APPROXIMATION DE Π PAR LA MÉTHODE DE MONTE CARLO AVEC MAXIMA

1 Présentation de la méthode

On considère un carré de côté 2 centré sur l'origine dans un repère orthonormé et le cercle unité. Un point généré au hasard dans le carré a une probabilité de $\pi/4$ de se trouver dans le cercle :

```
(% i1)  Aire_carre :2^ 2;
```
$(Aire\_carre) \quad 4$

```
(% i2)  Aire_cercle :%pi*1^ 2;
```
$(Aire\_cercle) \quad \pi$

```
(% i3)  Probabilite :Aire_cercle/Aire_carre;
```
$(Probabilite) \quad \frac{\pi}{4}$

Si nous générons un grand nombre de points, alors la loi des grands nombres montre que la proportion de points situés dans le cercle va se rapprocher de cette probabilité, ce qui permet d'obtenir une approximation de $\pi/4$. On multiplie donc ensuite par 4 pour obtenir une approximation de π.

2 Les bases du programme

2.1 Génération d'un point

Nous générons un point dans le carré à l'aide de la commande :

```
(% i4)  pt1 :[2*random(1.0)-1,2*random(1.0)-1];
```
$(pt1) \quad [0.8276191992257917\,,-0.6096307644044225]$

2.2 Test de l'appartenance du point au cercle

On teste si les coordonnées du point vérifient l'inéquation qui définit le disque unité :

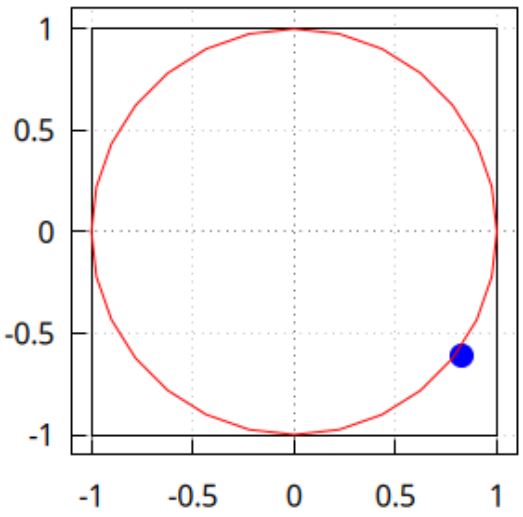
```
(% i6)  print( Somme des carrés des coordonnées du point : ,pt1[1]^ 2+pt1[2]^ 2)$
        if (pt1[1]^ 2+pt1[2]<1 or pt[1]^ 2+pt[2]^ 2=1) then print( Le point est à l'intérieur du cercle )
        else print( Le point n'est pas à l'intérieur du cercle )$
```
Somme des carrés des coordonnées du point : 1.0566032078354612
Le point est à l'intérieur du cercle

2.3 Représentation graphique

Représentons le carré, le cercle et le point pt1 :

```
(% i7)  wxdraw2d(
        title =concat( pt1 = ( , string(pt1[1]),  , , string(pt1[2]), ) ),
        color=black,proportional_axes =xy,
        grid=true, xaxis=true, yaxis=true,
        xrange=[-1.1,1.1], yrange=[-1.1,1.1],
        points_joined = true,
        point_type = none,
        points([[-1,-1], [-1, 1]]),  points([[-1,1], [1,1]]),
        points([[1,1], [1,- 1]]),  points([[1,-1], [-1,- 1]]),
        points_joined = false,
        color=blue,
        point_type=7, point_size=2,
        points([pt1]),
        color=red,
        parametric(cos(t), sin(t), t, 0, 2*%pi)
        );
```

pt1 = (0.8276191992257917 , -0.6096307644044225)



(%t7)

(%o7)

3 Premier programme

Ce premier programme prend comme paramètre le nombre de points générés.Pour chaque point, on incrémente un compteur lorsque ce point est à l'intérieur du cercle.On affiche chaque point généré, la valeur du compteur.A la fin de la boucle, on affiche le compteur final et l'approximation de π obtenue.

```
(% i8)  approx_pi_mc(n) :=block(
        [compteur,i ],
        compteur :0 ,
        for i :1 thru n do (
        pt :[2*random(1.0)-1,2*random(1.0)-1],
        print( point = ,pt, somme = ,pt[1]^ 2+pt[2]^ 2),
        if (pt[1]^ 2+pt[2]^ 2<1 or pt[1]^ 2+pt[2]^ 2=1 ) then compteur : compteur +1,
        print( compteur = ,compteur)
        ),
        print( compteur final = ,compteur),
        print( approximation de pi = ,float(4*compteur/n))
        )$
```

