Appunti del Corso di Metodi Numerici della Fisica Teorica

Parte I - Introduzione ai Metodi Monte-Carlo

Massimo D'Elia - a.a 2016/2017

NOTA: questi appunti sono per ora da considerarsi in forma preliminare ed incompleti, coprendo solo una parte limitata degli argomenti trattati a lezione. Nelle parti coperte sarà possibile trovare ulteriori approfondimenti non trattati a lezione.

Contents

1 Generalità sui metodi Monte-Carlo

I metodi Monte Carlo si applicano ad una grande quantità di problematiche fisiche. Nella maggioranza dei casi un metodo Monte Carlo si applica ad un problema in cui si è arrivati a calcolare una quantità del tipo

$$\langle F \rangle = \int Dq\ p(q) F(q)\ , \tag{1}$$

dove $q \equiv \{q^1, q^2, \ldots, q^k\}$ è un insieme di variabili stocastiche distribuite secondo la distribuzione di probabilità $Dq\ p(q) \equiv dq^1\ \ldots dq^k\ p(q^1, \ldots q^k)$ e si vuole calcolare il valor medio di una funzione $F(q)$ delle variabili stocastiche. L'approccio Monte Carlo al problema è il seguente:

- Con un opportuno algoritmo si produce un campione di N copie delle variabili stocastiche, $q_1\ \ldots q_N$ ($q_i \equiv \{q_i^1\ \ldots q_i^k\}$), estratte secondo la data distribuzione di probabilità $Dq\ p(q)$;
- su questo campione misuriamo $f_1 = F(q_1);\ \ldots f_N = F(q_N)$ e calcoliamo la media campionaria

$$\bar{f} = \frac{1}{N} \sum_{i=1}^{N} f_i\ ; \tag{2}$$

- le variabili f_i, funzioni di variabili stocastiche, sono esse stesse variabili stocastiche, e così lo è anche la media campionaria $\bar{f}$. Di quest'ultima conosciamo la distribuzione di probabilità nel limite di grandi N, che ci è data dal *Teorema del Limite Centrale*: indipendentemente dalla distribuzione di probabilità da cui siamo partiti (purché questa abbia la varianza ben definita, il che per esempio non è vero per la distribuzione Lorentziana) e assumendo che le copie del campione siano stocasticamente indipendenti l'una dall'altra, la variabile $\bar{f}$ nel limite di grande N è distribuita in modo normale

$$P(\bar{f}) = \frac{1}{\sqrt{2\pi\tilde{\sigma}^2}} e^{-\frac{\bar{f}-\langle F \rangle}{2\tilde{\sigma}^2}} \tag{3}$$

  dove $\tilde{\sigma}^2 = \sigma^2/N$ e $\sigma^2 \equiv \langle F^2 \rangle - \langle F \rangle^2$ è la varianza dell'osservabile originaria. Riguardo alla classe di distribuzioni per cui vale il teorema, questa esclude ovviamente distribuzioni che abbiano la media o la varianza non definite (per esempio divergenti) [1];

- la media campionaria $\bar{f}$ ha quindi una probabilità di circa il 68% di essere nell'intervallo $\langle F \rangle\ \pm \delta$ con $\delta = \sqrt{\tilde{\sigma}^2}$, cioè entro una deviazione standard dal valor medio $\langle F \rangle$. Possiamo quindi affermare, viceversa, che misurando la media campionaria otteniamo una stima del valor medio $\langle F \rangle = \bar{f} \pm \delta$, con errore che decresce come $1/\sqrt{N}$;
- per determinare l'errore standard δ bisognerebbe conoscere la varianza $\sigma^2 \equiv \langle F^2 \rangle - \langle F \rangle^2$. Una stima di σ^2 potrebbe essere ottenuta calcolando

$$\frac{1}{N} \sum_{i=1}^{N} (f_i - \langle F \rangle)^2 \tag{4}$$

[1] Se si considera lo spazio di tutte le possibili distribuzioni di probabilità di una singola variabile stocastica, è possibile dimostrare che, definendo l'operazione che assegna una nuova variabile stocastica come media aritmetica di N estrazioni indipendenti di quella originale, si definisce un'applicazione dallo spazio delle distribuzioni in se stesso che, nel limite in cui N tende ad infinito, può convergere verso varie possibili distribuzioni che sono lasciate invariate (punti fissi), a parte una trasformazione dei parametri, da questa trasformazione. La distribuzione gaussiana è un punto fisso a cui convergono tutte le distribuzioni a varianza finita. La distribuzione Lorentziana, cioè

$$p(x)dx = \frac{a}{\pi} \frac{dx}{(x - x_0)^2 + a^2}$$

è invece un punto fisso per quelle a varianza divergente.

che coincide con σ^2 nel limite $N \to \infty$, sempre per il teorema del limite centrale. Se invece di $\langle F \rangle$ usiamo $\bar{f}$ nella (4), che è la sola cosa che possiamo fare in pratica, otteniamo una sottostima di σ^2, in quanto i dati del campione fluttuano meno intorno alla media dei dati stessi che intorno alla media vera (c'è un grado di libertà in meno), per cui la stima va corretta moltiplicando per il fattore correttivo $N/(N-1)$. Infine otteniamo

$$\delta = \sqrt{\frac{\sum_{i=1}^{N}(f_i - \bar{f})^2}{N(N-1)}} = \sqrt{\frac{\overline{f^2} - \bar{f}^2}{N-1}} \ . \tag{5}$$

ESERCIZIO: Verificare che la stima della varianza effettuata su un campione finito di dati va corretta per il fattore $N/(N-1)$.

Consideriamo la varianza calcolata sul campione

$$\overline{f^2} - \bar{f}^{\,2} = \frac{1}{N}\sum_{i=1}^{N}(f_i - \bar{f})^2 = \frac{1}{N}\sum_{i=1}^{N}\left(f_i - \left(\frac{1}{N}\sum_{j=1}^{N} f_j\right)\right)^2$$

Vogliamo vedere quale sia il valore medio atteso per tale quantità al variare del campione considerato. Supponiamo di prendere un insieme di campioni indipendenti, tutti della stessa grandezza N, ed indichiamo con $\langle\cdot\rangle_c$ la media sull'insieme dei campioni. Tale media non è altro che la media sulla distribuzione di probabilità del processo stocastico che genera il campione stesso: assumendo che tale processo consista in N estrazioni indipendenti, tale distribuzione si può scrivere

$$\prod_{j=1}^{N} Dq_j \ p(q_j) \tag{6}$$

cioè come il prodotto di N distribuzioni identiche, per cui risulta semplice calcolare le medie su tale distribuzione, e quindi anche la varianza della media campionaria, a partire dai valori medi sulla singola distribuzione, che continueremo ad indicare con il solito simbolo $\langle\cdot\rangle$. In particolare:

$$\langle\frac{1}{N}\sum_{i=1}^{N}(f_i - \bar{f})^2\rangle_c = \frac{1}{N}\left[\sum_{i=1}^{N}\langle f_i^2\rangle_c + \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{N}\langle f_j f_k\rangle_c - \frac{2}{N}\sum_{i=1}^{N}\sum_{j=1}^{N}\langle f_i f_j\rangle_c\right]$$

La media sull'insieme dei campioni di una funzione del singolo elemento del campione coincide con la media sulla distribuzione, quindi ad esempio $\langle f_i^2\rangle_c = \langle F^2\rangle$. Invece la media di funzioni che coinvolgono più elementi dello stesso campione dipende in generale dalle correlazioni esistenti fra tali elementi, che però in questo caso sono nulle. Infatti, se i dati all'interno di uno stesso campione sono fra loro stocasticamente indipendenti, come nel nostro caso, abbiamo

$$\langle f_i f_j\rangle_c = (1-\delta_{ij})\langle f_i\rangle_c\langle f_j\rangle + \delta_{ij}\langle f_i^2\rangle = (1-\delta_{ij})\langle F\rangle^2 + \delta_{ij}\langle F^2\rangle$$

Applicando questo risultato all'espressione precedente otteniamo

$$\overline{f^2} - \bar{f}^{\,2} = \frac{1}{N}\left[N\langle F^2\rangle - \frac{N^2-N}{N}\langle F\rangle^2 - \langle F^2\rangle\right] = \frac{N-1}{N}(\langle F^2\rangle - \langle F\rangle^2) = \frac{N-1}{N}\sigma^2$$

ESEMPIO: Il primo metodo Monte-Carlo

I metodi Monte Carlo hanno iniziato ad essere usati nella metà del secolo scorso, precisamente in occasione del progetto Manhattan a Los Alamos: in quel caso il processo fisico che si voleva studiare era quello della diffusione di neutroni attraverso la materia. In realtà, mentre l'utilizzo pratico è stato chiaramente legato alla nascita di primi calcolatori elettronici (FERMIAC, dedicated analog computer, 1947), la nascita dell'idea di metodo Monte Carlo nasce molto prima, nel 1777, quando Georges-Louis Leclerc, Conte di Buffon, propose un metodo per determinare il valore di π gettando a caso degli aghi su un parquet [1], che ora andremo ad analizzare.

Supponiamo di avere degli aghi di lunghezza l che gettiamo a caso su un parquet diviso in fasce di larghezza a, con $a > l$. Qual è la probabilità che un ago una volta caduto intersechi la linea di separazione fra due fasce contigue? Le variabili stocastiche che caratterizzano *come* è caduto l'ago posso essere considerate, per quanto concerne il nostro problema, la distanza b del centro dell'ago dalla più vicina linea di separazione fra due fasce (b è distribuita uniformemente nell'intervallo $[0, a/2]$) e l'angolo ϕ formato dall'ago con la perpendicolare alla direzione delle fasce ($-\pi/2 \leq \phi \leq \pi/2$ con distribuzione uniforme). Le due variabili stocastiche b e ϕ hanno quindi una densità di probabilità uniforme $p(b, \phi) = 2/(\pi a)$. La probabilità P che l'ago intersechi la linea di separazione è data dal valor medio su questa distribuzione della funzione $\chi(b, \phi)$ che vale 1 se c'è intersezione e 0 altrimenti. È facile rendersi conto che avere intersezione equivale a $b < l/2 \cos\phi$, per cui

$$\begin{aligned} P = \langle\chi\rangle &= \int_0^{a/2} db \int_{-\pi/2}^{\pi/2} d\phi p(b,\phi)\chi(b,\phi) = \frac{2}{\pi a}\int_{-\pi/2}^{\pi/2} d\phi \int_0^{l/2\cos\phi} db = \\ &= \frac{2}{\pi a}\int_{-\pi/2}^{\pi/2} d\phi \frac{l}{2}\cos\phi = \frac{2l}{\pi a}\,. \end{aligned} \tag{7}$$

Dunque se effettuiamo N lanci di ago e contiamo il numero di intersezioni n, avremo

$$\lim_{N\to\infty}\frac{n}{N} = P = \frac{2l}{\pi a} \tag{8}$$

e potremo usare il nostro *esperimento* per dare una stima del valore di π.

Il metodo proposto da Buffon è un tipico esempio di Monte Carlo, anche se fatto *a mano*, e ci da l'occasione per iniziare a discutere dell'efficienza dei metodi M.C. Supponiamo per semplicità che sia $l = a$ e di voler determinare π con un errore sulla seconda cifra decimale, cioè ad esempio $\pi = 3.14 \pm 0.01$. Quante volte dobbiamo lanciare il nostro ago? Sappiamo che l'errore dato dal M.C. per la stima di P è pari a $\delta P = \sqrt{\sigma^2/N}$ con $\sigma^2 = \langle\chi^2\rangle - \langle\chi\rangle^2$. Essendo χ una funzione che vale 0 o 1, abbiamo che $\chi^2 = \chi$, e quindi $\langle\chi^2\rangle = \langle\chi\rangle = P$. Abbiamo quindi $\sigma^2 = P(1-P) = \frac{2}{\pi}(1-\frac{2}{\pi}) \simeq 0.23$. Essendo $P = 2/\pi$, abbiamo, dalle leggi di propagazione degli errori, che $\delta P/P = \delta\pi/\pi$, e quindi $\delta\pi = \frac{\pi}{P}\sqrt{\sigma^2/N}$. Imponendo $\delta\pi = 0.01$ si ottiene $N \sim 5.6 \cdot 10^4$. Lo sforzo necessario a lanciare 50-mila volte un ago sul pavimento sembra sproporzionato rispetto al modesto risultato raggiunto: esistono sicuramente metodi molto più efficienti per determinare π. Resta quindi il problema di capire quando un metodo Monte-Carlo sia veramente utile dal punto di vista pratico.

2 Metodi Monte-Carlo e metodi di integrazione diretta

Supponiamo di dover calcolare numericamente il valore medio di una funzione reale di una variabile reale $f(x)$ su un intervallo $[a, b]$

$$\langle f\rangle \equiv \frac{1}{|b-a|}\int_a^b f(x)dx\,. \tag{9}$$

il problema è equivalente al calcolo del valor medio di f per una distribuzione uniforme della variabile stocastica x in $[a, b]$. Se procediamo con un metodo Monte-Carlo possiamo estrarre N numeri x_i ($i = 1, \ldots N$) distribuiti a caso in $[a, b]$, valutare la media campionaria

$$\bar{f} = \frac{1}{N}\sum_i f(x_i)$$

che ci darà una stima di $\langle f\rangle$ con errore $\propto 1/\sqrt{N}$.

D'altra parte possiamo usare anche un metodo di integrazione diretta, cioè dividere $[a, b]$ in N intervalli di uguale ampiezza $\delta = |b-a|/N$ e valutare l'integrale in forma approssimata su ognuno degli intervalli, con metodi più o meno sofisticati. Quello più semplice (metodo dei rettangoli) è

$$\langle f\rangle \sim \frac{1}{|b-a|}\sum_{j=0}^{N-1}\delta f(a+\delta j)$$

si verifica facilmente che su ogni intervallo δ si commette un errore di ordine δ^2 (si considera la funzione costante sull'intervallo, come prima correzione si dovrebbe considerare il suo sviluppo di Taylor troncato al prim'ordine) per cui l'errore totale è al più di ordine $N\delta^2 \propto 1/N$.

Quindi valutare la funzione f su di un insieme ordinato di punti porta ad errore che decresce più rapidamente al crescere dello sforzo numerico effettuato (che è proporzionale ad N) che non nel caso del metodo Monte-Carlo. Inoltre per l'integrazione diretta possiamo usare metodi più sofisticati, ad esempio usando il metodo dei trapezi l'errore sul singolo intervallo diventa dell'ordine di δ^3 e quindi l'errore decresce come $1/N^2$. Il metodo Monte-Carlo non è quindi il più efficiente.

Le cose cambiano se aumentiamo il numero di variabili di integrazione. Supponiamo di voler calcolare il valor medio su di un certo dominio di una funzione di D variabili reali. Il metodo Monte-Carlo procede nello stesso modo: vengono estratte N D-uple di variabili stocastiche uniformemente a caso sul dominio e la funzione viene valutata su queste D-uple; l'errore sarà comunque sempre proporzionale a $1/\sqrt{(N)}$. Supponiamo invece di adottare un metodo a griglia: dividiamo il dominio D-dimensionale in N ipercubetti di lato $\delta \propto 1/N^{1/D}$. Usando il metodo più semplice in cui si approssima la funzione con una costante su ogni ipercubetto, l'errore elementare compiuto è in questo caso di ordine $\delta^D\delta$, cioè il volume dell'ipercubetto per il primo termine trascurato nello sviluppo di Taylor; usando un metodo più sofisticato si può arrivare ad un errore per ipercubetto dell'ordine di δ^{D+k} dove k dipende solo dal metodo di integrazione e non dalla dimensionalità. Se la funzione è abbastanza regolare in generale gli errori si sommano in modo coerente e quindi l'errore totale è dell'ordine di $N\delta^{D+k} \propto N^{-k/D}$.

Il risultato è che per ogni fissato metodo di integrazione (fissato k) esisterà sempre una dimensionalità D del sistema abbastanza grande per cui $k/D < 1/2$ e quindi l'errore del metodo di integrazione diretta decresce più lentamente con N che quello di un metodo Monte-Carlo.

Di fatto non è facile realizzare metodi di integrazione diretta che raggiungano grandi valori di k senza che questo comporti un considerevole sforzo numerico aggiuntivo. I metodi Monte-Carlo iniziano ad essere competitivi già quando D è $O(10)$ e sono di gran lunga preferibili quando D aumenta di vari ordini di grandezza, come nei problemi di meccanica statistica e teoria di campo che andremo a considerare.

