Simulation de lois probabilistes sur ordinateur

L'objectif est d'obtenir expérimentalement, sur ordinateur, des variables aléatoires obéissant à des lois classiques, alors que l'ordinateur donne seulement, grâce à son générateur de nombres aléatoires, une loi uniforme.

Supposons que nous n'ayons à notre disposition que la fonction *rand*() qui ramène un nombre entier au hasard entre 0 et RAND_MAX (valant 32767, ou 65535) suivant une loi uniforme. Commençons par fabriquer la fonction *rand01*() qui ramène un nombre réel (un flottant) entre 0 et 1, 1 non compris, suivant une loi uniforme sur [0 1[.

```
float rand01(void)
  {  return (float)rand()/((float)RAND_MAX+1.) ;}
```

En cas de besoin, fabriquons la fonction *random*(*n*) qui ramène un nombre entier au hasard entre 0 et *n* – 1, suivant la loi discrète uniforme sur {0, 1, 2, …, *n*-1}.

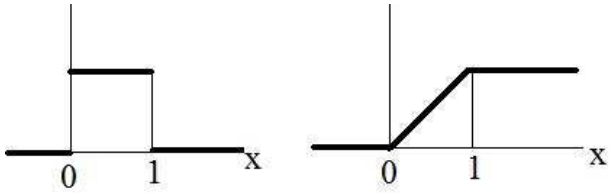
```
int random(int n)
  { return rand01()*n ; }
```

Pour de faibles valeurs de *n*, on peut aussi bien faire

```
int random(int n)
  { return rand() % n ; }
```

Rappelons qu'une variable *U* qui suit une loi uniforme sur [0 1] admet pour densité $d(x) = 1$ sur [0 1] et 0 ailleurs.

Sa fonction de répartition $F(x)$ est telle que $F(x) = p(X \leq x) = x$ sur [0 1], $F(x) = 0$ pour x négatif, et $F(x) = 1$ pour $x \geq 1$. Au cas où l'on veut une variable *V* qui suit une loi uniforme sur [*a b*] avec $b > a$, il suffit de prendre $V = (b - a)\,U + a$.



Densité et fonction de répartition de la loi uniforme sur [0 1]

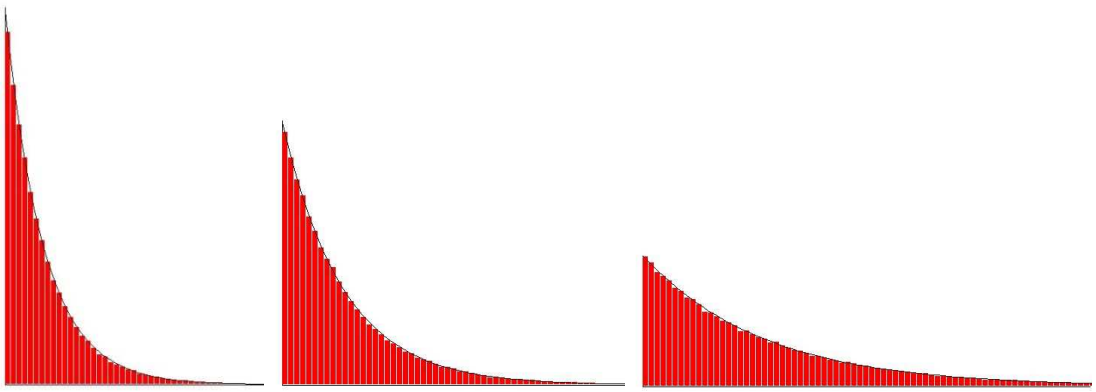
Comment avoir une loi exponentielle à partir de la loi uniforme ?

Rappelons qu'une variable aléatoire continue *X* obéit à une loi exponentielle de paramètre λ (réel positif) lorsque sa densité est $d(x) = \lambda e^{-\lambda x}$ pour x réel ≥ 0, et 0 pour x négatif. Elle a comme fonction de répartition $F(x)$ telle que : $F(x) = p(X \leq x) = 1 - e^{-\lambda x}$ pour x entre 0 et 1, $F(x) = 0$ pour x négatif et $F(x) = 1$ pour $x \geq 1$. Son espérance est $E(X) = 1 / \lambda$ et sa variance $V(X) = 1 / \lambda^2$.