Pour 10 points générés aléatoirement :

```
(% i9)  approx_pi_mc(10)$
```

point = $[-0.03521895029676081 , -0.28911988568142766]$ somme = 0.0848306827564475
compteur = 1
point = $[-0.49552224662923816 , -0.16887455455703426]$ somme = 0.27406091208132427
compteur = 2
point = $[0.22514777850764922 , 0.6836253106718426]$ somme = 0.5180350875581028
compteur = 3
point = $[0.011178889287623672 , 0.3569239900370289]$ somme = 0.12751970222965806
compteur = 4
point = $[-0.9272133682001562 , 0.2505950366836145]$ somme = 0.9225225025795405
compteur = 5
point = $[0.9874629157389636 , -0.9168215076911004]$ somme = 1.8156446869246778
compteur = 5
point = $[0.26525077873320235 , 0.18630436303395292]$ somme = 0.10506729130405719
compteur = 6
point = $[0.30931553222901487 , -0.0585786519326712]$ somme = 0.09910755694036778
compteur = 7
point = $[-0.896152892670965 , -0.9953442392493721]$ somme = 1.7938001616494494
compteur = 7
point = $[-0.36878460742817687 , 0.7204590524896228]$ somme = 0.6550633329901996
compteur = 8
compteur final = 8
approximation de pi = 3.2

4 Amélioration du programme avec un graphique

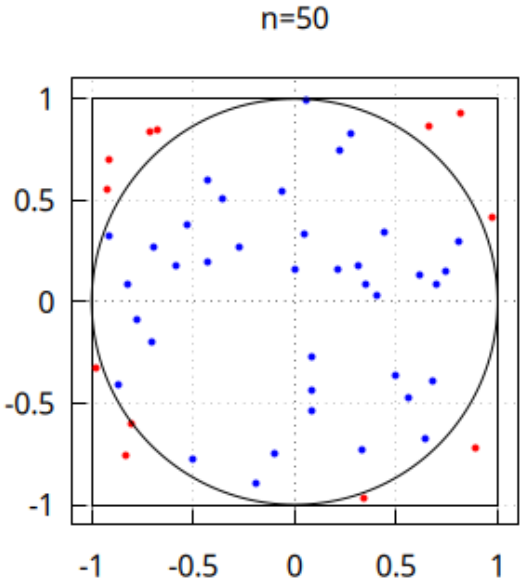
On supprime les affichages intermédiaires et on rajoute le graphique illustrant l'approximation. On colore les points de manière différente selon qu'ils soient à l'intérieur ou à l'extérieur du cercle.

```
-> approx_pi_mc2(n) :=block(
[compteur,i , pts_in,pts_out],
compteur :0 ,pts_in  : [],  pts_out : [],
for i :1 thru n do (
pt :[2*random(1.0)-1,2*random(1.0)-1],
if (pt[1]^ 2+pt[2]^ 2<1 or pt[1]^ 2+pt[2]^ 2=1 ) then
(compteur : compteur +1,
pts_in  : endcons(pt, pts_in)
) else pts_out  : endcons(pt, pts_out)
),
print( Nombre de points à l'intérieur du cercle  = ,compteur),
print( Approximation de pi = ,float(4*compteur/n)),
wxdraw2d(
title=concat( n= ,string(n)),
color=black,proportional_axes =xy,
grid=true, xaxis=true, yaxis=true,
xrange=[-1.1,1.1], yrange=[-1.1,1.1],
points_joined = true,
point_type = none,
points([[-1,-1], [-1, 1]]),  points([[-1,1], [1,1]]),
points([[1,1], [1,- 1]]),  points([[1,-1], [-1,- 1]]),
points_joined = false,
color=blue,
point_type=7, point_size=0.5,
points(pts_in),
color=red,
points(pts_out),
color=black,nticks=1000,
parametric(cos(t), sin(t), t, 0, 2*%pi)
)
)$
```

On teste avec 50 points par exemple :

```
(% i11) approx_pi_mc2(50)$
```

Nombre de points à l'intérieur du cercle = 38
Approximation de pi = 3.04



(%t11)

