Simulation d'une loi gaussienne

Ruidong PAN & Hengshuo LI

Introduction

Tous les langages de programmation possèdent un générateur de nombres pseudo-aleatoires qui suit la loi uniforme sur l'interval $[0, 1]$. Le but de ce projet est de simuler une loi gaussienne en utilisant deux variables aléatoires uniformes sur $[0, 1]$ independantes pour générer de nombres pseudo-aleatoires qui suit la loi $N(0; 1)$.

Partie théorique

Soient U et V deux variables aléatoires uniformes sur $[0, 1]$ independantes:

$$f_U(u) = \begin{cases} 1 & u \in [0,1] \\ 0 & sinon \end{cases} \qquad f_V(v) = \begin{cases} 1 & v \in [0,1] \\ 0 & sinon \end{cases}$$

alors $X = \sqrt{-2log(u)}cos(2\pi V)$ et $Y = \sqrt{-2log(u)}sin(2\pi V)$ sont variables aléatoires indépendantes et de la loi $N(0; 1)$.

Preuve

Soient X et Y deux variables aléatoires indépendantes avec $X \sim N(0; 1)$ et $Y \sim N(0; 1)$, alors on a

$$p(X, Y) \overbrace{=}^{indépendantes} p(X) \times p(Y) = \frac{1}{\sqrt{2\pi}} e^{-\frac{X^2}{2}} \times \frac{1}{\sqrt{2\pi}} e^{-\frac{Y^2}{2}} = \frac{1}{2\pi} e^{-\frac{X^2+Y^2}{2}}$$

En Transformant par coordonnées polaires avec $X = Rcos\theta$ et $Y = Rsin\theta$ où $\theta \in [0, 2\pi]$, on obtient :

$$\frac{1}{2\pi}e^{-\frac{X^2+Y^2}{2}} = \frac{1}{2\pi}e^{-\frac{R^2}{2}}$$

Or on sait que : $\iint_D f(x,y)dxdy = \int_\beta^\alpha d\theta \int_{\rho_1(\theta)}^{\rho_2(\theta)} f(\rho cos\theta, \rho sin\theta)\rho d\rho$

Alors on a :

$$\int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{1}{2\pi}e^{-\frac{X^2+Y^2}{2}} dXdY = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \frac{1}{2\pi}e^{-\frac{R^2}{2}} Rd\theta dR = 1$$

On a donc aussi la fonction de répartition pour R et θ :

$$F_R(R \leqslant r) = \int_0^r \int_0^{2\pi} \frac{1}{2\pi}e^{-\frac{R^2}{2}} Rd\theta dR$$

$$= \int_0^r \left(\int_0^{2\pi} \frac{1}{2\pi} Re^{-\frac{R^2}{2}} d\theta \right) dR$$

$$= \int_0^r Re^{-\frac{R^2}{2}} dR$$

$$= -\left[e^{-\frac{R^2}{2}}\right]_0^r = 1 - e^{-\frac{r^2}{2}}$$

$$F_\theta(\theta \leqslant \phi) = \int_0^\phi \int_0^{\infty} \frac{1}{2\pi}e^{-\frac{R^2}{2}} Rd\theta dR$$

$$= \int_0^\phi \left(\int_0^{\infty} \frac{1}{2\pi} Re^{-\frac{R^2}{2}} dR \right) d\theta$$

$$= \int_0^\phi \frac{1}{2\pi} \left[-e^{-\frac{R^2}{2}}\right]_0^{\infty} d\theta$$

$$= \int_0^\phi \frac{1}{2\pi} d\theta = \frac{\phi}{2\pi}$$

Soient U_1 et U_2 deux variables aléatoires independantes uniformes sur $[0,1]$, on peut prendre $\theta = 2\pi U_2$, car $F_\theta(\theta \leqslant \phi) = \frac{\phi}{2\pi}$, θ est uniformes sur $[0, 2\pi]$.