3 Metodi per campionare una data distribuzione di probabilità

Illustreremo ora una serie di metodi ed algoritmi per estrarre campioni di variabili stocastiche distribuite secondo un'assegnata distribuzione di probabilità, aventi la caratteristica che ogni singola estrazione all'interno di un dato campione è indipendente dalle altre. La classe di tali algoritmi è in genere applicabile solo a sistemi con numero di gradi di libertà limitato e si contrappone ai metodi basati su processi che esplorano dinamicamente lo spazio delle variabili stocastiche, come i processi (catene) di Markov che studieremo in seguito.

La base di partenza di un qualsiasi algoritmo numerico che voglia generare una qualche distribuzione di probabilità, è di essere già in grado di estrarre dei numeri a caso (numeri random) con qualche distribuzione di partenza; la più semplice distribuzione è quella uniforme in un dato intervallo. Partiremo quindi preliminarmente da una discussione sui generatori di numeri (pseudo)-random con distribuzione uniforme nell'intervallo $[0, 1]$, che sono un ingrediente essenziale di qualsiasi altro algoritmo e sulla cui correttezza (nel senso che andremo a discutere) si basa la correttezza dell'intera simulazione.

3.1 Generatori di numeri (pseudo)-casuali

Abbiamo visto che in un metodo M.C. è essenziale estrarre a caso delle variabili stocastiche reali distribuite in un certo modo assegnato, ad esempio in modo uniforme. Nell'esempio proposto da Buffon tali variabili vengono generate dal lancio casuale dell'ago sul pavimento. In generale la

generazione di numeri random può essere associata solo a fenomeni fisici di tipo stocastico, come ad esempio il decadimento radioattivo, il rumore termico nei circuiti elettronici o il tempo di arrivo dei raggi cosmici, tuttavia non in modo efficiente per le richieste tipiche dei metodi M.C.

Un esempio di sequenza di numeri random prodotta a partire da un fenomeno fisico stocastico è quella generata da Frigerio e Clark (1975) usando una sorgente α ed un rivelatore acceso per intervalli di 20 ms, in cui la media di decadimenti attesi era 24.315. Per ogni intervallo veniva generato un bit 1 per un numero pari di decadimenti ed uno 0 per un numero dispari[2] ed in tal modo venivano generati dei numeri casuali di 31-bit. La sequenza così prodotta consisteva di $2.5 \cdot 10^6$ numeri random. Sebbene sia in teoria possibile registrare tali sequenze per poi leggerle ed usarle durante una simulazione numerica M.C., ciò non sarebbe molto pratico anche perché la quantità di numeri random necessari in una simulazione M.C. può essere facilmente più grande di vari ordini di grandezza. Per questo motivo risulta necessario trovare un modo di produrre efficientemente una sequenza di numeri random attraverso una parte stessa del codice con cui si effettua la simulazione M.C.

L'idea di produrre dei numeri random attraverso un algoritmo può sembrare insensata, ed in effetti lo è: l'esecuzione di un algoritmo è un processo assolutamente deterministico. Quelli che descriveremo saranno in realtà dei generatori di numeri pseudo-random, che producono cioè sequenze di numeri che, pur essendo deterministiche, possono essere scambiate a tutti gli effetti per sequenze di numeri casuali. Come vedremo la frase "*a tutti gli effetti*" dipende dal problema considerato, e generatori di numeri pseudo-random che vanno bene per la simulazione di un dato sistema fisico possono rivelarsi inadeguati per altri problemi. Vi sono alcune richieste basilari: ad esempio nel caso di un generatore di numeri x_i, distribuiti uniformemente fra 0 ed 1, le medie calcolate sulla sequenza $\frac{1}{N}\sum_i x_i^p$ devono coincidere per grandi N con i momenti della distribuzione uniforme $\int_0^1 dx\ x^p = \frac{1}{p+1}$.

L'idea alla base dei primi generatori fu quella di generare una successione di numeri interi $X_0,\ X_1,\ \ldots X_k,\ \ldots$ compresi fra 0 ed un numero m attraverso una funzione

$$X_{k+1} = f(X_k)\ ; \tag{10}$$

da questa successione si può poi estrarre un sequenza di numeri distribuiti fra 0 ed 1 definendo $x_k = X_k/m$. Un tale generatore può essere molto efficiente e rapido, ma ha chiaramente alcuni possibili punti deboli a cui bisogna stare attenti (e che in realtà possono riguardare qualsiasi tipo di generatore):

- **periodo finito:** se ad un certo punto si passa attraverso un numero già estratto in precedenza, verrà ripetuta esattamente la stessa sequenza, essendo X_{k+1} funzione solo di X_k, per cui il generatore avrà un certo periodo finito. Avere la stessa sequenza di numeri random ripetuta più di una volta all'interno della stessa simulazione chiaramente può falsare i risultati di quest'ultima, per cui il periodo deve essere il più grande possibile;

- **correlazioni:** chiaramente un generatore del tipo (10) può dare luogo a correlazioni più o meno nascoste fra numeri successivi della sequenza che possono falsare i risultati della simulazione.

Una caratteristica importante di un generatore random è la rapidità con cui il numero viene prodotto: questo problema è attuale oggi tanto quanto lo era ai tempi dei primi calcolatori. È per questo che, dopo alcune altre proposte precedenti, ebbe grande successo la semplice proposta di Lehmer [3, 2]. L'idea di base è che se prendo un numero intero m (chiamato modulo), un numero $n < m$, e moltiplico n per un terzo numero molto grande a, a meno di casi particolari il resto

[2] In realtà, essendo la probabilità di avere un numero pari leggermente superiore a quella di avere un numero dispari, la procedura era leggermente diversa per correggere il bias

della divisione am/n sarà distribuito fra 0 ed $m-1$ in modo abbastanza casuale. I cosiddetti *generatori congruenti lineari* rappresentano una leggera variazione di tale idea, in cui prima di fare la divisione per m viene sommato un terzo numero, cioè la funzione in Eq. (10) è

$$X_{k+1} = mod(aX_k + c, m) \tag{11}$$

dove $mod(n_1, n_2)$ indica l'operazione n_1 modulo n_2, cioè appunto il resto della divisione di n_1 per n_2. Come già detto, m è detto *modulo*, mentre a e c sono detti rispettivamente *moltiplicatore* ed *incremento*. E' chiaro che il periodo di tale generatore non può essere maggiore di m, ed in particolare è pari ad m solo se vengono verificate contemporaneamente le seguenti condizioni (teorema di Hull-Dubell [2]): *i)* c è primo rispetto ad m; *ii)* ogni divisore primo di m lo è anche di $(a-1)$, questo inoltre deve essere vero anche per la divisione per 4.

La proposta di Lehemer venne implementata per la prima volta con $a = 23$ e $m = 10^8 + 1$ sul calcolatore ENIAC (Electronic Numerical Integrator and Computer) alla fine degli anni '40 dello scorso secolo. Questo era il primo calcolatore programmabile per scopi generali (general purpose), ed aveva la peculiarità di essere basato sul sistema di numerazione decimale ad 8 cifre significative, da cui si intuisce la scelta del modulo m. Su questo calcolatore, che assorbiva una quantità di energia comparabile al consumo della città di Filadefia (vicino alla quale era posto), il generatore produsse qualche milione di estrazioni di numeri random.

Come già detto il periodo non è l'unico l'unico parametro di qualità di un generatore: ci possono essere correlazioni più o meno nascoste. Queste possono essere evidenziate ad esempio disegnando n-uple di numeri estratti in successione nello spazio n-dimensionale: per numeri casuali genuini queste dovrebbero disporsi a caso, mentre per un generatore congruente si dispongono su di un insieme finito di iperpiani paralleli il cui numero dipende dalla bontà del generatore. Si pensi al generatore in Eq. (11): è facile convincersi che le coppie x_k, x_{k+1} si dispongono tutte su $a+1$ rette parallele nella porzione di piano $[0,1] \times [0,1]$: per a molto grande l'effetto non dovrebbe essere visibile se non dopo l'estrazione di un numero enorme di numeri random, ma può capitare che per valori particolari di a il numero di rette parallele si riduca ulteriormente. Uno dei primi generatori congruenti distribuiti dall'IBM, il generatore RANDU, si rivelò essere affetto da questo problema. Un generatore congruente di buona qualità si ottiene per esempio scegliendo $a = 16807$, $c = 0$ ed $m = 2^{31} - 1$ (che è un numero primo di Mersenne).

Esistono molte varianti ai generatori congruenti, ad esempio i generatori ritardati alla Fibonacci (Lagged Fibonacci Generators, LFG), in cui

$$X_k = f(X_{k-q}, X_{k-p})$$

con q e p costanti intere positive: il numero estratto al passo k dipende, secondo una certa regola da specificare, da due numeri precedenti della successione. I generatori LFG possono raggiungere periodi molto grandi ma possiedono una dinamica molto meno prevedibile che non per i semplici generatori congruenti: la loro bontà viene di solito verificata "sul campo". Un esempio di tale generatore è quello proposto da Kirkpatrick e Stoll nel 1981 e noto come R250: $q = 250$, $p = 103$ e la regola di Eq. (3.1) è uno XOR (OR esclusivo) fra le cifre binarie dei due interi; tale generatore ha un periodo pari a $2^{250} - 1$ ed è ritenuto un generatore di alta qualità. Un articolo di qualche anno fa (Alan M. Ferrenberg, D. P. Landau e Y. Joanna Wong, Physical Review Letters 69, 3382 - 3384 (1992)) ha tuttavia rivelato con grande sorpresa che simulazioni con grande statistica del modello di Ising in due dimensioni su un reticolo 16×16 ed usando l'algoritmo a cluster di Wolff e il generatore R250 per la produzione di numeri casuali, producono risultati sbagliati (ad esempio per l'energia o il calore specifico).

Terminiamo qui la discussione sui generatori di numeri casuali anche se la letteratura sull'argomento è molto vasta. Una discussione tecnica più approfondita e con esempi può essere trovata per esempio in uno dei capitoli del Numerical Recipes. La considerazione finale è che ogni generatore di numeri pseudo-casuali è in realtà un impostore, necessario per la vostra simulazione e più o meno abile nel non farsi scoprire: per ogni dato generatore esisterà un'applicazione che rivelerà

la truffa (come il modello ISING per il generatore R250), ma l'importante è che non si tratti del problema che state studiando voi.

3.2 Generatori di distribuzioni non uniformi mediante cambio di variabili

Supponiamo di avere a disposizione un buon generatore di numeri random uniformi fra 0 e 1 ed occupiamoci della generazione di numeri reali avente una distribuzione non uniforme assegnata.

Se x è distribuita come $p(x)dx$ e $y = y(x)$ è una sua funzione, che supponiamo essere invertibile, allora la distribuzione $\tilde{p}$ della variabile y si ottiene facilmente notando che le estrazioni della prima variabile fra x e $x + dx$ coincidono con quelle della seconda fra $y(x)$ e $y(x + dx)$ (per affermare questo è necessario che la funzione sia iniettiva), cioè

$$\tilde{p}(y)dy = p(x)dx \tag{12}$$

da cui $\tilde{p}(y) = p(x)/|dy/dx|$. In linea di principio data una distribuzione possiamo passare ad un'altra distribuzione mediante un cambio di variabili; il nostro problema è trovare il cambio di variabili opportuno. Supponiamo che x sia distribuita a caso fra 0 ed 1 ed integriamo entrambi i membri di Eq. (12) come segue:

$$\int_{y_0}^{y} \tilde{p}(y')dy' = \int_0^x dx' = x \tag{13}$$

se conosciamo la primitiva della funzione $\tilde{p}$ possiamo scrivere in forma esplicita anche il membro di sinistra e quindi ottenere una relazione fra x ed y da invertire.

Supponiamo di volere campionare un variabile y nel dominio $[0, \infty)$ secondo la distribuzione $\mu e^{-\mu y}$. In tal caso, ponendo $y_0 = 0$, l'Eq, (13) diventa

$$\int_0^y \mu e^{-\mu y'} dy' = 1 - e^{-\mu y} = x$$

da cui invertendo otteniamo $y = -\ln(1 - x)/\mu$: se scegliamo x a caso fra 0 ed 1, la variabile derivata y sarà distribuita come richiesto.

Un esempio notevole è fornito dalla distribuzione gaussiana. Supponiamo $\tilde{p}(y)dy = e^{-y^2}dy/\sqrt{\pi}$. Non conosciamo la primitiva di tale funzione, tuttavia consideriamo due variabili y_1 ed y_2 stocasticamente indipendenti fra loro (cioè la probabilità congiunta è il prodotto delle singole probabilità) distribuite come sopra

$$p(y_1, y_2)dy_1 dy_2 = \frac{1}{\pi} e^{-y_1^2} e^{-y_2^2} dy_1 dy_2$$

se passiamo in coordinate polari, $y_1 = r\cos(\phi)$, $y_2 = r\sin(\phi)$ la distribuzione diventa

$$p(r, \phi)drd\phi = \frac{1}{\pi} d\phi dr\, re^{-r^2} = \left(\frac{1}{2\pi} d\phi\right)\left(2re^{-r^2} dr\right)$$

cioè anche le variabili r e ϕ sono stocasticamente indipendenti fra loro con ϕ uniforme in $[0, 2\pi)$ ed r distribuito secondo $2re^{-r^2}dr$. Di quest'ultima funzione conosciamo la primitiva e possiamo quindi riscrivere l'Eq. (13) come

$$x = \int_0^r 2r' e^{-r'^2} dr' = 1 - e^{-r^2}$$

da cui $r = \sqrt{-\ln(1 - x)}$. Possiamo costruire quindi un algoritmo che fornisce una coppia di variabili gaussiane con media 0 e varianza $1/\sqrt{2}$.

- estraggo x_1 ed x_2 a caso in $[0, 1)$
- assegno $r = \sqrt{-\ln(1 - x_1)}$, $\phi = 2\pi x_2$;

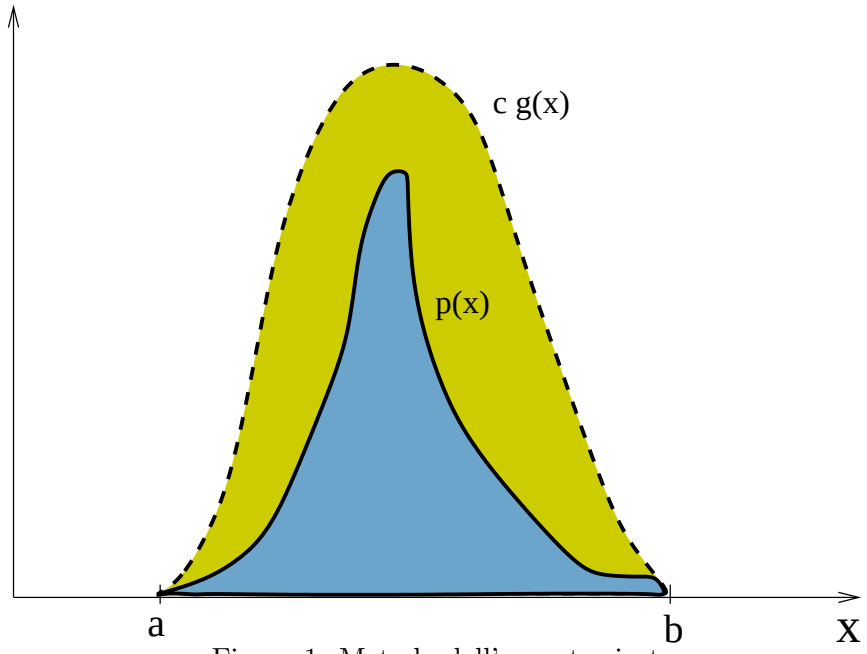

Figure 1: Metodo dell'accept reject.

- infine ricostruisco $y_1 = r\cos(\phi)$, $y_2 = r\sin(\phi)$.

E' chiaro che se voglio generare y distribuito in modo gaussiano con media y_0 e varianza σ^2, basterà predere le variabili generate con l'algoritmo precedente, moltiplicarle per $\sigma\sqrt{2}$ e quindi traslarle di y_0. Quello appena descritto è noto come algoritmo di Box-Müller.

