquisisce via via maggiore autorità. Il caso di domini multidimensionali è una generalizzazione di quello più semplice presentato nell'esempio. Questo metodo di risoluzione richiede che i valori generati casualmente siano il più possibile equamente distribuiti nel dominio di generazione, cercando di evitare la presenza di zone d'ombra all'aumentare del numero di dimensioni, come precedentemente accennato.

Bibliografia

[1] *Monte Carlo method*, Wikipedia (http://en.wikipedia.org/wiki/Monte_Carlo_method), [23 settembre 2012].

[2] N. Metropolis, *The beginning of the Monte Carlo method*, Los Alamos Science Special Issue, 1987.

[3] I. A. Cosma, L. Evers, *Review Course: Markov Chains and Monte Carlo Methods*, lecture notes, capitolo 2, (http://users.aims.ac.za/∼ioana/).

[4] *A Million Random Digits with 100.000 Normal Deviates*, Wikipedia (http://en.wikipedia.org/wiki/A_Million_Random_Digits), [23 settembre 2012].

[5] Tom Jennings, *Book review: A Million Random Digits with 100.000 Normal Deviates*, (http://www.wps.com/projects/million/index.html), [23 settembre 2012].

[6] RAND Corporation *A Million Random Digits with 100.000 Normal Deviates*, Free Press Publishers, Glencoe (Illinois), 1955.

[7] *Middle-square method*, Wikipedia (http://en.wikipedia.org/wiki/Middle_square_method), [23 settembre 2012].

[8] *Random number generation*, Wikipedia (http://en.wikipedia.org/wiki/Random_number_generation), [23 settembre 2012].

[9] *Lehmer random number generator*, Wikipedia (http://en.wikipedia.org/wiki/Lehmer_random_number_generator), [23 settembre 2012].

[10] *Linear congruential generator*, Wikipedia (http://en.wikipedia.org/wiki/Linear_congruential_generator), [23 settembre 2012].

[11] P. L'Ecuyer, *Handbook on Simulation*, capitolo 4, Jerry Banks Books, Wiley, 1998.

[12] S. K. Park, K. W. Miller, *Random Numbers Generators: good ones are hard to find*, Computing Practices, Ottobre 1988, Volume 31, Numero 10, E. H. Sibley Panel Editor.

[13] C. Moler, *Numerical Computing with MATLAB*, capitolo 9, SIAM, Philadelphia, 2008.

[14] *Hardware random number generator*, Wikipedia (http://en.wikipedia.org/wiki/Hardware_random_number_generator), [23 settembre 2012].

[15] *Statistics/Numerical Methods/Random Number Generation*, Wikibooks (http://en.wikibooks.org/wiki/Statistics/Numerical_Methods/Random_Number_Generation), [23 settembre 2012].

[16] D. E. Knuth, *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*, Addison-Wesley, Reading, Massachusetts.