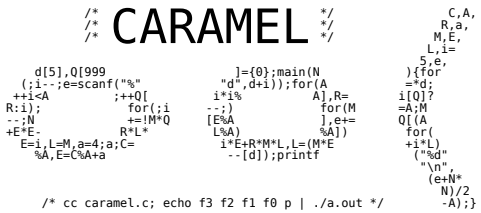
Peut-on calculer sur ordinateur ?

Paul Zimmermann



Bordeaux, 7 octobre 2010

(corrigé depuis, mais essayez 5.0 − 4.9 − 0.1)

Qu'est-ce qu'un calcul ?

données

résultat

Qu'est-ce qu'un résultat correct ?

Peut-on déduire quelque chose du résultat ?

Comment prouver que $\pi < 22/7$?

On peut commencer par calculer $22/7 - \pi$:

```
          GP/PARI CALCULATOR Version 2.4.3 (development)
   amd64 running linux (x86-64/GMP-4.2.4 kernel) 64-bit version
compiled: Jan 23 2009, gcc-4.3.2 20081105 (Red Hat 4.3.2-7) (GCC)
          (readline v5.2 enabled, extended help enabled)

? 22/7-Pi
%1 = 0.0012644892673496186802137595776399729593
```

Est-ce une preuve ?

L'utiliseriez-vous dans la preuve d'un théorème ?

L'accepteriez-vous en tant que relecteur ?

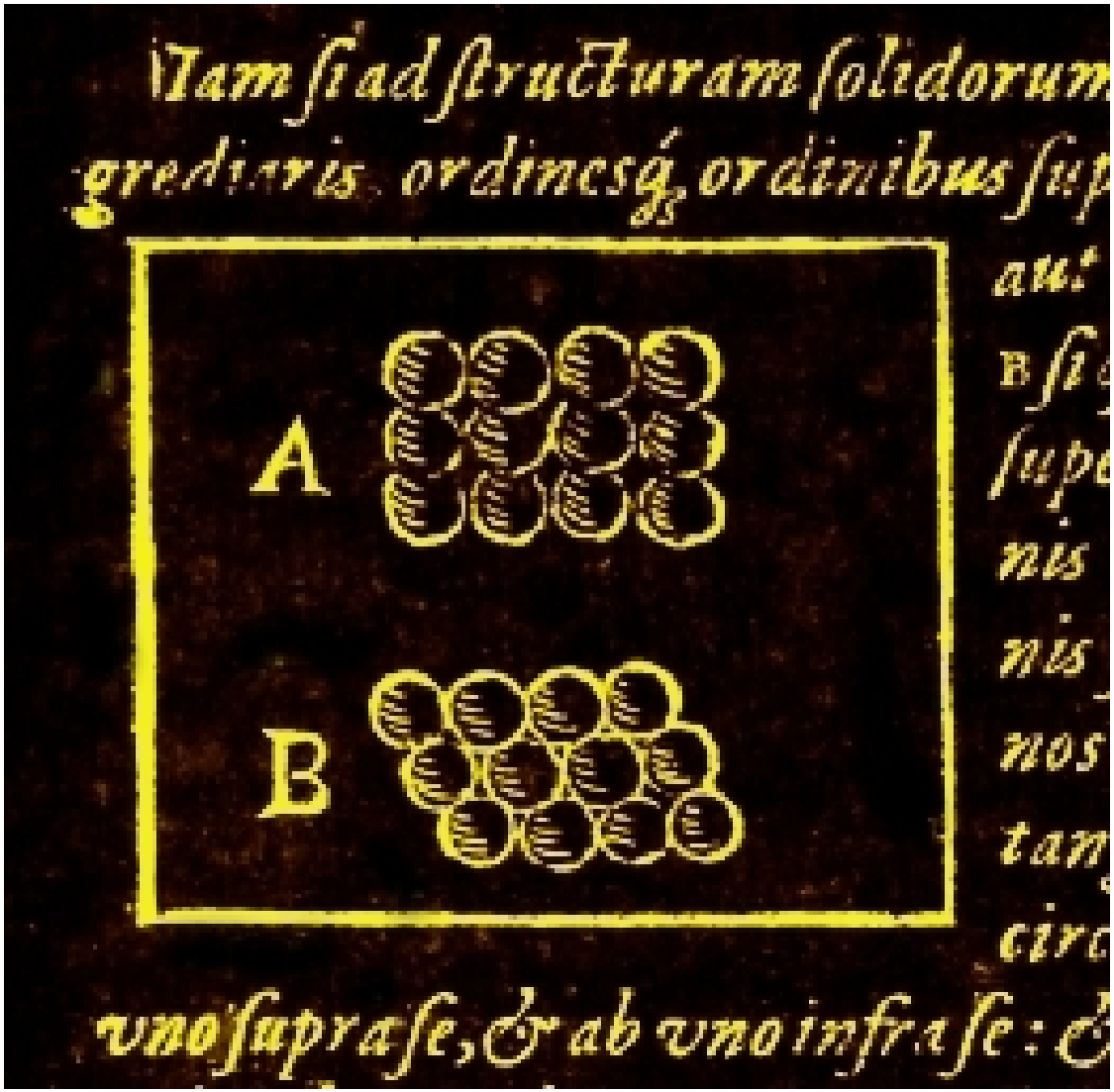
G. Tenenbaum, *Remarques sur les valeurs moyennes de fonctions multiplicatives,* L'Enseignement Mathématique, 2007.

$$r := \frac{4}{5} - \frac{9}{10}\sqrt{\frac{2}{3}} \approx 0,06517$$

```
? 4/5-9/10*sqrt(2/3)
%1 = 0.065153077165046570540814777588232582410
```

Quelle est la valeur exacte ?

Conjecture de Kepler (1611)



Empilement cubique à faces centrées : densité $\frac{\pi}{\sqrt{18}} \approx 0.74$

Conjecture de Kepler

Preuve « papier » par Thomas Hales en 1998 (300 pages, 40000 lignes de code), avec l'aide de 12 relecteurs pendant 5 ans. L'éditeur dit :

> *They checked many local statements in the proof, and each time they found that what you claimed was in fact correct. Some of these local checks were highly non-obvious at first, and required weeks to see that they worked out. The fact that some of these worked out is the basis for the 99% statement of Fejes Tóth that you cite.*

Thomas C. Hales, *A proof of the Kepler conjecture*, Annals of Mathematics, 2005, 1065–1185.

Page 1182 : *This completes the (abridged) proof of the Kepler conjecture.*

Received September 4, 1998. Revised February 14, 2003.

Exemple (crédit Roland Zumkeller) :

$$\forall 2.51^2 \leq x_1 \leq 2.696^2, 4 \leq x_2, x_3 \leq 2.168^2, 4 \leq x_4, x_5, x_6 \leq 2.51^2,$$

$$\frac{37}{50} < \frac{\pi}{2} + \operatorname{atan} \frac{\begin{array}{c} x_1x_3 + x_2x_5 - x_1x_4 - x_3x_6 + x_4x_6 \\ -x_2(x_1 - x_2 + x_3 - x_5 + x_4 + x_6) \end{array}}{\sqrt{D}}$$

où

$$\begin{aligned} D &= 4x_2(x_2x_5(x_1 - x_2 + x_3 - x_5 + x_4 + x_6) \\ &+ x_1x_4(x_2 - x_1 + x_3 + x_5 - x_4 + x_6) \\ &+ x_3x_6(x_2 + x_1 - x_3 + x_5 + x_4 - x_6) \\ &- x_1x_3x_5 - x_2x_3x_4 - x_2x_1x_6 - x_5x_4x_6) \end{aligned}$$

Plan

- quelques catastrophes
- méthodologies et outils
- la constante de Masser-Gramain

Quelques catastrophes

Maple 13 : signe de $\sin(2^{100})$?

```
> evalf(sin(2^100));
```

Maple 13 : signe de $\sin(2^{100})$?

```
> evalf(sin(2^100));
                    0.4491999480
```

Maple 13 : signe de $\sin(2^{100})$?

```
> evalf(sin(2^100));
                    0.4491999480

> evalf(sin(2^100),20);
```

Maple 13 : signe de $\sin(2^{100})$?

```
> evalf(sin(2^100));
                    0.4491999480

> evalf(sin(2^100),20);
              -0.58645356896925826300
```

Maple 13 : signe de $\sin(2^{100})$?

```
> evalf(sin(2^100));
                    0.4491999480

> evalf(sin(2^100),20);
              -0.58645356896925826300

> evalf(sin(2^100),30);
```

Maple 13 : signe de $\sin(2^{100})$?

```
> evalf(sin(2^100));
                    0.4491999480

> evalf(sin(2^100),20);
              -0.58645356896925826300

> evalf(sin(2^100),30);
          0.19988562165362573821513281152 5
```

Maple 13 : signe de $\sin(2^{100})$?

```
> evalf(sin(2^100));
                    0.4491999480

> evalf(sin(2^100),20);
              -0.58645356896925826300

> evalf(sin(2^100),30);
          0.19988562165362573821513281152 5

> evalf(sin(2^100),40);
```

Maple 13 : signe de $\sin(2^{100})$?

```
> evalf(sin(2^100));
                    0.4491999480

> evalf(sin(2^100),20);
              -0.58645356896925826300

> evalf(sin(2^100),30);
          0.19988562165362573821513281152 5

> evalf(sin(2^100),40);
    -0.87218360541826730978071977821347055932 43
```

Autre exemple en précision arbitraire

```
    |\^/|     Maple 10 (IBM INTEL LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2005
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> evalf(Int(exp(-x^2)*ln(x),x=17..42));
                                                  -126
                                0.2604007480 10
```

Autre exemple en précision arbitraire

```
    |\^/|     Maple 10 (IBM INTEL LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2005
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> evalf(Int(exp(-x^2)*ln(x),x=17..42));
                                                  -126
                              0.2604007480 10

> Digits:=20: evalf(Int(exp(-x^2)*ln(x),x=17..42));
```

Autre exemple en précision arbitraire

```
    |\^/|     Maple 10 (IBM INTEL LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2005
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> evalf(Int(exp(-x^2)*ln(x),x=17..42));
                                          -126
                          0.2604007480 10

> Digits:=20: evalf(Int(exp(-x^2)*ln(x),x=17..42));
                                            -126
                   0.34288028340847034512 10
```

Autre exemple en précision arbitraire

```
    |\^/|     Maple 10 (IBM INTEL LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2005
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> evalf(Int(exp(-x^2)*ln(x),x=17..42));
                                        -126
                          0.2604007480 10

> Digits:=20: evalf(Int(exp(-x^2)*ln(x),x=17..42));
                                          -126
                   0.34288028340847034512 10

> Digits:=50: evalf(Int(exp(-x^2)*ln(x),x=17..42));
```