Maintenant, voyons le lien avec la loi uniforme. Sur R+, la fonction F(x) est continue et strictement croissante, et réalise une bijection de R+ sur]0 1]. On a l'équivalence :

$y = 1 - e^{-\lambda x} \quad \leftrightarrow \quad x = -\dfrac{\ln(1-y)}{\lambda}$

$x \in \mathbf{R}+ \qquad\qquad y \in [0\ 1[$



Densité $d(x)$ de la loi exponentielle pour $\lambda = 1{,}5$, $\lambda = 1$, et $\lambda = 0{,}5$, avec la courbe théorique en noir, et les résultats expérimentaux en rouge (*voir programme ci-dessous*)

Avec $y \in [0\ 1[$,
$p(U < y) = y$, $p(-U > -y) = y$, $p(1 - U > 1 - y) = y$, $p(\ln(1 - U) > \ln(1 - y)) = y$,
$p(-\ln(1 - U) < -\ln(1 - y)) = y$, $p(-\ln(1 - U) / \lambda < -\ln(1 - y) / \lambda) = y$, et enfin :
$p(-\ln(1 - U) / \lambda < x) = 1 - e^{-\lambda x}$.

On trouve la fonction de répartition d'une loi exponentielle. Cela signifie que la variable aléatoire $-\ln(1 - U) / \lambda$ obéit à la loi exponentielle de paramètre λ. Et comme il y a autant de chances d'avoir un nombre au hasard U sur [0 1[que le nombre $1 - U$ pour des raisons de symétrie évidentes, la variable aléatoire $-\ln(U) / \lambda$ obéit à cette même loi. Cela donne le programme qui permet de créer une variable aléatoire exponentielle de paramètre λ donné.

```
float vaexpo(void)
{  return  - log( rand01())/lambda ;}
```

Programme allégé (sans les déclarations préliminaires) permettant d'avoir la fonction densité de la loi exponentielle, en passant du continu au discret sur des intervalles de longueur 0,1 :

```
int main ( int argc, char** argv )
{  srand(time(NULL));
   SDL_Init(SDL_INIT_VIDEO);
   ecran=SDL_SetVideoMode(800,600,32, SDL_HWSURFACE|SDL_DOUBLEBUF);
   blanc=SDL_MapRGB(ecran->format,255,255,255);
   noir=SDL_MapRGB(ecran->format,0,0,0);
   rouge=SDL_MapRGB(ecran->format,255,0,0);
   SDL_FillRect(ecran,0,blanc);
   ligne(xorig,yorig,xorig+600,yorig,noir);
   ligne(xorig,yorig,xorig,yorig-zoomy*lambda,noir);

  for(i=0; i<NBEXP; i++)
    {  vaexpo10=(int) (10.*vaexpo());   nbf [vaexpo10]++; }
```

```
  for(k=0; k<80;k++)
  { proba=(float)nbf[k]/(float)NBEXP;
    densite=proba / 0.1;
    rectangle=SDL_CreateRGBSurface(SDL_DOUBLEBUF,zoomx*0.1-1,
                                   densite*zoomy,32,0,0,0,0);
    position.x= xorig+0.1*zoomx*k; position.y=yorig-densite*zoomy;
    SDL_FillRect(rectangle,NULL,rouge);
    SDL_BlitSurface(rectangle,NULL,ecran,&position);
  }
  x=0.; y=lambda;   /* tracé de la courbe théorique */
   while(x<8.)
   {  newx=x+0.1;   newy = lambda*exp(-lambda*newx) ;
      ligne(xorig+zoomx*x, yorig - zoomy*y, xorig+zoomx*newx,yorig-zoomy*newy,noir);
      x=newx; y=newy;
   }
  SDL_Flip(ecran);
   pause(); return 0;
}
```

Quelques explications :

Imaginons que la variable aléatoire exponentielle *vaexpo*() valle 3,764987… On la transforme en 37, nombre entier, dans *vaexpo10*, et l'on augmente de 1 le nombre de fois où l'on tombe sur 37, avec *nbf* [37]++. Au terme des *NBEXP* expériences, la division de *nbf* [37] par *NBEXP* donne la probabilité de tomber entre 3,7 et 3,8. En divisant ce poids de probabilité par la longueur de la zone concernée, soit 0,1 ici, on a la densité en 3,7. Il ne reste plus qu'à tracer la barre rectangulaire verticale correspondante.

