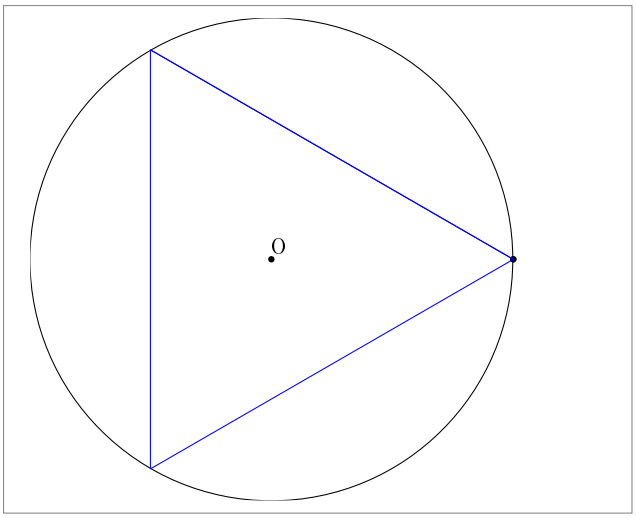
Le paradoxe de Bertrand traité avec Scilab

Paradoxe de Bertrand

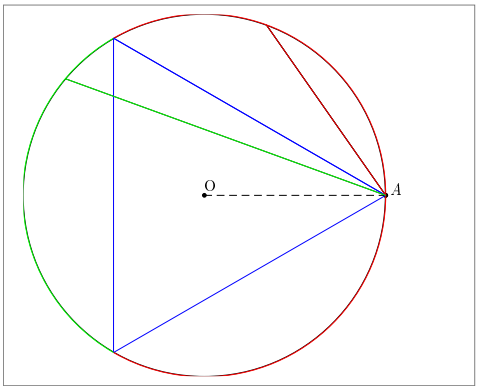
Pour cette activité, je me suis inspiré d'un document de Charles Suquet, Professeur à Lille 1, document qu'il a diffusé ces dernières années et que vous trouverez dans son intégralité ici `http://www-gat.univ-lille1.fr/~suquet/`, il vous suffit de "cliquer" sur *Probabiliés géométriques*.
Je le trouve **superbement bien fait** et vais essayer de l'exploiter pour des activités géométriques, statistiques et probabilistes au lycée l'an prochain.
Je n'aborde pas la partie probabiliste du paradoxe et vous invite à le faire pour donner une justification rigoureuse de chaque probabilité obtenue selon la méthode choisie de construction de la corde.[1]

Je vous rappelle l'idée.
On considère un cercle de rayon 1, par exemple le cercle trigonomètrique, et on y inscrit un triangle équilatéral, qui sera de côté $\sqrt{3}$.
On trace une corde et on cherche la probabilité que cette corde soit de longueur plus grande que $\sqrt{3}$.
Nous allons, grâce à Scilab, simuler de différentes façons, comme nous le suggère Charles dans son document, le tracé de cordes sur le cercle.
Nous allons observer l'évolution des fréquences empiriques en fonction du nombre de simulations, ceci pour chaque méhode de tracé de cordes choisi.On constatera des convergences différentes selon la méthode choisie, d'où le nom de

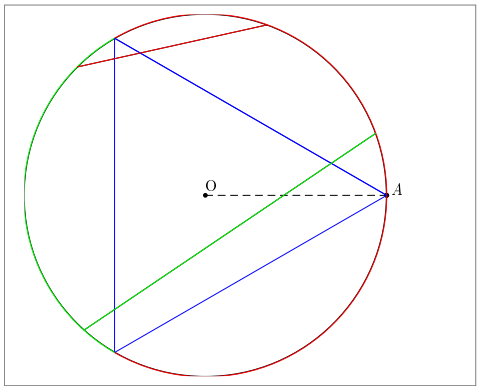
Paradoxe de Bertrand



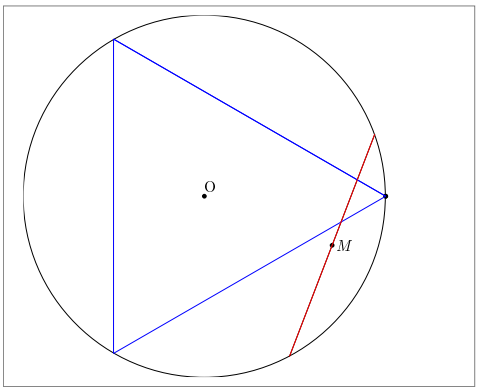
On définit la corde en choisissant un point M au hasard sur le cercle et en traçant la corde $[AM]$ correspondante.