Autre exemple en précision arbitraire

```
    |\^/|     Maple 10 (IBM INTEL LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2005
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> evalf(Int(exp(-x^2)*ln(x),x=17..42));
                                        -126
                          0.2604007480 10

> Digits:=20: evalf(Int(exp(-x^2)*ln(x),x=17..42));
                                          -126
                   0.34288028340847034512 10

> Digits:=50: evalf(Int(exp(-x^2)*ln(x),x=17..42));
                                                          -128
0.49076783443012876473973482836733778547443399549250 10
```

Autre exemple en précision arbitraire

```
    |\^/|     Maple 10 (IBM INTEL LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2005
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> evalf(Int(exp(-x^2)*ln(x),x=17..42));
                                          -126
                        0.2604007480 10

> Digits:=20: evalf(Int(exp(-x^2)*ln(x),x=17..42));
                                           -126
                  0.34288028340847034512 10

> Digits:=50: evalf(Int(exp(-x^2)*ln(x),x=17..42));
                                                              -128
0.49076783443012876473973482836733778547443399549250 10

> Digits:=100: evalf(Int(exp(-x^2)*ln(x),x=17..42));
```

Autre exemple en précision arbitraire

```
    |\^/|     Maple 10 (IBM INTEL LINUX)
._|\|   |/|_. Copyright (c) Maplesoft, a division of Waterloo Maple Inc. 2005
 \  MAPLE  /  All rights reserved. Maple is a trademark of
 <____ ____>  Waterloo Maple Inc.
      |       Type ? for help.
> evalf(Int(exp(-x^2)*ln(x),x=17..42));
                                          -126
                        0.2604007480 10

> Digits:=20: evalf(Int(exp(-x^2)*ln(x),x=17..42));
                                            -126
                  0.34288028340847034512 10

> Digits:=50: evalf(Int(exp(-x^2)*ln(x),x=17..42));
                                                              -128
0.49076783443012876473973482836733778547443399549250 10

> Digits:=100: evalf(Int(exp(-x^2)*ln(x),x=17..42));
0.49076783443012876473973482836733778547443399549250384\
                                                            -128
    2
```

Et en précision arbitraire ?

```
> Digits:=50: evalf(Int(exp(-x^2)*ln(x),x=17..42));
                                                          -128
0.49076783443012876473973482836733778547443399549250 10

> Digits:=100: evalf(Int(exp(-x^2)*ln(x),x=17..42));
0.4907678344301287647397348283673377854744339954925038\
                                                            -128
    1435498665013506331285359635525372050785062212 10
```

Et en précision arbitraire ?

```
> Digits:=50: evalf(Int(exp(-x^2)*ln(x),x=17..42));
                                                              -128
0.49076783443012876473973482836733778547443399549250 10

> Digits:=100: evalf(Int(exp(-x^2)*ln(x),x=17..42));
0.4907678344301287647397348283673377854744339954925038\
                                                                -128
    1435498665013506331285359635525372050785062212 10

> Digits:=150: evalf(Int(exp(-x^2)*ln(x),x=17..42));
```

Et en précision arbitraire ?

```
> Digits:=50: evalf(Int(exp(-x^2)*ln(x),x=17..42));
                                                              -128
0.49076783443012876473973482836733778547443399549250 10

> Digits:=100: evalf(Int(exp(-x^2)*ln(x),x=17..42));
0.4907678344301287647397348283673377854744339954925038\
                                                            -128
    1435498665013506331285359635525372050785062212 10

> Digits:=150: evalf(Int(exp(-x^2)*ln(x),x=17..42));
0.2565728500561051482917356395759084310260126667622196\
    16\
    0779728519476069516996149975630559922582634961886705\

                                                           -126
    5745187518615884166042779597226368281066261 3 10
```

Signe de $c = \exp(\pi\sqrt{163}) - 262537412640768744$?

Sage 4.5.3 :

```
sage: c=exp(pi*sqrt(163))-262537412640768744

sage: numerical_approx(c, digits=15)
448.000000000000

sage: numerical_approx(c, digits=30)
-5.96855898038484156131744384766e-13
```

Mathematica 6.0 :

```
In[1]:= c = Exp[Pi*Sqrt[163]]-262537412640768744;

In[2]:= N[c]

Out[2]= -480.

In[3]:= N[c, 20]

                                          -13
Out[3]= -7.4992740280181431112 10
```

```
sage: version()
'Sage Version 4.4.4, Release Date: 2010-06-23'
sage: f=sin(x)^2+cos(x)^2-1
```

```
sage: version()
'Sage Version 4.4.4, Release Date: 2010-06-23'
sage: f=sin(x)^2+cos(x)^2-1
sage:  f.nintegrate(x,0,1)
(-1.1195877839147071e-18, 6.4484713082106476e-18,
  8379, 5)
```

```
sage: version()
'Sage Version 4.4.4, Release Date: 2010-06-23'
sage: f=sin(x)^2+cos(x)^2-1
sage:  f.nintegrate(x,0,1)
(-1.1195877839147071e-18, 6.4484713082106476e-18,
  8379, 5)

sage: f.nintegrate?
OUTPUT:
  -- float: approximation to the integral
  -- float: estimated absolute error of the approximation
  -- the number of integrand evaluations
  -- an error code:
       0 -- no problems were encountered
       1 -- too many subintervals were done
       2 -- excessive roundoff error
       3 -- extremely bad integrand behavior
       4 -- failed to converge
       5 -- integral is probably divergent or slowly convergent
       6 -- the input is invalid
```

```
sage: f=exp(pi*sqrt(163))-262537412640768744
sage: f.nintegrate(x,0,1)
(-480.00000000000011, 5.3290705182007538e-12, 21, 0)

sage: n(f,120)
-7.4992399443085666632669017417356372e-13
```

```
sage: f=exp(pi*sqrt(163))-262537412640768744
sage: f.nintegrate(x,0,1)
(-480.00000000000011, 5.3290705182007538e-12, 21, 0)

sage: n(f,120)
-7.4992399443085666632669017417356372e-13
```

Est-ce un bug ?

missile Scud

anti-missile Patriot

L'échec de l'anti-missile Patriot

En 1991, pendant la guerre du Golfe, un anti-missile Patriot manque l'interception d'un missile Scud. Bilan : 28 soldats morts.

La position de l'anti-missile est calculée en multiples de dixièmes de seconde.

Un compteur totalise le nombre de dixièmes de seconde depuis le démarrage de l'horloge du Patriot.

Le format utilisé est en base 2 avec mantisse de 24 bits.

$\frac{1}{10}$ n'est pas exactement représentable en binaire.

Pour représenter $1/10$, on a utilisé (en binaire) :

$$0.000110011001100110011001100$$

L'erreur est d'environ 0.000000095s en décimal.

Après 100 heures, l'erreur totale est :

$$0.000000095 \cdot 10 \cdot 3600 \cdot 100 \approx 0.34\text{s}$$

Pendant ce temps, un Scud (1676 m/s) parcourt près de 600 mètres !

Si au lieu de tronquer $1/10$ on avait arrondi au plus proche :

$$0.000110011001100110011001101$$

l'erreur aurait été de 0.000000024 au lieu de 0.000000095

soit environ 140 mètres au lieu de 600 !

Élections du parlement au Schleswig-Holstein (1992)

Les partis ayant moins de 5% des voix n'ont pas de siège.

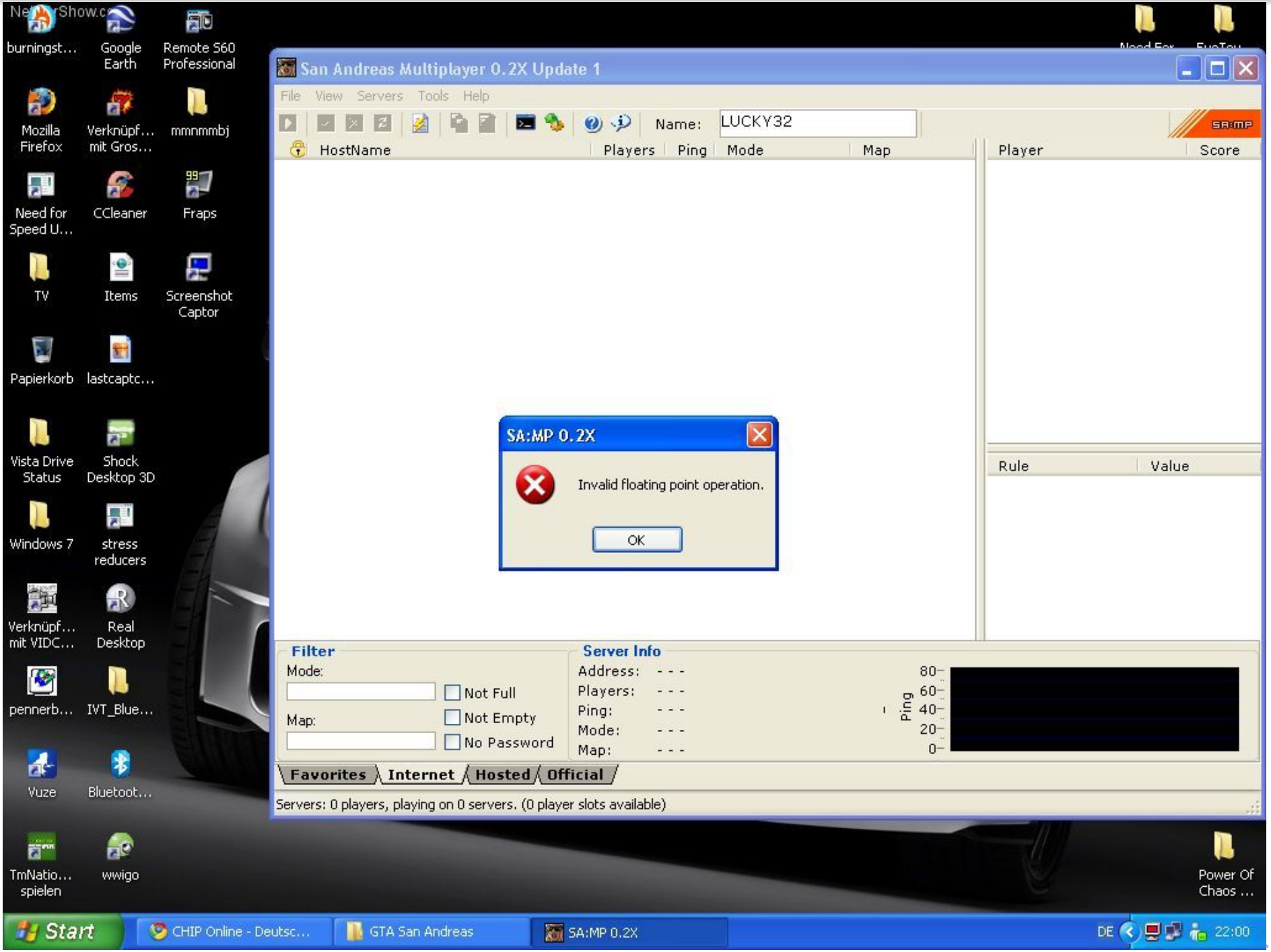
Au dépouillement, le parti écologiste obtient exactement 5%.