3.3 Il metodo "accept-reject" di von Neumann

Solo in pochi casi è possibile trovare il cambio di variabili opportuno per passare dalla distribuzione uniforme a quella voluta. Esistono però algoritmi che possono risolvere il problema in casi più generali, come quello formalizzato da John von Neumann e noto come "accept-reject".

Supponiamo di saper campionare una distribuzione $g(x)$ che abbia la seguente relazione con la nostra distribuzione $p(x)$: è definita sullo stesso dominio ed esiste una costante c tale che $cg(x) \geq p(x)$ per ogni x, come mostrato in Figura 1 (è ovvio che poiché sia g che p sono distribuzioni normalizzate, dobbiamo avere $c \geq 1$ e il segno uguale solo se $g(x) = p(x)$). Si noti che, se la $p(x)$ è limitata superiormente, è sempre possibile trovare la $g(x)$, ad esempio possiamo considerare la distribuzione $1/|b-a|$ costante sul dominio e porre $c = |b-a|max(p)$.

L'algoritmo per estrarre la variabile x secondo la data distribuzione è il seguente:

1. estraggo un numero x_p secondo la distribuzione $g(x)dx$ e un numero y scelto uniformemente a caso fra 0 e $cg(x)$.

2. se $y < p(x_p)$ allora considero $x = x_p$ come appartenente al mio campione (accept), altrimenti no (reject).

E' facile convincersi che il punto (x_p, y) scelto al passo 1) è distribuito a caso nella regione di piano con ordinate positive posta sotto la curva $cg(x)$, cioè $y > 0; y < cg(x)$, con x nel dominio della distribuzione. Al punto 2) invece seleziono solo i punti (x_p, y) distribuiti uniformemente sotto la curva $p(x)$: è altrettanto facile convincersi che l'ascissa di tali punti sarà ora distribuita proporzionalmente a $p(x)$.

Una proprietà importante di questo algoritmo è il numero medio di ripetizioni dell'algoritmo (cioè lo sforzo numerico) che dobbiamo compiere per effettuare una singola estrazione. Se A_{cg} è l'area compresa sotto la curva $cg(x)$ e A_p quella compresa sotto la curva $p(x)$, allora la probabilità

di accettare l'estrazione in una singola ripetizione dell'algoritmo (accettanza) è pari ad $\alpha = A_p/A_{cg}$. E' facile verificare che il numero medio di ripetizioni è uguale $1/\alpha$.

Questo mostra che, per quanto sia possibile scegliere $g(x)$ in modo abbastanza arbitrario (anche la distribuzione uniforme va bene se $p(x)$ è limitata), l'efficienza numerica dell'algoritmo sarà tanto migliore quanto più simili saranno le distribuzioni $f(x)$ e $g(x)$, cioè quanto più α è vicino ad uno.

Esempio: algoritmo per campionare $x \in [-1, 1]$ secondo la distribuzione $p(x) \propto \sqrt{1-x^2}\exp(\gamma\mu)$.

Scegliamo prima la variabile x distribuita secondo la distribuzione

$$g(x)dx = \frac{\gamma}{e^{\gamma} - e^{-\gamma}} e^{\gamma x} dx \,,$$

questo può essere fatto usando il metodo del cambio di variabili descritto nella Sezione 3.2, in particolare si verifica facilmente che se y è distribuito uniformemente in $[0, 1]$ allora

$$x = \frac{1}{\gamma} \ln\left(e^{-\gamma} + y(e^{\gamma} - e^{-\gamma})\right)$$

è distribuito come richiesto.

A questo punto estraiamo un secondo numero r a caso in $[0, 1]$ ed accettiamo x se $r < \sqrt{1-x^2}$ (si verifica facilmente che questo coincide con quanto descritto prima per un'opportuna scelta della costante c.

L'algoritmo descritto costituirà una parte di algoritmi più complessi per la simulazione numerica delle teorie di gauge su reticolo, ed è noto come algoritmo di Creutz. Notare che quando γ diventa molto grande, la variabile x viene inizialmente scelta molto vicina ad uno, ma viene quindi quasi sempre scartata perché $\sqrt{1-x^2}$ risulta molto piccolo, in particolare effettuando un calcolo approssimato del rapporto fra le aree nel caso in cui $\gamma \gg 1$ ci si rende conto che l'accettanza decresce come $1/\sqrt{\gamma}$. Quindi l'algortimo diventa molto inefficiente quando $\gamma \gg 1$. In tal caso è necessario ricorrere ad algoritmi alternativi.
ESERCIZIO: Verificare che l'accettanza dell'algoritmo di Creutz appena illustrato tende a zero come $1/\sqrt{\gamma}$.

3.4 Ripesamento (reweighting)

Un altro metodo molto generale che possiamo applicare nel caso in cui conosciamo una distribuzione $g(x)$ che assomiglia molto alla distribuzione $p(x)$ è quello del ripesamento, di cui di fatto abbiamo già discusso prima. Nell'espressione

$$\langle f \rangle_p = \int f(x)p(x)dx$$

non è necessariamente specificato chi sia la distribuzione e chi la funzione (osservabile) di cui effettuare la media, quindi possiamo riscrivere

$$\langle f \rangle_p = \int (f(x)p(x)/g(x))g(x)dx = \langle fp/g \rangle_g$$

cioè possiamo effettuare il campionamento con una distribuzione "sbagliata" e quindi porre riparo ridefinendo l'osservabile da mediare.

Come discusso in precedenza la scelta della distribuzione con cui viene effettivamente fatto il campionamento influenza la varianza dell'osservabile misurata, quindi una scelta sbagliata di $g(x)$ può rendere il Monte-Carlo molto inefficiente, ma una scelta oculata può addirittura migliorare l'efficienza rispetto al campionamento diretto con $p(x)$ (importance sampling).

4 Markov chain Monte-Carlo

In a few words, a *Markov process* (named after the Russian mathematician Andrej Markov) is a stochastic evolution process, defined within a given set of stochastic variables (which we will call *the system* in the following) and in terms of a (usually fictitious) time variable, in which it is only the present status of the system which determines its possible future evolution. It is common to consider a discrete time variable, so that the evolution proceeds step by step: in this case one speaks of a *Markov chain*.

Markov processes resemble closely the evolution determined by a set of first order differential equations, like Hamilton equations in phase space, or, in the case of Markov chains, by a set of first order recurrence relations[3]. In all these cases, the common idea is: tell me where you are now, and I will tell you where you are going in the future, no matter what you did in the past. The only, obvious difference is that the evolution rule is not deterministic, i.e. we are dealing with a stochastic process. Moreover we assume that the evolution rule keeps unchanged during the stochastic process, i.e. it is *time independent*. Actually, the simplest example of Markov process is the random walk discussed in Chapter 2.

The idea of exploiting Markov chains to sample a given distribution is in some sense dual to the idea at the basis of Statistical Mechanics. In the latter, one tries to rewrite time averages of some complex system at thermal equilibrium in terms of averages taken over a given probability distribution, representing the so-called statistical ensemble. In a Markov chain Monte-Carlo, the aim is instead to devise a stochastic dynamical process such that averages taken over the fictitious Monte-Carlo time coincide with averages over a given probability distribution. There is no compelling reason for adopting a Markov chain to do that, one could use a stochastic process with memory of the past as well, however the theory of Markov chains is much simpler, permits to predict their dynamics with relative ease, and leaves at the same time enough freedom for choosing and optimizing algorithms for a given problem.

In the following we will assume, for simplicity, that the set of all possible states of the system, i.e. of all the possible outcomes of the stochastic variables, which will be called *configuration space* in the following, is discrete, so that a given state can be enumerated by a index a: we will denote such space by Ω. Nothing dramatic happens when this is not true, just a change of some sums into some corresponding integrals. The Markov chain is then defined in term of a probability matrix W, which is usually called the *transition matrix*[4]

Definition
$W_{ab} \equiv P(b \to a)$ gives the probability that the system, starting from state "b", arrives in state "a" after one discrete time step of the stochastic process.

The elements of the transition matrix will be more or less explicitly encoded in the algorithm used to move from one state to the other during the computer implementation of the Markov chin. Those matrix elements have two obvious properties

$$0 \leq W_{ab} \leq 1 \quad \forall \quad a, b \in \Omega \quad ; \qquad \sum_{a \in \Omega} W_{ab} = 1 \quad \forall \quad b \in \Omega \, . \tag{14}$$

The first property simply states that W is a probability matrix, the second states that the stochastic process never stops, i.e. the system will surely reach some new state after each step.

It is quite easy to compute the probability of reaching a from b in k steps of the Markov chain: one has to consider the probability of a generic walk going from b to a, and then sum over all possible intermediate $k-1$ states, i.e.

$$P(b \to a, \ k \text{ steps}) = \sum_{c_1, c_2, \ldots c_{k-1}} W_{a c_{k-1}} \ldots W_{c_2, c_1} W_{c_1, b} = (W^k)_{ab} \, , \tag{15}$$

therefore the k-steps transition matrix is simply given by the k-th power of the original matrix.

Let us now call p_a the target probability distribution that needs to be sampled by our Monte-Carlo algorithm. The idea is to generate, by means of the Markov chain, a stochastic walk $a^{(0)}, a^{(1)}, a^{(2)}, \ldots, a^{(k)}, \ldots$, where $a^{(0)}$ is a given starting state, such that each state a is visited with frequency p_a during the walk; in this way, states visited during the walk can be taken as a representative sample for Monte-Carlo evaluation. That must be true independently of the starting state $a^{(0)}$, at least after waiting an appropriate relaxation time. We can turn this requirement into a simple mathematical statement, we need that

$$\lim_{k \to \infty} (W^k)_{ab} = p_a \quad \forall \ b \tag{16}$$

so that, independently of the starting state b, after many steps we end up in state a with probability p_a. Eq. (16) means that W^k, for large enough k, must tend to a matrix with all columns equal to p_a, so that the stochastic walk leads eventually to a good extraction probability and loses memory of the starting

[3]Those do not include many famous recurrence sequences, like the Fibonacci one.

[4]The literature often reports a definition of the transition matrix which is the transpose of the one adopted here.

position completely; how large must be k to effectively achieve that will a be a figure of merit of the chosen algorithm[5].

Before discussing how to construct an appropriate W which satisfies Eq. (16), let us introduce some additional properties that will be required in the following.

Definition
Ergodicity
A Markov chain is called *ergodic* if, for any couple of states $a, b \in \Omega$, there exists a k such that $(W^k)_{ab} \neq 0$

In other words, a Markov chain is ergodic if, starting from any state, we can end up in any other state of the system if we wait long enough. Tipically, when the chain is non-ergodic the state space Ω gets divided into a set of disconnected regions, meaning that it is not possible to move from one region to the other via the given stochastic process. That is as if the Markov chain could be reduced to a set of simpler Markov chains, each describing the stochastic process within one of the disconnected regions.

Definition
Period of a state
Consider a particular state "a" of the system, and the set of all integer numbers n such that $(W^n)_{aa} \neq 0$. The period of a, d_a, is defined as the greatest common divisor of all n's.
In other words, if we start from state a, future visits of a cannot take place but for integer multiples of d_a (not necessarily for all multiples of d_a).

Theorem 1 *If a Markov chain is ergodic, every state has the same period, $d_a = d \ \forall \ a$, so that we can define the period d of the chain itself.*
proof: *Let us consider a generic couple of states, a and b, and their associated periods, d_a and d_b. Since the chain is ergodic, it must be possible to go from a to b, and then back from b to a, in a finite number of steps. Let us consider one of these stochastic walks and let us call $\bar{k}$ the associated number of steps: by definition, $\bar{k}$ must be divisible by d_a.*

Now, the given loop can be modified as we want in the following way: as we reach b, before moving back to a, we add a new loop of n steps going from b to b: by definition, d_b is the GCD of all the possible values of n. In this way, we have constructed a new loop of $\bar{k} + n$ steps going from a to a: $\bar{k} + n$ must be divisible by d_a, and since also $\bar{k}$ is divisible by d_a, n must be divisible by d_a as well. Therefore we have proved that d_a is a divisor of all possible n such that $(W^n)_{bb} \neq 0$, so that $d_a \leq d_b$ by definition of GCD.

The above reasoning can be repeated, exchanging the role of a and b, to prove that $d_b \leq d_a$, so that finally we obtain $d_a = d_b$ for a generic couple of states.

Definition
Aperiodicity and regularity
An ergodic Markov chain which is aperiodic if $d = 1$. An aperiodic, ergodic Markov chain is also called "regular".

Regular Markov chains are important because some of the theorems we are going to discuss in the following hold only for them. It is easy to make sure that the chain is aperiodic: it suffices to take a small probability to remain in the same state at each step. Ergodicity is instead usually more difficult to prove.

4.1 Ensemble of Markov chains and some theorems on the spectrum of W

In order to investigate the properties of W, it is better to take a statistical mechanics attitude and discuss the behaviour of an ensemble of Markov processes, instead of a single one. To that purpose, we consider a very large (or infinite) set of identical copies of the same system (the ensemble), which are initially distributed over the configuration space Ω according to a starting distribution $\pi_a^{(0)}$, which is the fraction of the total number of copies living in state a. We imagine to start a Markov chain, described by the matrix

[5] A simple, obvious possibility would be to take $W_{ab} = p_a$ right from the beginning, i.e. to take a Markov chain that completely loses memory of the starting state b and selects the new state directly with probability p_a: such a transition matrix would work perfectly and indeed one has $W^k = W \ \forall \ k$ in this case. However, the point is that we are resorting to Markov chains just because we cannot sample p_a by simple means.

W, from each copy of the ensemble: at each step k of the chain we monitor how the copies are distributed over Ω, thus defining the distribution $\pi_a^{(k)}$ as a function of the step.

It is easy to understand how $\pi_a^{(k)}$ evolves step by step: the elementary process by which a copy ends up in state a at step $k+1$ starting from state b at step k happens with probability $W_{ab}\pi_b^{(k)}$; then we just need to sum over all possible starting states b:

$$\pi_a^{(k+1)} = \sum_b W_{ab}\ \pi_b^{(k)}\ . \tag{17}$$

Hence, the evolution of the distribution is simply given by a row by column multiplication by the matrix W, and we can give a simple expression for the probability distribution at step k:

$$\pi_a^{(k)} = \sum_b (W^k)_{ab}\ \pi_b^{(0)}\ . \tag{18}$$

Our requirement on the single process, Eq. (16), can now be rephrased as a requirement on the evolution of $\pi_a^{(k)}$. We want that, independently of the starting distribution $\pi_a^{(0)}$,

$$\lim_{k\to\infty} \pi_a^{(k)} = \lim_{k\to\infty} \sum_b (W^k)_{ab}\ \pi_b^{(0)} = p_a \tag{19}$$

i.e. the distribution must converge to an *equilibrium distribution* equal to p_a; the adjective "equilibrium" stems from the fact that, if previous equations holds, than such distribution must be left unchanged by a single step of the chain. It is easy to show that the two conditions, Eq. (16) and Eq. (19) are equivalent: if Eq. (16) holds, then Eq. (19) trivially follows; on the other hand, if Eq. (19) holds, then Eq. (16) follows by choosing a starting distribution all concentrated in a single state, $\pi_{b'}^{(0)} = \delta_{b'b}$, where δ is the Kronecker delta symbol.

The rephrasing of the problem permits now to discuss it in terms of the spectrum of the matrix W, i.e. in terms of its eigenvalues λ and associated eigenvectors v_a:

$$\sum_b W_{ab}\ v_b = \lambda v_a^{(\lambda)}\ . \tag{20}$$

Indeed, it is clear that p_a must be an eigenvector of W with $\lambda = 1$, which is nothing else but the equilibrium condition. Then, in a few words, what we need to prove, in order to guarantee that Eq. (19) holds, is that there is a unique eigenvector with $\lambda = 1$, and that all other eigenvectors have $|\lambda| < 1$, so that any deviation of the starting distribution from the equilibrium one is washed away exponentially fast in the Markov chain time k. One can show that *regular* Markov chains satisfy such conditions, and in the following we discuss some details on how one arrives to this result, even if more rigorous presentations can be found elsewhere [**?**]. However, such details can be skipped by the uninterested reader, since the important point is then to learn how to construct a Markov chain having a given equilibrium distribution.