$F_R(R \leqslant r) = 1 - e^{-\frac{r^2}{2}}$ donc on peut obtenir $R = F_R^{-1}(z) = \sqrt{-2ln(1-z)}$, d'après la méthode de la transformée inverse, la variable aléatoire $R = F_R^{-1}(z)$ a pour fonction de répartition $F_R(R \leqslant r)$, où z est une variable aléatoire de loi uniforme sur $[0; 1]$. Or si z est uniforme, alors $(1 - z)$ est aussi uniforme. Pour

notre cas, z est uniforme sur $[0,1]$, donc $(1-z)$ est aussi uniforme sur $[0,1]$. Et on peut prendre $U_1 = (1-z)$.

Enfin, on peut obtenir $X = Rcos\theta = \sqrt{-2lnU_1}cos(2\pi U_2)$ et $Y = Rsin\theta = \sqrt{-2lnU_1}sin(2\pi U_2)$ où X et Y sont deux variables aléatoires indépendantes de la loi $N(0;1)$.

Donc on peut conclure que on n'a besoin que de deux variables aléatoires independantes U_1 et U_2 qui suivent la loi uniforme sur $[0,1]$, alors on peut obtenir une variable aléatoire de la loi $N(0;1)$ soit par la formule $\sqrt{-2lnU_1}cos(2\pi U_2)$ soit par $\sqrt{-2lnU_1}sin(2\pi U_2)$. Cette conclusion s'appelle aussi la méthode de la transformation de Box-Muller.

CQFD.

Partie appliquée avec le code

Mise en oeuvre informatique

Nous utilisons langage C pour réaliser la programme.

Code pour créer les données entre 0 et 1

```
float rm(){
     float c= rand()/(double)(RAND_MAX + 1.0);
     return c;
}
```

Code permettant d'obtenir les données pour simuler une gaussienne

```
void gaussienne(float U[max] ,float V[max],float m,float n,float t,coord C[max]){
       n=sqrt(n);
       float tmp1, tmp2;
       for (int j=0; j< t; j++){
                tmp1 = -2*log(U[j]);
                tmp2 = 2*PI*V[j];
                C[j].x=m+n* sqrt(tmp1)*sin(tmp2);
                C[j].y=m+n* sqrt(tmp1)*cos(tmp2);     }

}
```

Ensuite nous traitons les données pour calculer sa fréquence et faire des histogrammes

Code permettant d'obtenir histogramme de données de gaussienne

```
void hist1(int t ,int nb,float m,float n,coord C[max] ){
       float xb[N]; int a[N];
       float da;
       da = 2.0 * SCOPE * sqrt(n) / nb;
       for (int i=0; i<=nb;i++) {
              xb[i]=m-SCOPE * sqrt((double)n)+ i * da;
       }
       for (int i=0; i<nb; i++) a[i]=0;
       for (int i =0; i< t;i++ ){
              for (int j=0;j<nb; j++){
              if (contain(C[i].x,xb[j],xb[j+1])==1) {
              a[j]+=1;
              break;
              }
              }

       }
       float mean, proba;
       for(int i=0 ; i<nb;i++){
              mean=0.5* ( xb[i] + xb[i+1]);
              proba=a[i]/(da*t);
              printf(" %f %f\n",mean,proba);
       }

}
```

Code pour histogramme des données aléatoires fabriqué par générateur de nombre aléatoires

```
void hist2(int t ,int nb , float U[max] ){
       float xbb[N] ;  int aa[N] ;
       float da = 1.0 / nb;
       for (int i=0;  i<=nb;i++)  xbb[i] = i*da;
       for (int i=0; i<nb; i++) aa[i]=0;

       for (int i =0; i< t;i++ ){

              for (int j=0;j<nb;j ++){
                     if (contain(U[i],xbb[j],xbb[j+1])== 0) continue;
                     aa[j]+=1;
                     break;

              }
```

```
        }

        float mean, proba;
        for(int i=0 ; i<nb;i++){
                mean=0.5* (xbb[i] + xbb[i+1]);
                proba=aa[i]/(t*da);
                printf(" %f %f\n", mean,proba);
        }



}
```