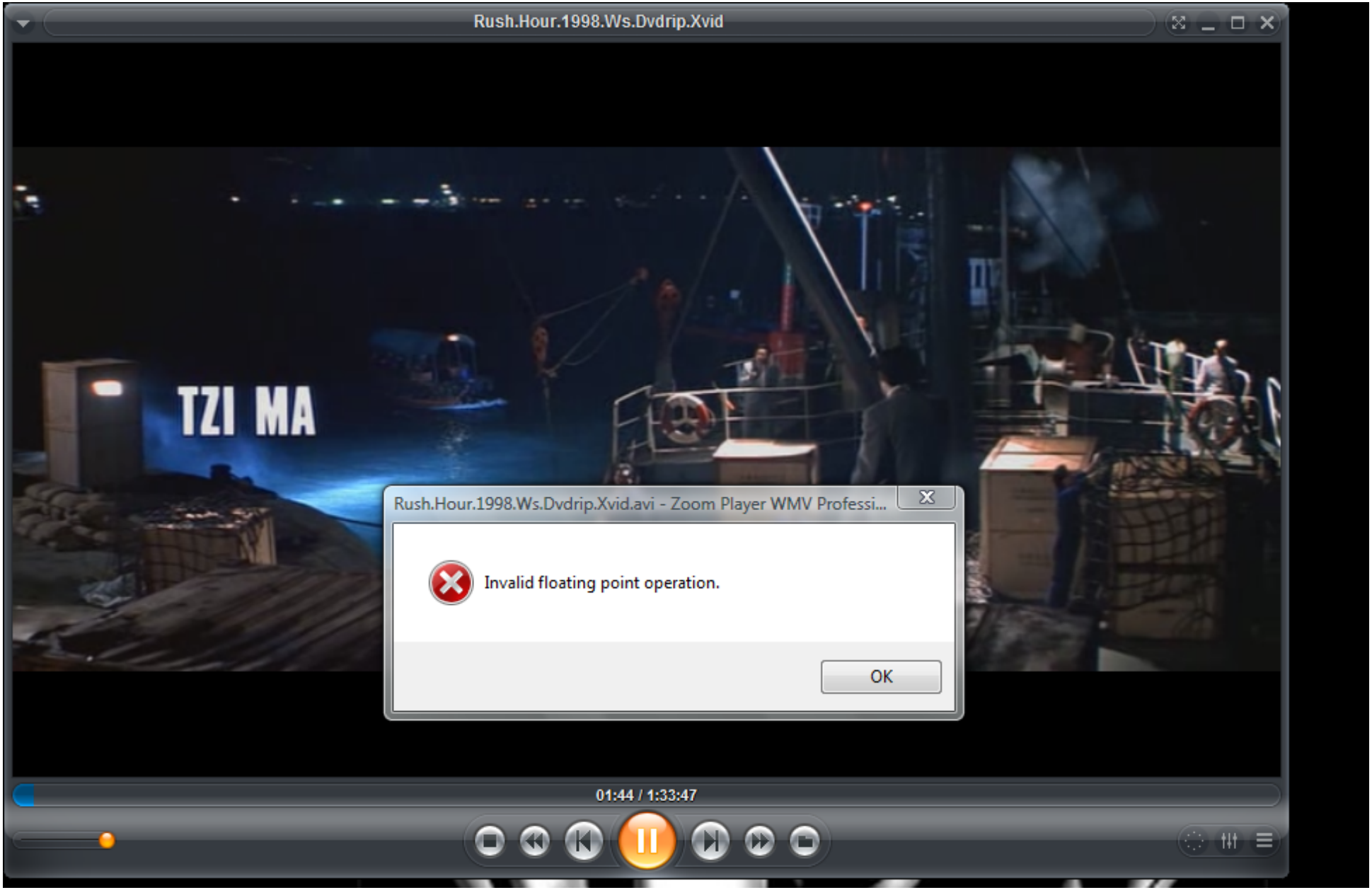
Après l'annonce officielle des résultats, on découvre que le parti écologiste n'a en fait que 4.97% des voix !

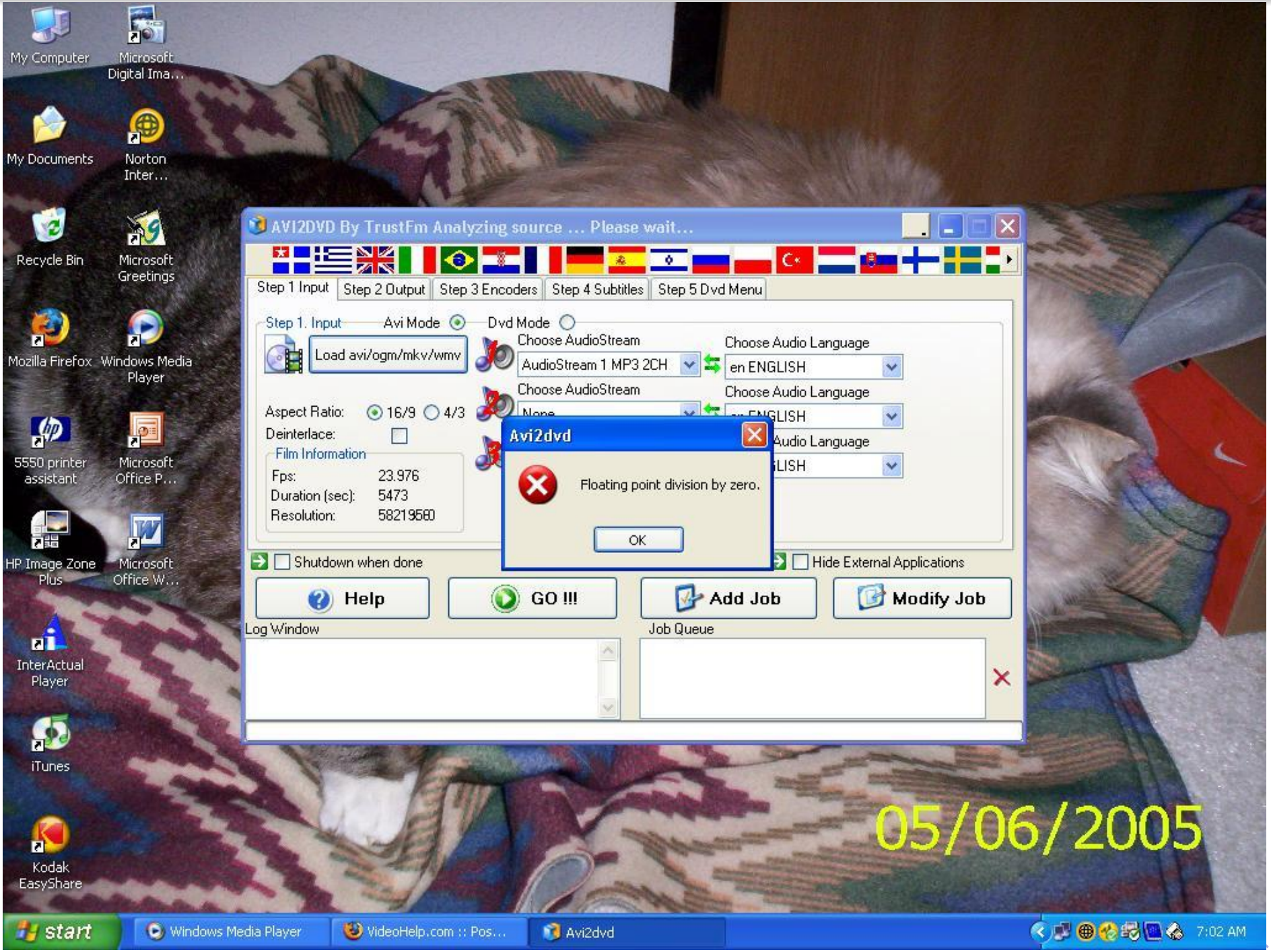
Le programme affichant les résultats arrondit à un seul chiffre après la virgule, et donc a arrondi 4.97 à 5.0.

L'erreur corrigée, le siège est retiré aux verts, et attribué au SPD, qui avec 45 sièges sur 89 obtient la majorité absolue.

Ce programme était utilisé depuis plusieurs années...


San Andreas Multiplayer 0.2X Update 1
File View Servers Tools Help
Name: LUCKY32
HostName Players Ping Mode Map
Player Score
Rule Value
SA:MP 0.2X
Invalid floating point operation.
OK
Filter
Mode:
Map:
Not Full
Not Empty
No Password
Server Info
Address: - - -
Players: - - -
Ping: - - -
Mode: - - -
Map: - - -
Ping
80
60
40
20
0
Favorites Internet Hosted Official
Servers: 0 players, playing on 0 servers. (0 player slots available)
Start
CHIP Online - Deutsc...
GTA San Andreas
SA:MP 0.2X
DE
22:00


Rush.Hour.1998.Ws.Dvdrip.Xvid
TZI MA
Rush.Hour.1998.Ws.Dvdrip.Xvid.avi - Zoom Player WMV Professi...
Invalid floating point operation.
OK
01:44 / 1:33:47


My Computer
Microsoft Digital Ima...
My Documents
Norton Inter...
Recycle Bin
Microsoft Greetings
Mozilla Firefox
Windows Media Player
5550 printer assistant
Microsoft Office P...
HP Image Zone Plus
Microsoft Office W...
InterActual Player
iTunes
Kodak EasyShare
AVI2DVD By TrustFm Analyzing source ... Please wait...
Step 1 Input
Step 2 Output
Step 3 Encoders
Step 4 Subtitles
Step 5 Dvd Menu
Step 1. Input
Avi Mode
Dvd Mode
Load avi/ogm/mkv/wmv
Choose AudioStream
AudioStream 1 MP3 2CH
Choose Audio Language
en ENGLISH
Aspect Ratio:
16/9
4/3
Deinterlace:
Film Information
Fps: 23.976
Duration (sec): 5473
Resolution: 58219580
Avi2dvd
Floating point division by zero.
OK
Shutdown when done
Hide External Applications
Help
GO !!!
Add Job
Modify Job
Log Window
Job Queue
05/06/2005
start
Windows Media Player
VideoHelp.com :: Pos...
Avi2dvd
7:02 AM

William Kahan, Arith 17 keynote talk

Values *Excel 2000* Displays for Several Expressions

Expression	1.23456789012345000E+00	<– Entered to help count digits
V = 4/3 displays …	1.33333333333333000E+00	Does *Excel* carry 15 sig. dec.?
W = V - 1	3.33333333333333000E-01	Whence comes the 15th 3 ?
X = W*3	1.00000000000000000E+00	Where went all 15 of the 9s ?
Y = X - 1	0.00000000000000000E+00	They all went away !
Z = Y*2^52	0.00000000000000000E+00	Really all gone.
(4/3 - 1)*3 - 1	0.00000000000000000E+00	Yes, gone.
((4/3 - 1)*3 - 1)	-2.22044604925031000E-16	(But not *ENTIRELY* gone !)
((4/3 - 1)*3 - 1)*2^52	-1.00000000000000000E+00	*Excel*'s arithmetic is *weird*.