Theorem 2 *There is always at least one eigenvector with* $\lambda = 1$.
proof: *The normalization condition* $\sum_a W_{ab} = 1$ *implies that* $\sum_a (W_{ab} - \delta_{ab}) = 0$*, which is a condition of linear dependence among the rows of the matrix* $W - \mathrm{Id}$*, meaning that* $\det(W - \mathrm{Id}) = 0$*, so that the system* $W_{ab}\ v_b = v_a$ *has at least one solution.*

Theorem 3 *If* λ *is an eigenvalue of* W*, then* $|\lambda| \leq 1$.
proof: *Let* v_a *be an eigenvector with eigenvalue* λ*. Then considering the equation* $\sum_b W_{ab}\ v_b = \lambda v_a$ *and taking the absolute value of both members we obtain*

$$|\lambda||v_a| = |\sum_b W_{ab}\ v_b| \leq \sum_b |W_{ab}\ v_b| = \sum_b W_{ab}\ |v_b|$$

where we have exploited the fact that $W_{ab} \geq 0$*. If we sum the above inequality over* a *we obtain*

$$|\lambda| \sum_a |v_a| \leq \sum_{a,b} W_{ab}\ |v_b| = \sum_b |v_b| \quad \Longrightarrow \quad |\lambda| \leq 1$$

where the property $\sum_a W_{ab} = 1$ *has been used.*

Theorem 4 *If v_a is an eigenvector with $\lambda \neq 1$, then $\sum_a v_a = 0$.*
proof: *Considering the equation $\sum_b W_{ab}\ v_b = \lambda v_a$ and taking the sum over the first index we obtain*

$$\lambda \sum_a v_a = \sum_{a,b} W_{a,b}\ v_b = \sum_b v_b$$

from which the thesis follows since $\lambda \neq 1$.

Theorem 5 *If v_a is an eigenvector associated with an eigenvalue $\lambda = 1$ and if W is an ergodic chain, then it is possible to take $v_a \geq 0\ \ \forall\ a$, i.e. v_a can be normalized to be a probability distribution over Ω.*
proof: *Let us suppose that an eigenvector v with $\lambda = 1$ exists, which has both negative and positive elements. Then, the elements of Ω can be divided into two sets, those for which $v_a < 0$, and the other ones. Let us now consider the equation $v_a = \sum_b W_{ab}\ v_b$ and the associated inequality*

$$|v_a| = |\sum_b W_{ab}\ v_b| \leq \sum_b |W_{ab}\ v_b| = \sum_b W_{ab}\ |v_b|\,,$$

if the inequality is strict for at least one state then, by summing over a, we obtain an absurdity. However, in order to have only equalities, we need that for each case only addends having the same sign appear both on the left and on the right hand side. Since we have supposed that v_a can take either signs, we need that W_{ab} only connects couples of states for which v_a has the same sign. However, that implies that Ω can be divided into two sets such that it is not possible to go from one set to the other, in any number of steps, by the given Markov chain. This is absurd, since W was supposed to be ergodic.

The argument above does not take into consideration the possibility that the eigenvector v_a is complex, but little changes in this case. Repeating the above reasoning, one can write $v_a = \rho_a \exp(i\phi_a)$ and divide Ω into different subsets corresponding to states with the same ϕ_a. If $\lambda = 1$, then W cannot have matrix elements between states with different values of ϕ_a, i.e. it is not possible to go from one set to the other in any number of steps. That leads again to conclude either that W is non-ergodic, or that ϕ_a must be a constant independent of a, so that the eigenvector can be rotated in order to have all of its elements are positive.

Theorem 6 *If W is ergodic, the eigenvector associated with $\lambda = 1$ is unique (apart from an overall normalization constant).*
proof: *Let us suppose that $v_a = \sum_b W_{ab}\ v_b$ and $v'_a = \sum_b W_{ab}\ v'_b$, where both v and v' are normalized, $\sum_a v_a = \sum_a v'_a = 1$, and with $v_a \neq v'_a$ for at least some states. Then also $w_a = v_a - v'_a$ is an eigenvector with $\lambda = 1$, with at least some non-zero elements, however we have $\sum w_a = 0$, meaning that w has elements with differents signs. This is absurd because of previous theorem, since W is ergodic.*

Theorem 7 *If W is regular, then $\lambda \neq 1 \implies |\lambda| < 1$.*
proof: *Let us suppose instead that an eigenvalue $\lambda \neq 1$ with $|\lambda| = 1$ exists, and let v_a be an eigenvector associated with λ. As we have done above, we can write*

$$|\lambda||v_a| = |\sum_b W_{ab}\ v_b| \leq \sum_b |W_{ab}\ v_b| = \sum_b W_{ab}\ |v_b|$$

but now the strict equality must hold for each a, otherwise, after summing over a, we would obtain $|\lambda| < 1$. That means that all elements v_b appearing in the sum $\sum_b W_{ab}\ v_b$ must have the same phase, i.e. they can be written as $v_b = \rho_b \exp(i\phi_b)$ with the same $\phi_b = \bar{\phi}(a)$ and, since we can write $\lambda = \exp(i\phi_\lambda)$ with $\phi_\lambda \neq 0$, the element v_a on the left hand side must have a different phase $\phi_a = \bar{\phi}(a) + \phi_\lambda$.

Therefore, if we divide the states of Ω in classes corresponding to elements v_a having the same complex phase, we conclude that the probability that the Markov chain leaves a state belonging to a given class unchanged is exactly zero. That cannot be applied to states for which $v_a = 0$, however we need that at least some states exist for which $v_a \neq 0$: that must be true, otherwise we would have $\lambda = 0$.

Now, since we have supposed $|\lambda| = 1$, the above reasoning can be applied to a generic power of W, i.e. for a generic number k of steps of the Markov chain, unless $\lambda^{\bar{k}} = 1$ for some $\bar{k}$. That means that, if the system starts from a state with $v_a \neq 0$, the Markov chain will not bring it back to the same state but for integer multiples of $\bar{k}$, so that the period of the chain must be $\bar{k}$ at least (with $\bar{k} > 1$) if $\bar{k}$ exists, or infinite otherwise.

This is an absurdity, since by hypothesis the chain is regular (ergodic and aperiodic), hence its period must be one.

Let us summarize the results achieved above and derive some further conclusions. A regular Markov chain has a unique equilibrium distribution, let us call it $\pi_a^{(EQ)}$, which is left unchanged by the Markov chain evolution. All other eigenvectors correspond to eigenvalues λ such that $|\lambda| < 1$ and are characterized by the fact that the sum of their elements gives zero.

These facts are essential not only to be sure that equilibrium exists, but also to understand that we will actually converge to it, independently of the starting distribution. To see that, let us write the starting distribution as $\pi_a^{(0)} = \pi_a^{(EQ)} + \Delta\pi_a^{(0)}$, where $\Delta\pi^{(0)}$ represents its distance from the equilibrium one. Since both $\pi_a^{(0)}$ and $\pi_a^{(EQ)}$ are normalized probabilities, we have $\sum_a \Delta\pi_a^{(0)} = 0$, meaning that it can be expanded over the eigenvectors[6] with $\lambda \neq 1$, moreover its elements are bounded by $|\Delta\pi_a^{(0)}| \leq 2$. We can write

$$\pi_a^{(k)} = (W^k)_{ab}\, \pi_b^{(0)} = (W^k)_{ab}\, (\pi_b^{(EQ)} + \Delta\pi_b^{(0)}) = \pi_a^{(EQ)} + \Delta\pi_a^{(k)} \tag{21}$$

where $\Delta\pi_a^{(k)} = (W^k)_{ab}\, \Delta\pi_b^{(0)}$. Since $\Delta\pi^{(0)}$ lives in a space of eigenvectors with eigenvalues $|\lambda| < 1$, the magnitude of its elements will be contracted after each iteration of W. Therefore, defining l_{exp} as the *sup* of $|\lambda|$ over all $\lambda \neq 1$, we have

$$|\Delta\pi_a^{(k)}| \leq 2 l_{exp}^k = 2\exp(-k/\tau_{exp}) \tag{22}$$

where we have defined the so-called *exponential autocorrelation time*

$$\tau_{exp} \equiv -\frac{1}{\log l_{exp}}\,. \tag{23}$$

In conclusion, we will converge exponentially fast to the equilibrium distribution, with a relaxation time which depends on the starting distribution but is at most equal to τ_{exp}. The initial part of the Markov chain, needed to achieve convergence towards equilibrium, is usually called the *thermalization phase*. The exponential autocorrelation time is surely finite for systems with a finite number of possible states, however it could arbitrarily large for systems with infinite states, since l_{exp} could be arbitrarily close to 1. This possibility is not only academic, since we will face it as a real problem in numerical simulations of statistical systems close to a critical point.

4.2 Equilibrium condition and the detailed balance principle

The remaining question that needs to be answered is how to fix $\pi_a^{(EQ)} = p_a$, i.e. how to construct a regular Markov chain that converges to an assigned equilibrium distribution p_a. The task is much simpler than finding $\pi_a^{(EQ)}$ for a given W, which corresponds to solving an eigenvalue problem. We need instead to find a W having an assigned eigenvector of unit eigenvalue: as we will see, the problem has in general infinite solutions, a fact that can be properly exploited in order to optimize the Monte-Carlo sampling.

The equilibrium condition, $\sum_b W_{ab}\, p_b = p_a$ can be rewritten, after some simple steps[7], as a set of equations, one for each state a

$$\sum_{b\neq a} W_{ab}\, p_b = \sum_{b\neq a} W_{ba}\, p_a \tag{24}$$

which have a very simple and intuitive meaning in terms of the invariance of the population of state a. The left hand side represents the fraction of copies of the system which, being originally in a state other than a, move to a after one iteration of the Markov chain. The right hand side reprents the fraction of copies which, being originally in state a, move to a different state during the same step. The equation then represents an exact balance between the incoming and outgoing copies, which leaves the population of a, hence p_a, unchanged.

Eq. (24) reprents a very loose set of constraints for W_{ab}, which leaves a large freedom[8] and are not easy to be checked on a given algorithm. Therefore, since one is not interested in determining all possible solutions, but just some of them which can be suitable for Monte-Carlo sampling, it is usual to adopt a more restrictive set of equations, which are a sufficient, but not necessary:

$$W_{ab}\, p_b = W_{ba}\, p_a \quad \forall\ a, b\ \in\ \Omega\,. \tag{25}$$

[6] We are not completely accurate on this point, since one would need to show the existence of a basis of eigenvectors for the transition matrix W, but this is not always the case.

[7] Exploiting the normalization condition for W, one can rewrite $p_a = p_a \sum_b W_{ba}$.

[8] Let us assume, to fix ideas, that Ω has a finite number N of states. Then W, taking into account normalization conditions, is written in terms $N^2 - N$ unknowns, which are poorly constrained by the N equations (24).

In practice, the balance between incoming and outgoing populations is not imposed as a whole on each state, but separately for each couple of states, checking that the total flux going from one state to the other is zero. This is known as the *detailed balance principle*, it guarantees the general balance and equilibrium condition (24), as can be checked by summing Eq. (25) over $b \neq a$, and is much more practical, since it is usually easier to check Eq. (25) having in mind two well definite states of the systems, than to check an integral condition like Eq. (24).

Of course, in this way we are missing a large number of possible algorithms which would satisfy just the general balance condition, however detailed balance is still loose enough to allow for a large variety of solutions and possible algorithm optimization. In the following we are going to analyze two classical schemes of algorithms, known respectively as the Metropolis and the heat-bath algorithms, which satisfy the detailed balance principle.

4.3 The Metropolis algorithm

There is a whole class of algorithms respecting the detailed balance principle which takes its name from the first author (Nicholas Metropolis) of a famous paper, dated 1953, where one of the first applications of Monte-Carlo simulations to the study of the statistical properties of matter was presented [**?**].

Let us describe the scheme of the typical Metropolis algorithm. In order to assign the elements of the matrix W, we need to describe the stochastic process by which we move from state a to state b in one elementary step of the chain:

1. Starting from a, a new, trial state $\tilde{b}$ is selected according to a *tentative* transition probability $A_{\tilde{b}a}$. The matrix A is properly normalized, so that it could define a Markov chain by itself, however it can be completely unrelated to the equilibrium distribution we are looking for. We will just impose that A is symmetric, i.e. $A_{\tilde{b}a} = A_{a\tilde{b}}$, as in the original paper, however even this constraint can can be relaxed by properly generalizing the algorithm;
2. If $p_{\tilde{b}} > p_a$, then we take $b = \tilde{b}$;
3. If instead $p_{\tilde{b}} < p_a$, then $b = \tilde{b}$ with probability $p_{\tilde{b}}/p_a$, and $b = a$ otherwise.

Last step is usually called the Metropolis, or acceptance, step. Notice that if the move is rejected, i.e. if $b = a$, the Markov chain step is considered as done anyway, as if state a had been extracted multiple consecutive times.

Later we will better explicit the various steps described above with a specific example. Now instead we verify that the algorithm satisfies detailed balance. In order to do that, we need consider a generic couple of states, a and b, and compute the two transition probabilities, W_{ab} and W_{ba} associated with the stochastic process described above. In order to fix ideas, let us assume that $p_a \leq p_b$. Then the probability of moving from a to b is simply given by $W_{ba} = A_{ba}$, since the move will be accepted for sure; instead the probability of moving from b to a will be $W_{ab} = A_{ab}\, p_a/p_b$, since the move is accepted with probability p_a/p_b. Then, exploiting the symmetry of the matrix A, we have:

$$W_{ab}\, p_b = A_{ab} \frac{p_a}{p_b}\, p_b = A_{ba}\, p_a = W_{ba}\, p_a \tag{26}$$

which is indeed the detailed balance principle. Of course, detailed balance is not enough and one should check, case by case, that the chain is also ergodic and aperiodic.

In some applications, it is not easy to build the tentative transition probability A_{ab} so that it is symmetric. In these cases, one can resort to a generalization of the algorithm, known as the Metropolis-Hastings algorithm, after W.K. Hastings who proposed it in 1970 [**?**]. It reads as follows:

1. Starting from a, a new, trial state $\tilde{b}$ is selected according to $A_{\tilde{b}a}$;
2. One then considers the ratio
$$r = \frac{p_{\tilde{b}}\, A_{a\tilde{b}}}{p_a\, A_{\tilde{b}a}}$$
and one takes $b = \tilde{b}$ with probability r or 1 if $r > 1$, and $b = a$ otherwise.

Detailed balance can be easily verified as we have done above for the standard case.

An important feature of the Metropolis-Hastings algorithm, which makes it suitable for a large variety of systems, is that we never need to know the exact value of the probability p_a (nor that of the

tentative transition probability), but just that of probability ratios, i.e. the algorithm is not affected by our ignorance about the overall normalization factor entering p_a. This is important especially for complex systems, for which an exact computation of the normalization constant could be difficult, think for instance of statistical systems, where in order to know the normalization constant of the Boltzmann distribution one needs to compute the partition function.

Example:
Suppose we want to generate a sample of real variables according to the normal distribution, $p(x) \propto \exp(-x^2/2)$. We already know the Box-Müller algorithm, which permits to perform independent extractions efficiently. However this is a great example where to learn how to implement the Metropolis algorithm in practice, and which gives us the opportunity to discuss the possible subtleties one should consider when treating a system characterized by a continuous (instead of discrete) set of states, which in this case are the possible values of x.

We have to generate a sequence of extractions $x_0, x_1, \dots, x_k, \dots$, starting from a given x_0 and assigning how we move from x_k to x_{k+1}. A possible algorithm is the following:

1. we choose $\tilde{x}$ with a uniform random distribution in the interval $[x_k - \delta, x_k + \delta]$;
2. if $\exp(-\tilde{x}^2/2) > \exp(-x_k^2/2)$ then $x_{k+1} = \tilde{x}$;
3. otherwise, we extract u randomly in $[0, 1]$ and if $u < \exp(-\tilde{x}^2/2 + x_k^2/2)$ then $x_{k+1} = \tilde{x}$, otherwise $x_{k+1} = x_k$.

It is apparent that, moving step by step, we can reach any point of the real line from any other point in a finite number of steps; moreover, at each step we have a non-negligible probabiity of remaining in the same point; therefore the chain is ergordic and aperiodic. The tentative transition probability is clearly symmetric in this case: the particular choice can of course be modified at will, one could take it Gaussian, or anything else. The parameter δ can be chosen freely, it is usually taken fixed during the Markov chain and, as we will see, it characterizes the efficiency of the algorithm. It is interesting to notice that step 2) could be avoided by doing step 3) directly, since if condition 2) holds then the acceptance test 3) is surely passed: in this way one would avoid an *if* condition, with the risk of extracting an unnecessary random variable u; what is better depends on efficiency considerations regarding the particular code implementation (and not the algorithm itself).