Somme de variables exponentielles

Nous allons montrer la propriété suivante :

Une somme $S_n = X_1 + X_2 + \ldots + X_n$ de n variables aléatoires ($n \geq 1$) obéissant toutes à une loi exponentielle de paramètre λ obéit à la loi gamma de paramètre n et λ, avec pour densité

$$d_n(x) = \frac{e^{-\lambda x}\lambda^n x^{n-1}}{(n-1)!}$$ pour x réel ≥ 0, et 0 sinon.

Pour la démonstration, faisons un raisonnement par récurrence :

- La formule est vraie pour $n = 1$, puisque pour $S_1 = X_1$, on a la loi exponentielle de paramètre λ, et l'on a bien $d_1(x) = \lambda e^{-\lambda x}$.
- Supposons la formule vraie à un rang n, et montrons qu'elle reste vraie au rang $n+1$:

$S_{n+1} = S_n + X_{n+1}$, où S_n a pour densité d_n par hypothèse de récurrence et X_{n+1} a pour densité d_1 ; ces deux variables étant indépendantes. La densité de S_{n+1} est alors :

$$D(x) = \int_{-\infty}^{+\infty} d_n(t)d_1(x-t)dt$$

Comme $d_n(t)$ est nulle pour t négatif, on a aussi $D(x)=\int_0^{+\infty} d_n(t)d_1(x-t)dt$, et comme $d_1(x-t)$ est nulle pour $t > x$, il reste $D(x)=\int_0^x d_n(t)d_1(x-t)dt$.

$$D(x)=\int_0^x \frac{e^{-\lambda t}\lambda^n t^{n-1}}{(n-1)!}\lambda e^{-\lambda(x-t)}dt=\frac{e^{-\lambda x}\lambda^{n+1}}{(n-1)!}\int_0^x t^{n-1}dt=\frac{e^{-\lambda x}\lambda^{n+1}}{(n-1)!}\frac{x^n}{n}$$

$$=\frac{e^{-\lambda x}\lambda^{n+1}x^n}{n!}=d_{n+1}(x)$$

On a bien trouvé la formule au rang n+1.

Loi de Poisson et lien avec la loi exponentielle

Rappelons qu'une variable aléatoire discrète X obéit à la loi de Poisson de paramètre λ (réel positif) lorsque : $p(X=k)=e^{-\lambda}\frac{\lambda^k}{k!}$ avec k entier naturel supérieur ou égal à 0. Son espérance est $E(X) = \lambda$ et sa variance $V(X) = \lambda$.

Reprenons la somme S_n de variables aléatoires exponentielles du paragraphe précédent, toutes avec le même paramètre λ, et introduisons la variable aléatoire discrète N, telle que :

$N =$ le plus grand des k (entiers ≥ 1) tels que $S_k \leq 1$, et $N = 0$ lorsque $X_1 > 1$.

1) Montrer que pour $n \geq 1$ l'évènement $N \geq n$ équivaut à $S_n \leq 1$.

L'évènement $N \geq n$ signifie que le plus grand des k tels que $S_k \leq 1$ est au moins égal à n, ce qui se résume à $S_n \leq 1$.

2) Montrer que N obéit à une loi de Poisson de paramètre λ. Pour cela :

a) Montrer que $p(S_{n+1} \leq 1)=1-\sum_{j=0}^{n} e^{-\lambda}\frac{\lambda^j}{j!}$ pour $n \geq 0$. On utilisera pour cela la formule de Taylor avec reste intégral :

$$e^{\lambda}=\sum_{j=0}^{n}\frac{\lambda^j}{j!}+\int_0^{\lambda} e^{\lambda-t}\frac{t^n}{n!}\,dt \quad (n \geq 0).$$

Divisons la formule précédente par e^{λ} : $1=\sum_{j=0}^{n} e^{-\lambda}\frac{\lambda^j}{j!}+\int_0^{\lambda} e^{-t}\frac{t^n}{n!}\,dt$ puis faisons le changement de variable $u = t / \lambda \leftrightarrow u = \lambda t$ dans l'intégrale :

$$1=\sum_{j=0}^{n} e^{-\lambda}\frac{\lambda^j}{j!}+\int_0^1 e^{-\lambda u}\frac{\lambda^{n+1}t^n}{n!}\,dt \quad \text{d'où} \quad \int_0^1 e^{-\lambda u}\frac{\lambda^{n+1}t^n}{n!}\,dt=1-\sum_{j=0}^{n} e^{-\lambda}\frac{\lambda^j}{j!}$$

Sous l'intégrale on retrouve la densité de S_{n+1}, et finalement :

$$p(S_{n+1} \leq 1)=1-\sum_{j=0}^{n} e^{-\lambda}\frac{\lambda^j}{j!}$$

b) Montrer que $p(N \le n) = \sum_{j=0}^{n} e^{-\lambda} \frac{\lambda^j}{j!}$ pour tout $n \ge 0$.