Du *Handbook of Floating-Point Arithmetic* :

Excel 2007 : $65536 - 2^{-37}$ s'affiche 100001

Mais si on ajoute 1 on obtient 65537

Maple 6.0 : 2147483648 affiche -2147483648, et -2147483648 affiche 2147483648
(`http://maple.bug-list.org/`, bug #1542)

Maple 7.0 : 1001!/1000! donne 1 au lieu de 1001

Parfois on n'a pas besoin d'erreur numérique...

Crash de Mars Climate Orbiter en 1999 : mètres vs pieds...

2004 : pont sur le Rhin entre l'Allemagne et la Suisse, définitions différentes du niveau de la mer...

Plus près de nous...

INRIA
INRIA
ROCQUENCOURT
DOMAINE DE VOLUCEAU
B.P. 105
78150 LE CHESNAY

Etat de frais définitif Agent
Provisoire

Informations relatives à l'agent

Identifiant Agent	**4645**	Service de l'agent	**SPACES-LOR PR**
Nom Prénom	**M. ZIMMERMANN PAUL**	Fonction	**PERSONNEL SCIENTIFIQUE**
Résidence familiale	**29 RUE DU PRIEURE**		
	54220 MALZEVILLE	Résidence adm.	**VILLERS LES NANCY**
Statut	**TITULAIRE**		
Groupe	**Groupe I**		
Indice	**820**		
Réductions			

Total Etat de Frais 9552

Frais du 22/11/2004 07:00 au 22/11/2004 22:20

Montant de l'avance théorique	**11,44**
Montant de l'avance accordée	**0,00**
Avance payée le	
Montant total des frais	**35,55**

Informations relatives à la mission 817

Mission	**817 / 1**	Durée	**du 22/11/2004 07:00 au 22/11/2004 22:20**
Service Gestionnaire	**UR LORRAINE**	Référence	**ALC / 1413549**
Lieux de mission	**LYON (FR)**		
Déplacement	**Mission**	Transport principal	**Train**
Service Bénéficiaire	**ARENAIRE PR**		
Itinéraire	**MALZEVILLE - NANCY - LYON - NANCY - MALZEVILLE**		
Motif	**Membre du jury de thèse de Sylvie Boldo**		
Commentaire	**TAXI Autorisé :Transport de matériel encombrant.**		

Date 1ère attrib.	Lieu	Type	Barème	Qté	Mnt. Unitaire	Tot.	Mnt. Unit. Devise	Taux Devise
22/11/2004	NANCY	Repas du soir	FRance PRovince	1,00	15,25	15,25		
22/11/2004	NANCY	Frais de taxi	FRance PRovince	1,00	18,90	18,89		
22/11/2004	LYON	Billets de bus, métro et RER		1,00	1,40	1,39		

Visa de l'agent

Visa de l'ordonnateur

/astre/id/xsl/FraisAgentTiers.xsl

Page 1

	Qté	Mnt. Unitaire	Tot.	Mnt.
vince	1,00	15,25	15,25	
vince	1,00	18,90	18,89	
	1,00	1,40	1,39	

Méthodologies et outils

Comment obtenir une preuve ?

Méthodes « calculatoires » :

- calcul avec arrondi : le standard IEEE 754
- arithmétique d'intervalles
- modèle RealRAM

Méthodes formelles : assistant de preuve

Le standard IEEE 754

Première version en 1985, révisé en 2008.

Définit 3 formats binaires (`binary32`, `binary64`, `binary128`) et 2 formats décimaux (`decimal64`, `decimal128`).

Impose l'arrondi correct pour $+, -, \times, \div$, la racine carrée, et les conversions binaire-décimal.

Recommande l'arrondi correct pour les fonctions

$$\exp x, \log x, \sqrt{x^2+y^2}, x^{-1/2}, (1+x)^n, x^{1/n}, x^n, x^y, \sin x, \cos x, \tan x$$

$$\sin(\pi x), \cos(\pi x), (\operatorname{atan} x)/\pi, \operatorname{asin} x, \operatorname{acos} x, \operatorname{atan} x, \sinh x, \cosh x$$

$$\tanh x, \operatorname{asinh} x, \operatorname{acosh} x, \operatorname{atanh} x$$

Le format `binary64`

Correspond au type `double` du langage C.

$$x = (-1)^s \cdot 1.b_1 b_2 \ldots b_{52} \cdot 2^e$$

Signe s encodé sur un bit : $s = 0 \Rightarrow x \geq 0$, $s = 1 \Rightarrow x \leq 0$

Exposant e encodé sur 11 bits. Soit $0 \leq v \leq 2047$ la valeur codée, on a $e = v - 1023$.

Mantisse $1.b_1 b_2 \ldots b_{52}$ encodée sur 52 bits, par la valeur entière de $b_1 b_2 \ldots b_{52}$.

± 0 sont encodés avec $v = 0$.
$\pm \mathrm{Inf}$ et NaN (Not A Number) sont encodés avec $v = 2047$.

Formats : un exemple

Soit l'approximation au plus proche de π :

$$\frac{\overbrace{7074237752028440}^{53\text{ bits}}}{2^{51}}$$

$s = 0$, $e = 1$, $v = e + 1023 = 1024$

$$m := b_1 b_2 \ldots b_{52} = 7074237752028440 - 2^{52} = 2570638124657944$$

$$s \cdot 2^{63} + v \cdot 2^{52} + m = 4614256656552045848$$

$$= (0100000000001001001000011111101101010100010001000010110100011000)_2$$

Exceptions

- *underflow* : $x/2.0$ pour $x = 2^{-1074}$
- *overflow* : $x \times 2.0$ pour $x = 2^{1023}$
- *division by zero* : $1.0/0.0$
- *inexact* : $1.0/5.0$
- *invalid* : $\sqrt{-1.0}$ ou $+\mathrm{Inf} - (+\mathrm{Inf})$

Arrondi correct

IEEE 754 définit 5 modes d'arrondi :

- `roundTiesToEven` : au plus proche avec arrondi pair
- `roundTiesToAway` : au plus proche avec arrondi *away*
- `roundTowardPositive` : vers $+\infty$
- `roundTowardNegative` : vers $-\infty$
- `roundTowardZero` : vers 0

Exemple en décimal avec 2 chiffres significatifs :

	-1.23	1.23	1.25	1.27
roundTiesToEven	-1.2	1.2	1.2	1.3
roundTiesToAway	-1.2	1.2	1.3	1.3
roundTowardPositive	-1.2	1.3	1.3	1.3
roundTowardNegative	-1.3	1.2	1.2	1.2
roundTowardZero	-1.2	1.2	1.2	1.2

Ce qu'apporte le standard IEEE 754

- un seul résultat correct
- $\Rightarrow$ reproductibilité (processeur, compilateur, système)
- arrondis dirigés : borne inférieure ou supérieure
- borne d'erreur sur l'arrondi
- possibilité de prouver des théorèmes

Bornes d'erreur

$$y = \circ(x)$$

Si $y = (-1)^s \cdot 1.b_1 b_2 \ldots b_{52} \cdot 2^e$, on définit

$$\mathrm{ulp}(y) := 0.00 \ldots 001 \cdot 2^e = 2^{e-52}$$

- arrondi au plus proche : $|y - x| \leq \frac{1}{2}\mathrm{ulp}(y)$

Bornes d'erreur

$$y = \circ(x)$$

Si $y = (-1)^s \cdot 1.b_1 b_2 \ldots b_{52} \cdot 2^e$, on définit

$$\mathrm{ulp}(y) := 0.00 \ldots 001 \cdot 2^e = 2^{e-52}$$

- arrondi au plus proche : $|y - x| \leq \frac{1}{2}\mathrm{ulp}(y)$
- arrondis dirigés : $|y - x| \leq \mathrm{ulp}(y)$
  - vers zéro : $|y| \leq |x|$

Bornes d'erreur

$$y = \circ(x)$$

Si $y = (-1)^s \cdot 1.b_1 b_2 \ldots b_{52} \cdot 2^e$, on définit

$$\mathrm{ulp}(y) := 0.00 \ldots 001 \cdot 2^e = 2^{e-52}$$

- arrondi au plus proche : $|y - x| \leq \frac{1}{2}\mathrm{ulp}(y)$
- arrondis dirigés : $|y - x| \leq \mathrm{ulp}(y)$
  - vers zéro : $|y| \leq |x|$
  - vers $-\infty$: $y \leq x$

Bornes d'erreur

$$y = \circ(x)$$

Si $y = (-1)^s \cdot 1.b_1 b_2 \ldots b_{52} \cdot 2^e$, on définit

$$\mathrm{ulp}(y) := 0.00 \ldots 001 \cdot 2^e = 2^{e-52}$$

- arrondi au plus proche : $|y - x| \leq \frac{1}{2}\mathrm{ulp}(y)$
- arrondis dirigés : $|y - x| \leq \mathrm{ulp}(y)$
  - vers zéro : $|y| \leq |x|$
  - vers $-\infty$: $y \leq x$
  - vers $+\infty$: $y \geq x$

Lemme de Sterbenz (1974)

Lemme. *Si x et y sont deux flottants tels que $y/2 < x < 2y$, alors :*

$$z \leftarrow \circ(x - y)$$

est exact.

```
sage: R=RealField(42)

sage: x=R(catalan)
sage: y=R(euler_gamma)
sage: x, y
(0.915965594177, 0.577215664902)

sage: z=x-y
sage: z
0.338749929276

sage: x.exact_rational() - y.exact_rational()
1489837944587/4398046511104

sage: z.exact_rational()
1489837944587/4398046511104
```

FastTwoSum (Knuth, 1973)

