Les nombres de Bernoulli, quelques séries, quelques premiers et l'Inverseur version 2003.

Conférence du Séminaire de Combinatoire, vendredi le 21 février 2003.

Au programme :

Résultats pour B(n), B(750000).
Conjecture de Agoh-Giuga, n=49999.
La formule :

$$\frac{-1}{2}\left[\tfrac{n}{4}+\tfrac{1}{2}\right]+\frac{1}{2}\left[\tfrac{n}{4}\right]+\left\{\sum_{p=2}^{n+1}-\frac{\left[\frac{n}{p-1}\right]}{p}\right\}=\left\{\sum_{k=2}^{n}\{B_k\}\right\}$$

p est premier, n est pair.
[ ] est la fonction plancher,
{ } est la partie fractionnaire.

Inverseur 2003, programme interactif Maple.
Algorithme pour les séries plus rapide.

On commence par le calcul des B(n), la formule connue est celle qui utilise les binomiaux mais…

On peut la refaire avec ceci,

on prend la formule $\frac{x}{e^x-1}$ qu'on développe en série de Taylor en x=0. En collectant les termes de même degré on retombe sur ses pattes avec une formule de récurrence : $(B+1)^{p+1}-B^{p+1}=0$ ou on remplace l'exposant par l'indice.

$$\frac{x}{e^x-1}=\sum_{k=0}^{\infty}B_k\frac{x^k}{k!}$$

Donc si on isole un seul nombre de Bernoulli ont est forcé d'évaluer tous les précédents. La formule exponentielle donne la formule avec les binomiaux. Cette formule de récurrence est élégante mais peu pratique : on ne peut guère de rendre à plus de n=10000. B(10000) est bien assez grand pour mesurer le diamètre d'un abat-jour (référence à pi), en termes de calcul ce n'est pas assez rapide.

D'autres formules existent comme

$$\frac{B_{4n+2}}{8n+4}=\sum_{k=0}^{\infty}\frac{k^{4n+1}}{e^{2k\pi}-1}$$

qui est explicite oui mais qui demande pas mal de calculs si n est de l'ordre de 250000 par exemple.

L'une des définitions est la suivante (simplifiée).

$$B(n) = \frac{2n!}{2^n \pi^n} \zeta(n) \quad (1)$$

On suppose ici n toujours pair et le résultat positif. Si n=10000, la partie gauche du membre de droite donne plusieurs milliers de décimales qui sont exactes. Le résultat vrai a 27678 décimales. Il n'y a que $\zeta(n)$ à évaluer exactement. Déjà avec les quelques premiers termes de $\zeta(n)$ on a une très bonne approximation. Cette formule peut se trouver si on fait le ratio B(n+2)/B(n) qui tend assez vite sur une constante assez facile à déterminer, le reste de l'expression se devine en faisant du Bootstrapping simple. (amorçage simple).

C'est là qu'intervient la formule de V-S-C (Von-Staudt-Clausen). Elle permet d'avoir la partie fractionnaire de B(n) même si n est très grand.

Par exemple : la partie fractionnaire de B(16) est

$\frac{1}{2}+\frac{1}{3}+\frac{1}{5}+\frac{1}{17}=\frac{47}{510}$ au signe près.

Il y a plusieurs formulations : c'est la somme des inverses des nombres+1 qui divisent n et qui sont premiers.
--on pose la liste des diviseurs de n
--on pose la liste +1
--si p est premier et fait partie de la liste+1 on fait la somme des inverses.
--on applique une règle si n = 0 mod 4.

En pratique, c'est très rapide, on est limité par la taille de n, lorsque n ~ 10^40.

Dans la formule (1), il suffit donc d'évaluer $\zeta(n)$ jusqu'à ce que ce soit exact à l'entier près.