The algorithm seems a trivial implementation of the Metropolis scheme, however there are some important details we should discuss more in deep. The stochastic variable is continuous and, in order to make the verification of detailed balance sound, it is better to discuss in terms of transition probabilities between infinitesimal intervals. Let us consider two intervals, $[x, x + dx]$ and $[y, y + dy]$: the given distribution, p, assigns to them a probability $p(x)dx$ and $p(y)dy$, respectively. If we start from the first interval, the probability of selecting a point in the second interval is $A(y, x)dy = dy\,\theta(\delta - |x - y|)/(2\delta)$, where the θ-function is defined such that $\theta(x) = 1$ for $x \geq 0$ and $\theta = 0$ for $x < 0$. Notice that $A(y, x) = A(x, y)$, however this does not imply a symmetry for the tentative transition probability, since in general $dx \neq dy$. The parameter needed for acceptance of the transition $x \to y$ is therefore that of Hastings' generalization:

$$r = \frac{p(y)dy\,A(x, y)dx}{p(x)dx\,A(y, x)dy} = \frac{p(y)}{p(x)} \tag{27}$$

which is indeed the one we have used. Now, assuming $p(y) < p(x)$, detailed balance is verified since

$$A(x, y)dx\,p(y)dy = \frac{p(y)}{p(x)}\,A(y, x)dy\,p(x)dx = A(y, x)dy\,p(y)dx = A(x, y)dy\,p(y)dx\,.$$

The above reasoning may seem a more cumbersome way to find the same simple result. However, the important point is that the simplification leading to $r = p(y)/p(x)$ takes place because *the measure used to define the probability distribution and the measure used to define the tentative transition probability are the same.* To better appreciate this point, you should consider the following variation of the exercise above. Suppose the real line has not a "flat" measure, i.e. that the measure of intervals be given by $\mu(x)dx$, where $\mu(x)$ is some positive function: that could be the case after some change of variable has been performed, or when one considers multivariate systems. Suppose we want to approach the same problem, i.e. to construct a Markov chain for an equilibrium probability distribution $p(x)$, meaning now that the probability of finding the stochastic variable in an interval $[x, x + dx]$ is $p(x)\mu(x)dx$. We can proceed as above, and choose the tentative new variable $\tilde{x}$ uniformly distributed in the interval $[x_k - \delta, x_k + \delta]$. What is the correct acceptance parameter now? Where does the non-uniform measure $\mu(x)$ enter? The point now

is that the tentative transition probability has not changed, i.e. the probability of selecting the tentative change in $[y, y+dy]$ is still $A(y, x_k)dy$, as if the measure were flat, hence from Eq. (27) one obtains $r = (p(\tilde{x})\mu(\tilde{x}))/(p(x_k)\mu(x_k))$. However, were we able to select the tentative transition proportionally $A(y, x_k)\mu(y)dy$, the acceptance parameter would be $r = p(\tilde{x})/p(\tilde{x}_k)$.

In order to further discuss this point, we refer the reader to exercises 1 and 2.

4.4 Heat-Bath Algorithms

We are going to describe a class of algorithms which can be useful for multivariate system with a large number of stochastic variables. The general idea is to perform a Markov chain step in which most variables are kept fixed and just a few of them are changed; in particular, completely independent values are extracted for those variable, according to the conditional probability obtained from the original probability distribution for fixed values of the unchanged variables. Therefore, the unchanged variables play the role, for the subset of modified variables, of a thermal heat-bath which fixes their distribution function (hence the name of the algorithm).

An algorithm of this kind is practicable only if the conditional distribution of the subset variables can be sampled easily and efficiently; from this point of view, it is not really important that the subsystem be small, but just that its distribution can be easily sampled. Moreover, in most cases at each step of the Markov chain the modified subset is chosen randomly, in order to ensure ergodicity. We are now going to discuss the algorithm more in detail and prove that it respects detailed balance.

Let us rewrite the index labelling the possible states in the form $a \equiv (\bar{a}, \alpha)$ where α parametrizes the possible states of the subsystem, and $\bar{a}$ those of the remaining set of variables. Therefore, the sum over all states can be rewritten as $\sum_a = \sum_{\bar{a}} \sum_\alpha$ and the probability distribution can be written as $p_a = p(\bar{a}, \alpha)$. We can define a probability distribution for the $\bar{a}$ variables, by integrating over the α variables

$$\bar{p}(\tilde{a}) = \sum_\alpha \; p(\tilde{a}, \alpha)\,. \tag{28}$$

The conditional probability, $P(\alpha|\bar{a})$, can be written in terms of p and $\bar{p}$:

$$p(\bar{a}, \alpha) = \bar{p}(\bar{a}) P(\alpha|\bar{a}) \tag{29}$$

from which we obtain

$$P(\alpha|\bar{a}) = \frac{p(\bar{a}, \alpha)}{\bar{p}(\bar{a})} = \frac{p(\bar{a}, \alpha)}{\sum_{\alpha'} \; p(\bar{a}, \alpha')}\,. \tag{30}$$

The elementary step of the Markov chain is built as follows:

1. $\bar{a}_{k+1} = \bar{a}_k$
2. α_{k+1} is independent of α_k and extracted according to $P(\alpha_{k+1}|\bar{a}_{k+1})$

so that the transition probability can be written as:

$$W_{(\bar{b},\beta),(\bar{a},\alpha)} = P(\beta|\bar{b})\, \delta_{\bar{b}\bar{a}}\,. \tag{31}$$

Detailed balance is now checked by noting that

$$W_{(\bar{b},\beta),(\bar{a},\alpha)} \; p(\bar{a}, \alpha) = \delta_{\bar{b}\bar{a}}\, P(\beta|\bar{b}) \; P(\alpha|\bar{a}) \; \bar{p}(\bar{a}) \tag{32}$$

where Eq. (29) has been used: this is exactly equal to the result obtained by computing $W_{(\bar{a},\alpha),(\bar{b},\beta)} \; p(\bar{b}, \beta)$.

Example:
Suppose we have to devise a Markov chain for sampling N stochastic variables distributed according to

$$p(y_1, y_2, \; \ldots \; y_N) \; dy_1 \; dy_2 \; ...dy_N \propto \exp\left(-\prod_j \; y_j^2\right) \prod_j dy_j\,.$$

A possible algorithm, based on the heat-bath scheme, is the following. Let $y_j^{(k)}$ be the values of the stochastic variables at step k of the Markov chain.

1. select i at random in $1, ..., N$
2. $y_j^{(k+1)} = y_j^{(k)}$ for $j \neq i$, while $y_i^{(k+1)}$ is extracted according the Gaussian distribution proportional to $\exp(-Ay_i^2)$, where $A = \prod_{j \neq i} (y_j^{(k+1)})^2$.

4.5 Composition of Markov Chains

It is usual to deal with problems where one can devise several algorithms (Markov chains), each having some good features, but failing on some particular aspect. In this respect, it is important to notice that Markov chains can be combined to create new Markov chains. Suppose W^A and W^B be two regular Markov chains having the same equilibrium distribution p_a, i.e. $\sum_b W^A_{ab}\ p_b = p_a$ and $\sum_b W^B_{ab}\ p_b = p_a$. We can define a new Markov chain W, in which at each step we select the step of Markov chain W^A with probability $w > 0$, and the one of Markov chain W^B with probability $(1 - w)$, that means

$$W_{ab} = wW^A_{ab} + (1 - w)W^B_{ab}\,. \tag{33}$$

It is trivial to check that p_a is an equilibrium distribution for W too. In order to verify that W is a good Markov chain, we just need to prove that it is regular. Now, taken a generic couple of states, we know that at least two paths exists, leading from one state to the other, both for W^A and W^B: each path is also a possible path for the newly defined chain W, since the probability that we always select W^A (or W^B) along each path is non-negligible. Also the number of paths leading from one state to itself includes at least all those one could find for W^A and W^B separately, therefore the G.C.D. of their lengths cannot increase, i.e. W will be aperiodic as well.

The argument above can be easily repeated for the composition of an arbitrary number of Markov chains: the choice of the different weights by which each chain is selected at each step is a matter of optimization.

5 Advanced Error Analysis Techniques

In Section 1 we have learned how to estimate the statistical error on the Monte-Carlo evaluation of some average over a finite sample: that was actually the basic approach, which can be safely used only when one is interested in computing the average of a given function of the stochastic variables, and assuming the sample is composed of statistically independent draws. We are now going to explore two different ways in which the basic approach could not work, thus requiring a more careful thinking. The first is a consequence of the correlation existing between consecutive draws of a Markov chain Monte-Carlo: they are not statistically independent, each draw moving away from the previous one according to the assigned transition probability. The second is related to the fact that sometimes one is interested in less trivial functionals of the probability distribution, which are not just simple averages of some function of the stochastic variables: combinations of different averages, like cumulants, are a typical example. The way to correctly estimate the error on the Monte-Carlo evaluation obtained for such functionals is non-trivial: standard error propagation from the errors obtained for the single averages reveals to be the wrong choice; moreover, in some cases, even the estimate obtained for the functional in the limit of asymptotically large sample size is wrong, meaning that a *bias* appears and one should consider also a systematic error, in addition to the statistical one.

Often these two kind of problems combine, and various kinds of advanced error analysis techniques have been developed to properly deal with them. We are going to expose some of them and, in order to make the discussion as simple as possible, our playground will be the normal distribution for a single stochastic variable.

5.1 Error estimate in the presence of autocorrelations

Let us consider a Metropolis algorithm for sampling a stochastic variable x distributed according to

$$p(x)\,dx \propto \exp\left(\frac{(x-\mu)^2}{2\sigma^2}\right)\,dx \tag{34}$$

with $\mu = 5$ and $\sigma^2 = 1$. The algorithm has been already outlined in Section 4.3. We produce a sequence $x_0, x_1, \ldots$ of draws where x_{k+1} is chosen starting from x_k as follows: $x^{(t)}$ is selected randomly in $[x_k - \delta, x_k + \delta]$, then $x_{k+1} = x^{(t)}$ if $p(x^{(t)}) > p(x_k)$, otherwise $x_{k+1} = x^{(t)}$ with probability $p(x^{(t)})/p(x_k)$ and $x_{k+1} = x_k$ otherwise.

On the left-hand side of Fig. 2 we show a sequence produced by 15000 iterations of the algorithm above, starting from $x_0 = 0$ and adopting $\delta = 0.1$. The starting point is quite far from equilibrium (it is five standard deviations from the average of the normal distribution), hence we need some iterations to achieve equilibrium: Fig. 2 shows that, in our case, the thermalization phase lasts for at least 1000 iterations. It

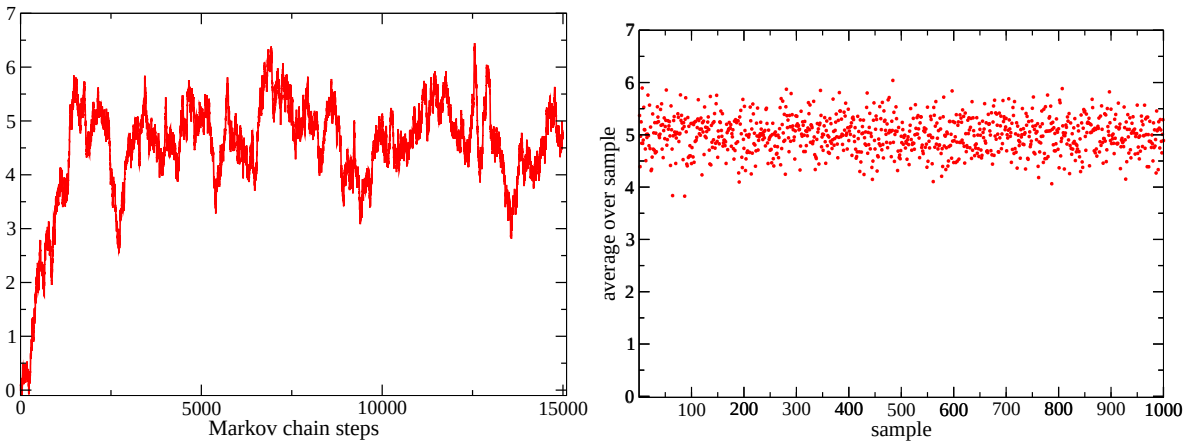

Figure 2: Left: sequence of 15000 Metropolis steps with $\delta = 0.1$ for sampling the normal distribution in Eq. (34). Right: sample averages $\tilde{x}$ for 1000 different samples extracted by the same algorithm, each made up of 10 thousands steps.

is usually wise to be generous in estimating the duration of such phase: the statistical error will not be much affected by discarding a larger number of iterations, while the damage of including draws which are still not properly distributed could be much worse. Therefore, we have decided to discard the first 5000 iterations, taking the last $N = 10000$ draws as our Monte-Carlo sample. From now on, $x_i,\ i = 1, 10000$ will refer to such sample.

We will now estimate the average $< x >$ as outlined in Section 1,

$$\langle x \rangle = \tilde{x} \pm \sigma \,,$$

and compare it to the known result $\langle x \rangle = \mu = 5$. We have

$$\tilde{x} = \frac{1}{N} \sum_{i=1}^{N} x_i \quad ; \qquad \sigma_{\tilde{x}}^2 = \frac{1}{N} \frac{1}{N-1} \sum_{i=1}^{N} (x_i - \tilde{x})^2 \tag{35}$$

and the result we get is

$$\langle x \rangle = 4.721 \ \pm \ 0.007 \tag{36}$$

which clearly indicates that something wrong is going on: we are about 40 standard deviations off the true average $\mu = 5$. Either the Markov chain algorithm is not reproducing the correct distribution, or our estimate of the standard deviation of the sample mean is largely underestimated.

In order to better understand this point, we have produced 1000 different samples as the one above (in practice, we have iterated the Markov chain for 10 million steps and divided it in 1000 different sequences), to see how the sample average $\tilde{x}$ is actually distributed. Results for the averages obtained from the different samples are reported in the right-hand side of Fig. 2, from which we learn two things: sample averages are evenly distributed around $\mu = 5$, as expected, however their standard deviation is very far from $\sigma_{\tilde{x}} = 0.007$ estimated above, and rather of the order of 0.1 or larger, i.e. at least one order of magnitude larger than our wrong estimate. Eq. (35) returns the wrong result because its derivation in Section 1 assumes statistically independent draws, which is not true for a Markov chain Monte-Carlo. Hence, we should work out again how the sample average is expected to fluctuate in this case.

The quantity we are interested in is

$$\sigma_{\tilde{f}}^2 = \langle (\tilde{f} - \langle f \rangle)^2 \rangle_s \tag{37}$$

where in our case $f(x) = x$ and by $\langle \cdot \rangle_s$ we mean the *average over the extraction probability of a particular sample*. In the case of N independent draws, such an extraction probability is simply the product of N independent probability distributions $\prod_{k=1}^{N} p(x_k)$; if instead the sample is draw as a piece of a Markov chain, the sample extraction probability is given by the Markov process itself.

Eq. (37) can be rewritten as follows

$$\begin{aligned}\sigma_{\bar{f}}^2 &= \left\langle \left(\frac{1}{N}\sum_j f_j - \langle f\rangle \right)^2 \right\rangle_s = \left\langle \left(\frac{1}{N}\sum_j (f_j - \langle f\rangle) \right)^2 \right\rangle_s \\ &= \left\langle \frac{1}{N^2}\sum_i\sum_j \delta f_i\, \delta f_j \right\rangle_s = \frac{1}{N^2}\sum_i\sum_j \langle \delta f_i\, \delta f_j\rangle_s \end{aligned} \tag{38}$$

where $\delta f_i \equiv f_i - \langle f\rangle = f(x_i) - \langle f\rangle$ measures the deviation (fluctuation) of f at the i-th step from its average value. For independent draws we have $\langle \delta f_i \delta f_j\rangle_s = \sigma_f^2\, \delta_{ij}$ and one recovers the result of Section 1. When this is not true one defines the so-called *autocorrelation function* for f

$$C_f(i,j) \equiv \frac{\langle \delta f_i\, \delta f_j\rangle_s}{\sigma_f^2}\,, \tag{39}$$

which serves as a measure of the correlation between two different draws of f and is normalized so that $C_f(i,j) = 1$ when $i = j$. We can state some useful properties of C_f and also predict, in the case of a regular Markov chain, its asymptotic behaviour in the limit of large separation between i and j.

