

LES DECIMALES DE π

BERNARD EGGER

La génération de suites de nombres pseudo aléatoires est un enjeu essentiel pour la simulation.

Si comme le dit B Ycard dans le cours écrit pour le logiciel SEL, « *Paradoxalement, le problème de démontrer qu'une suite particulière est aléatoire est indécidable en général* », le test du Chi-deux peut nous permettre de rejeter éventuellement une hypothèse. Nous allons l'appliquer aux 1250 premières décimales de π .

Bien entendu le nombre 1250 est « petit ». Il permet simplement de donner une image de ce qui est faisable. Il a l'avantage de permettre des traitements informatiques rapides tant sur Derive que sur Excel.

Nous partons de l'hypothèse que les décimales de π ont une distribution uniforme. C'est-à-dire que pour 1250, on en attend théoriquement 125 pour chacune des valeurs de 0 à 9. Qu'en est-il exactement ?

La première tâche est de saisir ces 1250 décimales.

Pour cela nous allons à nouveau utiliser le logiciel de calcul formel Derive.

Celui-ci permet d'afficher π avec bien plus de 1250 décimales.

```
π
3.1415926535897932384626433832795028841971693993751058209749445923078164062862089986280348~
2534211706798214808651328230664709384460955058223172535940812848111745028410270193852110~
5559644622948954930381964428810975665933446128475648233786783165271201909145648566923460~
3486104543266482133936072602491412737245870066063155881748815209209628292540917153643678~
9259036001133053054882046652138414695194151160943305727036575959195309218611738193261179~
3105118548074462379962749567351885752724891227938183011949129833673362440656643086021394~
9463952247371907021798609437027705392171762931767523846748184676694051320005681271452635~
6082778577134275778960917363717872146844090122495343014654958537105079277968925892354201~
9956112129021960864034418159813629774771309960518707211349999998372978049951059731732816~
0963185950244594553469083026425223082533446850352619311881710100031378387528865875332083~
8142061717766914730359825349042875546873115956286388235378759375195778185778053217122680~
6613001927876611195909216420198938095257201065485863278865936153381827968230301952035301~
852968995773622599413891249721775283479131515574857242454150695950829533168617278558890~
7509838175463746493931925506040092770167113900984882401285836160356370766010471018194295~
5596198946767837449448255379774726847104047534646208046684259069491293313677028989152104~
7521620569660240580381501935112533824300355876402474964732639141992726042699227967823547~
816360093417216412199245863150302861829745570674983850549458858692699569092721079750930~
2955
```

Ci-dessus par exemple, il y en a 1500.

Bien entendu, on pourrait envisager de saisir avec patience tous ces chiffres dans les 1250 premières lignes de la colonne A du tableur. Mais l'entreprise a de quoi rebuter.

C'est donc en utilisant les fonctions de texte des tableurs que nous allons opérer.

Pour comprendre la méthode commençons par ne prélever que les 10 premières

décimales.

```
PrecisionDigits := 11
Precision := Approximate
π = 3.1415926535
u := STRING(π)
u = 3.1415926535
v := DELETE(DELETE(u, 1), 1)
v = 1415926535
w := APPEND(a, v)
w
a1415926535
```

On a utilisé les fonctions de traitement des chaînes de caractères que contient Derive.

On copie dans le presse papier la chaîne a1415926535 retournée par Derive.

On colle cette chaîne dans la cellule D2 du tableur par exemple.

D
a1415926535

La lettre « a » sert tout simplement à empêcher le logiciel d'évaluer 1415926535 comme un nombre ce qu'il ne manquerait pas de faire.

Il ne reste plus qu'à extraire chacun des chiffres et à les ranger dans les cellules B1,..., B10, la colonne A contenant les entiers de 2 à 11.

La fonction **STXT** qui extrait une sous chaîne d'une chaîne donnée, à partir d'un certain rang, et d'une longueur donnée, remplit parfaitement ce rôle.