Mais, on remarque que $\zeta(n)$ peut aussi s'écrire, un certain L. Euler a trouvé ça il y a quelques temps…

$$\zeta(n) = \prod_{\substack{p \\ \text{premier}}} \frac{1}{1-p^{-n}}$$

Étant donné qu'il y a moins de premiers que d'entiers il est facile d'imaginer qu'en utilisant les premiers on arrive plus vite. (Détails informatiques en moins)

On optimisera ensuite le calcul de l'inverse,
avoir une valeur de π avec suffisamment de précision et il suffit de tronquer au bon endroit. C'est simple pourtant mais c'est grâce aux valeurs des constantes facilement disponibles que c'est maintenant possible…
Il y a beaucoup de progrès comme ça qui arrivent grâce au simple fait que quelque chose soit automatisé plutôt que manuel…

…qui donne les résultats suivants avec un pc, 1600 Mhz et 768 Mb de mémoire.

B(10000) : 27678 chiffres en 3.2 secondes.
B(400000) : 2000000 de chiffres en 16023 secondes
B(750000) : 3400000 chiffres en 75000 secondes.

Un certain Kellner se serait rendu à B(1000000) en décembre 2002 (le résultat est incertain).

Mais pourquoi s'acharner à calculer les Bernoulli aussi loin ?

Les nombres de Bernoulli fournissent des premiers probables intéressants comme 6*B(2*2153)/2153 et 2153 est un premier irrégulier et on sait pas exactement pourquoi c'est premier. En calculant facilement B(n) on peut chercher plus loin. Qui dit Bernoulli dit aussi nombres d'Euler, E(454) et E(510) sont premiers et le TNP n'explique pas exactement ces nombres, 1 chance sur 2500.
La conjecture de Agoh : n*B(n-1) = -1 mod n si et seulement si n est premier a été vérifiée jusqu'à n = 49999. Agoh est équivalente à la conjecture de Giuga.

La formulation de Giuga est plus intéressante en fait :

$$n \mid \sum_{k=1}^{n-1} k^{n-1} + 1 = s(n)$$

et ce si et seulement si n est premier. Le lien est avec les nombres de Carmichael est que s(n) = -1 mod n si et seulement si n est un nombre de Carmichael. Par exemple 561 en est un. Il est impair, composé et $2^{560} = 1 \mod 561$. Ces nombres font planter les programmes de primalité probabilistes comme les premières versions de isprime() de Maple.
Une autre formulation est : le nombre 561 est pseudo-premier en n'importe quelle base, c'est à cause de l'existence de ces nombres que les tests de pseudo-primalité prennent plusieurs bases différentes.

Rappel : $a^{(n-1)} = 1 \mod n$ signifie <souvent> que n est premier. En répétant le test avec différents a on pseudo-prouve que n est premier. Il doit (a) être sans facteurs carrés et premier à n (évidemment).

Trouvé récemment : ζ(4305)/ζ(-1) est premier, probablement. Y-en a t-il d'autres ? et pourquoi 2153 est il irrégulier aussi…?

On peut encore améliorer le calcul de ζ(n) si on considère la formule sommatoire d'Euler-Maclaurin.

On peut encore améliorer le calcul si on prend les premiers sur disque puisque la fonction ithprime de maple est assez lente et boguée si la les premiers sont dans [ ].

On ne gagne pas en calcul si on garde le résultat de B(n) pour calculer B(n+2) même si on a une bonne approximation car le terme principal de B(n) donne déjà au-delà de 1/3 des décimales.

Une conséquence de la formule de V-S-C.

Rappel : on avait

$$\frac{-1}{2}\left[\tfrac{n}{4}+\tfrac{1}{2}\right]+\frac{1}{2}\left[\tfrac{n}{4}\right]+\left\{\sum_{p=2}^{n+1}-\frac{\left[\frac{n}{p-1}\right]}{p}\right\}=\left\{\sum_{k=2}^{n}\{B_k\}\right\}$$

Se justifie par la décomposition de { } de B(n) par la formule de V-S-C. En effet pour chaque k on a une somme simple d'inverses de premiers.

Par exemple $\{ B(100) \} = \frac{-1}{2}+\frac{-1}{3}+\frac{-1}{5}+\frac{-1}{11}+\frac{-1}{101}$

Donc pour les diviseurs de 100 on retrouvera une somme d'inverses qui parcourera les mêmes premiers, en collectant les termes semblables (premiers) on peut expliquer.
Principe simple : les sommes sont indépendantes, on a qu'à raisonner avec les diviseurs de 100 (ou de n).