First of all, $C_f(i,j)$ is, by definition, symmetric under exchange of i and j and, for a time-independent Markov process, can only depend on the distance between i and j, i.e. $C_f(i,j) = C_f(|i-j|)$. Moreover, one has $|C_f(|i-j|)| \leq 1$, has can be easily proven by writing

$$2\,\delta f_i\, \delta f_j = \pm(\delta f_i^2 + \delta f_j^2) \mp (\delta f_i \mp \delta f_j)^2$$

then taking the average $\langle\cdot\rangle_s$ of both sides.

Let us now compute $C_f(|i-j|)$ in terms of the transition matrix of the Markov process. In order to properly compute the autocorrelation function, we should state what is the probability of extracting state a at step i and then state b at state j (setting $j > i$ in order to fix ideas). Assuming the Markov chain has reached equilibrium, the probability of being in state a at step i is simply the sampling probability p_a. The probability of moving to b after $j-i$ steps is instead given by the transition matrix $(W^{|j-i|})_{ba}$. Therefore, the sought probability is $(W^{|j-i|})_{ba}\, p_a$ and we can write:

$$\langle \delta f_i\, \delta f_j\rangle_s = \sum_a\sum_b (W^{|j-i|})_{ba}\, p_a\, \delta f(a)\, \delta f(b)\,. \tag{40}$$

We can exploit the fact that, since the Markov chain is regular, for $|j-i| \gg 1$ $(W^{|j-i|})_{ba}$ is expected to converge exponentially fast to the equilibrium transition matrix, i.e.

$$(W^{|j-i|})_{ba} = p_b + R_{ba}(|j-i|) \quad \text{with} \quad |R_{ba}(|j-i|)| \sim O(e^{-|j-i|/\tau_{exp}}) \tag{41}$$

so that

$$\begin{aligned}\langle \delta f_i\, \delta f_j\rangle_s &= \sum_a\sum_b p_b\, p_a\, \delta f(a)\, \delta f(b) + \sum_a\sum_b R_{ba}(|j-i|)\, p_a\, \delta f(a)\, \delta f(b) \\ &= \langle \delta f\rangle\langle \delta f\rangle + \sum_a\sum_b R_{ba}(|j-i|)\, p_a\, \delta f(a)\, \delta f(b) \\ &= \sum_a\sum_b R_{ba}(|j-i|)\, p_a\, \delta f(a)\, \delta f(b) \lesssim O(e^{-|j-i|/\tau_{exp}}) \end{aligned} \tag{42}$$

where we have used the fact that, by definition, $\langle \delta f\rangle = 0$. Therefore, the autocorrelation function is expected to vanish exponentially fast for large separation, with a decay time which is at most τ_{exp}, but can be even shorter and depends on the particular function f one is considering.

We can now go back to Eq. (38) and rewrite it as follows

$$\sigma_{\bar{f}}^2 = \frac{1}{N^2}\sum_{i=1}^N\sum_{j=1}^N \langle \delta f_i\, \delta f_j\rangle_s = \sigma_f^2 \frac{1}{N^2}\sum_{i=1}^N\sum_{j-i} C_f(|j-i|) \tag{43}$$

where we have rewritten the double sum in a different way. For a finite sample, the range of the sum over the difference $j-i$ depends on i, however if the sample is large enough, $N \gg \tau_{exp}$, we commit a negligible

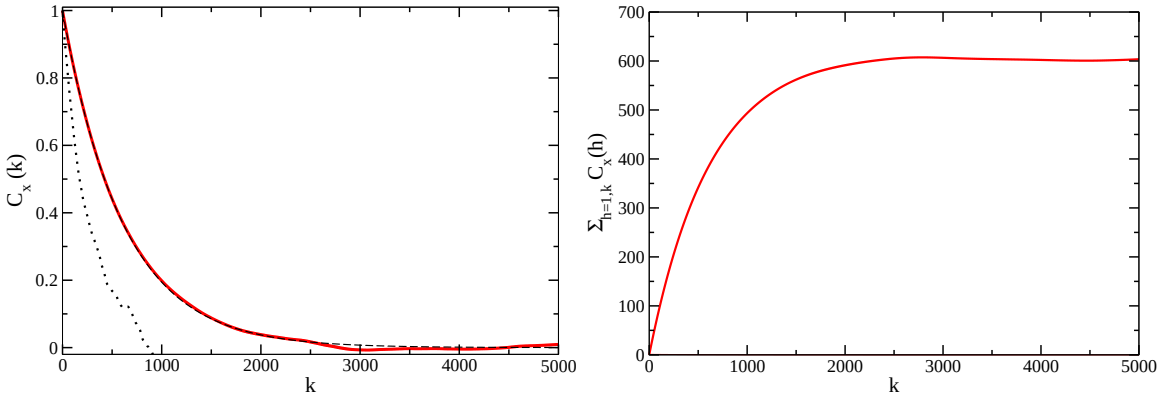

Figure 3: Left: autocorrelation function $C_x(k)$ for the Metropolis algorithm with $\delta = 0.1$, computed from a sample of 10 millions steps; the dashed line is the function $\exp(-k/\tau)$ with $\tau = 600$, the dotted line is the autocorrelation function computed on a subsample of just 10 thousands steps. Right: integral of the autocorrelation function showed on the left. In particular we show $\sum_{h=1}^{k} C(h)$ as a function of k: the asymptotic value yields the integrated autocorrelation time τ_{int}.

error by extending the sum from $-\infty$ to $+\infty$, since $C_f(|j-i|)$ vanishes exponentially fast on both sides. Hence, the two sums become decoupled and we can write

$$\begin{aligned} \sigma^2_{\bar{f}} &\simeq \sigma^2_f \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j-i=-\infty}^{+\infty} C_f(|j-i|) \\ &= \sigma^2_f \frac{1}{N} \sum_{j-i=-\infty}^{+\infty} C_f(|j-i|) = \frac{\sigma^2_f}{N}(1+2\tau_{int}) \end{aligned} \tag{44}$$

where we have introduced the so-called *integrated autocorrelation time*

$$\tau_{int} \equiv \sum_{k=1}^{\infty} C_f(k) \tag{45}$$

which for an exponentially vanishing function $C_f(k)$ coincides with its decay time, i.e. if $C_f(k) = e^{-k/\tau}$ (with $\tau \gg 1$) then $\tau_{int} \simeq \tau$. Eq. (44) has a simple interpretation: the variance of the sample mean is not σ^2_f/N but instead σ^2_f/N_{eff}, with $N_{eff} = N/(1+2\tau_{int})$, as if our sample were composed of N_{eff} effective independent draws. This is a reasonable result, since draws are correlated among themselves over a range of the order of τ_{int}.

Let us now go back to our original exercise. The left-hand side of Fig. 3 shows the autocorrelation function of the variable x, for $|i-j|$ up to 5000 and for the Metropolis algorithm with $\delta = 0.1$ discussed above. In practice, the function has been computed from a sample of $N' = 10$ millions iterations of the Markov chain as follows

$$C_x(k) = \frac{1}{N'-k} \sum_{i=1}^{N'-k} (x_i - \tilde{x})(x_{i+k} - \tilde{x})$$

where $\tilde{x}$ is the sample average evaluated on the same data set. In principle one could also give an estimate of the statistical error on the determination of $C_x(k)$, but we skip the discussion of this point. The exponential decay of the autocorrelation function is clearly visible. On the right-hand side of Fig. 3 we also show the "integral" of the autocorrelation function up to a maximum k, as a function of k: the integral clearly reaches an asymptotic value corresponding to $\tau_{int} \simeq 600$; the function $\exp(-k/\tau_{int})$ also reproduces $C_x(k)$ quite well, as can be appreciated from the figure on the left-hand side.

Now, taking into account the result obtained for τ_{int}, we must revisit our previous estimate of the average on the sample with $N = 10000$ data, in particular of its error. That now becomes

$$\langle x \rangle = 4.721 \pm 0.007\sqrt{2\tau_{int}+1} = 4.72 \pm 0.24 \tag{46}$$

which is perfectly compatible with the exact average $\mu = 5$, being at slightly more than one estimated standard deviation from it. The error reported above is still not completely accurate, since the correct standard deviation of the sample average is $\sqrt{\sigma_x^2(2\tau_{int}+1)/N} \simeq 0.35$, which is very close to the standard deviation observed for the distribution of sample averages on the right-hand side of Fig. 2; the reason is that σ_x^2 itself has been estimated from the finite sample and is affected by a statistical fluctuations itself.

Actually, we have been cheating a little bit. The autocorrelation function and its integral have been estimated on a very large sample of 10 millions draws, of size much bigger than τ_{int} itself, rather than on the original sample of 10 thousands draws: this is the reason we obtain so clean results for $C_x(k)$ and its integral in Fig. 3. This is usually not possible in real life, where the sample size is constrained by computational budget limits. Therefore, in Fig. 3 we have also reported the autocorrelation function obtained from the original small sample of $N = 10000$ data: results are quite different, we would have missed the correct autocorrelation time by roughly a factor 2, leading to an estimated standard deviation for the sample average of about 0.16 (instead of 0.24). This is not dramatic, the true average is still within 2 standard deviations. After all, in this game we are rolling dices, and we can just be a bit unlucky: two standard deviations could just be a manifestation of bad luck, 40 standard deviations (what we obtained before considering autocorrelations) is the clear manifestation of an evil bug somewhere.

It is interesting to notice that, in this particular simple case, the autocorrelation time of the Metropolis algorithm can also be estimated analitically. In order to simplify the calculation, we will do it in the limit of small δ, and consider a normal distribution with zero average and unit variance, the shift in the average μ being completely irrelevant to our purposes. The autocorrelation function after 1 Metropolis step can be written as

$$\begin{aligned}
\langle x_i\, x_{i+1}\rangle_s &= \frac{1}{\sqrt{2\pi}}\frac{1}{2\delta}\int_{-\infty}^{\infty} dx \int_{-\delta}^{\delta} dy\; e^{-x^2/2}\, x\,[(x+y)\,p_{acc} + x\,(1-p_{acc})] \\
&= \frac{1}{\sqrt{2\pi}}\frac{1}{2\delta}\int_{-\infty}^{\infty} dx \int_{-\delta}^{\delta} dy\; e^{-x^2/2}\, x\,[x + y\,p_{acc}] \\
&= 1 + \frac{1}{\sqrt{2\pi}}\frac{1}{2\delta}\int_{-\infty}^{\infty} dx \int_{-\delta}^{\delta} dy\; e^{-x^2/2}\, x\, y\, p_{acc}(x,y)
\end{aligned}$$

where x is the value at the starting step, $x+y$ is the tentative new value at the next step and p_{acc} is the probability of accepting it, which depends both on x and y, as explicited in the last line; moreover we have already used the fact that $\langle x^2\rangle_s = \sigma_x^2 = 1$. Assuming $\delta \ll \sigma_x = 1$, we can now limit ourselves to the case of x and $x+y$ both strictly positive or negative: different combinations will give negligible corrections to our computation. Taking the positive case first, we have $p_{acc}(x,y) = 1$ for $y < 0$, while for $y > 0$ we can write

$$p_{acc}(x,y) = \exp(-(x+y)^2/2 + x^2/2) \simeq \exp(-xy) \simeq 1 - xy$$

where have we exploited the fact that $y \ll 1$ and that x is of $O(1)$. Therefore we obtain

$$\int_{-\delta}^{\delta} dy\; x\, y\, p_{acc}(x,y) \simeq \int_{-\delta}^{0} dy\; x\, y + \int_{0}^{\delta} dy\; x\, y\,(1 - x\, y) = -x^2\frac{\delta^3}{3}\,,$$

and exactly the same result is obtained for $x < 0$. Finally we have

$$\langle x_i\, x_{i+1}\rangle_s \;\simeq\; 1 - \frac{1}{\sqrt{2\pi}}\,\frac{1}{2\delta}\,\frac{\delta^3}{3}\,\int_{-\infty}^{\infty} dx\, e^{-x^2/2}\, x^2 \;= 1 - \frac{\delta^2}{6}\,.$$

From that, assuming $\langle x_i x_{i+k}\rangle_s = \exp(-k/\tau)$, we derive

$$\exp(-1/\tau) = 1 - \frac{\delta^2}{6} \quad \Longrightarrow\; \tau \simeq \frac{6}{\delta^2} \tag{47}$$

which for $\delta = 0.1$ gives $\tau \simeq 600$, which is perfectly consistent with what we have found numerically.

5.1.1 Data blocking techniques

A faster and more practical way to correctly estimate the statistical error on standard sample averages is based on the idea of repeating the naive computation, Eq. (35), on a modified sample obtained from the original one by averaging adjacent draws in a given block.

Let us consider again the sample of N draws x_i and, assuming N is even, let us define a new sample of blocked data $x_j^{(1)}$, $j = 1, N/2$, with

$$x_j^{(1)} \equiv \frac{x_{2j-1} + x_{2j}}{2} \; ; \tag{48}$$

obviously we have $\overline{x^{(1)}} = \tilde{x}$, i.e. the average sample estimate is left unchanged by the blocking operation. We can now apply again the naive definition for the variance of the sample average to the blocked sample

$$\begin{aligned} \sigma^2_{(1)} &= \frac{1}{N/2}\frac{1}{N/2-1}\sum_{j=1}^{N/2}(x_j^{(1)} - \tilde{x})^2 \simeq \frac{1}{N/2}\frac{1}{N/2}\sum_{j=1}^{N/2}\frac{1}{4}(\delta x_{2j-1} + \delta x_{2j})^2 \\ &\simeq \frac{1}{N}\frac{1}{N}\sum_{j=1}^{N/2}(\delta x_{2j-1}^2 + \delta x_{2j}^2 + 2\delta x_{2j-1}\delta x_{2j}) \simeq \sigma^2_{(0)} + \frac{1}{N}\sigma_x^2\langle \delta x_{2j-1}\delta x_{2j}\rangle \\ &\simeq \sigma^2_{(0)}\,(1 + C_x(1)) \end{aligned} \tag{49}$$

where we have defined $\sigma^2_{(0)}$ as the naive estimate of $\sigma^2_{\tilde{x}}$ on the original sample, $\delta x \equiv x - \tilde{x}$, and we have made use of the standard definition of autocorrelation function and of approximations valid for $N \gg 1$. We thus see that, in case of statistically independent draws, the naive estimate of the variance of the average remains unchanged for the blocked sample, while it grows in the case of positive autocorrelation, which is the usual case.

We can now iterate such a procedure, i.e. define a new sample $x_l^{(2)}$, $l = 1, N/4$, by blocking pairs of the sample $x_j^{(1)}$, and so on. After k steps of iteration, a sample $x_i^{(k)}$ of $N/2^k$ data[9], corresponding to averages over blocks of 2^k entries of the original sample, will be obtained, with an associated naive estimate $\sigma^2_{(k)}$. We expect that the sequence $\sigma^2_{(k)}$ be an increasing function of k, until a value of k is reached such that 2^k is of the order of the autocorrelation time of the original sample, so that adjacent blocks are effectively decorrelated and further iterations do not change the naive σ^2 estimate further. The saturation value of $\sigma^2_{(k)}$ can be taken as a correct estimate of the true variance of the sample average. Indeed we have

$$\sigma^2_{(k)} \simeq \frac{2^k}{N}\left(\frac{2^k}{N}\sum_{j=1}^{N/2^k}\left(\frac{x_{2^k(j-1)+1} + \ldots x_{2^k j}}{2^k} - \tilde{x}\right)^2\right) \tag{50}$$

and, when 2^k is large enough so that different blocks can be considered as statistically independent, the quantity in big brackets can be taken as the correct variance of the average for samples of size 2^k, which is expected to scale as $1/2^k$ independently of the autocorrelation time (which just sets the prefactor), so that multiplying this quantity by $2^k/N$ will give the correct expected variance for averages over samples of size N.