- Pour $n = 0$, $p(N \le 0) = p(N = 0) = p(X_1 > 1) = 1 - p(X_1 \le 1) = 1 - (1 - e^{-\lambda})$ $= e^{-\lambda}$ grâce à la fonction de répartition de la loi exponentielle. La formule est bien vraie au rang 0.
- Pour $n \ge 1$, on sait que

$p(N \ge n+1) = p(S_{n+1} \le 1) = 1 - \sum_{j=0}^{n} e^{-\lambda} \frac{\lambda^j}{j!}$. Par complémentarité

$$p(N \le n) = \sum_{j=0}^{n} e^{-\lambda} \frac{\lambda^j}{j!}$$

c) En déduire que $p(N = n) = e^{-\lambda} \frac{\lambda^n}{n!}$

$p(N = n) = p(N \le n) - p(N \le n-1) = e^{-\lambda} \frac{\lambda^n}{n!}$. La variable aléatoire N suit bien une loi de Poisson de paramètre λ.

Comment avoir la loi de Poisson à partir de la loi uniforme

On commence par voir ce qui se passe avec le premier tirage X_1, qui suit la loi exponentielle de paramètre λ, en utilisant le lien entre la loi exponentielle et la loi uniforme sur [0 1]. Cela revient à faire un premier tirage U_1 de la loi uniforme, et de prendre $-\ln(U_1) / \lambda$. Si $-\ln(U_1) / \lambda > 1$, ce qui revient à $\ln U_1 < -\lambda$ ou $U_1 < e^{-\lambda}$, on ramène 0.

Sinon on cherche le plus grand k tel que $S_k \le 1$, c'est-à-dire $\sum_{j=0}^{k} -\frac{\ln U_j}{\lambda} \le 1$, ou bien :

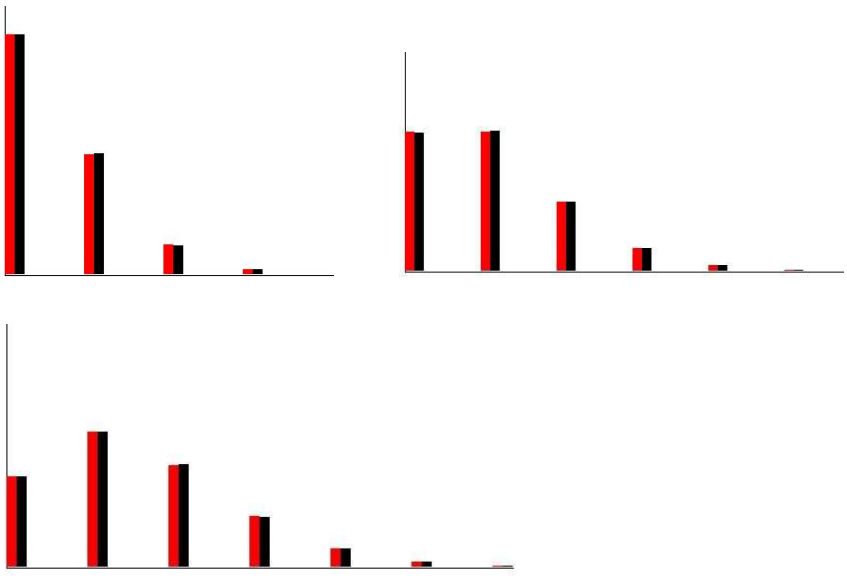
$\sum_{j=1}^{k} \ln U_j \ge -\lambda$, soit $\ln \prod_{j=1}^{k} U_j \ge -\lambda$ ou encore $\prod_{j=1}^{k} U_j \ge e^{-\lambda}$