B1	=	=STXT(\$D\$1;A1;1)		
A	B	C	D	
2	1		a1415926535	
3	4			

Les résultats retournés ne sont pas des nombres mais des « lettres ». On peut les conserver sous cette forme, ou bien les transformer en chiffre à l'aide de **CNUM** :

=	=CNUM(B1)	
C		
1		a14

On peut même combiner les deux opérations avec en B1 : **=CNUM(STXT(\$D\$2 ;A1 ;1)**
C'est ce que nous ferons par la suite.

Pour clarifier la mise en page et en vue d'une utilisation pour un autre test, nous allons découper le problème en 5 étapes, en extrayant des chaînes de 260 décimales dont nous placerons les 250 premières valeurs en 5 colonnes.

La colonne A contient les entiers de 2 à 251.

Il reste maintenant à extraire les différentes chaînes dans Derive.

```
Precision := Approximate
PrecisionDigits := 1600
u := STRING(pi)
u
3.141592653589793238462643383279502884197169399375105820974944592307816406286208899862803~
48253421170679821480865132823066470938446095505822317253594081284811174502841027019385~
21105559644622948954930381964428810975665933446128475648233786783165271201909145648566~
92346034861045432664821339360726024914127372458700660631558817488152092096282925409171~
53643678925903600113305305488204665213841469519415116094330572703657595919530921861173~
81932611793105118548074462379962749567351885752724891227938183011949129833673362440656~
64308602139494639522473719070217986094370277053921717629317675238467481846766940513200~
05681271452635608277857713427577896091736371787214684409012249534301465495853710507922~
79689258923542019956112129021960864034418159813629774771309960518707211349999998372978~
0499510597317328160963185950244594534690830264252230825334468503526193118817101000313~
78387528865875332083814206171776691473035982534904287554687311595628638823537875937519~
57781857780532171226806613001927876611195909216420198938095257201065485863278865936153~
38182796823030195203530185296899577362259941389124972177528347913151557485724245415069~
59508295331168617278558890750983817546374649393192550604009277016711390098488240128583~
61603563707660104710181942955596198946767837449448255379774726847104047534646208046684~
25906949129331367702898915210475216205696602405808315019351125338243003558764024749647~
3263914199272604269227967823547816360093417216412199245863150302861829745557067498385~
05494588586926995690927210797509302955321165344987202755960236480665499119881834797753~
566369807426542527862551818417574672890977772793800
```

On commence par éliminer le « 3. ».

```
v := DELETE(DELETE(u, 1), 1)
```

On extrait les 250 premières décimales, puis on ajoute la lettre « a » devant :

```
w := v
VECTOR(k, k, 1, 250)
t := APPEND(a, w)
t
a141592653589793238462643383279502884197169399375105820974944592307816406286208998628034~
82534211706798214808651328230664709384460955058223172535940812848111745028410270193852~
11055596446229489549303819644288109756659334461284756482337867831652712019091
```

On copie le résultat dans le presse papiers et on le colle dans le tableur, en cellule G1 par exemple.

En cellule B1, on entre la formule : **=CNUM(STXT(\$G\$1 ;A1 ;1))**.

Puis on copie cette formule, on sélectionne la première plage et l'on fait **CTRL V**.

On retourne à Derive pour les 250 décimales suivantes :

La procédure est totalement identique.

```
w := v
VECTOR(k, k, 251, 500)
t := APPEND(a, w)
t
a456485669234603486104543266482133936072602491412737245870066063155881748815209209628292~
54091715364367892590360011330530548820466521384146951941511609433057270365759591953092~
18611738193261179310511854807446237996274956735188575272489122793818301194912
```

(remarque : l'instruction `t :=APPEND(a,w)` n'est en réalité pas à réécrire, puisqu'elle s'applique à la nouvelle valeur de `w`).