Code pour histogramme des doneées gausienne en dimension deux pour vérifier indépendances

```
void hist3(int t ,int nb,float m,float n,coord C[max]  ){
        coord xc[N][N]; int ac[N][N];
        float da  =2 * SCOPE * sqrt((double)n)/nb;
        for (int i=0; i<=nb; i++) {
               for (int j=0;j<=nb;j++){
                       xc[i][j].x=m-SCOPE * sqrt(n)+i*da;
                       xc[i][j].y=m-SCOPE * sqrt(n)+j*da;
               }
        }
        for (int i=0 ; i<nb ; i++)
                for (int j =0; j< nb ; j++)
                        ac[i][j]=0;

        for (int i =0; i< t ;i++ ){
                for (int j=0;j<nb; j++){
                        if (contain(C[i].x,xc[j][0].x,xc[j+1][0].x)==1){
                           for (int k=0; k<nb; k++){
                               if (contain(C[i].y , xc[j][k].y , xc[j][k+1].y)==1) {
                                       ac[j][k]+=1;
                                       break;
                               }
                           }
                           break;
                        }

                }
        }

        float meanx,meany,proba;
```

```
        for(int i=0 ; i<nb;i++ ){
                for (int j=0;j<nb;j++){
                        meanx=0.5*(xc[i][j].x + xc[i+1][j].x);
                        meany=0.5*(xc[i][j].y+xc[i][j+1].y);
                        proba=ac[i][j]/(da*da*t);
                        printf(" %f %f %f\n", meanx, meany, proba);}
                }
}
```

Code de main program

```
int main()
{       srand(time(NULL));
        float U[max],V[max],X[max]; coord C[max];
        float m,n;
        int t,nb;
        int cho;
        scanf("%f %f %d %d %d",&m,&n,&t,&nb,&cho);
        for (int i=0; i < t; i++){
        U[i]=rm();
        V[i]=rm();}
        gaussienne(U,V,m,n,t,C);
        switch(cho){
                case 1 :
                hist1(t,nb,m,n,C);
                break;
                case 2 :
                hist2(t,nb,U);
                break;
                case 3 :
                hist3(t,nb,m,n,C);
                break;
        }
        return 0;
}
```

L'analyse des données

Loi de uniforme

Nous pouvons utiliser le fonction `srand()` et `rand()` pour obtenir des données entre (0,1). Et nous les transformons en images de fonction densité de probabilité et comparons avec la loi de uniforme.

Ce sont les données nous en prenons cent mille. Elles ne peuvent pas strictement obéir à la distribution uniforme et elles fluctuent entre (-0.1,0.1). Peut-être le

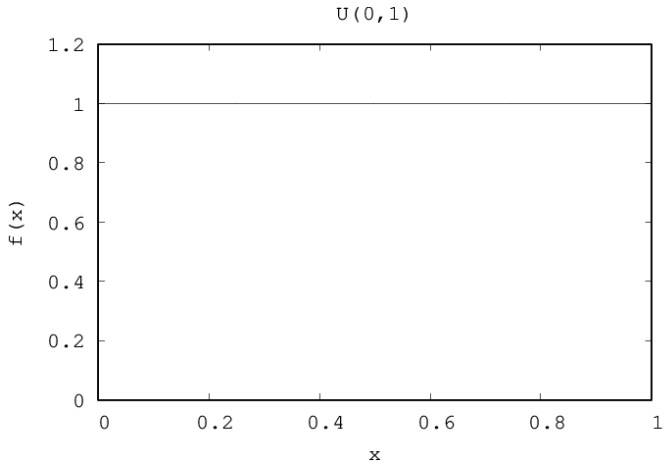

Figure 1: la fonction de densité de loi de uniforme U(0,1)