Théorème. *Si* $|a| \geq |b|$, *et :*

$$\begin{aligned} s &\leftarrow \circ(a+b) \\ t &\leftarrow \circ(s-a) \\ u &\leftarrow \circ(b-t) \end{aligned}$$

alors

$$a+b = s+u.$$

```
sage: R=RealField(53,sci_not=1)

sage: a=R(pi)
sage: b=R(exp(1))

sage: s=a+b
sage: t=s-a
sage: u=b-t

sage: s, u
(5.85987448204884e0, 4.44089209850063e-16)

sage: a.exact_rational()+b.exact_rational()\
     -s.exact_rational()-u.exact_rational()
0
```

Calcul de la base interne (Malcolm, 1972)

Sage 4.4.4 :

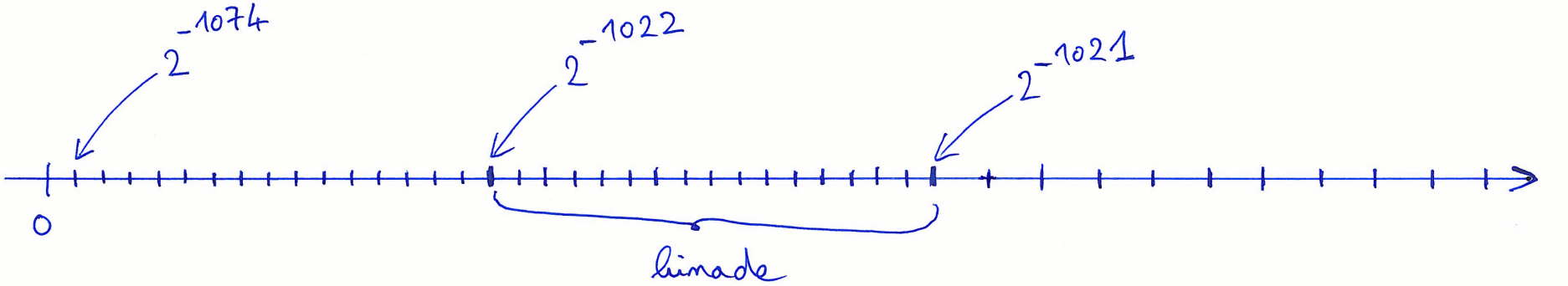
```
sage: A = 1.0
sage: B = 1.0
sage: while ((A + 1.0) - A) - 1.0 == 0.0:
....:         A = 2.0 * A
sage: while ((A + B) - A) - B <> 0.0:
....:         B = B + 1.0
sage: B
2.00000000000000
```

Maple 13 :

```
> A := 1.0:
> B := 1.0:
> while (evalf(A + 1.0) - A) - 1.0 = 0.0 do
     A := 2.0 * A
  od:
> while ((A + B) - A) - B <> 0.0 do
     B := B + 1.0
  od:
> B;
                                10.0
```

Subtilités de IEEE 754 : Gradual underflow

Le plus petit nombre *normalisé* positif est 2^{-1022}.
Entre 0 et 2^{-1022}, les nombres sont *régulièrement* espacés de 2^{-1074}.



Conséquence : si $x \neq y$, alors $|x - y| \geq 2^{-1074}$, donc $x - y$ ne peut pas être arrondi à zéro.
Sans nombre dénormalisé : soit $x = 2^{-1022}$ et $y = x + 2^{-1074}$, alors $y - x = 2^{-1074}$ est arrondi à zéro (au plus proche).

Gradual underflow : conséquences

Avec *gradual underflow*, le code ci-dessous ne peut pas faire de division par zéro :

```
if x <> y then
   z = 1.0 / (x - y)
else
   ...
```

Subtilités de IEEE 754 : conversion décimal-binaire

Le réel 0.1 n'est pas exactement représentable en binaire. Il est donc arrondi, au plus proche à :

```
sage: a=RealField(53)(0.1)
sage: a.exact_rational()
3602879701896397/36028797018963968
```

Conséquence : si on écrit `sin(0.1)` dans un programme C, on ne calcule pas le sinus de 0.1, mais celui de

$$3602879701896397 \cdot 2^{-55} \approx 0.10000000000000005551115123125$$

Conversion binaire-décimal

```
int main()
{
  double x = 0.1;
  double y = 0.3;
  printf ("3*x-y=%e\n", 3*x-y);
}

[zimmerma@coing tmp]$ gcc e.c; ./a.out
3*x-y=5.551115e-17
```

Double arrondi

Quand on calcule `sin(0.6)` avec p bits de précision :

1. 0.6 est arrondi à x sur p bits ;
2. puis sin x est arrondi sur p bits.

Conséquence : on ne peut pas calculer directement l'arrondi correct de sin 0.6 sur p bits.

Exemple avec $p = 11$: 0.6 est arrondi à 0.60009765625.
sin 0.60009765625 est arrondi à 0.564**94140625**
sin 0.6 serait arrondi à 0.564**453125**

Mise en œuvre de IEEE 754 en langage C

```
#include <stdio.h>
#include <fenv.h>
int main()
{
  double x = 3.0;
  fesetround (FE_DOWNWARD);
  printf ("1/3 arrondi vers -Inf: %.17f\n", 1.0 / x);
  fesetround (FE_UPWARD);
  printf ("1/3 arrondi vers +Inf: %.17f\n", 1.0 / x);
}

tarte% gcc -O0 -frounding-math ex.c -lm
tarte% ./a.out
1/3 arrondi vers -Inf: 0.33333333333333331
1/3 arrondi vers +Inf: 0.33333333333333337
```

Mais...

```
#include <stdio.h>
#include <math.h>
#include <fenv.h>
int main()
{
  double x = 3.0;
  fesetround (FE_DOWNWARD);
  printf ("sin(3) arrondi vers -Inf:        %.17f\n", sin (x));
  fesetround (FE_UPWARD);
  printf ("sin(3) arrondi vers +Inf:        %.17f\n", sin (x));
  fesetround (FE_TONEAREST);
  printf ("sin(3) arrondi au plus proche : %.17f\n", sin (x));
  fesetround (FE_TOWARDZERO);
  printf ("sin(3) arrondi vers 0 :          %.17f\n", sin (x));
}
```

```
tarte% gcc -O2 -frounding-math ex2.c -lm ; ./a.out
sin(3) arrondi vers -Inf:        2.78837698122124511
sin(3) arrondi vers +Inf:        2.78837698122124511
sin(3) arrondi au plus proche : 2.78837698122124511
sin(3) arrondi vers 0 :          2.78837698122124511
```

D'une machine à l'autre...

```
#include <math.h>
int main()
{
  double x = 0.2522464;
  printf ("sin(%.7f)=%.16e\n", x, sin (x));
  x = 1267650600228229401496703205376.0; /*2^100*/
  printf ("sin(2^100)=%.16e\n", sin (x));
}
```

Core 2 Duo (64 bit)/Fedora Core 12/gcc 4.4.4 :

```
$ gcc -O0 bug10709.c -lm; ./a.out
sin(0.2522464)=2.4957989804940914e-01
sin(2^100)=-8.7218360541826734e-01
```

Pentium 4 (32 bit)/Fedora Core 10/gcc 4.3.2 :

```
$ gcc -O0 bug10709.c -lm; ./a.out
sin(0.2522464)=2.4957989804940911e-01
sin(2^100)=7.1256751265005913e-01
```

Core 2, Linux 2.6.32, gcc 4.4.4, libc 2.11.2 :

```
Testing function sin for exponent 10.
 rounding mode MPFR_RNDZ:
      outside range for x=1.72483495297764 53e2
           f(x)=3.6751672462495377
      not between -1.0000000000000000
                and 1.0000000000000000
```

Core 2, Linux 2.6.32, gcc 4.4.4, libc 2.11.2 :

```
Testing function cbrt for exponent 0.
 rounding mode MPFR_RNDD:
      monotonicity not respected for
                  x=1.9405836340591431e-1
             f(x-)=5.7895408352384725e-1
      not <= f(x)=5.7895408352384714e-1
```

Core 2, Linux 2.6.32, gcc 4.4.4, libc 2.11.2 :

```
Testing function sinh for exponent 0.
 rounding mode MPFR_RNDN:
      1.6420340235345066 ulp(s) for
      x=-7.2825481099151501e-1
 rounding mode MPFR_RNDZ:
      2.6626207558438182 ulp(s) for
      x=-6.1148603439700744e-2
      wrong directed rounding for
      x=4.7926047436246777e-1 [1.0388485498260707]
```

Arrondi correct

- en double précision : IEEE 754 (pour les quatre opérations et la racine carrée) ;
- efficace, facilité de mise en œuvre
- facile uniquement pour opérations atomiques

Outils : `fenv.h` en C, Maple ($+, -, \times, \div$), NTL ($+, -, \times, \div, \sqrt{x}$ au plus proche), decNumber ($+, -, \times, \div, \sqrt{x}, x^n$), GNU MPFR (toutes les fonctions de ISO C99), MPC pour les complexes en précision arbitraire (avec A. Enge)

La bibliothèque GNU MPFR

- implante IEEE 754 en précision arbitraire
- *garantit* l'arrondi correct pour les fonctions mathématiques
- uniquement opérations atomiques (comme IEEE 754)

Preuve que $\pi < 22/7$ avec GNU MPFR

```
#include "mpfr.h"
int main (int argc, char *argv[])
{
  mpfr_t x;
  mpfr_init2 (x, atoi (argv[1]));
  mpfr_const_pi (x, GMP_RNDU);
  mpfr_mul_ui (x, x, 7, GMP_RNDU);
  mpfr_sub_ui (x, x, 22, GMP_RNDU);
  mpfr_printf ("7pi-22 <= %RUf\n", x);
  mpfr_clear (x);
}

$ gcc pi.c -lmpfr; ./a.out 12
7pi-22 <= -0.0078125
```

Ou plus simplement en Sage...

```
sage: RealField(12,rnd='RNDU')(7*pi-22)
-0.00781

sage: RealField(12,rnd='RNDU')(7*pi-22).n(100)
-0.0078125000000000000000000000000
```

Arithmétique d'intervalles

- principe : on représente x par un intervalle $[a, b]$ tel que $a \leq x \leq b$
- facilité de mise en œuvre, perte d'un facteur 2 à 4
- problème de décorrélation : $x^2 - x$ sur $[0, 2]$ donne $[0, 4] - [0, 2] = [-2, 4]$; $x(x - 1)$ donne $[0, 2] \times [-1, 1] = [-2, 2]$; l'intervalle optimal est $[-0.25, 2]$

Outils : `filib++` et INTLAB en précision fixe (CoStLy pour les complexes), MPFI en précision arbitraire.
Voir `http://www.cs.utep.edu/interval-comp/`
En cours de standardisation :
`http://grouper.ieee.org/groups/1788/`