Et ainsi de suite... En définitive, on a :

	A	B	C	D	E	F	G
1	2	1	4	9	5	3	a1415926535897
2	3	4	5	8	1	8	a4564856692346
3	4	1	6	3	8	0	a9833673362440
4	5	5	4	3	7	9	a5187072113499
5	6	9	8	6	0	5	a3809525720107

Il ne reste plus qu'à comptabiliser le nombre d'occurrences de chaque chiffre.

	H
0	117
1	143
2	124
3	126
4	112
5	129
6	116
7	120
8	129
9	134

En colonne I, on fait apparaître les effectifs théoriques pour une distribution uniforme, c'est-à-dire 125 et l'on applique le test du Chi-deux.

=TEST.KHIDEUX(H7:H16;I7:I16)		
G	H	I
8	129	125
9	134	125
		0,71730697

Le résultat obtenu est très bon. On ne peut pas rejeter l'hypothèse que la répartition des 1250 premières décimales de π soit uniforme.

Mais bien sûr nous n'avons pas prouvé qu'il s'agit d'une suite « aléatoire » : nous aurions par exemple obtenu le même pourcentage avec un nombre dont les 117 premiers chiffres après la virgule auraient été des « 0 », les 143 suivants des « 1 », etc.

LE TEST DU POKER

Pour tester la qualité d'un générateur de nombres pseudo aléatoires, plusieurs moyens sont possibles. Le test d'uniformité est un des plus fréquent : nous l'avons utilisé pour la suite des décimales du nombre π .

Il consiste par exemple à vérifier que pour un grand nombre de nombres pseudo aléatoires compris entre 0 et 1, comme ceux retournés par la fonction **ALEA**, le premier chiffre après la virgule qui est compris entre 0 et 9 a pour chacune des valeurs possibles une fréquence « autour de 10% ».

Si par exemple on place dans la plage A1 :E1000 la formule matricielle **{ENT(10*ALEA)}** (rappelons que pour introduire une telle formule, on sélectionne la plage, puis en ligne de formule, on entre cette formule sans accolade et l'on tape sur **CTRL MAJ ENTRÉE**), et si l'on compte le nombre d'apparition de chaque chiffre à l'aide de l'instruction **NB.SI**, le test du Khi-deux appliqué par rapport aux valeurs théoriques d'une distribution uniforme, à savoir 500 pour chaque chiffre, ne permet pas de rejeter l'hypothèse d'une distribution uniforme

=TEST.KHIDEUX(G1:G10;H1:H10)					
D	E	F	G	H	
9	7	0	478	500	
4	3	1	542	500	
9	0	2	501	500	
7	1	3	468	500	
4	3	4	493	500	
3	7	5	522	500	
7	4	6	510	500	
4	9	7	481	500	
7	6	8	504	500	
7	3	9	501	500	
1	9				
1	1		0,478		

Mais ce test d'uniformité n'est pas suffisant pour considérer que l'on a un « bon générateur ». En effet il donnerait le même résultat si par exemple pour l'écran ci-dessus, tous les 0 correspondaient aux 478 premières valeurs, tous les 1 aux 542 valeurs suivantes...

Un test plus efficace est celui dit du « poker ».

On utilisera le livre d'Arthur Engel, *Les certitudes du hasard*, pour le décrire :

Les chiffres à tester sont regroupés par bloc de 5. Il existe 10^5 tels blocs et chacun a donc la probabilité 10^{-5} d'apparaître. Ces blocs de 5 chiffres sont regroupés en 7 catégories dont les probabilités sont calculées. On compare alors ces probabilités avec les fréquences observées.

Le tableau ci-dessous donne la description de ces 7 catégories, lointainement inspirées du poker :

N°	DESCRIPTION	TYPE	EXEMPLE	PROBABILITÉ
1	Chiffres distincts	abcde	34679	0,3024
2	Une paire	aabcd	05609	0,5040
3	Deux paires	aabbc	43354	0,1080
4	Un triplet	aaabc	67669	0,0720
5	Paire-triplet	aaabb	77676	0,0090
6	Quadruplet	aaaab	22212	0,0045
7	Quintuplet	aaaaa	77777	0,0001

On laisse au lecteur le soin de vérifier ces probabilités.