5 Approximation avec de grands nombres

On supprime les résulats intermédiaires, les listes et les graphiques pour parcourir rapidement la boucle et on calcule l'erreur par rapport à la valeur de π :

```
(% i12)  approx_pi_mc3(n) :=block(
         [compteur,i ],
         compteur :0 ,
         for i :1 thru n do (
         pt :[2*random(1.0)-1,2*random(1.0)-1],
         if (pt[1]^ 2+pt[2]^ 2<1 or pt[1]^ 2+pt[2]^ 2=1 ) then compteur : compteur +1
         ),
         print( Pour n= ,n, approximation de pi = ,float(4*compteur/n),
         écart à pi =  ,abs(float(%pi-4*compteur/n)))
         )$
```

```
(% i13)  approx_pi_mc3(100)$
```

Pour n= 100 approximation de pi = 3.0 écart à pi = 0.14159265358979312

```
(% i14)  for i :1 thru 20 do approx_pi_mc3(5000*i)$
```

Pour n= 5000 approximation de pi = 3.1328 écart à pi = 0.008792653589793087
Pour n= 10000 approximation de pi = 3.112 écart à pi = 0.029592653589793017
Pour n= 15000 approximation de pi = 3.1597333333333335 écart à pi = 0.018140679743540 39
Pour n= 20000 approximation de pi = 3.1442 écart à pi = 0.00260734641020699
Pour n= 25000 approximation de pi = 3.13328 écart à pi = 0.008312653589793051
Pour n= 30000 approximation de pi = 3.1401333333333334 écart à pi = 0.001459320256459673
Pour n= 35000 approximation de pi = 3.1325714285714286 écart à pi = 0.009021225018364554
Pour n= 40000 approximation de pi = 3.1392 écart à pi = 0.0023926535897929035
Pour n= 45000 approximation de pi = 3.1451555555555557 écart à pi = 0.0035629019657625882
Pour n= 50000 approximation de pi = 3.13824 écart à pi = 0.0033526535897929755
Pour n= 55000 approximation de pi = 3.147781818181818 écart à pi = 0.006189164592024898
Pour n= 60000 approximation de pi = 3.1317333333333335 écart à pi = 0.009859320256459636
Pour n= 65000 approximation de pi = 3.1424 écart à pi = 8.0734641020674410^{-4}

Pour n= 70000 approximation de pi = 3.1453142857142855 écart à pi = 0.003721632124492391
Pour n= 75000 approximation de pi = 3.1306133333333332 écart à pi = 0.010979320256459868
Pour n= 80000 approximation de pi = 3.1422 écart à pi = 6.0734641020676610^{-4}
Pour n= 85000 approximation de pi = 3.150964705882353 écart à pi = 0.009372052292559996
Pour n= 90000 approximation de pi = 3.1416 écart à pi = 7.34641020683213210^{-6}
Pour n= 95000 approximation de pi = 3.1441263157894737 écart à pi = 0.002533662199680542
Pour n= 100000 approximation de pi = 3.13812 écart à pi = 0.0034726535897933175

On remarque que la convergence est lente et qu'il faut générer beaucoup de points pour obtenirune approximation correcte.

6 Amélioration de la présentation des résultats

On définit une matrice pour présenter les résultats, et on limite le nombre de décimales

(% i44)
```
approx_pi_mc4(n) :=block(
[compteur,i ,entetes],
fpprintprec : 5, kill(a),
entetes : matrix([ n , approx pi , erreur ]),
compteur :0 ,
for i :1 thru n do (
pt :[2*random(1.0)-1,2*random(1.0)-1],
if (pt[1]^ 2+pt[2]^ 2<1 or pt[1]^ 2+pt[2]^ 2=1 ) then compteur : compteur +1,
a[i,1] :i,
a[i,2] :float(4*compteur/n),
a[i,3] :abs(float(%pi-4*compteur/n))),
resultat :genmatrix(a,n,3),
resultat : addrow(entetes, resultat),
return(resultat)
)$
```

(% i46)
```
approx_pi_mc4(10);
```

(%o46)

n	$approxpi$	$erreur$
1	0.4	2.7416
2	0.8	2.3416
3	1.2	1.9416
4	1.2	1.9416
5	1.6	1.5416
6	2.0	1.1416
7	2.4	0.74159
8	2.8	0.34159
9	3.2	0.058407
10	3.2	0.058407

On définit maintenant une fonction avec n0 comme nombre de départ, n1 comme arrivée, et un pas pour les itérations :