Utilisation de MPFI en Sage

```
sage: x=RIF(0,2)
sage: x.str(style='brackets')
'[0.00000000000000000 .. 2.00000000000000000]'

sage: (x^2-x).str(style='brackets')
'[-2.00000000000000000 .. 4.00000000000000000]'

sage: (x*(x-1)).str(style='brackets')
'[-2.00000000000000000 .. 2.00000000000000000]'

sage: x=RIF(-1,1)
sage: (x^2).str(style='brackets')
'[0.00000000000000000 .. 1.00000000000000000]'
sage: (x*x).str(style='brackets')
'[-1.00000000000000000 .. 1.00000000000000000]'
```

Preuve que $\pi < 22/7$ avec MPFI

```
sage: RIF12 = RealIntervalField(12)
sage: RIF12(7*pi-22).str(style='brackets')
'[-0.015625 .. -0.0078125]'

sage: RIF(4/5-9/10*sqrt(2/3)).str(style='brackets')
'[0.065153077165046369 .. 0.065153077165046814]'

sage: RIF(sin(2^100)).str(style='brackets')
'[-0.87218360541826734 .. -0.87218360541826722]'
```

Modèle RealRAM

- principe : évaluer un arbre d'expressions récursivement, avec une erreur absolue d'au plus 2^{-n} à la racine
- le logiciel gère automatiquement la propagation des erreurs, et détermine la précision optimale pour chaque branche, avec recalcul au besoin
- facilité de mise en œuvre
- efficacité dépend beaucoup de l'implantation
- ne peut pas décider de l'égalité à 0

Outils : RealLib, iRRAM

Utilisation de iRRAM (Norbert Müller)

```
REAL x=REAL(1)-REAL(9)/REAL(10)-REAL(1)/REAL(10);
REAL y=REAL(5)-REAL(49)/REAL(10)-REAL(1)/REAL(10);
REAL z=REAL(22)/REAL(7)-pi();
REAL t=REAL(4)/REAL(5)-REAL(9)/REAL(10)
       *sqrt(REAL(2)/REAL(3));
REAL u=sin(power(REAL(2),REAL(100)));
REAL v=exp(pi()*sqrt(REAL(163)))-REAL(262537412)
       *REAL(1000000000)-REAL(640768744);
REAL w=sin(REAL(17))*sin(REAL(17))
       +cos(REAL(17))*cos(REAL(17))-REAL(1);

printval("1-9/10-1/10          = "              ,DP,x);
printval("5-49/10-1/10         = "              ,DP,y);
printval("22/7-pi              = "              ,DP,z);
printval("4/5-9/10*sqrt(2/3)= "                 ,DP,t);
printval("sin(2^100)           = "              ,DP,u);
printval("exp(pi*sqrt(163))-262537412640768744= ",DP,v);
printval("sin(17)^2+cos(17)^2-1="                ,DP,w);
```

```
Length of mantissa: 20

1-9/10-1/10          =    0

5-49/10-1/10         =    0

22/7-pi              = +.12644892673496186802E-0002

4/5-9/10*sqrt(2/3)= +.65153077165046570541E-0001

sin(2^100)           = -.87218360541826730978E+0000

exp(pi*sqrt(163))-262537412640768744=
                       -.74992740280181431112E-0012

sin(17)^2+cos(17)^2-1=   0
```

```
Length of mantissa: 10

1-9/10-1/10          =    0

5-49/10-1/10         =    0

22/7-pi              = +.1264489267E-0002

4/5-9/10*sqrt(2/3)= +.6515307717E-0001

sin(2^100)           = -.8721836054E+0000

exp(pi*sqrt(163))-262537412640768744=    0

sin(17)^2+cos(17)^2-1=   0
```

Preuve formelle

- principe : formaliser en termes informatiques la méthode employée (arrondi correct, intervalles, ...)
- réutiliser un « noyau flottant » indépendant de l'application
- vérification de la preuve automatique
- il n'y a qu'à vérifier que l'énoncé « formel » correspond bien à l'énoncé mathématique

Outils : HOL Light, PVS, Coq, ...
Succès : théorème des 4 couleurs (Gonthier, Werner), division flottante de l'Itanium (Harrison)

Conjecture de Kepler : Hales a lancé en 2003 le projet Flyspeck pour obtenir une preuve formelle.

La constante de Masser-Gramain

(travail en cours avec Guillaume Melquiond)

Constante d'Euler-Mascheroni :

$$\gamma = \lim_{n\to\infty} \left(\sum_{k=1}^{n} \frac{1}{k} - \log n \right) \approx 0.577.$$

Soit d_k la longueur du plus petit segment de $\mathbb{R}$ qui contienne au moins k points de $\mathbb{Z}$. On a $d_k = k - 1$.

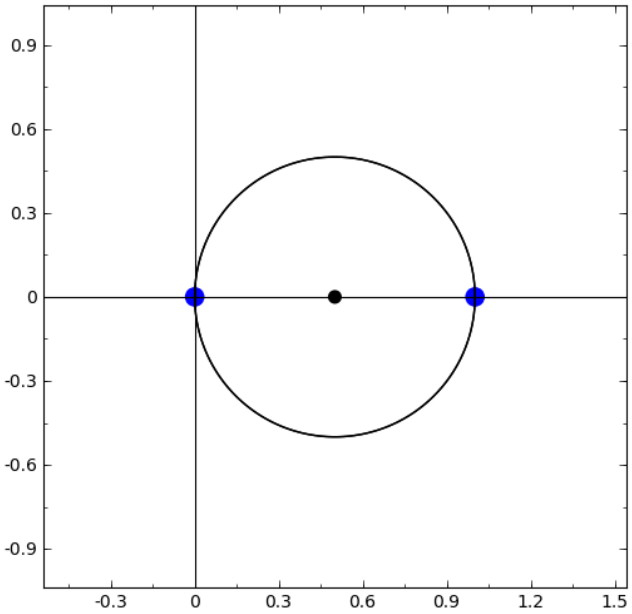
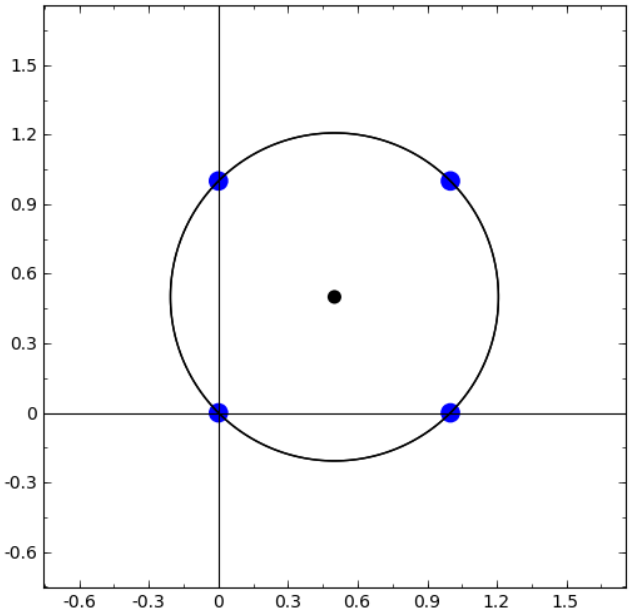
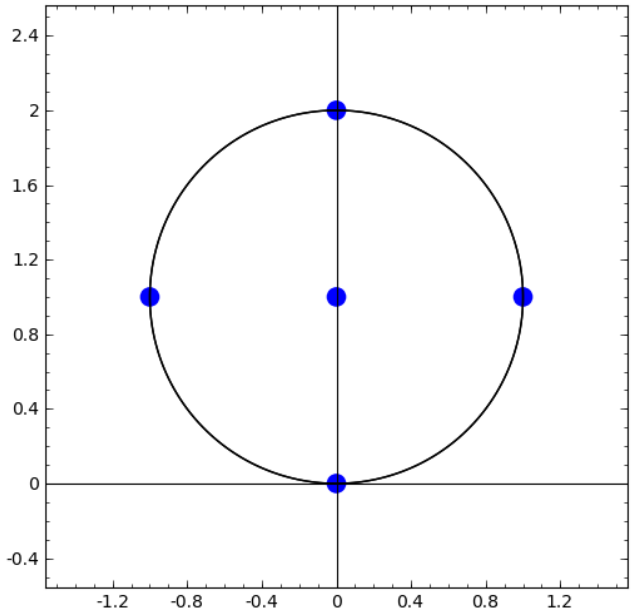
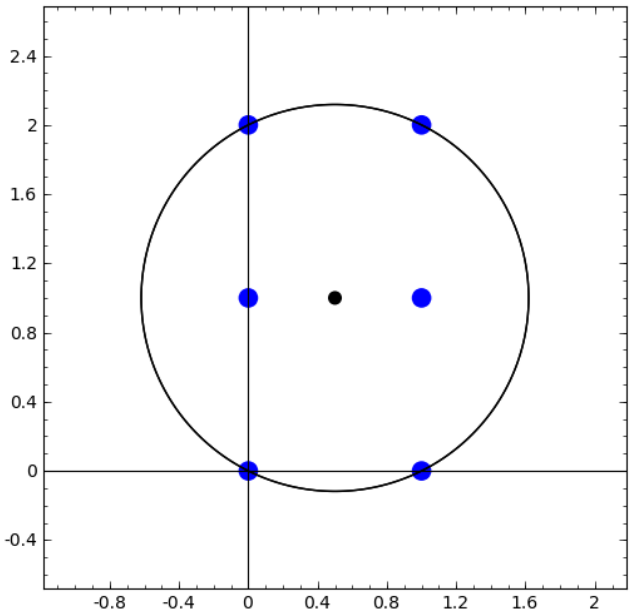
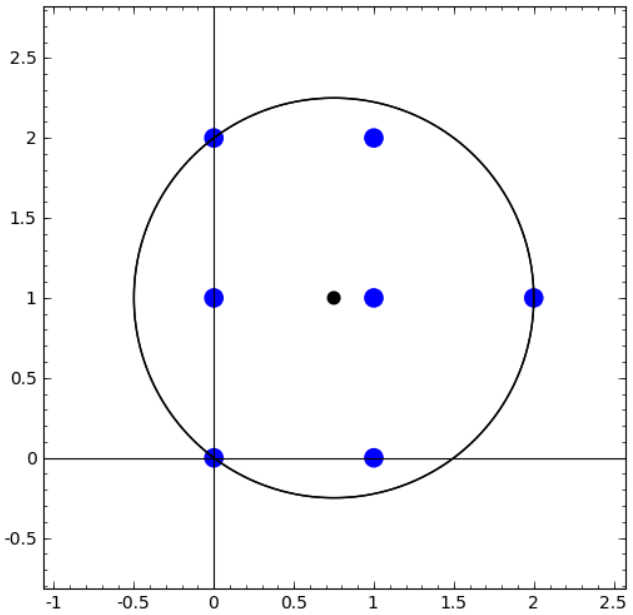
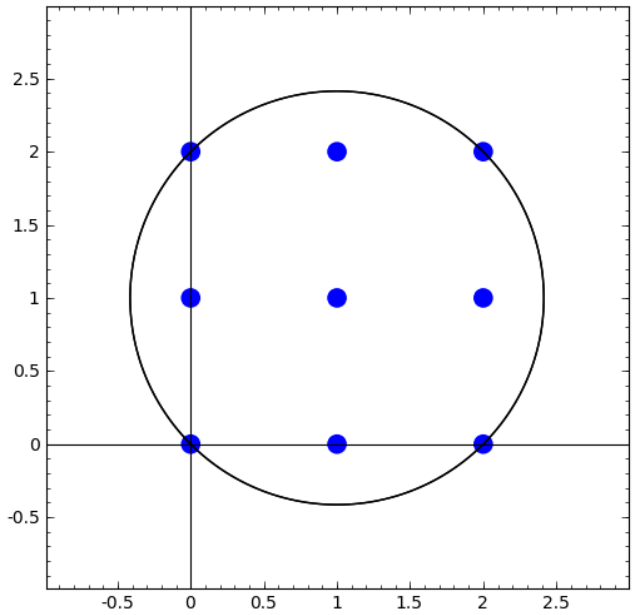
$$\gamma = \lim_{n\to\infty} \left(\sum_{k=2}^{n} \frac{1}{d_k} - \log n \right).$$