On retrouvera p, [n/(p-1)] fois.

La règle associée à la formule de V-S-C qui dit que la somme est si n = 0 mod 4 alors on a **–somme** sinon on a **1-somme**, en somme. En fait les termes ne font que changer le signe puisque 1/p est équivalent à (p-1)/p mod 1. Ce qui explique le terme avec [n/4] et [n/4 + ½] qu'on retrouve une fois sur 2.
Le terme [n/4] et [n/4 + ½] est [-1/2, 0, -1/2, 0, -1/2, 0…].

Il est surprenant de voir une formule aussi simple et on se demande : ne serait-il pas possible d'évaluer des séries en ne prenant que des sommes d'inverses de premiers ?

Il y a moins de premiers que d'entiers (La Palisse).

Évaluer la n'ième décimale (en base a) de 1/p est facile avec le petit théorème de Fermat car

$$a^p \equiv a \bmod p$$

Par exemple la série harmonique peut s'écrire (avec les nombres premiers) : est la somme our n=100 des fractions

```
-51/64,-62/81,-23/49,-10/11,-4/13,-15/17,14/19,
4/23,26/29,7/31,20/37,22/41,23/43,25/47,1/53,1/5
9,1/61,1/67,1/71,1/73,1/79,1/83,1/89,1/97
```

(trivialement),
la somme à partir de 97 jusqu'à 53 est simple puisque le numérateur est toujours 1, ensuite le numérateur est 3/2… jusqu'à 2.

En ré-écrivant la série harmonique uniquement en termes des premiers on obtient qu'il ne reste que
4 types de sommes : **H(2)**, **H(3)**, **1/p** et **H*(2^k)**.

Mais qui dit série harmonique dit aussi une série du type $\sum_{k=1}^{\infty} \frac{1}{k 2^k}$

En procédant par récurrence on peut raisonner de la façon suivante pour cette somme.

Si on la somme de façon classique (de k=1 à n) on a
n termes à évaluer et chaque terme prend log(n) pour être évalué : n*log(n) opérations pour évaluer.

Si on utilise que les premiers on a n/log(n) termes qui prennent une certaine quantité à évaluer.
Si on peut réussir à le faire en log(n) * k étapes on aura une évaluation de la somme en n*k étapes (!).

Par exemple la somme jusqu'à 100 donne (ré-écrite avec les premiers) :

```
-16/81,-7/25,-44/49,-5/11,-11/13,-13/17,-8/19,
-21/23,-9/29,20/31,33/37,16/41,6/43,30/47,27/53,
30/59,31/61,34/67,36/71,37/73,40/79,42/83,45/89,
49/97
```

On ignore le terme avec les puissances de 2 (assez costaud) étant donné qu'en base 2 il est évaluable facilement.

On applique alors le même raisonnement qu'avec la série harmonique : les termes de 97 à 53 sont simples, c'est (p-1)/2. Les termes entre 47 et 37, la première moitié et le premier tiers sont (formule harmonique avec un terme en 2^k).

Finalement, mis-à-part la série harmonique, le fait d'avoir des termes avec 2^k ne fait que déplacer le point décimal (en base 2).

En examinant la série harmonique sommée avec les premiers on calcule que le nombre d'opérations est en fait le même c'est lorsqu'on a à faire avec d'autres séries qu'il y aurait un gain.

Pour avoir les nombres premiers on triche en fait on les enregistrant sur disque par tranche, la plus grande possible à l'avance on peut s'en servir pour toutes les séries ou bien on peut le faire (sur la mouche) à mesure en y allant à l'envers à partir de la fin.

En examinant les séries connues on imagine qu'il serait possible d'évaluer par exemple la somme

$$\sum_{n=1}^{\infty} \frac{n2^{n}}{\binom{2n}{n}} = \pi + 3$$

En effet la somme pour 100 nous donne le même comportement des termes avec les premiers à partir de 97.