D'où la fonction ramenant une variable aléatoire qui suit la loi de Poisson de paramètre λ :

```
int vapoisson(void)
{ float product; int k;  /* on se donne en global expo = exp(-lambda) ;  */
   k=0;
   product=rand01();
  while(product>=expo) {k++; product*=rand01();}
  return k;
}
```

Programme (allégé) donnant l'histogramme de la loi de Poisson, d'une part grâce à la formule théorique $p(N = k) = e^{-\lambda} \frac{\lambda^k}{k!}$, et d'autre part grâce à la simulation par le biais de la loi uniforme :

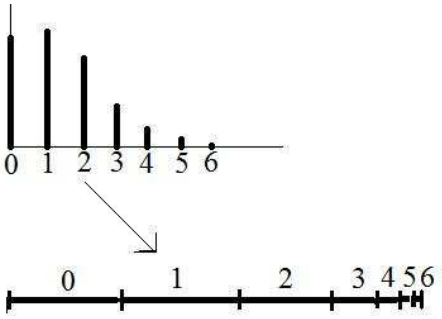
```
int main ( int argc, char** argv )
{
   expo= exp(-lambda);
   srand(time(NULL));
   SDL_Init(SDL_INIT_VIDEO);
   ecran=SDL_SetVideoMode(800,600,32, SDL_HWSURFACE|SDL_DOUBLEBUF);
   blanc=SDL_MapRGB(ecran->format,255,255,255);
   noir=SDL_MapRGB(ecran->format,0,0,0);
   rouge=SDL_MapRGB(ecran->format,255,0,0);
   SDL_FillRect(ecran,0,blanc);
   ligne(xorig,yorig,xorig+500,yorig,noir);
   ligne(xorig,yorig,xorig,yorig-300,noir);   /* repère */
   y=expo*zoomy;         /* histogramme de la loi de Poisson théorique */
   for(k=0;k<8; k++)
     {   rectangle= SDL_CreateRGBSurface(SDL_DOUBLEBUF,10,y,32,0,0,0,0);
         position.x= xorig+zoomx*k; position.y=yorig-y;
        SDL_FillRect(rectangle,NULL,rouge);
        SDL_BlitSurface(rectangle,NULL,ecran,&position);
        y*=lambda/(float)(k+1);
     }
  for(i=0; i<NBEXP; i++) { nbf [vapoisson()]++; }
  for(k=0; k<8;k++)         /* histogramme experimental */
    { proba=(float)nbf[k]/(float)NBEXP;
      rectangle=SDL_CreateRGBSurface(SDL_DOUBLEBUF,10,proba*zoomy,
                                          32,0,0,0,0);
      position.x= xorig+zoomx*k+10; position.y=yorig-proba*zoomy;
      SDL_FillRect(rectangle,NULL,noir);
      SDL_BlitSurface(rectangle,NULL,ecran,&position);
    }
  SDL_Flip(ecran);
  pause(); return 0;
}
```



Histogramme de la loi de Poisson pour $\lambda = 0{,}5$, $\lambda = 1$, et $\lambda = 1{,}5$, théorique en rouge, et expérimental en noir.

Une méthode générale pour simuler une loi quelconque

Cette méthode vaut aussi bien pour une loi discrète que pour une loi continue. Nous allons traiter un exemple, celui de la loi de Poisson, qui est une loi discrète, étant entendu qu'il en serait de même pour une loi continue, qu'il suffit de discrétiser.



On part de l'histogramme de la loi de Poisson avec ses probabilités pour chaque valeur entière *k*, soit :