La constante δ

Généralisation en deux dimensions :

$$\delta = \lim_{n\to\infty} \left(\sum_{k=2}^{n} \frac{1}{\pi r_k^2} - \log n \right)$$

où r_k est le rayon du plus petit disque de $\mathbb{R}^2$ contenant au moins k points de $\mathbb{Z}^2$.

Historique

δ introduite par Masser (CRAS, 1980).

Conjecture (Gramain, 1982)

$$\delta = 1.822825...$$

Gramain et Weber (Math. of Comp., 1985) avec 175 heures de calcul sur un 6502 en BASIC :

$$1.811447299 < \delta < 1.897327117.$$

Objectif : calculer la 2e décimale après la virgule...

Borne supérieure

$$r_k < \sqrt{\frac{k-1}{\pi}}.$$

Découle d'un résultat classique de Pólya et Szegö (1976) : si un domaine D du plan d'aire A est compact, alors il existe une translation de D qui contient au moins $\lfloor A \rfloor + 1$ points entiers.

En prenant pour D un disque de rayon $\sqrt{\frac{k-1}{\pi}}$, soit d'aire $k-1$, on a au moins k points par translation.

Bornes inférieures

Pour $k \geq 1$:

$$\frac{\sqrt{\pi(k-1)+4}-2}{\pi} < r_k$$

Pour $k \geq 6$:

$$\frac{\sqrt{\pi(k-6)+2}-\sqrt{2}}{\pi} \leq r_k$$

La deuxième borne est meilleure que la première pour $k \geq 76$.

Existence de la limite

Theorem (Gramain, Masser, 1985)

La suite $\delta_n = \sum_2^n 1/(\pi r_k^2) - \log n$ *est croissante et bornée.*

En effet, $\delta_{n+1} - \delta_n = 1/(\pi r_{n+1}^2) - \log(1 + 1/n)$. Comme $r_{n+1} < \sqrt{\pi/n}$:

$$\delta_{n+1} - \delta_n > \frac{1}{n} - \log(1 + 1/n) > 0.$$

La borne $r_k > \frac{\sqrt{\pi(k-1)+4}-2}{\pi}$ donne pour $k \geq 38$:

$$\frac{1}{\pi r_k^2} < \frac{1}{k} + \frac{3}{k^{3/2}},$$

ce qui permet de conclure.

Un disque *minimal* D_k est un disque centré en $(x, y) \in \mathbb{R}^2$, contenant au moins k points entiers, et de rayon minimal. Son périmètre est noté Γ_k.

Proposition (Gramain, Weber, 1985)

Pour $k \geq 3$, si Γ_k ne contient pas 3 points entiers, alors Γ_k a un diamètre avec deux points entiers.

On parle alors de *disque exceptionnel* (aucun pour $k \leq 1500$).

Proposition (Gramain, Weber, 1985)

Si Γ_k contient au moins 3 points entiers, alors il admet un triangle inscrit avec sommets entiers et angles $\leq \pi/2$.

Proposition

r_k^2 est rationnel.

Trivial pour les disques exceptionnels ($r_k = m + 1/2$).
On peut supposer qu'un des trois points entiers du bord est l'origine, soit O.
Comme le triangle a des angles $\leq \pi/2$, on peut prendre les deux autres points B et C dans le premier quadrant.
Le centre du cercle circonscrit au triangle (OBC) est (x, y) avec :

$$x = \frac{y_C|B|^2 - y_B|C|^2}{D}, y = \frac{x_B|C|^2 - x_C|B|^2}{D},$$

et

$$D = 2(x_B y_C - x_C y_B), \quad r^2 = \frac{|B|^2|C|^2|B - C|^2}{D^2}$$

Le calcul de Gramain et Weber

1. Calcul exact de r_k^2 jusque $k = 1401$.
2. Encadrement de r_k pour $1401 \leq k \leq 1.364 \cdot 10^7$
3. Borne supérieure $\sqrt{(k-1)/\pi}$ et borne inférieure de Chaix pour $k > 1.364 \cdot 10^7$:

$$k < \pi r^2 + 30.84274723 r^{2/3}$$

qui donne

$$\frac{1}{\pi r_k^2} < \frac{1}{k} + \frac{21.05893628}{k^{5/3}}$$

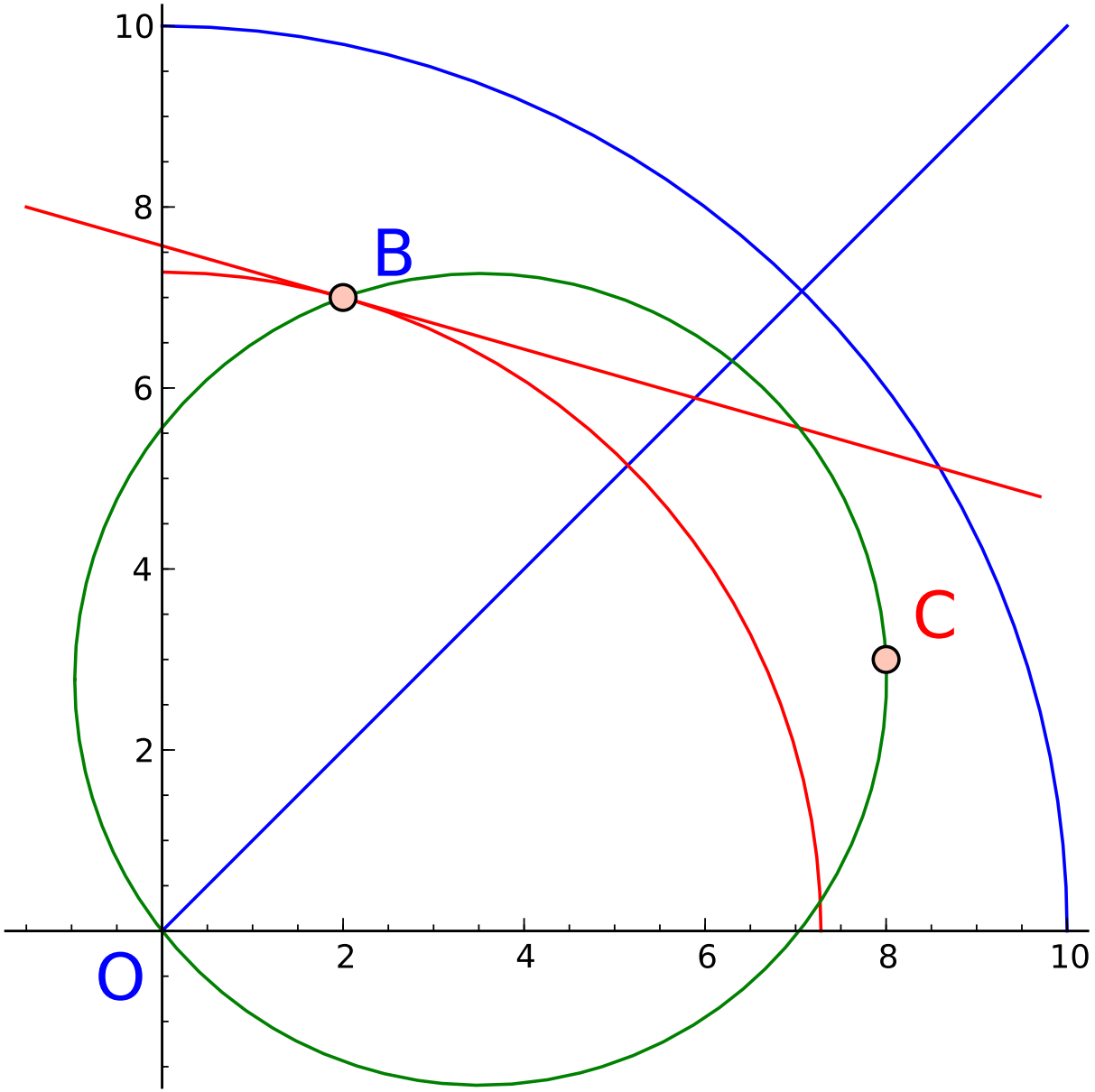
Soit $s_n = \sum_2^n 1/(\pi r_k^2)$.

