Programmation en langage C

É. Carry

2 mars 2015

1 Nombre aléatoire

Les trois méthodes qui suivent génèrent des nombres aléatoires de répartition uniforme sur l'intervalle [0, 1].

1.1 `urandom`

Sous les systèmes Unix, il existe un périphérique permettant de générer des nombres aléatoires `/dev/urandom`. Ces nombres sont des entiers compris entre 0 et 255 (1 octet). Dans un terminal, testez la commande :

```
login@machine$ cat /dev/urandom
```

Pour stopper cette commande tapez C-c.
Pour générer un fichier d'octets aléatoires utilisez la commande `dd` (disk to disk), le fichier d'entrée étant `/dev/urandom` et le fichier de sortie `alea.dat` (dans votre répertoire de travail). Pour la syntaxe de la commande `dd` tapez dans un terminal : `man dd`.
Compilez et exécutez le programme `alea1.c` qui crée un fichier `alea1.dat` contenant 200000 nombres réels aléatoires à partir du fichier `alea.dat` .
Modifiez ce programme de telle façon qu'il réalise la même fonction mais directement à partir du fichier de périphérique `/dev/urandom`.

1.2 Rand

La fonction `rand()` (bibliothèque `stdlib`) permet de générer des entiers aléatoires compris entre 0 et `RAND_MAX`.
En vous inspirant du programme précédent, écrivez un programme `alea2.c` qui permet à l'aide de la fonction `rand` de générer un fichier `alea2.dat` de 200000 nombres réels aléatoires compris entre 0 et 1.

1.3 gsl

En utilisant la fonction `gsl_rng_uniform (r)` et en vous inspirant du programme `alea2.c`, réécrivez le programme précédent pour créer un fichier de 200000 valeurs aléatoires que vous appellerez `alea3.dat`.

2 Histogramme

2.1 Graphe

Téléchargez et compilez le programme `histo1.c`. Pour tester ce programme vous devez taper la commande suivante :

```
cat alea1.dat | ./histo1 0 1 100 > histo1.dat
```

`histo1.dat` est un fichier créé par le programme `histo1`, contenant la fréquence pour chaque classe de l'histogramme. Dans cette ligne de commande on utilise 100 classes égales sur l'intervalle [0,1].
Pour visualiser cet histogramme, ouvrez **octave** et tapez les commandes suivantes :

```
load "histo1.dat"
bar(histo1(:,3))
print ("histo1.png","-dpng")
```

2.2 Caractéristiques

Pour une répartition uniforme sur l'intervalle [0,1] la fréquence moyenne est égale à N/M où N est le nombre de valeur et M le nombre de classe, la déviation standard et l'écart maximum sont nuls. Les caractéristiques d' un histogramme peuvent être obtenues par les fonctions suivantes :

```
double gsl_histogram_max_val (const gsl_histogram * h) //fréquence maximale
double gsl_histogram_min_val (const gsl_histogram * h) //fréquence minimale
double gsl_histogram_mean (const gsl_histogram * h)    //valeur moyenne
double gsl_histogram_sigma (const gsl_histogram * h)   //déviation standard
```

Modifiez le programme `histo1.c` en utilisant les fonctions précédentes pour afficher la valeur moyenne, la déviation maximale et la déviation standard.
Appliquez ce programme aux fichiers `alea1.dat`, `alea2.dat` et `alea3.dat`. Conclusion ?

3 distribution

3.1 distribution normale

La loi normale ou gaussienne a la densité de probabilité suivante :

$$p(x)dx = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-x^2/2\sigma^2)dx$$

La commande "gsl-randist *mean N type sigma*" permet de générer N valeurs aléatoires suivant une distribution *type* de valeur moyenne *mean* et d'écart type *sigma*.
exemple : gsl-randist 0 100000 gaussian 30—gsl-histogram -100 100 100¿histogram.dat
cette commande génère 100000 valeurs aléatoires de répartition gaussienne .
Modifiez le programme précédent pour que celui-ci affiche la valeur moyenne , l'écart type, la médiane et les quartiles de la distribution calculés à l'aide des fonctions suivantes :

```
double gsl_stats_variance (const double data[], size_t stride, size_t n)
double gsl_stats_variance (const double data[], size_t stride, size_t n)
void gsl_sort (double * data, size_t stride, size_t n)
double gsl_stats_median_from_sorted_data (const double sorted_data[],
size_t stride, size_t n)
double gsl_stats_quantile_from_sorted_data (const double sorted_data[],
size_t stride, size_t n, double f)
```

3.2 Distribution de Laplace

Répétez les opérations du paragraphe précédent pour une distribution de Laplace.

$$p(x)dx = \frac{1}{2a}\exp(-|x/a|)dx$$

avec $a = 0,5$
Vous devez utiliser l'instruction :

```
double gsl_ran_laplace (const gsl_rng * r, double a)
```