Si ça marche il nous faut :

1) Une liste des premiers disponible (extrèmement facile à générer même pour jusqu'à 10^15).
2) La règle pour chaque terme en termes des premiers et de la série harmonique.
3) Une procédure récursive.

Le gain se trouverait dans le fait que n/log(n) << n et l'application du petit théorème de Fermat aux premiers et l'évaluation au n'ième chiffre facilement.

Inverseur : version 2003.

Le nouveau programme Maple permet d'accéder l'inverseur à partir d'une session Maple et de passer par le fil du téléphone grâce aux 'package' Sockets.
Le programme est portable, il fonctionne sur la plupart des machines grand public et unix.
Les calculs avancés sont faits à partir de la station de l'usager, aucune charge sur les serveurs du LaCIM sauf pour les requêtes (lookup).
On peut ajuster la profondeur de recherche par la précision numérique demandée (de 16 à …).
À la limite, l'inverseur pourrait ne pas avoir de page principale mais le jaune et bleu est joli finalement.

Il y a 6 fonctions, x peut être soit réel flottant ou une expression Maple évaluable en point flottant.

pilook(x) : fait une requête à l'inverseur et retourne la plus courte réponse (s'il y a lieu) avec x.
pismart(x) : fait une requête avancée de pilook 140 fois avec x, ça prend 10-15 secondes à exécuter (internet).
pidev(x) : développe x de 17 façons différentes et essaye voir si la série tronquée des termes ne serait pas une fraction rationelle, comme avec la suite de Fibonacci.
pialg(x) : teste si x est algébrique de degré 2-5 selon la précision, peut aller jusqu'à 100 décimales et de degré 12 facilement si on augmente la précision courante (Digits).
pilll(x) : tente de trouver une combinaison linéaire de constantes dont x serait solution, 30 constantes sont essayées par groupe de 5.
piyves(x) : retourne un version du dessin en ASCII de Yves Chiricota qui ne demande aucun calcul mais qui est amusant. On peut appeler la fonction sans argument.

```
       Plouffe's Inverter

  1XS1?
 ?XXXXXXXSSXSSOIOXXXXXXXSSIIOSOC
  ;;?1IC1?;;;;          :;;:
 SXXXXXXXXXXXXXXXXXSXXXSOSOSOCIII1
 :?!!!!;!?!;::  ::;;;;;;;;!!!;!?
 ?IXXXXXXXXXXXSXSSSSSSSSSSOSSSXXX=
?XXXXXXXXXXXXXXXXXXSSSSSCXX1IIICI?
     =X               :?  S? :
     11               ;1  X:  ::
     I=                =  X  ;  :::
  ??;OI?=?;;           ?  C
 :;;;S=!!!?!:          ?  ??       ;??
     I                 :  :S          !!
     ?                  !  O!
     !                  :? :C
     ?                   !; !:
     !                    !  ?
    ;;                     =; :
    !:;                     ;  :
    !;?                         !
    ? I                          !
    ?:XI                          :!
    ;?XX     ???!;!!=I?         ?!   ;!
    ; XXC    !!?!;!!!!;!          ?O=?!!
    ; OXX:                      :;; ;1XXX
      ;C?      :::                !SSO?!?
              ?OII1CCOOO=           ?CXXX
              !!!;!;;;;;:              !?
        YC

   Copyright (c) 1986-2003, Simon Plouffe
```