A l'aide de ce tableau théorique et du test du χ^2 , on peut examiner si les décimales de π remplissent bien ce test.

Nous allons reprendre les décimales, que nous avons déjà rangées en paquets de 5 puisque nous les avons écrites en 250 rangées de 5 colonnes.

Il suffit de compter combien de ces paquets contiennent 5 chiffres distincts, une paire,... Bien évidemment ce rangement n'est pas tout à fait aléatoire, mais il nous suffira dans un premier temps (créer un rangement en paquets de 5 plus aléatoire ne présente pas de difficulté majeure de programmation : il suffit de placer les 1250 décimales dans un tableau ; on désigne aléatoirement un indice du tableau et l'on range le contenu dans une cellule. On place alors la dernière case du tableau dans la case maintenant vide, et l'on recommence avec une décimale de moins jusqu'à ce que l'on ait épuisé les décimales).

Il faut donc apprendre au tableur à compter chacun des cas.

Le principe sera le suivant : dans un premier temps nous mettons à 0 les cellules qui contiendront le nombre d'occurrences de chaque type : paire, brelan... (ces valeurs seront rangées tout au long de l'exécution du programme dans un tableau **pok** à 7 cases).

Pour chacune des 250 lignes on place les cinq cellules dans un tableau nommé **table**, tableau que l'on trie de façon à ce que les cellules de même valeur soient placées côte à côte.

```

Sub poker()
Dim table(5) As Variant
Dim table2(5) As Variant
Dim resul(5) As Variant
Dim pok(7) As Variant

For i = 1 To 7
    pok(i) = 0
Next i

```

Appel de la procédure par le mot clé **sub**.
Zone de définition de tableaux avec leur dimension entre parenthèses. On ne fixe pas le type de contenu de chaque tableau (c'est ce que signifie **Variant**)

Initialisation du tableau **pok**.

```

For k = 1 To 250
For i = 1 To 5
    table(i) = Cells(k, i).Value
    resul(i) = 0
    table2(i) = 0
Next i
i = 1
Do While i <= 5
    u = table(i)
    j = i + 1
    Do While j <= 5
        If table(j) = u Then
            v = table(i + 1)
            table(j) = v
            table(i + 1) = u
            j = j + 1
            i = i + 1
        Else
            j = j + 1
        End If
    Loop

```

Pour chacune des 250 lignes, on procède de la même façon.

On range dans **table** les contenus de la ligne et l'on trie **table** de façon à ce que les cellules de même contenu soient contiguës.

Pour chaque valeur différente de **table**, on compte le nombre de fois où elle apparaît, à l'aide d'un tableau **table2**.

```

i = i + 1
Loop
i = 1
Do While i <= 5
    u = table(i)
    For j = 1 To 5
        If table(j) = u Then
            table2(j) = 1
        Else
            table2(j) = 0
        End If
    Next j
    s = table2(1) + table2(2) + table2(3) + table2(4) + table2(5)
    resul(s) = resul(s) + 1
    i = i + s
Loop

```

```

If resul(1) = 5 Then
    pok(1) = pok(1) + 1
End If
If resul(5) = 1 Then
    pok(7) = pok(7) + 1
End If
If resul(2) = 2 Then
    pok(3) = pok(3) + 1
End If
If resul(2) = 1 Then
    If resul(3) = 1 Then
        pok(5) = pok(5) + 1
    Else
        pok(2) = pok(2) + 1
    End If
End If
If resul(3) = 1 And resul(2) = 0 Then
    pok(4) = pok(4) + 1
End If
If resul(4) = 1 Then
    pok(6) = pok(6) + 1
End If
For i = 1 To 7
    Cells(i, 8).Value = pok(i)
Next i
Next k

```

Le tableau **resul** contient 5 cases : la première indique le nombre de valeurs qui n'apparaissent qu'une seule fois dans **table**, la deuxième le nombre de celles qui apparaissent deux fois, la troisième trois fois, etc.

