Segnali con noise

Sia f_0 [m], m=0, 1, ..., N-1, il campionamento del segnale in arrivo sul rivelatore; il segnale campionato in uscita f[m] differisce da f_0 [m] per quantità che variano in modo casuale. Si hanno due tipi di noise:

- **noise di lettura** o noise elettronico (read-out noise); tale noise è addittivo e la relazione tra segnale con e senza noise è data da:

$$f[m] = f_0[m] + w[m] \quad ;$$

- **noise di conteggio** ; f[m] è un numero intero che è la realizzazione di una variabile aleatoria con valor medio (valore atteso) f_0 [m].

Noise di lettura

Per ogni m, w[m] è la realizzazione di una variabile aleatoria con le seguenti proprietà:

- per m diverso da m', w[m] e w[m'] sono realizzazioni di variabili aleatorie **indipendenti**;

- w[m] è la realizzazione di una variabile aleatoria Gaussiana (o normale) con media 0 e varianza σ; quindi la sua densità di probabilità è data da:

$$p(x) = \frac{1}{\sqrt{2p}\,s} e^{-\frac{x^2}{2s^2}} \quad , \quad x = w[m] \; .$$

Noise di conteggio

Per ogni m, f[m] è la realizzazione di una variabile aleatoria con le seguenti proprietà:

- per m diverso da m', f[m] e f[m'] sono realizzazioni di variabili aleatorie **indipendenti**;

- f[m] è un numero intero, realizzazione di una variabile aleatoria poissoniana con valor medio f_0 [m].

Distribuzione di Poisson

$$p(n) = \frac{e^{-l}\, l^{n}}{n!} \; ; n = 0,1,.....; \; \sum_{n=0}^{+\infty} p(n) = 1$$

$$\sum_{n=0}^{+\infty} n\; p(n) \;=\; l \quad ; \; \sum_{n=0}^{+\infty} (n - l\,)^2 \; p(n) \;=\; l \quad .$$

Generazione di numeri casuali

La generazione di numeri casuali è utile sia per le simulazioni di processi fisici sia per alcune applicazioni del calcolo numerico.

Esempi di processi fisici intrinsecamente aleatori sono: decadimenti radioattivi, rumore, sistemi turbolenti (per esempio il cesto delle palline del lotto).

Metodi numerici che utilizzano tali tecniche sono, per esempio, le simulazioni di tipo Montecarlo.

Noi utilizzeremo tali tecniche per generare rumore gaussiano e poissoniano.

Occorre, quindi un metodo per generare variabili aleatorie che segnuano distribuzioni di probabilità note.

Per fare ciò servono sequenze di numeri casuali.

Generazione di sequenze pseudo-casuali

Generare numeri veramente casuali al calcolatore è molto difficile, allora ci si accontenta di generare sequenze pseudo-casuali, nel senso che i numeri della sequenza :

- sono numeri interi
- sono distribuiti uniformemente in un dato intervallo
- hanno un periodo di ripetizione molto lungo
- hanno un basso livello di correlazione (tra un elemento della sequenza e il successivo)

Generatore in ambiente C

Il linguaggio C fornisce un generatore di numeri casuali (pseudo-casuali) uniformemente distribuiti nell'intervallo [0,RAND_MAX].

Il valore di RAND_MAX , che nelle macchine a 32 bit vale solitamente $2^{32}-1$, è un valore costante predefinito nelle librerie di sistema (verificare il contenuto di stdlib.h) e va usato come una costante di tipo long int.

La funzione che genera numeri casuali si chiama

long int random()

Quindi un programma che utilizzi la generazione di numeri causuali deve includere **stdlib.h** e ad ogni chiamata di **random**, avremo un numero della sequenza.

Seme della sequenza

Esiste una funzione che permette di posizionarsi all'interno della sequenza in un punto "a caso" definendo il seme della sequenza.

Questa funzione è:

void srandom(long int)

L'argomento della funzione deve essere un numero intero grande ed è chiamato il seme della sequenza.

Seme uguali portano a sequenze uguali

Un modo di rendere la sequenza diversa ad ogni esecuzione di un programma è quello di mettere come seme il tempo macchina.