Exemple d'exécution en Maple.

```
read inverter;

#### on tape une valeur :

.9182739187231231712;
pilook(%);

> int(1.0/(1+x^3),x=0..1);
 0.83564888482647210533371O3

> pilook(%);
"K = .83564888482647210533371O3 ... no match found"

%;
 0.83564888482647210533371O3

> pismart(%);
"K*11/12 = .76601144424266096555901l ... no match
found"
"K*9/10 = .752083963438248948003393 ... no match found"
"K*8/9 = .742798976235307602966314 ... no match found"
"K*7/8 = .731192742231630921669965 ... no match found"
"K*6/7 = .716270441369760902860374 ... no match found"
"K*5/6 = .696374040220600877780919 ... no match found"
"K*4/5 = .668519078611776842669682 ... no match found"
"K*7/9 = .649949104205894152595525 ... no match found"
"K*3/4 = .626736636198540790002827 ... no match found"
"K*5/7 = .596892034474800752383645 ... no match found"
"K*7/10 = .584954193785304737335972 ... no match found"
"Sum(1/P(n),n=1..infinity)    P(n)=(2nd    degree    polynomials)
5570992321764807   =   sum(1/(18*n^2-21*n+5),n=1..inf)   K*2/3   =
.557099232176480702224735"
"K*5/8 = .522280530165450658335689 ... no match found"
"Psi(a/b)-Psi(c/d) a/b and c/d = k/120,k=1..119 5013893089588326
= Psi(1/6)-Psi(2/3) K*3/5 = .501389308958832632002262"
"K*7/12 = .487461828154420614446643 ... no match found"
"K*4/7 = .477513627579840601906916 ... no match found"
"K*5/9 = .464249360147067251853946 ... no match found"
"K*1/2 = .417824424132360526668552 ... no match found"
"K*4/9 = .371399488117653801483157 ... no match found"
"K*3/7 = .358135220684880451430187 ... no match found"
"K*5/12 = .348187020110300438890460 ... no match found"
"K*2/5 = .334259539305888421334841 ... no match found"
"K*3/8 = .313368318099270395001414 ... no match found"
"Sum(1/P(n),n=1..infinity)    P(n)=(2nd    degree    polynomials)
2785496160882403   =   sum(1/(36*n^2-42*n+10),n=1..inf)   K*1/3   =
.278549616088240351112368"
"K*2/7 = .238756813789920300953458 ... no match found"
"K*1/4 = .208912212066180263334276 ... no match found"
"Sum(1/P(n),n=1..infinity)    P(n)=(2nd    degree    polynomials)
1856997440588269   =   sum(1/(54*n^2-63*n+15),n=1..inf)   K*2/9   =
.185699744058826900741578"
"K*1/6 = .139274808044120175556184 ... no match found"
"K*1/7 = .119378406894960150476729 ... no match found"
"K*1/8 = .104456106033090131667138 ... no match found"
"K*1/9 = .928498720294134503707892e-1 ... no match found"
"K*1/12 = .696374040220600877780919e-1 ... no match found"
"K+11/12 = 1.75231551493138772000377 ... no match found"
"K+9/10 = 1.73564884826472105333710 ... no match found"
"K+8/9 = 1.72453773715360994222599 ... no match found"
"K+7/8 = 1.71064884826472105333710 ... no match found"
"K+6/7 = 1.69279170540757819619425 ... no match found"
"K+5/6 = 1.66898218159805438667044 ... no match found"
"K+4/5 = 1.63564884826472105333710 ... no match found"
"K+7/9 = 1.61342662604249883111488 ... no match found"
"K+3/4 = 1.58564884826472105333710 ... no match found"
"K+5/7 = 1.54993456255043533905139 ... no match found"
"K+7/10 = 1.53564884826472105333710 ... no match found"
"K+2/3 = 1.50231551493138772000377 ... no match found"
"K+5/8 = 1.46064884826472105333710 ... no match found"
"K+3/5 = 1.43564884826472105333710 ... no match found"
"K+7/12 = 1.41898218159805438667044 ... no match found"
"K+4/7 = 1.40707741969329248190853 ... no match found"
"K+5/9 = 1.39120440382027660889266 ... no match found"
"K+1/2 = 1.33564884826472105333710 ... no match found"
"K+4/9 = 1.28009329270916549778155 ... no match found"
"K+3/7 = 1.26422027683614962476567 ... no match found"
"K+5/12 = 1.25231551493138772000377 ... no match found"
"K+2/5 = 1.23564884826472105333710 ... no match found"
"K+3/8 = 1.21064884826472105333710 ... no match found"
"K+1/3 = 1.16898218159805438667044 ... no match found"
"K+2/7 = 1.12136313397900676762282 ... no match found"
"K+1/4 = 1.08564884826472105333710 ... no match found"
"K+2/9 = 1.05787107048694327555932 ... no match found"
"K+1/6 = 1.00231551493138772000377 ... no match found"
"K+1/7 = .97850599112186391047996O ... no match found"
"K+1/8 = .96064884826472105333703 ... no match found"
"K+1/9 = .94675995937583216444821l ... no match found"
"K+1/12 = .91898218159805438667043O ... no match found"
"K-11/12 = .810178184019456133295l4e-1 ... no match found"