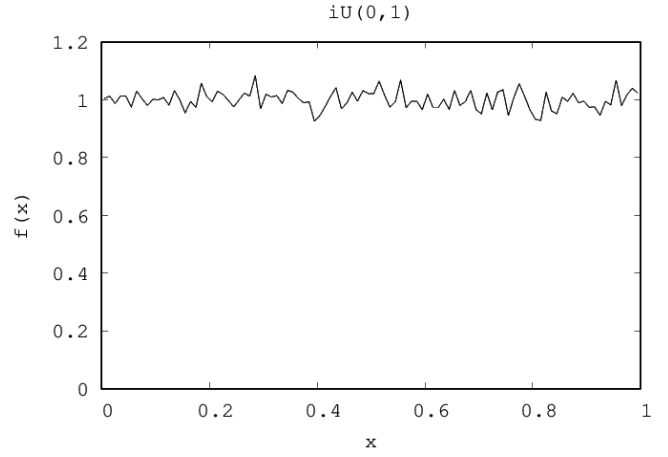

Figure 2: les données nous simulons loi de uniforme

méthode que nous utilisons n'est pas si aléatoire, mais ca ira.

Loi de normale (méthode de gaussienne)

La but de ce projet est que on simule les variables aléatoire gaussienne et on traite les données pour faites en une image de distributions probabilitées pour vérifier s'il s'agit d'un le loi de normale.

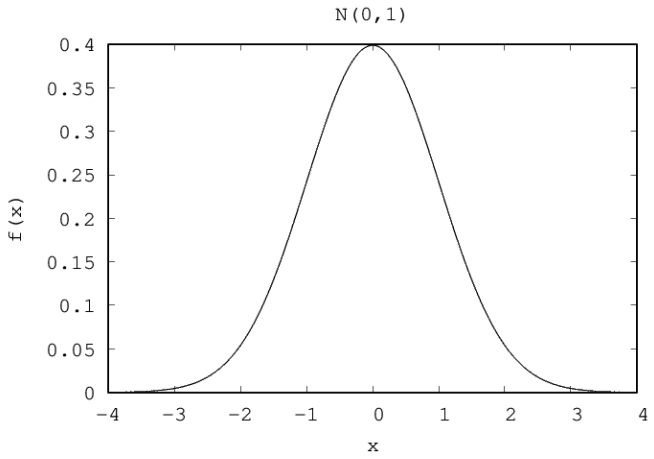

Figure 3: la loi de normale

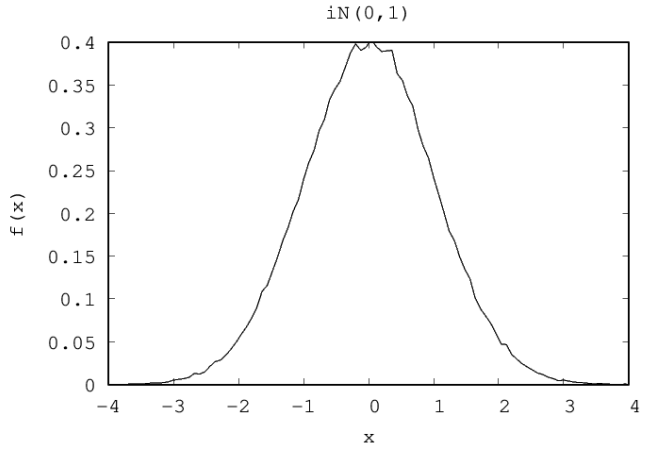

Figure 4: les données nous simulons avec les variable gaussienne

Nous constatons des deux images sont cohérent. Donc ca nous permet de dire la simulation de gaussienne est bien réussite. Simultanément, on simuler les

variable aléatoire gaussienne avec moyenne et varience(comparer avec la fonction de densitées de loi normale avec le même moyenne et varience), ce sont des images.