Esempio

Vediamo un esempio di generazione di 10000 numeri pseudo-casuali, distribuiti uniformemente tra [0,RAND_MAX]

```
#include <stdlib.h>

main () {
int n=10000,i;
long int x[10000];

for (i=0;i<n;i++)
x[i]=random();
}
```

```
#include <stdlib.h>
#include <time.h>

main () {
int n=10000,i;
long int x[10000];
srandom(time());
for (i=0;i<n;i++)
x[i]=random();
}
```

Esempio II

Vediamo un esempio di generazione di 10000 numeri pseudo-casuali, distribuiti uniformemente tra [0,1] (le variazioni rispetto al programma precedente sono evidenziate in blu).

```
#include <stdlib.h>
#include <time.h>

main () {
int n=10000,i;
double x[10000];
srandom(time());
for (i=0;i<n;i++)
x[i]=(double) random()/RAND_MAX;
}
```

Esempio III

.... oppure distribuiti in un intervallo generico [xmin,xmax]

```
#include <stdlib.h>
#include <time.h>

main () {
int n=10000,i;
double x[10000],xmin,xmax;
srandom(time());
.......
for (i=0;i<n;i++)
x[i]=(xmax-xmin)*((double) random()/RAND_MAX)-xmin;
}
```

Distribuzioni di probabilità arbitrarie

Abbiamo visto come generare eventi con distribuzione di probabilità uniforme in un intervallo generico [xmin,xmax]; la maggior parte delle simulazioni necessitano, tuttavia, di eventi con distribuzioni di probabilità non uniformi.

E' quindi importante imparare a generare eventi distribuiti secondo una generica distribuzione di probabilità a partire da eventi con densità di probabilità uniforme.

Ci sono differenti metodi che sono più o meno efficienti a seconda del tipo di distribuzione di probabilità da generare:

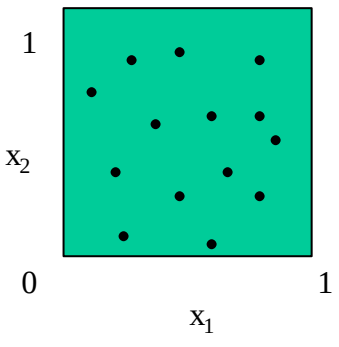
- Metodi di trasformazione
- Metodi di reiezione

Metodi di trasformazione: distribuzione gaussiana

Per i dettagli matematici del metodo si veda : Numerical Recipes in C par. 7.2 (il link al libro è nella pagina web del corso).

Questo metodo si basa sul generare una coppia di numeri, ognuno con distribuzione uniforme in [0,1].

Possiamo vedere questa coppia x_1,x_2 come le coordinate di un punto nello spazio cartesiano con distribuzione uniforme nel quadrato di lato 1.

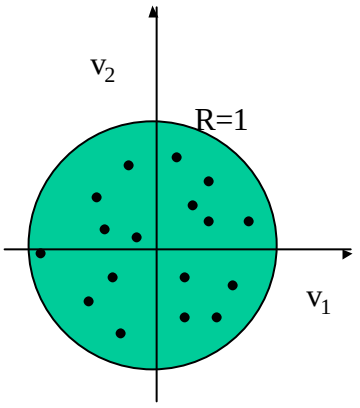


Si dimostra che le seguenti trasformazioni definiscono altre due variabili con distribuzione gaussiana con media 0 e varianza 1:

$$y1 = \sqrt{-2\ln x_1}\cos(2\pi x_2)$$
$$y2 = \sqrt{-2\ln x_1}\sin(2\pi x_2)$$

Per evitare di utilizzare le funzioni trigonometriche, si generano, in effetti due variabili v_1, v_2 corrispondenti alle coordinate di punti interni al cerchio di raggio 1 centrato sull'origine, si ha:

$$R^2 = v_1^2 + v_2^2$$
$$\sin(2\pi x_2) = v_1 / R$$
$$\cos(2\pi x_2) = v_2 / R$$