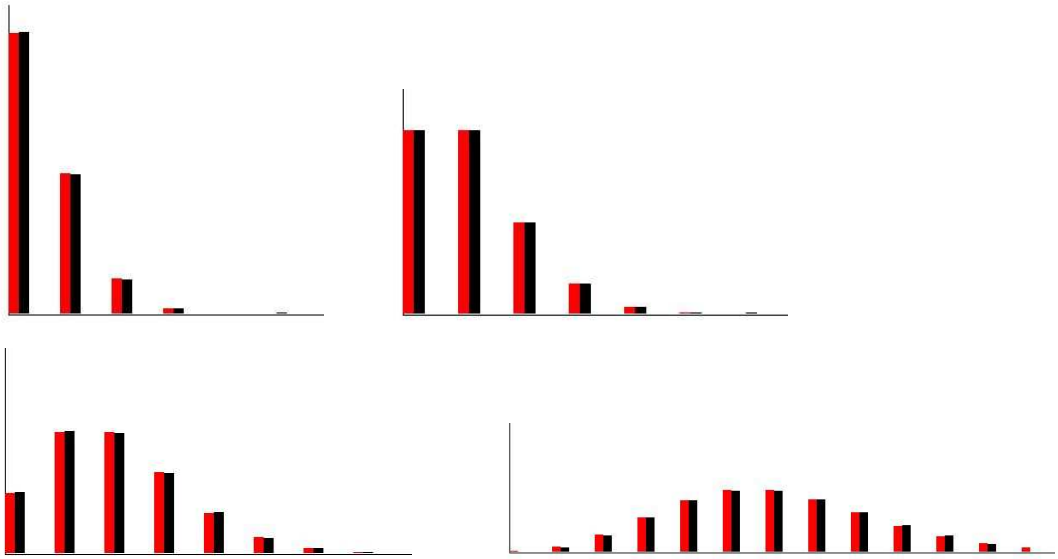
$$p(X = k) = e^{-\lambda}\frac{\lambda^k}{k!}.$$

Sachant que la somme de ces probabilités vaut 1, on multiplie ces probabilités par 1000, la longueur de chaque barre de l'histogramme devenant *length*[*k*]= 1000.**p*(*X* = *k*). Puis on met ces barres en succession l'une derrière l'autre, la longueur totale étant 1000, comme indiqué sur le dessin. Il suffit alors de tirer un nombre au hasard entre 0 et 999, soit *random*(1000), suivant la loi uniforme pratiquée par l'ordinateur, et de noter le numéro de la zone où tombe ce nombre (voir dessin). Ce numéro suit la distribution de la loi de Poisson. On en déduit le programme :

```
On se donne lambda et le nombre d'expériences NBEXP
int main ( int argc, char** argv )
{   int i,k,vaU1000; float y,proba;
    expo= exp(-lambda);
    srand(time(NULL));
    SDL_Init(SDL_INIT_VIDEO);
    ecran=SDL_SetVideoMode(800,600,32, SDL_HWSURFACE|SDL_DOUBLEBUF);
    SDL_FillRect(ecran,0,blanc);
    ligne(xorig,yorig,xorig+690,yorig,noir);   ligne(xorig,yorig,xorig,yorig-300,noir);
    y=expo;  length[0]=1000.*expo;     /* y correspond à p(X = k) */
    for(k=0;k<15; k++)
    {  rectangle= SDL_CreateRGBSurface(SDL_DOUBLEBUF,10,y*zoomy,32,0,0,0,0);
       position.x= xorig+zoomx*k; position.y=yorig-y*zoomy;
       SDL_FillRect(rectangle,NULL,rouge);
       SDL_BlitSurface(rectangle,NULL,ecran,&position);
       y*=lambda/(float)(k+1); length[k+1]=1000.*y;
    }
    for(i=0;i<NBEXP;i++)
    {
       vaU1000=random(1000);
       k=0;
       while(length[k]==0) k++;
       while(length[k]>0 && vaU1000 - length[k]>=0) { vaU1000-=length[k]; k++; }
       nbf[k]++;
    }
   for(k=0; k<12;k++)
   {
    proba=(float)nbf[k]/(float)NBEXP;
    rectangle= SDL_CreateRGBSurface(SDL_DOUBLEBUF,10,proba*zoomy,32,0,0,0,0);
       position.x= xorig+zoomx*k+10; position.y=yorig-proba*zoomy;
       SDL_FillRect(rectangle,NULL,noir);
```

```
        SDL_BlitSurface(rectangle,NULL,ecran,&position);
    }
    SDL_Flip(ecran);
     pause(); return 0;
}

float rand01(void) {  return (float)rand()/((float)RAND_MAX+1.) ; }
int random(int n) {   return (int)( rand01()*(float)n) ; }
```



Distribution de la loi de Poisson (théorique et expérimentale) pour $\lambda = 0{,}5$, $\lambda = 1$, $\lambda = 2$, $\lambda = 6$

Remarque finale : à propos de la loi de Poisson

La loi de Poisson, du nom de Siméon Denis Poisson (années 1800), intervient essentiellement dans deux configurations :