(% i47)

```
approx_pi_mc5(n0, n1, pas) := block(
[compteur, i, j, entetes, resultat, n_vals, nb_vals],
fpprintprec : 5,
kill(a),
entetes : matrix([ n , approx pi , erreur ]),

/* liste des valeurs de n : n0, n0+pas, n0+2*pas, ..., n1 */
n_vals : makelist(n0 + k*pas, k, 0, floor((n1-n0)/pas)),
nb_vals : length(n_vals),

for j : 1 thru nb_vals do (
compteur : 0,
for i : 1 thru n_vals[j] do (
pt : [2*random(1.0)-1, 2*random(1.0)-1],
if (pt[1]^ 2 + pt[2]^ 2 <= 1) then compteur : compteur + 1
),
a[j,1] : n_vals[j],
a[j,2] : float(4*compteur / n_vals[j]),
a[j,3] : abs(float(%pi - 4*compteur / n_vals[j]))
),

resultat : genmatrix(a, nb_vals, 3),
resultat : addrow(entetes, resultat),
return(resultat)
)$
```

(% i53)

```
approx_pi_mc5(1000,150000,10000);
```

(%o53)

n	$approxpi$	$erreur$
1000	3.108	0.033593
11000	3.1444	0.002771
21000	3.1398	0.0017831
31000	3.1468	0.0052461
41000	3.1571	0.015481
51000	3.137	0.004573
61000	3.1429	0.0012926
71000	3.1475	0.0059003
81000	3.1488	0.0072469
91000	3.1358	0.0057685
101000	3.1378	0.0038105
111000	3.139	0.0026017
121000	3.1428	0.0012503
131000	3.1419	2.699410^{-4}
141000	3.1459	0.0043364

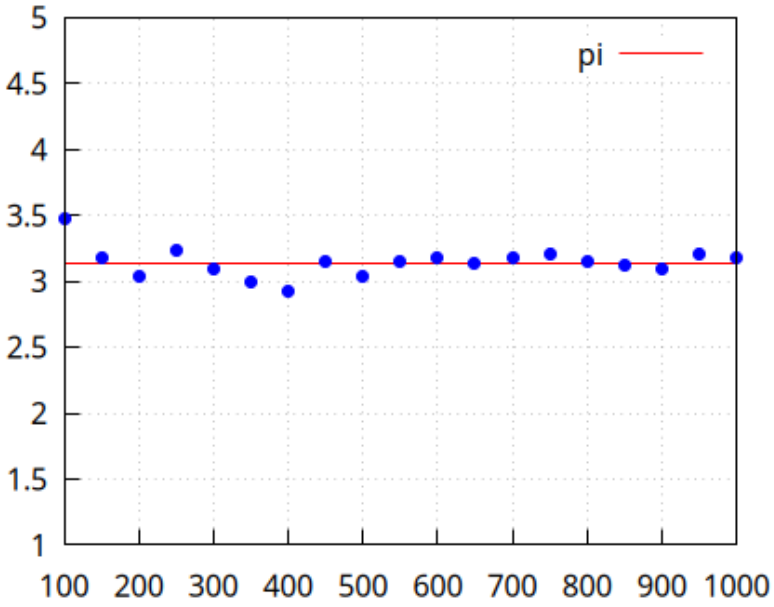
7 Etude graphique de la convergence vers Π

(% i80)
```
approx_pi_mc6(n0, n1, pas) := block(
[compteur, i, j, n_vals, nb_vals,pts],
fpprintprec : 5,
kill(a),
entetes : matrix([ n , approx pi , erreur ]),

/* liste des valeurs de n : n0, n0+pas, n0+2*pas, ..., n1 */
n_vals : makelist(n0 + k*pas, k, 0, floor((n1-n0)/pas)),
nb_vals : length(n_vals),

for j : 1 thru nb_vals do (
compteur : 0,
for i : 1 thru n_vals[j] do (
pt : [2*random(1.0)-1, 2*random(1.0)-1],
if (pt[1]^ 2 + pt[2]^ 2 <= 1) then compteur : compteur + 1
),
a[j,1] : n_vals[j],
a[j,2] : float(4*compteur / n_vals[j]),
a[j,3] : abs(float(%pi - 4*compteur / n_vals[j]))
),
pts : makelist([a[j,1], a[j,2]], j, 1, nb_vals),
wxdraw2d(
grid=true, xaxis=true, yaxis=true,
yrange=[1,5],
/* la droite y = pi */
key= pi ,
color = red,
explicit(%pi, x, n0, n1),
key= ,
/* les points de simulation */
color = blue,
point_size = 1,
point_type = 7,
points(pts)
)
)$
```

(% i81)
```
approx_pi_mc6(100,1000,50) ;
```



(%$t81$)

(%$o81$)