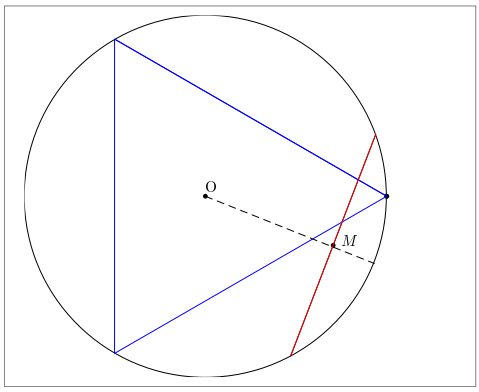
On choisit deux points M et N sur le cercle et cela définit la corde $[MN]$ correspondante.



On choisit un point M au hasard à l'intérieur du cercle, et on définit la corde de telle sorte que le point M soit le milieu de la corde, comme suit



Le milieu de la corde est choisi au hasard de façon que la distance OM suive la loi uniforme sur $[0;1]$.



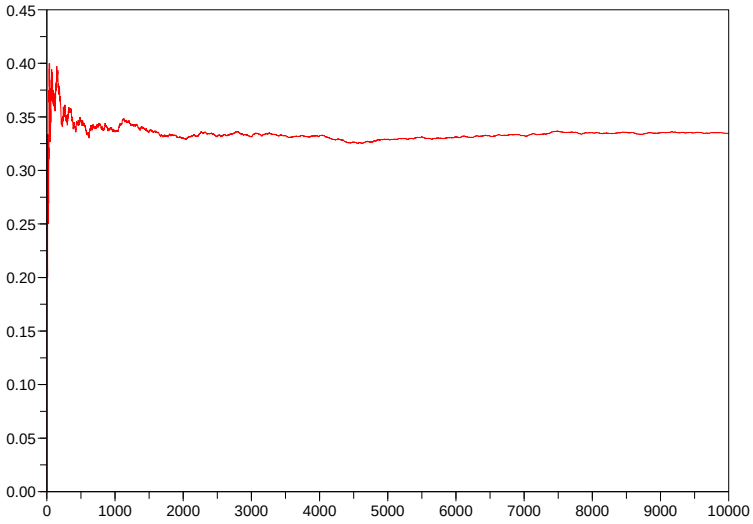
Aide

Si l'on veut programmer la première simulation, il faut
- définir le nombre de points que l'on va simuler au total, par exemple $n = 100$,
- puis simuler n points appartenant au cercle trigo(on pourra pour cela simuler les affixes correspondants),
- puis calculer les distances entre chaque point simulés et le point A d'affixe 1,
- puis tester si chaque distance calculée est supérieure à $\sqrt{3}$,
- cela permet de calculer, en fonction du nombre de points simulés, la fréquence des points dont la distance au point A est supérieure à $\sqrt{3}$,
- reste à tracer l'évolution de cette fréquence en fonction du nombre de points simulés.

Voici ce que l'on obtient pour chaque méthode de simulation
- La première méthode : Le point A d'affixe 1 est fixé et il suffit de choisir au hasard un point sur le cercle pour définir la corde

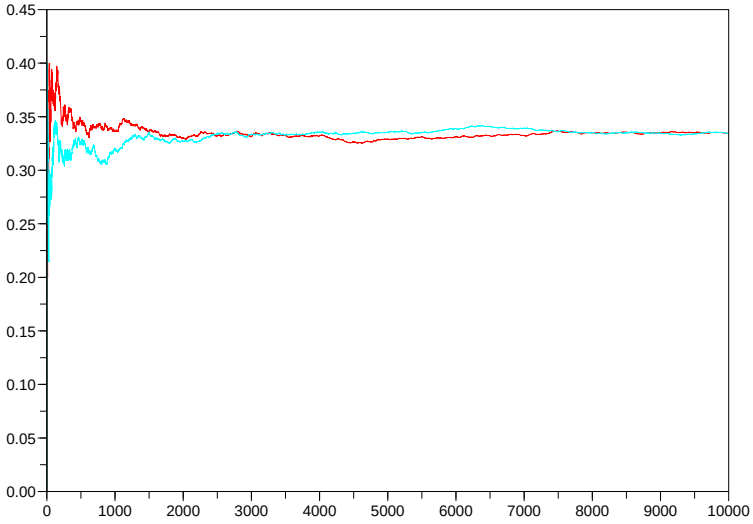
```
// On fixe le nombre de simulations
→ N=10000;
→ x=rand(N,1);
→ xx=[1:N];
// On simule N points sur le cercle trigo
→ for i=1:N,M(i)=exp(%i*2*%pi*x(i));end
// On détermine les cordes de longueur supérieures ou égales à √3
→ for i=1:N,
→  if abs(M(i)-1)>=sqrt(3) then
→  eff(i)=1;
→  elseif abs(M(i)-1)<sqrt(3) then
→  eff(i)=0;
→  end,
```

```
→ end
// On calcule les fréquences correspondantes
→ for i=1:N, f(i)=(sum(eff(1:i))/i);end
→ plot2d(xx,f,style=5)
```