En fonction des valeurs contenues dans **resul**, on peut savoir la structure de **table** et modifier le contenu de **pok**.

Affichage sur l'écran du tableur des résultats.

Et redémarrage de la boucle.

Au bout de quelques secondes, on obtient :

G	H
rien	78
une paire	121
deux paires	24
un brelan	24
un full	2
un carré	1
une quinte	0

En colonne I, on place les probabilités théoriques données plus haut et les effectifs qui correspondent pour 250 lots (le test du Khi-deux ne s'applique impérativement qu'à des effectifs).

I	J
0,3024	75,6
0,504	126
0,108	27
0,072	18
0,009	2,25
0,0045	1,125
0,0001	0,025

Pour appliquer le test du Khi-deux, il faut également que les effectifs de chaque modalités soient supérieurs ou égaux à 5. On procède donc à un regroupement des dernières lignes et l'on applique le test.

G	H	I	J
rien	78		75,6
paire	121		126
2 paires	24		27
autre	27		21,4
test khideux	0,5573		

On peut considérer que les 1250 décimales de π ont passées avec succès le test du poker. Mais est-ce bien sûr ? Notre rangement initial y est-il pour quelque chose ?

Montrons comment nous pourrions distribuer de façon plus aléatoire les 1250 décimales. On commence par renommer la feuille de travail qui contient les 1250 décimales (par exemple « poker et pi ») et une autre feuille, par exemple « poker 2 ». On introduit dans l'éditeur Visual Basic le programme suivant :

```
Sub tri()
Dim decimale(1250) As Variant
Sheets("poker et pi").Select
For k = 1 To 5
  For i = 1 To 250
    decimale(i + 250 * (k - 1)) = Cells(i, k)
  Next i
Next k
```

On entre dans le tableau **decimale** les 1250 décimales qui se trouvent dans la feuille « poker et pi ».

On active la deuxième fenêtre et on initialise les deux variables i et k

```
Sheets("poker 2").Select
i = 1
k = 1
```

Et enfin le corps du programme avec le remplacement des 1250 décimales en 5 colonnes et 250 lignes

```
Do While i <= 1250
  a = Int(Rnd() * (1250 - i + 1)) + 1
  b = decimale(a)
  decimale(a) = decimale(1250 - i + 1)
  q = Int(i / 250)
  r = i - q * 250
  If r = 0 Then
    r = 250
  End If
  Cells(r, k).Value = b
  If r = 250 Then
    k = k + 1
  End If
  i = i + 1
Loop
End Sub
```

Comme on peut le constater ci-dessous, si l'on fait fonctionner l'ensemble des programmes (**tri** puis **poker**) on trouve des résultats tout à fait différents à chaque exécution.

Il est difficile d'en conclure quelque chose concernant les décimales du nombre π , ne serait-ce que du fait du relativement petit nombre de décimales choisies. Toutefois, nous rencontrons une fois de plus la difficulté évoquée plus haut concernant la preuve statistique.

rien	91	0,3024	75,6
une paire	117	0,504	126
deux paires	30	0,108	27
un brelan	11	0,072	18
un full	1	0,009	2,25
un carré	0	0,0045	1,125
une quinte	0	0,0001	0,025
rien	91		75,6
une paire	117		126
deux paires	30		27
autres	12		21,4
test khideux		0,0413	

rien	77	0,3024	75,6
une paire	118	0,504	126
deux paires	34	0,108	27
un brelan	17	0,072	18
un full	2	0,009	2,25
un carré	2	0,0045	1,125
une quinte	0	0,0001	0,025
rien	77		75,6
une paire	118		126
deux paires	34		27
autres	21		21,4
test khideux		0,5018	

Même en tenant compte de l'inévitable fluctuation d'échantillonnages, il semble difficile de conclure sur le comportement des 1250 premières décimales de π vis à vis du test du poker.