Funzione per la generazione di numeri con distribuzione di probabilità gaussiana

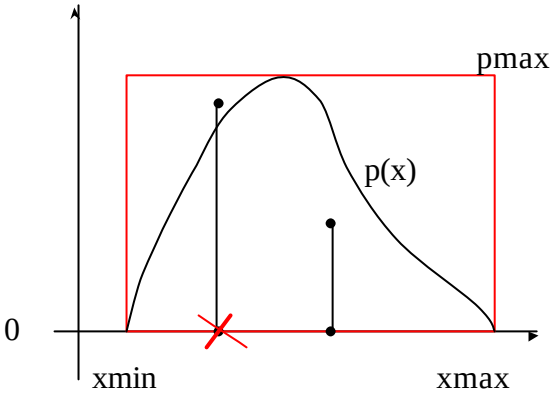
```
#include <math.h>

float gausdev ()
{
float v1,v2,r2,fac;
do {
v1=2.0*(float)random()/RAND_MAX-1.0;
v2=2.0*(float)random()/RAND_MAX-1.0;
r2=v1*v1+v2*v2;
} while (r2>=1.0|| r2 == 0.0);
fac=sqrt(-2.0*log(r2)/r2);
return v2*fac;
}
```

cerca fino a che il punto non sta nel cerchio di raggio 1

Metodo di reiezione

Supponiamo che la distribuzione di probabilità da generare, p(x), sia definita in un intervallo [xmin,xmax] e che in tale intervallo sia compresa tra 0 e pmax.

Si genera prima x uniformemente tra xmin e xmax, poi si genera un secondo numero y tra 0 e pmax, se y<p(x) accettiamo il valore x, altrimenti lo rigettiamo.



Funzione per la generazione di numeri con distribuzione poissoniana

Per i dettagli matematici del metodo si veda : Numerical Recipes in C par. 7.3 (il link al libro è nella pagina web del corso).

Non si riporta qui il codice della funzione, perchè è piuttosto lungo e i dettagli esulano dallo scopo della lezione.(Trovate il codice alla pagina del corso.)

La funzione è:

float poidev(float xm)

e restituisce un numero intero (come floating-point) che è una realizzazione di una variabile aleatoria con distribuzione di probabilità poissoniana con media xm.

Noise gaussiano

Ora che abbiamo imparato a generare dei numeri casuali con una distribuzione di probabilità assegnata, usiamo questo strumento per “sporcare” un segnale con rumore gaussiano. Questa operazione simula il comportamento dei circuiti elettrici.

Il noise gaussiano è addittivo e quindi basta aggiungere ad ogni valore del segnale il risultato della chiamata alla funzione gausdev opportunamente scalato.

Infatti possiamo cambiare la deviazione standard del rumore moltiplicando il risultato della funzione per la deviazione standard voluta; per esempio, se data[i] è il valore corrente del segnale, la sua versione rumorosa sarà data da:

```
data_noisy[i]= data[i]+sigma*gausdev();
```

Noise possoniano

Per applicare il noise poissoniano, la procedura è fondamentalmente diversa. Infatti, abbiamo detto che il rumore poissoniano è intrinseco di eventi di tipo conteggio, e la sua varianza è uguale alla media.

Quindi per ottenere un rumore più o meno importante, si deve scalare il segnale prima di applicare il rumore.

Infatti il rumore poissoniano non è addittivo, ma se data[i] è il valore corrente del segnale la sua versione rumorosa sarà data da:

```
data_noisy[i]=poidev(data[i]);
```

Esercitazione

Scrivere un programma che generi N numeri casuali

1. con distribuzione uniforme tra xmin e xmax
2. con distribuzione gaussiana media 0 e varianza 1
3. con distribuzione poissoniana con media 10,100,1000

calcoli, in ogni caso il valore medio, la varianza e la deviazione standard.

$$\overline{X} = \frac{1}{N}\sum_{i=0}^{N-1} x[i] \qquad \text{media}$$

$$s^2 = \frac{1}{N-1}\sum_{i=0}^{N-1}\left(x[i]-\overline{X}\right)^2 \qquad \text{varianza}$$

$$s = \sqrt{s^2} \qquad \text{deviazione standard}$$

Esercitazione seconda parte

- Si campioni una sinusoide (seno o coseno) sul suo periodo, con un numero di punti a scelta, e si visualizzi il segnale e la sua versione rumorosa con un rumore gaussiano con deviazione standard pari al 1% del valore massimo del segnale.
- Si consideri il seguente segnale nell'intervallo [-T,T]

$$f(t) = 1 + 2\sqrt{(T^2 - t^2)}$$

Lo si campioni con un numero di punti a scelta.

Si calcoli il fattore di moltiplicazione in modo tale che il rumore poissoniano nel punto di massimo sia il 2%.

Si visualizzi il segnale prima e dopo l'applicazione del noise poissoniano.