– La deuxième méthode : On choisit au hasard deux points sur le cercle pour définir la corde

```
→ N=10000;
→ x1=rand(N,1);
→ y1=rand(N,1);
→ xx1=[1:N];
→ for i=1:N,M1(i)=exp(%i*2*%pi*x1(i));P1(i)=exp(%i*2*%pi*y1(i));end
→ for i=1:N,
→  if abs(M1(i)-P1(i))>=sqrt(3) then
→  eff1(i)=1;
→  elseif abs(M1(i)-P1(i))<sqrt(3) then
→  eff1(i)=0;
→  end,
→ end
→ for i=1:N, f1(i)=(sum(eff1(1:i))/i);end
→ plot2d(xx1,f1,style=4)
```

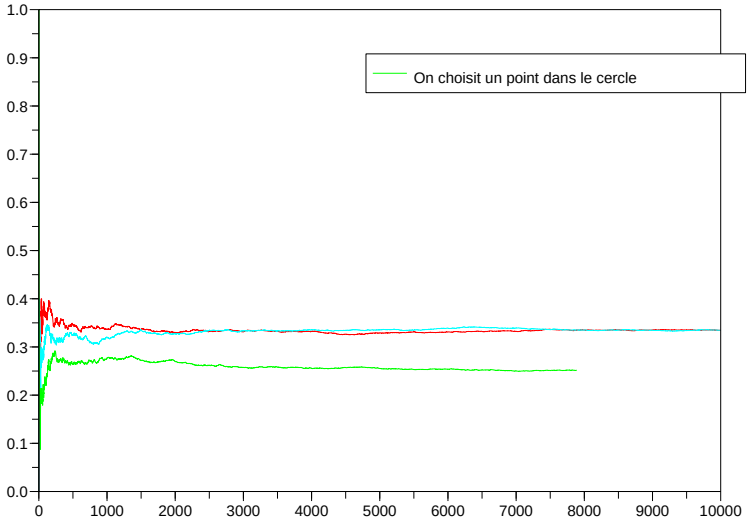


– La troisième méthode : On choisit un point au hasard à l'intérieur du cercle

```
→ N=10000;
// je simule N points dans le carré de côté 2 et de centre O
→ for i=1:N,x2(i)=2*rand()-1;y2(i)=2*rand()-1;end
// je compte ceux qui sont dans le cercle trigo
→ k=0;
→ for i=1:N,l2(i)=(x2(i))^2+(y2(i))^2;end
→ for i=1:N,if sqrt(l2(i))<1 then k=k+1;z1(k)=x2(i);z2(k)=y2(i);end,end
→ n=length(z1)
 n  =
```

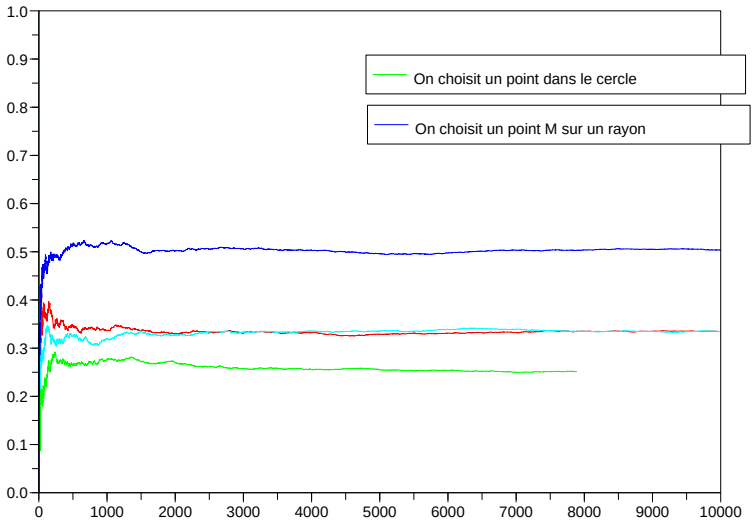
7889

```
→ xx2=[1:n];
// je compte le nombre de cordes correspondantes dont la longueur convient
→ for i=1:n,ll(i)=(z1(i))^2+(z2(i))^2;end
→ for i=1:n,
→  if 2*sqrt(1-ll(i))>=sqrt(3) then
→  eff2(i)=1;
→  elseif 2*sqrt(1-ll(i))<sqrt(3) then
→  eff2(i)=0;
→  end,
→ end
→ for i=1:n,f2(i)=(sum(eff2(1:i)))/i;end
// je mets une légende au graphique dont je choisis la couleur du tracé
→ legends(["On choisit un point dans le cercle "],[3])
→ plot2d(xx2,f2,style=3)
```