- Elle s'applique aux phénomènes considérés comme rares, par exemple le taux de pannes dans une entreprise de production de pièces, et le nombre de défectuosités des pièces produites. En effet, la distribution de Poisson, foncièrement asymétrique mais de moins en moins, comme l'indiquent les histogrammes ci-dessus, tend à ressembler à la loi binomiale lorsque son paramètre λ augmente. Plus précisément, on démontre qu'une suite de variables aléatoires X_n obéissant à la loi binomiale $B(n, p)$ où p varie avec n de façon que np tende vers une limite λ pour n infini, converge en loi vers la loi de Poisson de paramètre λ. On considère que l'on obtient une bonne approximation de la loi binomiale par la loi de Poisson dès que l'effectif n est supérieur ou égal à 50, que la probabilité d'occurrences p d'un évènement (rare) est inférieur ou égal à 0,1, et que np est inférieur ou égal à 15. Voici un exemple :

Une entreprise fabrique des pièces dont 1% en moyenne sont défectueuses. Lorsque 400 pièces sont sorties, on désire savoir la probabilité qu'il y en ait k défectueuses parmi elles (avec k entier ≥ 0).

En prenant pour X le nombre de pièces défectueuses parmi les 400, cette variable suit une loi binomiale de paramètres $n = 400$, et $p = 0{,}01$. D'où

$p(X = k) = C_{400}^{k}\, 0{,}01^{k}\, 0{,}99^{400-k}$. Un tel nombre est difficile à calculer. Aussi utilise-t-on l'approximation par une loi de Poisson de paramètre $np = 4$, puisqu'on est bien

dans ce cas, avec $n \geq 50$, $p \leq 0{,}1$ et $np \leq 15$. Finalement $p(X = k) = e^{-4}\frac{4^k}{k!}$. Par exemple $p(X = 4) = 0{,}195$: il y a 19 chances sur 100 de tomber sur 4 pièces défectueuses parmi les 400.

- Elle s'applique aussi à des phénomènes qui s'inscrivent dans une certaine durée, par exemple le nombre de communications dans un réseau téléphonique durant un temps *t*, ou le nombre de personnes sortant d'un aéroport pendant un temps *t* (sous réserve que cet aéroport fonctionne nuit et jour). En appelant ce nombre *X*, on considère qu'il suit une loi de Poisson de paramètre proportionnel à *t*, soit *a t*. En effet, on constate notamment que la probabilité que ce nombre soit nul, soit $p(X = 0)$ peut être élevée lorsque le temps *t* choisi est bref, et que cette probabilité devient faible lorsque *t* est grand, ce qui correspond à l'évolution de l'histogramme de la loi de Poisson.

Prenons l'exemple de l'aéroport : le nombre *X* de voyageurs sortant de l'aéroport pendant un intervalle de temps *t* obéit à la loi de Poisson de paramètre *a t*, soit $p(X = k) = e^{-at}\frac{(at)^k}{k!}$. Maintenant, l'intervalle de temps *t* étant donné, de l'instant 0 à l'instant *t* par exemple, on désire connaître le temps *T* (à partir de l'instant 0) qu'il faut attendre pour voir sortir le premier voyageur. L'évènement $X = 0$, signifiant qu'aucun voyageur n'est sorti pendant le temps *t*, équivaut au fait que le temps d'attente de sortie du premier voyageur est $T > t$, *T* étant une variable aléatoire continue.

$p(T > t) = p(X = 0) = e^{-at}$, d'où $p(T \leq t) = 1 - e^{-at}$, avec *t* positif, la probabilité $p(T < 0)$ étant évidemment nulle. On reconnaît la loi exponentielle de paramètre *a*. On vient d'établir un lien entre la loi de Poisson et la loi exponentielle comme temps d'attente.

- Enfin signalons que la somme de deux variables de Poisson indépendantes, de paramètres respectifs λ et μ, suit aussi une loi de Poisson de paramètre $\lambda + \mu$. [1]

[1] La fonction génératrice de la loi de Poisson de paramètre λ est $G(z) = e^{\lambda(z-1)}$. On sait que la fonction génératrice associée à la somme de deux variables aléatoires indépendantes est le produit de leurs fonctions génératrices, soit ici $e^{\lambda(z-1)}e^{\mu(z-1)} = e^{(\lambda+\mu)(z-1)}$. On trouve bien une loi de Poisson de paramètre $\lambda + \mu$.