We show two examples of application of the blocking procedure in Fig. 4, both cases refer to the Gaussian distribution sampled via Metropolis with $\delta = 0.1$ as discussed above. On the left hand side, the starting sample size is $N = 10^7$. The logarithmic scale on the vertical axis clearly shows that, in the first steps, the naive error grows exponentially with k: this is clearly understandable from Eq. (49), since in the presence of strong autocorrelation between adjacent blocks, $C(1) \simeq 1$ and the naive variance estimate approximately doubles at each blocking step, i.e. the naive error grows as the square root of the block size. Such exponential growth stops approximately around $k = 10$, after which a stable plateau is reached. Both the block size after which the growth stops, $2^{10} \sim O(10^3)$, and the ratio between the plateau value and naive error at $k = 0$ (slightly larger than 30) are compatible with the autocorrelation time that we have already found, $\tau \sim 600$; however the blocking procedure can be a much more practical method. It should be clear that the particular choice of blocks succession, scaling by a factor 2 at each step, is completely immaterial, and that any other succession would work equally well.

Let us stress that results might not be always so clear: the second example shown on the right hand side of Fig. 4, which is based on a sample of just 10^4 data, shows that in this case the plateau region is barely reached and the correct statistical error is not clearly identifiable; the reason is that the sample size is not so large, and in fact only $O(10)$ times larger than τ.

[9] We are assuming that N is divisible by 2^k. When this is not true, one can discard some entries of the sample (for instance, the last ones) so as to reduce N to the closest multiple of 2^k. Typically, that will not affect the estimate of $\sigma^2_{(k)}$ in a relevant way.

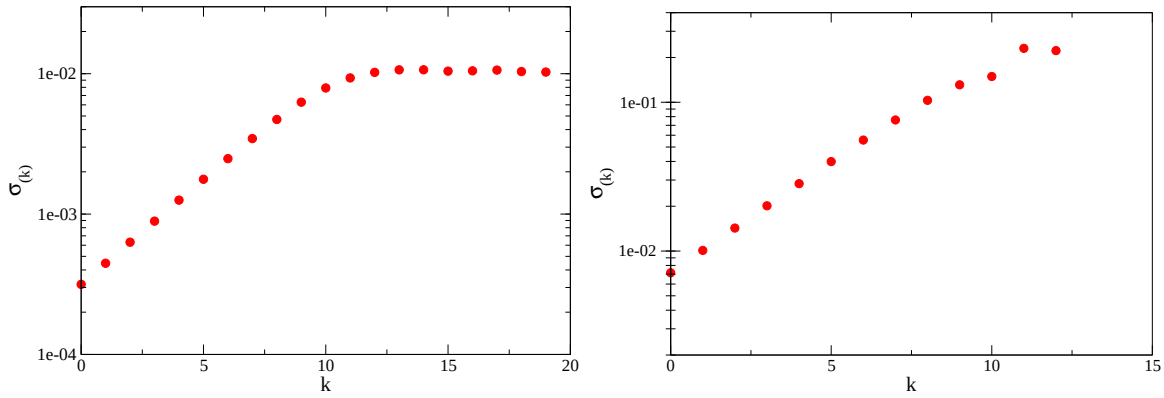

Figure 4: Naive error estimate for the sample average as a function of the blocking iteration and for two different sample sizes. Both samples have been drawn according to the Gaussian distribution with unit variance, via the Metropolis algorithm with $\delta = 0.1$. The sample size is $N = 10^7$ for the left figure, and $N = 10^4$ for the right one.

5.1.2 Markov chain optimization: looking for the highest possible efficiency

Previous discussion has shown that the efficiency of a given Monte-Carlo algorithm, i.e. the numerical effort required to attain a given statistical accuracy, can be strongly dependent on its parameters. For the case we have analyzed, small values of δ, with $\delta \ll \sigma$, are not advisable: the Metropolis test acceptance is very high, however we move very little steps at each Markov chain iteration and, as a consequence, autocorrelation times are large and of the order of $1/\delta^2$ (i.e., according to Eq. (44), statistical errors grow as $1/\delta$ at fixed sample size N).

On the other hand, it is clear that very large values of δ are also not advisable. Indeed, if $\delta \gg \sigma$, only a fraction of tentative moves proportional to σ/δ will have a chance to be accepted during the Metropolis step, so that, even if accepted steps will be very effective in updating the system, the autocorrelation time will be very large and proportional to δ, because of a very low acceptance proportional to $1/\delta$. At fixed N, statistical errors will scale as $\sqrt{\delta}$.

We conclude that some optimal choice of δ must exist, which is expected to be of the order of the variance of the sampled distribution, σ. In order to further clarify the issue, we have produced different samples of equal size $N = 10^6$, hence requiring the same numerical effort, corresponding to different values of δ chosen in a wide range. On the left-hand side of Fig. 5 we report the error on the sample average $\tilde{x}$ (estimated via the blocking procedure described above) as a function of δ. The logarithmic scale permits to better appreciate the behaviour of the error proportional to $1/\delta$ or to $\sqrt{\delta}$ for small or large values of δ, respectively; on the other hand, the minimum error, corresponding to the maximum efficiecency, is reached for $\delta \sim 3 - 4\ \sigma$.

The same data are shown on right-hand side of Fig. 5 as a function of the Metropolis test acceptance and show that the optimal acceptance in this case is around 40%. It is interesting to notice that the error does not change much around the optimal value of δ, so that no fine tuning of the parameter is really needed and values of δ which are reasonably close to the optimal one will work well.

5.2 Error estimate for non-trivial estimators: The Jackknife, and the Bootstrap

We are now going to analyze an issue which is not related to the problem of autocorrelation, even if one has usually to deal with it and with autocorrelations at the same time. We shall make use of the Gaussian distribution (with unit variance and zero average) as a laboratory also in this case; however, in order to clearly decouple the different kinds of problems, we will first start with sets of statistically independent data, produced via the Box-Müller algorithm.

Let us consider a sample of $N = 10^4$ data and the problem of estimating the quantity $\langle x^4 \rangle / (3 \langle x^2 \rangle^2)$ on such sample. The main difference with respect to previous examples is that now we are not interested in a simple average of some function of the stochastic variable over the sample, but in a combination of averages, a ratio in this case. Let us assume for the moment that the corresponding combination of sample

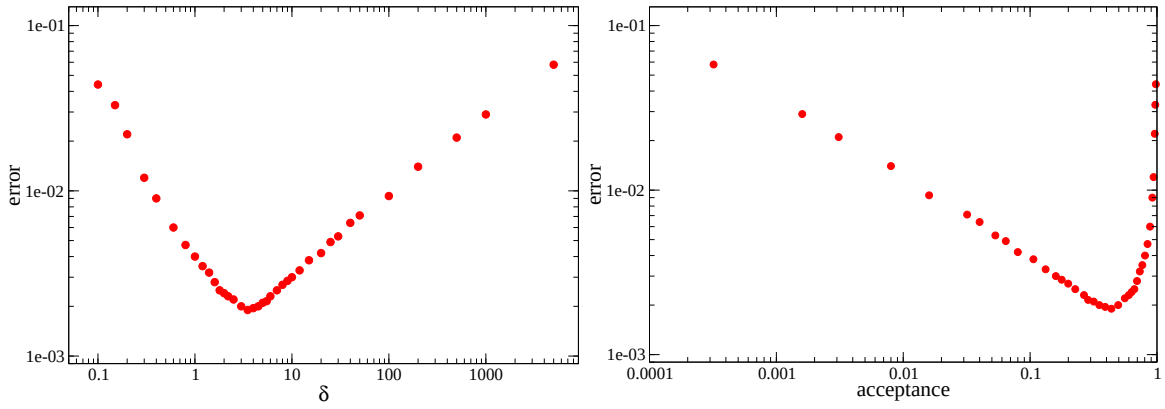

Figure 5: Left: Statistical error as a function of δ for samples of size $N = 10^6$ of the Gaussian distribution with unit variance. Right: the statistical error is reported as a function of the Metropolis acceptance.

averages, i.e.

$$E_N[x_1, x_2, \ldots, x_N] = \frac{\overline{x^4}}{3\,\overline{x^2}^{\,2}}\,, \tag{51}$$

is the correct estimator, in the sense that, if we consider many or an infinite number of draws of the same sample, then $\langle E_N \rangle_s = \langle x^4 \rangle / (3\langle x^2 \rangle^2)$; even if later we will reconsider this hypothesis, by now we just want to understand how to correctly compute the statistical uncertainty σ_{E_N}.

The first naive way one usually tries in this situation is to first estimate the statistical errors on single sample averages, and then compute the ratio and its error by standard error propagation. We have tried to do that on our sample (which is not so useful to report graphically) obtaining

$$\langle x^4 \rangle = 2.96(10) \quad ; \quad \langle x^2 \rangle = 1.000(14)$$

i.e. a relative error 3.4 % on $\langle x^4 \rangle$ and 2.8 % on $\langle x^2 \rangle^2$. When taking the ratio, we are not allowed to sum relative errors in quadrature, as one could do for independent quantities, since both averages have been estimated on the same sample, so we have to take the straight sum of relative errors to finally obtain

$$\frac{\langle x^4 \rangle}{3\langle x^2 \rangle^2} = 0.99(6)\,. \tag{52}$$

Is that a correct estimate? It is largely compatible with the expected value, which for a Gaussian distribution with zero average is 1, hence for sure we have not underestimated the statistical error σ_{E_N}, however we could have overestimated it.

In order to clarify this point, we have repeated the same estimate for 10^3 independent samples of the same size, to see how E_N is really distributed; results are reported on the left-hand side of Fig. 6. It is already visible by eye that 0.06 is an overestimate for σ_{E_N}, and indeed a direct computation yields $\sigma_{E_N} = 0.016$, i.e. almost a factor 4 lower. Such a bad estimate of the statistical error is equivalent to losing a factor 14 in numerical effort, so we should understand where we did go wrong and how to get a correct estimate. The wrong point is of course the use standard error propagation: $\overline{x^4}$ and $\overline{x^2}^{\,2}$ are computed on the same sample and indeed we adopted a conservative standard sum of the relative uncertainties. However, actually, the existing correlation between the two sample averages acts in the opposite direction: if the sample is such that $\overline{x^4}$ has a fluctuation towards higher values, the same will typically happen for $\overline{x^2}$ too, so that fluctuations will partially cancel in the ratio E_N.

How to get the correct estimate for the standard deviation σ_{E_N} is non-trivial. Of course, also in this case we cannot think of repeating the sample extraction several times and measuring the fluctuations of E_N from sample to sample directly. Instead, we have to devise some method which relies just on the single sample we have at our disposal. We will illustrate two different methods, known as the Jackknife and the Bootstrap. The common idea is that of extracting fictitious new samples starting from the original one, and to study how the estimator fluctuates over the new samples: this idea, which is known under the general name of *resampling techniques*, does not require much new numerical effort, since just the originally sampled system states are needed; that might seem a bit suspicious, since it sounds like creating new information from nothing, but it works very well.

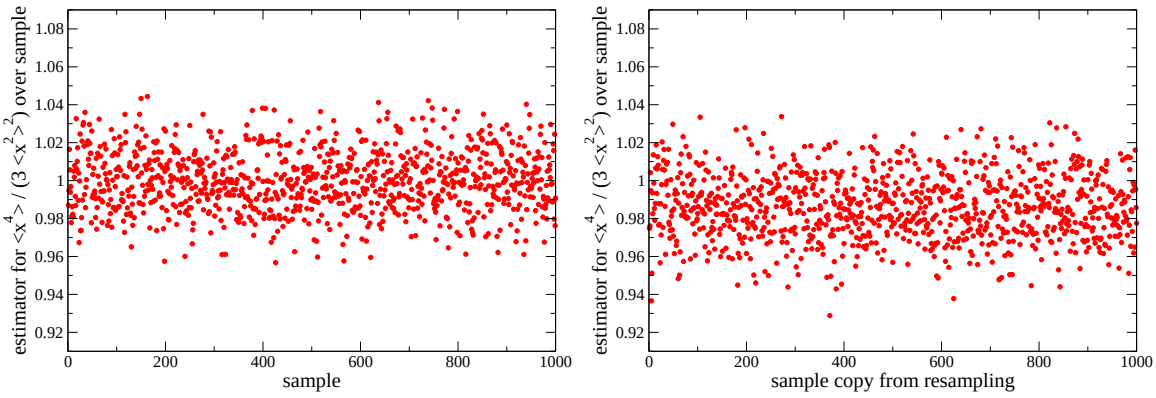

Figure 6: Left: Estimates of $\overline{x^4}/(3\,\overline{x^2}^{\,2})$ over 10^3 independent samples with $N = 10^4$, in order to estimate the true variance of the estimator. Right: bootstrap estimates of the same quantity over 10^3 samples with $N = 10^4$ which have been resampled starting from the first sample of the left figure.

5.2.1 The Bootstrap

Let us consider a generic estimator over the finite sample made up of N independent draws. In our chosen simplified playground, i.e. the Gaussian distribution for one real stochastic variable, this is a generic function of N real variables, $F_N(x_1, x_2, \ldots, , x_N)$, which is usually symmetric for generic permutations of the sample variables and converges, for $N \to \infty$, to some functional $F[p(x)]$ of the probability distribution function ($p(x)$ according to which the sample is drawn. The quantity E_N defined in Eq. (51) is one example, but one could think of any other combination of sample averages (the sample variance, for instance) or also of other less standard functions, like the median over the distribution.

Our problem is that of estimating how F_N fluctuates over different draws of the sample, in order to assign a statistical error to our estimate of F over a single sample. In particular, the statistical error is defined as

$$\sigma_{F_N} = \sqrt{\langle F_N^2 \rangle_s - \langle F_N \rangle_s^2} \tag{53}$$

where by $\langle \cdot \rangle_s$, as usual, we mean the average over the sample probability distribution, $P(x_1, x_2, \ldots x_N)$, which under the assumption of independent draws is defined as

$$P(x_1, x_2, \ldots x_N)\, dx_1 \ldots dx_N = p(x_1)\, p(x_2) \ldots p(x_N)\, dx_1 \ldots dx_N\,. \tag{54}$$

Our difficulties are that, on one hand, for a generic F we do not have a simple estimator of σ_{F_N} itself, as it happens for simple average values, and that, on the other hand, we have to rely on one single sample, i.e. we cannot afford the sampling of the distribution over samples, Eq. (54), directly: each new sample would require a new run of the Markov chain and a numerical effort equal to that needed for the original sample.

The idea of the bootstrap, originally proposed by Efron [**?**], is the following. Since the N elements of the single sample at our disposal are distributed according to $p(x)$, why do not we use them as our *dice box* to produce new *fake* samples? The extraction of the new j-th sample, $x_k^{(j)}$ with $k = 1, \ldots N$, will proceed as follows: we extract N integer random numbers $j_1, j_2, \ldots, j_N$, each uniformly distributed between 1 and N, then $x_k^{(j)} = x_{j_k}$, where x_{j_k} is the j_k-th element of the original sample. The new samples are fake in the sense that they are not genuinely drawn from the original distribution function $p(x)$, but just by a finite sample of it, however they are very cheaply produced, since no new Monte-Carlo generation is needed.

For each new sample, we can define a new estimate of F_N

$$F_N^{(j)} \equiv F_N(x_1^{(j)}, x_2^{(j)}, \ldots x_N^{(j)}) \tag{55}$$

we can then consider M new fake samples, measure the variance of $F_N^{(j)}$ on them and take it as an estimate of σ_{F_N}, i.e. assume that

$$\sigma_{F_N} \simeq \sqrt{\frac{1}{M}\sum_{j=1}^{M}(F_N^{(j)})^2 - \left(\frac{1}{M}\sum_{j=1}^{M} F_N^{(j)}\right)^2}\,. \tag{56}$$

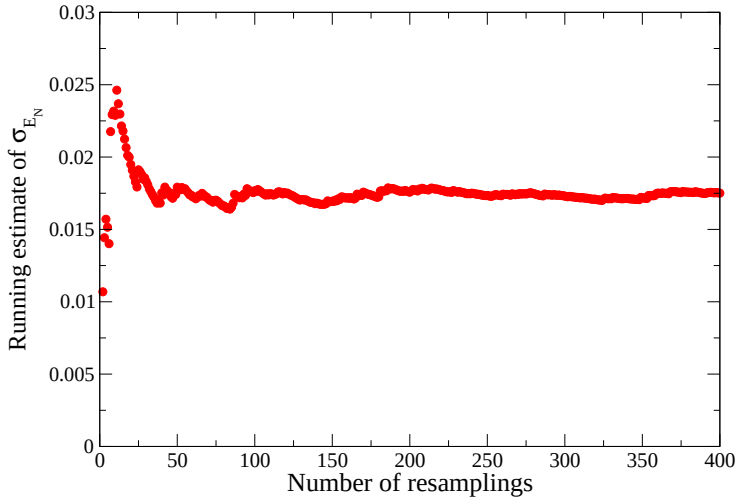

Figure 7: Bootstrap estimate of the statistical uncertainty on the estimator E_N in Eq. (51), as a function of the number M of resamplings appearing in Eq. (56).