Lemme

Pour $n \geq 1.364 \cdot 10^7$, on a

$$s_n - \log n + \frac{1}{2n} < \delta < s_n - \log(n + \frac{1}{2}) + \frac{31.58840442}{n^{2/3}}$$

Calcul exact de r_k



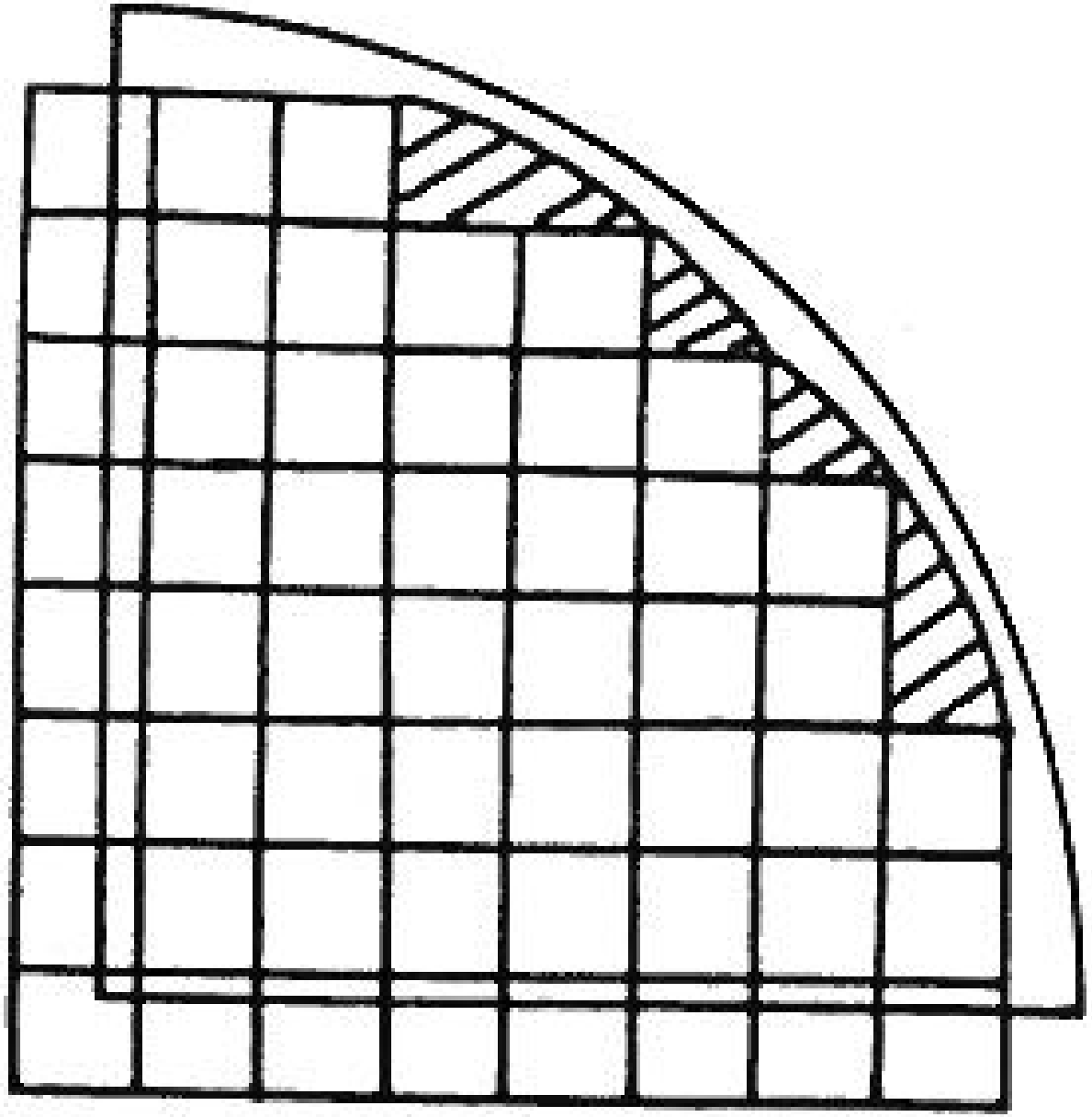
k	r_k^2
298862	95119.46000432795
298863	95119.46000918466
⋮	⋮
547821	348725/2
⋮	⋮
547868	348725/2
⋮	⋮
999999	3920141408851265/12316023458

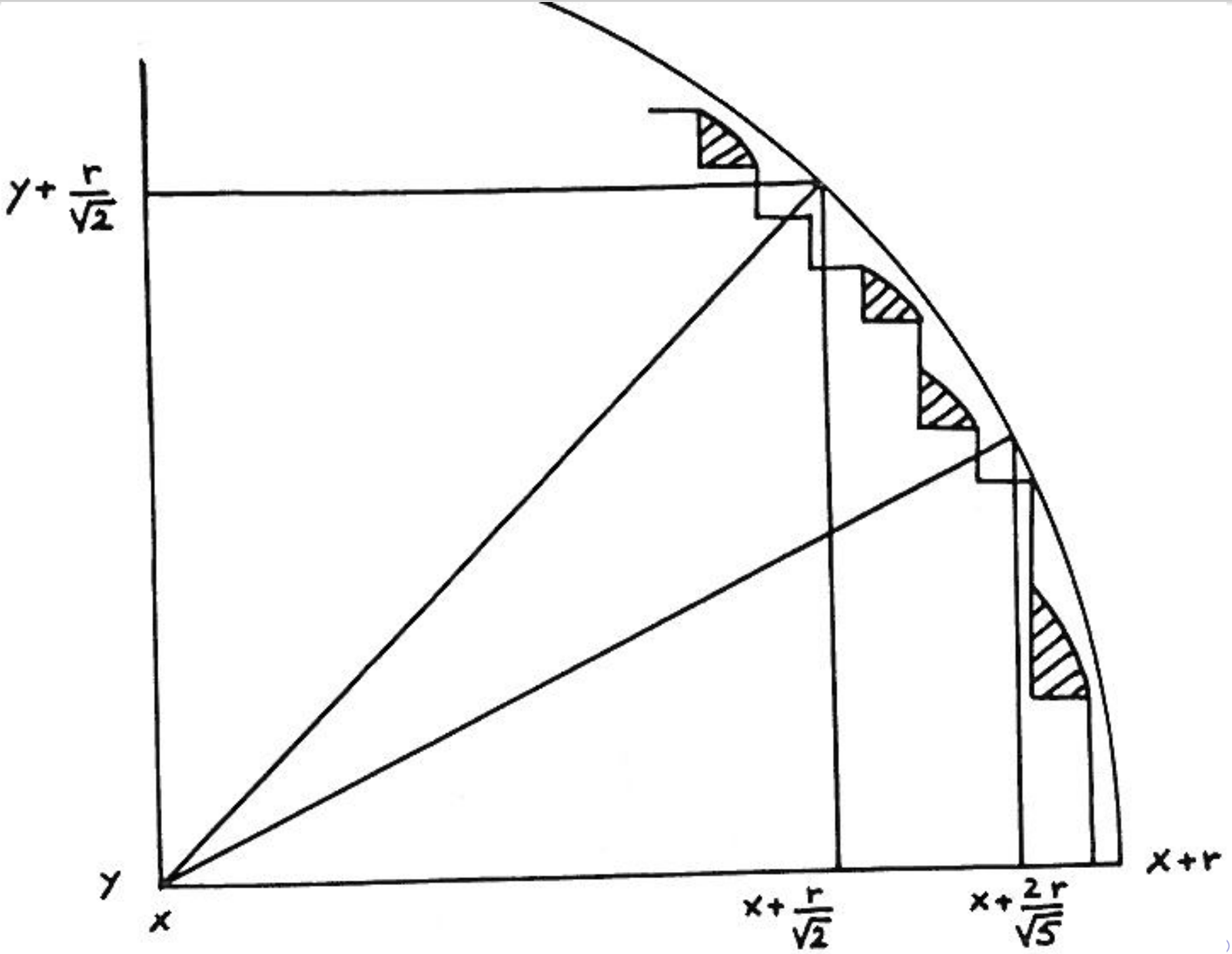
Encadrement de r_k : borne inférieure

Méthode de Gramain et Weber : on minore l'aire entre les carrés au bord du disque. Donne une meilleure borne que celles analytiques pour $k \geq 2798$.

Borne de Chaix : meilleure que celle de Gramain et Weber pour $k \geq 13647034$.

Notre méthode : bisection et arithmétique d'intervalles.




$y + \frac{r}{\sqrt{2}}$
y
x
$x + \frac{r}{\sqrt{2}}$
$x + \frac{2r}{\sqrt{5}}$
$x + r$

Borne inférieure pour r_k par bisection et arithmétique d'intervalles

Lemme

Soit une partition de $[0,1]^2$ *en rectangles* (x,y)*, où* x *et* y *sont des intervalles réels, et* r *un réel. Si pour tout rectangle* (x,y)*, le nombre de points entiers du disque de rayon* r *centré en* (x,y) *est inférieur à* k*, alors* $r < r_k$*.*

```
n = 0
for u from floor(x-r) to ceil(x+r) do
   s = max(sqrt(r^2 - (u-x)^2))
   n += ceil(y+s) - floor(y-s) + 1
```

Algo : on fait de la bisection sur les coordonnées du centre dans $[0,1]^2$ jusqu'à obtenir une borne inférieure assez fine.
Exemple pour $k = 999999$: $564.174913450156736 9 < r_k$.
Vraie valeur 564.177306337818, borne de Gramain-Weber 563.801436860656963 3.

Borne supérieure pour r_k par bisection

Lemme

Soient x et y deux réels dans $[0,1]$, et r un réel. Si le nombre de points entiers du disque de rayon r centré en (x,y) est $\geq k$, alors $r_k \leq r$.

Algo : on fait de la bisection sur les coordonnées du centre dans $[0,1]^2$ jusqu'à obtenir une borne supérieure assez fine.

Exemple pour $k = 999999$:

$$r_k \leq 564.179049593301$$

Vraie valeur 564.177306337818, borne analytique 564.1890193578907429.

Nos résultats (travail en cours)

1. Calcul exact de r_k^2 pour $k < 10^6$:

$$15.63523662136671 < s_{999999} < 15.63523662136672$$

2. Encadrement de r_k pour $10^6 \leq k < 216771893$:

$$21.01400837 < s_{216771892} < 21.01403684$$

3. Bornes analytiques pour $216771893 \leq k$ (avec lemme) :

$$s_{216771892} - 19.19433562 < \delta < s_{216771892} - 19.19422686$$

D'où :

$$1.8197275 < \delta < 1.8200998.$$

ce qui invaliderait la conjecture de Gramain ($\delta \approx 1.822825$).

Conclusion

On peut toujours calculer sur ordinateur...

Pour *utiliser* le résultat d'un calcul, il faut une sémantique précise du calcul.

Plusieurs méthodologies disponibles : arrondi exact, arithmétique d'intervalles, RealRAM, preuve formelle...

Références

Accuracy and Stability of Numerical Algorithms, Higham, SIAM, 2002.

Handbook of Floating-Point Arithmetic, Brisebarre, de Dinechin, Jeannerod, Lefèvre, Melquiond, Muller, Revol, Stehlé, Torres, Birkhäuser, 2009.

Arithmétique exacte, conception, algorithmique et performances d'une implémentation informatique en précision arbitraire, Valérie Ménissier-Morain, 1994.

The SIAM 100-Digit Challenge : A Study in High-Accuracy Numerical Computing, SIAM, 2004.

Gourdon and Sebah, Numbers, constants and computation, `numbers.computation.free.fr`

Logiciels

PARI/GP : `pari.math.u-bordeaux.fr`

GNU MPFR : `www.mpfr.org`

MPC : `mpc.multiprecision.org`

MPFI : `mpfi.gforge.inria.fr`

Sage : `sagemath.org`

```
> evalf(sin(2^100));
                    0.4491999480
```

```
> evalf(sin(2^100), 20);
            -0.58645356896925826300
```