– La quatrième méthode : On choisit un point au hasard sur un rayon

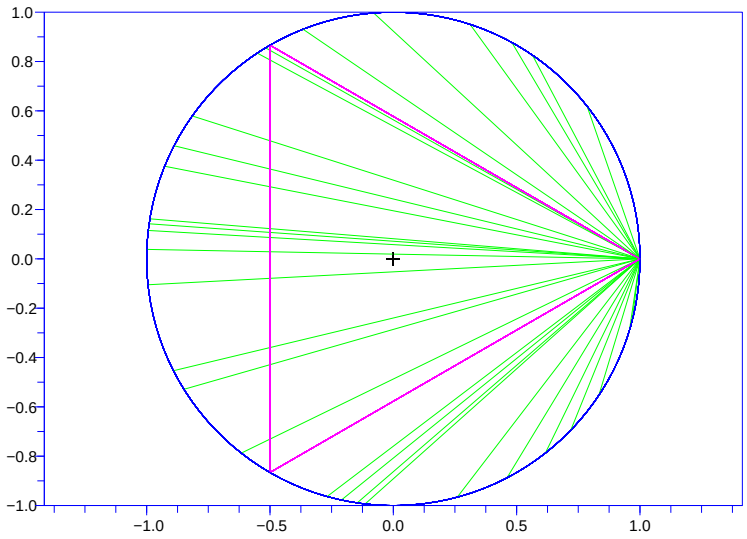
```
→ N=10000;
→ x3=[1:N];
→ for i=1:N, tt(i)=rand();l3(i)=2*sqrt(1-(tt(i))^2);end
→ for i=1:N,
→ if  l3(i)>=sqrt(3) then eff3(i)=1;
→ elseif  l3(i)<sqrt(3) then eff3(i)=0;
→ end,
→ end
→ for i=1:N, f3(i)=(sum(eff3(1:i)))/i;end;
→ legends(["On choisit un point M sur un rayon"],[2])
→ plot2d(x3,f3,style=2)
```



Voici un programme qui trace 30 cordes successivement sur le cercle trigo, le point A d'affixe 1 étant

fixé, et calcule la fréquence des cordes de longueur supérieure ou égale à $\sqrt{3}$
(j'espère mettre à profit très prochainement les conseils de Christophe Poulain pour proposer une figure obtenue grâce à Metapost)

```
→ clf;
→ c=0;
→ for i=1:30,
→ z=exp(%i*rand()*2*%pi);
→ if abs(z-1)>sqrt(3) then
→    c=c+1;
→ end;
→ x=[real(z) 1];
→ y=[imag(z) 0];
→ xset("color",3)
→ plot2d(0,0,-1,"031"," ",[-1,-1,1,1])
→ xpoly(x(1:2),y(1:2),"lines",1),
→ xset("color",6)
→ t=exp(%i*0*2*%pi);
→ tt=exp(%i*(1/3)*2*%pi);
→ ttt=exp(%i*(2/3)*2*%pi);
→ u=[real(t) real(tt) real(ttt)];
→ v=[imag(t) imag(tt) imag(ttt)];
→ xpoly(u(1:3),v(1:3),"lines",1)
→ xset("color",2)
→ xarc(-1,1,2,2,0,360*64)
→ end
```




```
→ c/30

 ans  =
```

0.4333333

En voici un autre qui trace 30 cordes successivement sur le cercle trigo, en choisissant deux points au hasard sur le cercle

```
→ clf;
→ for i=1:30,
→ z=exp(%i*rand()*2*%pi);
→ zz=exp(%i*rand()*2*%pi);
→ x=[real(z) real(zz)];
→ y=[imag(z) imag(zz)];
→ xset("color",3)
→ plot2d(0,0,-1,"031"," ",[-1,-1,1,1])
→ xpoly(x(1:2),y(1:2),"lines",1),
→ xset("color",6)
→ t=exp(%i*0*2*%pi);
→ tt=exp(%i*(1/3)*2*%pi);
→ ttt=exp(%i*(2/3)*2*%pi);
→ u=[real(t) real(tt) real(ttt)];
→ v=[imag(t) imag(tt) imag(ttt)];
→ xpoly(u(1:3),v(1:3),"lines",1)
→ xset("color",2)
→ xarc(-1,1,2,2,0,360*64)
→ end
```