It is important to stress that, for each new sample, the random integers j_k are extracted independently, i.e. it is allowed to have multiple repetitions of some elements of the original sample. Therefore the resampled samples are not just a reshuffling of the original sample, which would lead to exactly the same estimates of F_N under the assumption that it is symmetric under permutations.

In order to illustrare how the bootstrap method works, we have considered the original sample with $N = 10^4$, on which the estimate in Eq. (52) was based, and exploited bootstrap resampling to produce 10^3 new samples of the same size[10]. On each of these samples we have evaluated the estimator E_N in Eq. (51), obtaining the results reported on the right-hand side of Fig. 6, where they can be directly compared to results obtained by a genuine draw of independent samples. Two things are worth noticing: *i)* the bootstrap estimates fluctuate around a value which is a bit lower than 1, this is a reflection of the fact that the bootstrap samples are actually drawn from an approximate representation of the original distribution, consisting in a sample for which indeed we have $E_N \simeq 0.99$; *ii)* nevertheless, fluctuations around the mean value are very similar in size to those obtained for genuine independent samples, so that we expect to obtain a reasonable estimate of σ_{E_N}, at least to the extent of accuracy needed for a statistical error.

In Fig. 7 we report bootstrap estimates of σ_{E_N}, obtained according to Eq. (56), as a function of the number of resamplings M. One can see that just a few tens of resamplings are needed to converge to an estimate $\sigma_{E_N} \sim 0.017$, which is very close to the value $\sigma_{E_N} \simeq 0.016$ we have found before.

What we have found, in particular the number of resamplings which are needed to get a reasonable estimate of the statistical error, is quite general and independent of the particular problem and sample size under investigation. The interested reader can found more details about the bootstrap method in plenty of existing literature (see, e.g., Refs. [**?**, **?**, **?**, **?**, **?**]). Maybe it is worth concluding by discussing the origin of the name of the method: this is related to the famous sentence *pull oneself up by one's bootstraps*, which clearly summarizes the kind of impossible task the method seems to achieve. Of course, there is no magic at all and, as we have already stressed above, the central point is the assumption that the real sample one has at disposal be a good and faithful empirical realization of the full statistical population, i.e. of the theoretical distribution p. The assumption is better and better as the real sample size increases. For small real sample size issues, the reader is referred to the specialistic literature on the topic [**?**, **?**, **?**, **?**, **?**].

5.2.2 The Jackknife

Let us start in this case by the origin of the name, which is related to the famous homonymous tool and is due to the many practical purposes this method can easily adapt to. It was originally introduced by Tukey [??] as a tool to correct the presence of a bias in some estimators (a problem that we will discuss later in

[10]In principle, the size of the resampled samples could be different from that of the original one. Indeed, to the extent the original sample is considered as a good representation of the theoretical reference distribution, one could make use of it to produce samples of arbitrary size N' and study in this way the predicted fluctuations for the estimator $F_{N'}$.

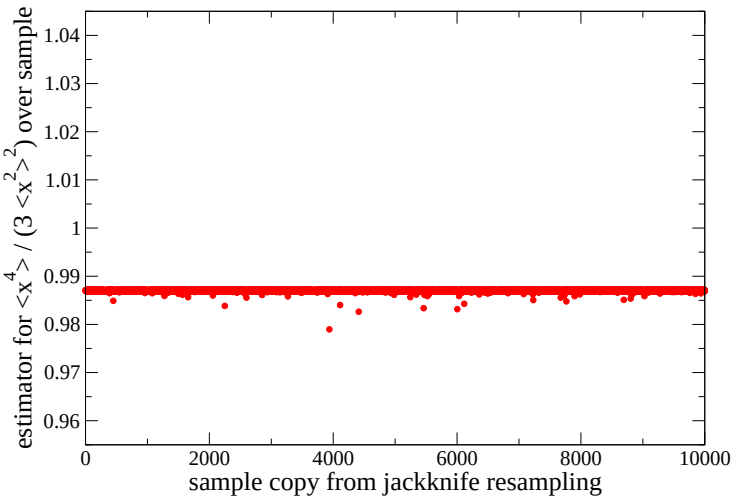

Figure 8: Estimates of $\overline{x^4}/(3\,\overline{x^2}^{\,2})$ over all jackknife subsamples obtained from a sample with $N = 10^4$.

this Chapter), then it was realized it could also be a practical way to obtain other useful information, e.g. on the variance of a given estimator.

Analogously to the bootstrap, also the jackknife is based on the production of several new samples starting from the original one, however that happens now in a much more systematic way. In particular, one creates N new samples $y_k^{(j)}$ of size $N-1$, i.e. $j = 1, N$ and $k = 1, N-1$, by removing in each case one of the elements of the original sample:

$$\begin{aligned} y_k^{(j)} &= x_k \quad \text{for } k < j \\ y_k^{(j)} &= x_{k+1} \quad \text{for } k \geq j\,. \end{aligned} \tag{57}$$

On each new sample one makes an evaluation of the F_{N-1} estimator:

$$F_J^{(j)} \equiv F_{N-1}(y_1^{(j)}, y_2^{(j)}, \ldots\, y_{N-1}^{(j)}) \tag{58}$$

and then compute the variance of such estimates

$$\sigma^J_{F_{N-1}} \equiv \sqrt{\frac{1}{N}\sum_{j=1}^{N}(F_J^{(j)})^2 - \left(\frac{1}{N}\sum_{j=1}^{N} F_J^{(j)}\right)^2}\,. \tag{59}$$

How can that be related to the statistical uncertainty on F_N we are interested in? The relation in this case is more involved that in the bootstrap case. In Fig. 8 we report the jackknife estimates $E_J^{(j)}$ for the estimator in Eq. (51), starting from the same starting sample with $N = 10^4$ already used as a playground for the bootstrap. It is well visible that fluctuations of F_{N-1} over the jackknife subsamples are strongly reduced, indeed we obtain $\sigma^J_{E_{N-1}} \simeq 1.7 \times 10^{-4}$; the reason is also well clear: the subsamples are very similar to each other, each pair differing only for the substitution of one single element. We will now give an euristic argument to establish a relation between σ_{F_N} and $\sigma^J_{F_{N-1}}$, thus obtaining the jackknife formula to evaluate the statistical uncertainty.

The information contained in σ_{F_N} is how F_N fluctuates as a consequence of the sample distribution explicited in Eq. (54), i.e. for independent fluctuations of each sample element x_k according to $p(x)$. As we have already said, the jackknife subsamples differ from each other for the substitution of just one element, therefore, also due to the symmetry of F under permutations of its arguments, we can consider $\sigma^J_{F_{N-1}}$ as the estimate of the fluctuations induced on F_{N-1} due the fluctuation of just one of the sample elements, keeping the others fixed. Actually, we should consider the usual bias correction for the variance estimate, i.e. take $\sqrt{N/(N-1)}\,\sigma^J_{F_{N-1}}$ in its place.

It is quite reasonable to assume that the fluctuations of F_{N-1} when all variables fluctuate independently be the *incoherent* sum of those due to single variable fluctuations, i.e. that fluctuations sum in quadrature. The reasoning is in some sense similar to that leading to the formula for error propagation in the presence

of statistically independent variables, i.e.

$$\delta F_{N-1} = \sqrt{\sum_{k=1}^{N-1} \left(\frac{\partial F_{N-1}}{\partial y_k} \delta y_k \right)^2} .$$

Based on this assumption, and given the symmetry of F_{N-1} under permutation of its arguments, we can state that

$$\sigma_{F_{N-1}} = \sqrt{N-1} \, \sqrt{\frac{N}{N-1}} \, \sigma^J_{F_{N-1}} .$$

The final relation to σ_{F_N} is obtained taking for granted that the statistical error scales as $1/\sqrt{N}$, so that

$$\sigma_{F_N} = \sqrt{\frac{N-1}{N}} \, \sigma_{F_{N-1}} = \sqrt{N-1} \, \sigma^J_{F_{N-1}} . \tag{60}$$

Applying the formula above to our particular example, we obtain $\sigma_{E_N} \simeq 10^2 \, \sigma^J_{E_{N-1}} \simeq 0.017$, in very good agreement with the estimates obtained before.

Also for the jackknife method we refer to the specialized literature for more details [**?**, **?**, **?**]. In most cases the bootstrap and the jackknife will work equally well. In general the bootstrap is preferable when the starting sample size, N, is large, since usually a few tens of resamplings will suffice for the bootstrap, while the jackknife will require the estimate on N subsamples anyway. In addition, there are a few estimators for which the euristic argument given above is wrong and the jackknife does not work, while the bootstrap is still valid: a relevant counterexample is given by the median estimator.

5.2.3 Extension to the case of correlated data sets

We have illustrated the bootstrap and the jackknife resampling techniques for the case of statistically independent data sets. However, in most cases one has to apply these techniques to data produced by Markov chain Monte-Carlo: should we modify resampling to take into account autocorrelations, and how? In order to answer both questions, we have considered again the Metropolis algorithm with $\delta = 0.1$, and produced a sample of $N = 10^7$ draws according to the Gaussian distribution with zero average and unit variance. When applied to this sample in the same way described above, both the jackknife and the bootstrap return an estimate

$$\frac{\langle x^4 \rangle}{3 \langle x^2 \rangle^2} = 1.0073 \pm 0.0005$$

which is around 14 standard deviations off the expected result. Therefore, we conclude that both techniques go wrong in estimating the fluctuations of E_N in this case: it is clear that the reason must be the presence of autocorrelations among data, however it is interesting to discuss where exactly to two procedures fail, and how to modify them.

In the case of bootstrap, the hypothesis that the original sample might be a good representation of the full distribution function is completely unaffected by the presence of autocorrelations among data. However, the procedure followed for resampling from this distribution should be reconsidered and matched to its purpose, which is that of reproducing the fluctuations that would be obtained for the estimator F_N if the original sample were drawn several independent times. Indeed, the standard bootstrap resampling, in which each new element is drawn randomly from the original set, reproduces the sampling of statistically independent data, so that the error estimate we get is wrong and actually the same we would obtain if F_N were measured on data without correlation.

We must modify the resampling procedures so that the new artificial samples contain the same degree of correlation of the original data set. The recipe to do that is simple and intuitive: instead of selecting each element randomly, we have to select random blocks of consecutive draws of the original sample, so that correlations up to the chosen block size present in the original data sample will be inherited by the artifical samples. A study of the error estimate as a function of the block size will clarify what is true error.

One can devise various different recipes to implement this idea. The original data sample could be divided in N/M blocks, of size M each, and then one could perform N/M random draws (with repetitions) of such blocks; or one could instead perform N/M random draws of single elements of the original data sample, and take each time the first M consecutive elements starting from the selected one. The data that we are going to illustrate have been obtained following the second recipe, however it should be clear that

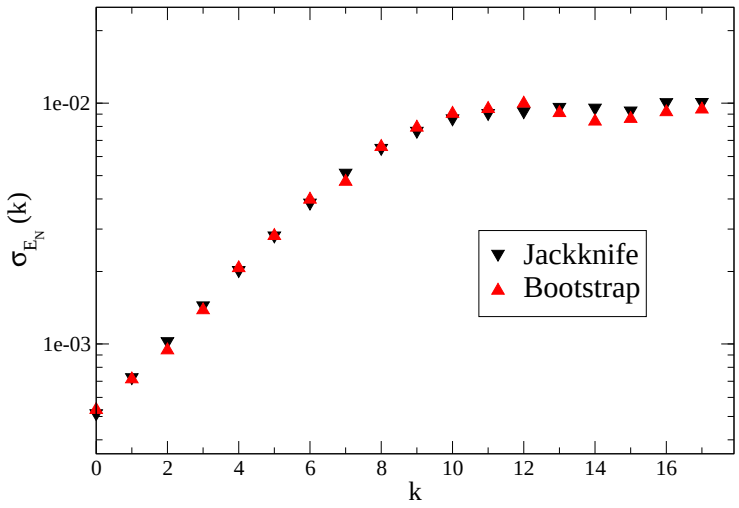

Figure 9: Bootstrap and jackknife estimates for the error on the estimator E_N in Eq. (51), as a function of the blocksize 2^k. The original data set ($N = 10^7$) has been drawn according to the Gaussian distribution with unit variance and zero average, using the Metropolis algorithm with $\delta = 0.1$.

the choice is immaterial, since the purpose is just that the new samples have built-in autocorrelations.

In the case of the jackknife, the error estimate goes wrong in a similar way. The quantity $\sigma^J_{F_{N-1}}$ defined in Eq. (59), i.e. how the estimator fluctuates when changing a single data entry, will give similar results (at fixed N) independently of the degree of autocorrelation existing between data. However, the assumption used to obtain the final estimate, Eq. (60), was that all entries in the data set fluctuate independently, so that single fluctuations giving rise to the global fluctuation of F_{N-1} will sum up in quadrature (i.e. incoherently), giving rise to a factor $\sqrt{N-1}$. This assumption is not true when data are correlated: fluctuations will sum coherently over lengths of the order of the autocorrelation time, and incoherently over larger distances; as a consequence, the global fluctuation of F_{N-1} will be larger.

Also in this case, the solution is to divide the data set in N/M blocks of size M each, then applying a modified jackknife recipe in which N/M new samples are created, and in each of them an entire block is removed. Apart from that, everything proceeds in the same way, substituting N with N/M where appropriate. If the original data are not correlated, the final error estimate will remain unaltered: indeed, the fluctuations of the jackknife estimates, σ^J, will increase by roughly a factor $\sqrt{M}$ with respect to the standard jackknife, since M statistically independent data are removed at the same time; however this factor is recovered when using N/M in place of N in Eq. (60). Instead, when data are correlated over a length much larger than M, σ^J will increase by a factor M with respect to the standard case (because of the almost complete correlation existing between the removed data), so that the final error estimate will increase by roughly a factor $\sqrt{M}$. Hence, also in this case one has to repeat the modified jackknife for several, increasing block sizes, and look for a plateau of the corresponding error estimate as a function of M.

In Fig. 9, we show results obtained for the error on $\langle x^4\rangle/(3\langle x^2\rangle^2)$, adopting the modified versions of the bootstrap and of the jackknife described above, as a function of the blocksize 2^k (in logarithmic scale). One can appreciate a very good agreement between the two procedures for each value of k, hence also on the final estimate, so that we can finally quote $\langle x^4\rangle/(3\langle x^2\rangle^2) = 1.007 \pm 0.010$, which is in good agreement with the theoretical expectation[11]

[11]The reader might notice that the ratio between the naive error, 0.0005, and the real one, 0.010, points to an autocorrelation time of the order of 200, which is a bit lower than the one found before, $\tau \simeq 600$. However, in that case the relevant autocorrelation time was that of the stochastic variable x itself, while for the quantity $\langle x^4\rangle/(3\langle x^2\rangle^2)$ the relevant autocorrelation times are those of x^2 and x^4, which are shorter.

5.3 Biased Estimators

References

[1] Histoire de l'Acad. Roy. des. Sciences (1733), 43-45

[2] T.E. Hull and A.R. Dobell, SIAM Review Vol. 4, no. 3, (1962) 230-254

[3] D.H. Lehmer, *"Mathematical methods in large-scale computing units"* Annals of the Computation Laboratory of Harvard University, Vol. 26 (1951)

[4] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller and E. Teller, "Equation of state calculations by fast computing machines," J. Chem. Phys. **21** (1953) 1087. doi:10.1063/1.1699114

Exercise 1

Devise a Metropolis algorithm to uniformly cover the circle of unitary radius. Do it in cartesian coordinates x, y, first, and then in polar coordinates r, ϕ.

Exercise 2

Devise a Metropolis algorithm to uniformly cover the sphere of unitary radius. Do it in cartesian coordinates x, y, z, first, and then in spherical coordinates r, θ, ϕ.