"K-9/10 = .643511517352789466628971e-1 ... no match found"
"K-8/9 = .532400406241678355517l6e-1 ... no match found"
"K-7/8 = .393511517352789466628971e-1 ... no match found"
"K-6/7 = .214940088781360895200l0e-1 ... no match found"
"K-5/6 = .231551493138772000377l0e-2 ... no match found"
"K-4/5 = .356488482647210533371O3e-1 ... no match found"
"K-7/9 = .578710704869432755593l5e-1 ... no match found"
"K-3/4 = .856488482647210533371O3e-1 ... no match found"
"K-5/7 = .121363133979006767622817 ... no match found"
"K-7/10 = .135648848264721053337103 ... no match found"
"K-2/3 = .168982181598054386670436 ... no match found"
"K-5/8 = .210648848264721053337103 ... no match found"
```

```
"K-3/5 = .23564884826472105333710 3 ... no match found"
"K-7/12 = .25231551493138772000377 0 ... no match found"
"K-4/7 = .26422027683614962476567 4 ... no match found"
"K-5/9 = .28009329270916549778154 7 ... no match found"
"K-1/2 = .33564884826472105333710 3 ... no match found"
"K-4/9 = .39120440382027660889265 9 ... no match found"
"K-3/7 = .40707741969329248190853 2 ... no match found"
"K-5/12 = .41898218159805438667043 6 ... no match found"
"K-2/5 = .43564884826472105333710 3 ... no match found"
"K-3/8 = .46064884826472105333710 3 ... no match found"
"K-1/3 = .50231551493138772000377 0 ... no match found"
"K-2/7 = .54993456255043533905138 9 ... no match found"
"K-1/4 = .58564884826472105333710 3 ... no match found"
"K-2/9 = .61342662604249883111488 1 ... no match found"
"K-1/6 = .66898218159805438667043 6 ... no match found"
"K-1/7 = .69279170540757819619424 6 ... no match found"
"K-1/8 = .71064884826472105333710 3 ... no match found"
"K-1/9 = .72453773715360994222599 2 ... no match found"
"K-1/12 = .75231551493138772000377 0 ... no match found"
"2*K = 1.6712976965294421066742 1 ... no match found"
"Mixed constants with 5 operations 2506946544794163 =
(Pi+sr(3)*ln(2))/sr(3) 3*K = 2.5069465447941631600113 1"
"7*K = 5.8495419378530473733597 2 ... no match found"
"ln(K) = .17954679209895017647151 0 ... no match found"
"ln(1+K) = .60739801479086606858996 2 ... no match found"
"exp(K-1) = .84844403696844075348403 2 ... no match found"
"exp(1-K) = 1.17862811974385602922406 ... no match found"
"exp(K) = 2.30631000815574690233048 ... no match found"
"cos(K) = .67069655245405054248251 6 ... no match found"
"sin(K) = .74173184812724000306384 5 ... no match found"
"sin(Pi*K) = .49368700711284300285018 3 ... no match found"
"cos(Pi*K) = .86963966043871510075724 7 ... no match found"
"exp(-K) = .43359305404031756036207 9 ... no match found"
"K^(1/2) = .91413830915497740851876 1 ... no match found"
"K^(1/3) = .94190681604041662661908 2 ... no match found"
"K^(3/2) = .76389862520001638099003 2 ... no match found"
"K^(2/3) = .88718845010339524818611 4 ... no match found"
"1/K^(1/2) = 1.09392636758040749219060 ... no match found"
"1/K^(1/3) = 1.06167614775715842313823 ... no match found"
"1/K^(3/2) = 1.30907422400212294041996 ... no match found"
"1/K^(2/3) = 1.12715624271647968426528 ... no match found"
"1/(1-K) = 6.08453296153776868349910 ... no match found"
"1/(1+K) = .54476650092708190888795 6 ... no match found"
"(1+K)^2 = 3.36960669413559689726927 ... no match found"
"(K-1)/(1+K) = .89533001854163817775909 5e-1 ... no match found"
"(1+K)/(K-1) = 11.169065923075537366998 2 ... no match found"
"trunc(K)+1-K = .16435115173527894666289 7 ... no match found"
"1/K = 1.19667489768766480796016 ... no match found"
"K^2 = .69830899760615479059507 1 ... no match found"
"1/K^2 = 1.43203081075578303594539 ... no match found"
"K^3 = .58354110958247510187153 0 ... no match found"
"abs(K-1) = .16435115173527894666289 7 ... no match found"
"K-2 = 1.16435115173527894666290 ... no match found"
"K-3 = 2.16435115173527894666290 ... no match found"
"1+K = 1.83564884826472105333710 ... no match found"
"K+2 = 2.83564884826472105333710 ... no match found"
"1-1/K = .19667489768766480796016 ... no match found"
"K/Pi = .26599528978075911930186 9 ... no match found"
"arctan(K) = .69610325434328225917135 9 ... no match found"
"GAMMA(K) = 1.12646782232185819118302 ... no match found"
"Psi(K) = .88573870124687756729078 8 ... no match found"
"Pi*K = 2.62526828268921950069389 ... no match found"
```