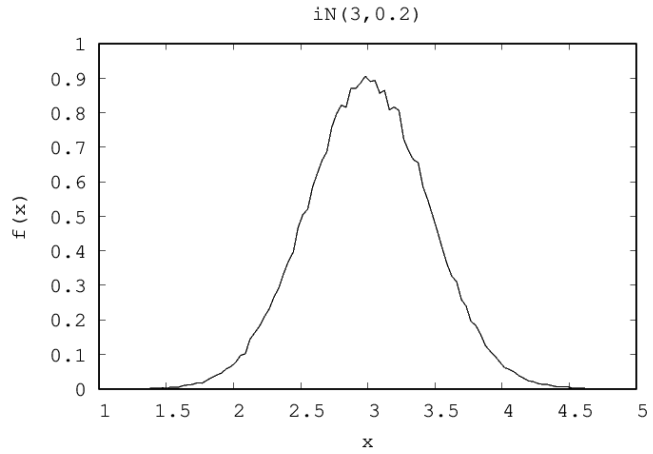

Figure 5: iN(3,0.2)

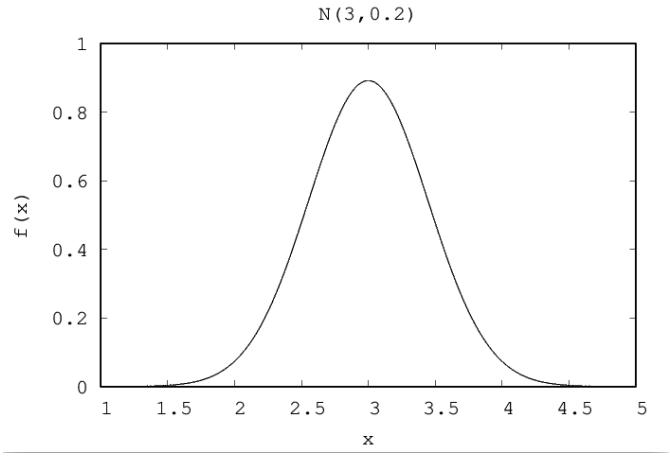

Figure 6: N(3,0.2)

Bien que leurs formes soient similaires, mais leurs échelles d'abscisses sont différentes. C'est dommage que je ne puisse pas les mettre dans une seule image.

La loi de normale en dimension 2(méthode de gausienne)

On veut vérifier des deux varaibles X et Y s'il sont indépendant. Pendant ce temps on veut aussi vérifier la démonstration qu'on fait avant. La variable X

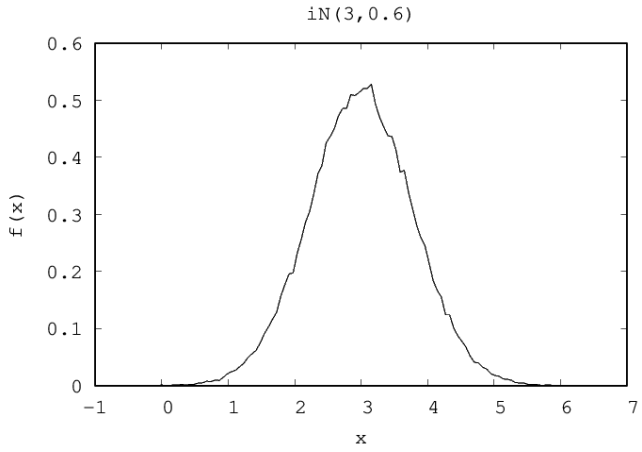

Figure 7: iN(3,0.6)

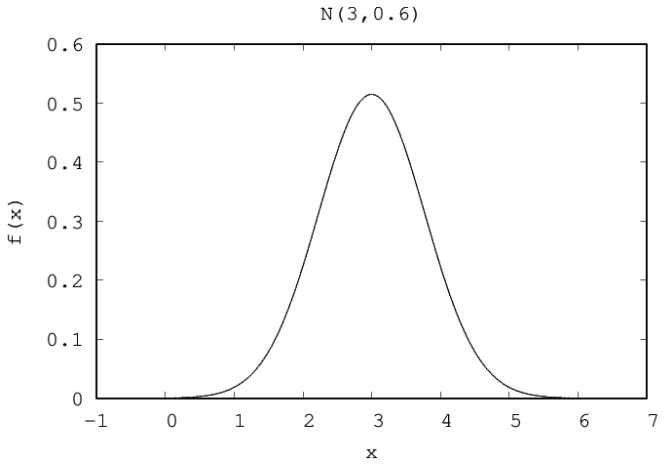

Figure 8: N(3,0.6)

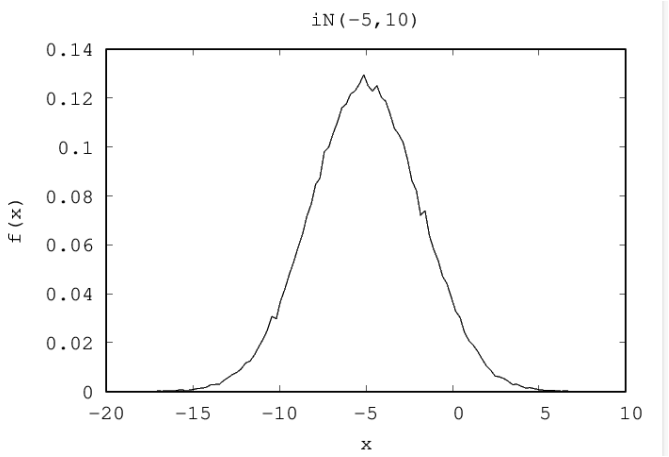

Figure 9: iN(-5,10)

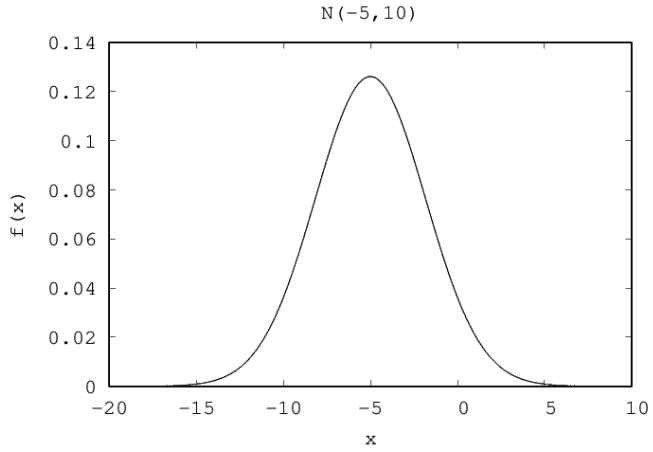

Figure 10: N(-5,10)

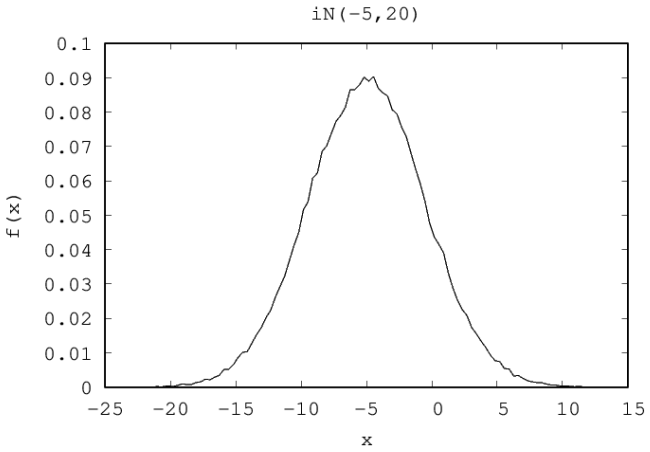

Figure 11: iN(-5,20)