On recoupe la réponse comme étant une combinaison linéaire en Pi*sqrt(3) et log(2), ce que Maple évalue symboliquement facilement.

Comme on le voit l'algorithme est assez bête mais ça marche. Les variations essayées sont **x op F12** ou F12 est l'ensemble de Farey d'ordre 12 et **op** est une opération (+, ., /,*, exponentiation). Les autres sont des fonctions usuelles avec x et leur inverses si c'est faisable.

Le nom pilll vient de Plouffe Inverter Lentra-Lenstra-Lovasz du nom de l'algorithme duquel la programme est construit.

Pour l'algorithme LLL (pilll) il y a une règle simple pour le nombre de coefficients nécessaires selon la précision en vigueur qui rend le calcul automatique :

Nombre de coefficients = [sqrt(Digits)] donne une règle du pouce.

En d'autres mots 100 décimales de précision permettent de tester avec 10 nombres à la fois (donc 9 inconnues d'un coup).

Références

Conjecture de Agoh :
http://mathworld.wolfram.com/AgohsConjecture.html

Conjecture de Giuga :
http://mathworld.wolfram.com/GiugasConjecture.html

Résultats avec les nombres de Bernoulli et formules connues :
http://www.lacim.uqam.ca/%7Eplouffe/ber750000.txt
http://numbers.computation.free.fr/Constants/constants.html
http://mathworld.wolfram.com/BernoulliNumber.html

Programme pour l'inverseur :
http://www.lacim.uqam.ca/%7Eplouffe/inverter.txt

Description générale de l'Inverseur :
http://www.lacim.uqam.ca/%7Eplouffe/articles/Certitude_sans_demonstration.pdf
http://www.sciencenews.org/sn_arch/11_9_96/mathland.htm

Pour réussir à craquer quelle est l'expression asymptotique de x/(exp(x)-1) il suffit d'utiliser la méthode de craquage suivante.

On développe x/(exp(x)-1) en série de Taylor,

On récupère un terme sur 2,

On met en valeur absolue,

On récupère les coefficients qu'on multiplie par n! pour s'apercevoir que les coefficients croissent rapidement et ne sont pas du type c^n

On voit que ce n'est pas constant on fait l'hypothèse que a(n+1)/a(n) pourrait l'être,
Et ça ne l'est pas, on fait alors l'hypthèse que ça croît comme un polynôme et on effectue les premières différences suivies des secondes différences et on récupère la constante = .20264236728467554288... qui une fois passée dans l'inverseur de Plouffe trouve que c'est 2/Pi^2 et qui nous permet de trouver le terme principal du développement asymptotique de $\frac{x}{e^x - 1}$ qui est environ :

(1) $$B(n) = \frac{2n!}{2^n \pi^n}$$