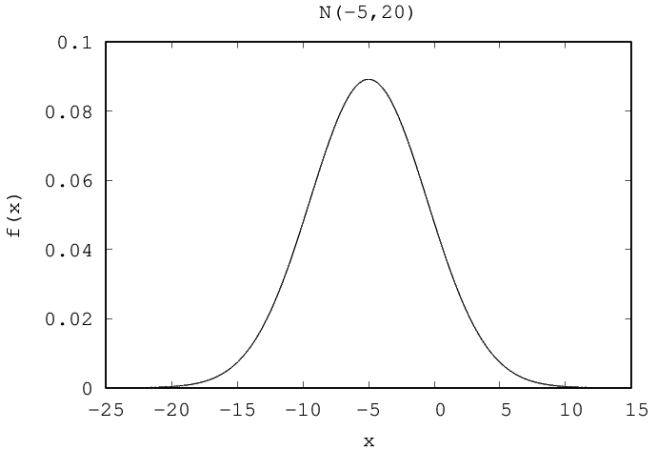

Figure 12: N(-5,20)

est dans un dimension et Y est dans l'autre dimension, on vois que s'il peut fabriquer la fonction de densités dans dimension deux. voici les résultat:

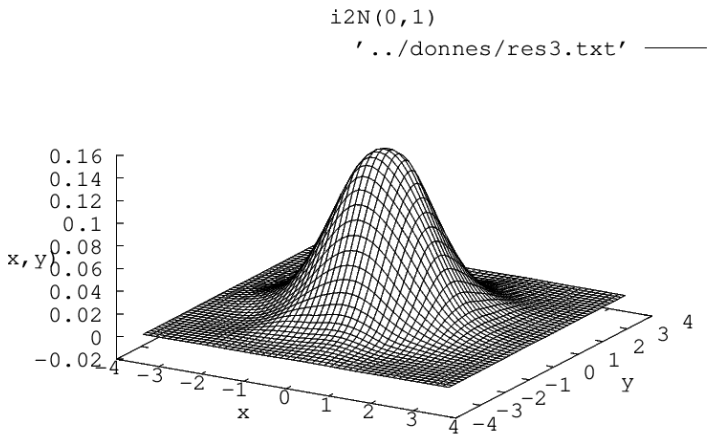

Figure 13: 2iN(0,1)

Elle est bien symétrique, uniforme. on la compare avec le fonction de densité de loi de normale en deux dimension:

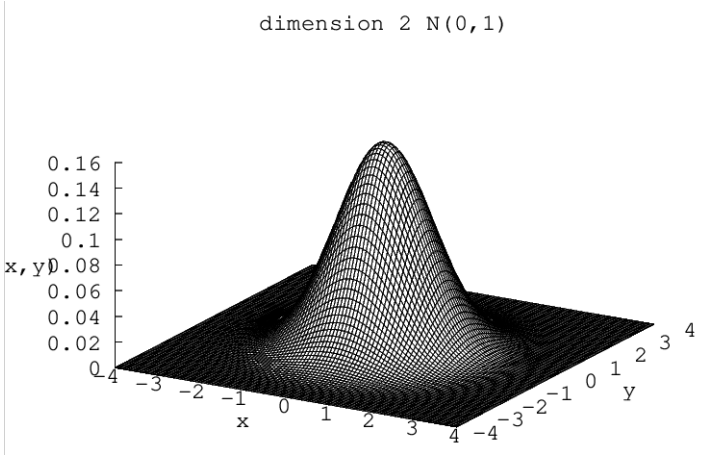

Figure 14: 2N(0,1)

Nous constatons ce sont identique. Donc on peut dire que la démonstration a la

raisons et X et Y sont bien ondépendant.

Conclusion

Nous mettons les projet en oeuvre en 3 partie. D'aborde on démonstre la méthode gaussienne comment il se connect avec la loi de uniforme. Et puis nous programmeons la code et nous débuggeons. Au final, nous traitons des données en images.
Nous avons compris comment utiliser les générateur de nombres aléatoire pour obtient les varaibles gausiennes. la précision dépende du caractère aléatoire des nombre générée par générateur. C'est une méthode très pratique, simple et pas beaucoup de calcul pour PC.
Nous avons compris comment utiliser gnuplot pour faire des images avec des données.
Nous constatons les limite mémoire quand on teste les data, pour mon ordinateur il peut pas traiter au plus de un million de deux integer(8 octets) pour tableaux.

References

[1] RO-projet2.pdf

[2] http://fr.wikipedia.org/wiki/Méthode_de_Box-Muller