Sage : une alternative libre à Magma, Maple, Mathematica et Matlab

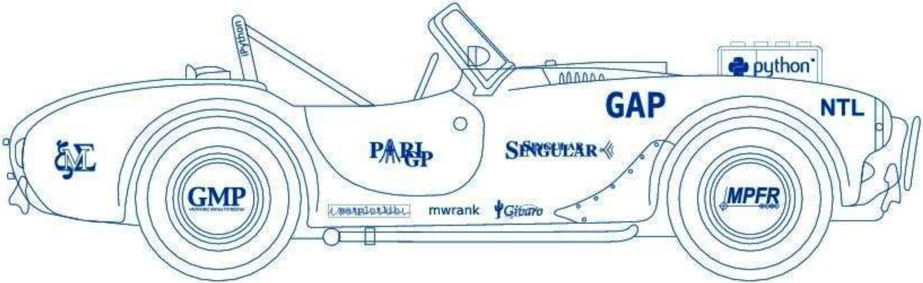
Paul Zimmermann

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE
INRIA
centre de recherche NANCY - GRAND-EST

Journée Plume « Les alternatives libres aux outils propriétaires de maths », 4 février 2010

SAGE

Building »The Car«



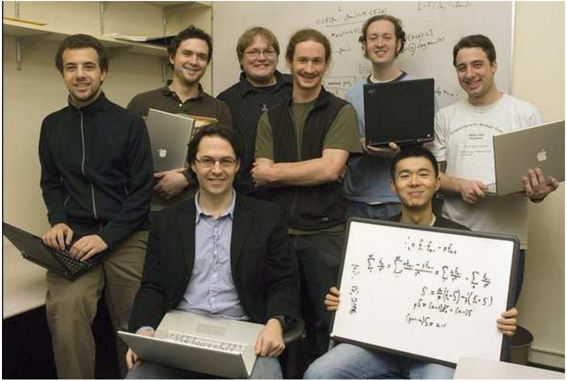
»Every free computer algebra system I've tried has reinvented many times the wheel without being able to build the car.«

Plan

- Historique
- Calculer
- Programmer
- Communiquer avec d’autres logiciels
- Contribuer

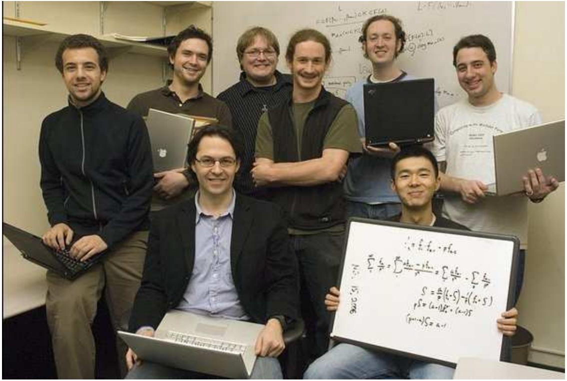
Historique

Cf `http://wstein.org/mathsoftbio/history.pdf`



Historique

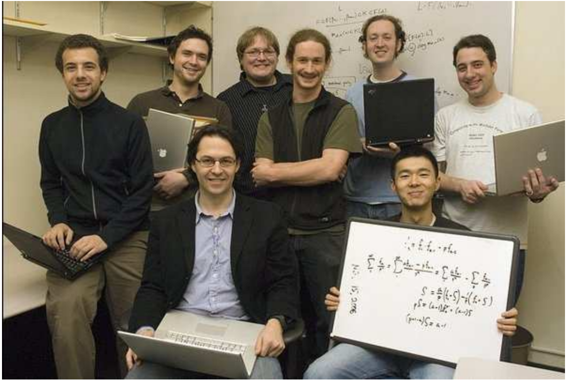
Cf `http://wstein.org/mathsoftbio/history.pdf`



1997, graduate student at Berkeley, découvre Linux.

Historique

Cf `http://wstein.org/mathsoftbio/history.pdf`



1997, graduate student at Berkeley, découvre Linux.

Ken Ribet : “Existe-t-il un premier *p* tel que l’algèbre de Hecke de niveau *p* est ramifiée en *p* ?”

Trouvé un article de Hijikata avec algorithme, mais nécessite de calculer des nombres de classe d'un grand nombre de corps quadratiques. Comment les calculer ?

Trouvé un article de Hijikata avec algorithme, mais nécessite de calculer des nombres de classe d'un grand nombre de corps quadratiques. Comment les calculer ?

WS a entendu parler de Pari : il l'installe sur son ordinateur, et youpi !, Pari peut calculer rapidement les nombres de classe. (premier logiciel mathématique libre pour WS).

1999

“The algebraic number theory algorithms implemented [. . .] were amazing, deep, and very fast”

1999

“The algebraic number theory algorithms implemented [. . .] were amazing, deep, and very fast”

“There was a major bug [. . .] the function `qfbclassno`, silently returned wrong answers”

1999

“The algebraic number theory algorithms implemented [. . .] were amazing, deep, and very fast”

“There was a major bug [. . .] the function `qfbclassno`, silently returned wrong answers”

“in 1999, David Kohel [. . .] told me about implementing algorithms related to his thesis in Magma”

1999

“The algebraic number theory algorithms implemented [. . .] were amazing, deep, and very fast”

“There was a major bug [. . .] the function `qfbclassno`, silently returned wrong answers”

“in 1999, David Kohel [. . .] told me about implementing algorithms related to his thesis in Magma”

“David had implemented code for computing with quaternion algebras, and this was the only implementation of that algorithm in the world”

1999

“The algebraic number theory algorithms implemented [. . .] were amazing, deep, and very fast”

“There was a major bug [. . .] the function `qfbclassno`, silently returned wrong answers”

“in 1999, David Kohel [. . .] told me about implementing algorithms related to his thesis in Magma”

“David had implemented code for computing with quaternion algebras, and this was the only implementation of that algorithm in the world”

“But my algorithm fundamentally relied on exactly the computations in rational quaternion algebras that David Kohel had implemented in Magma”

1999

"The algebraic number theory algorithms implemented [. . .] were amazing, deep, and very fast"

"There was a major bug [. . .] the function `qfbclassno`, silently returned wrong answers"

"in 1999, David Kohel [. . .] told me about implementing algorithms related to his thesis in Magma"

"David had implemented code for computing with quaternion algebras, and this was the only implementation of that algorithm in the world"

"But my algorithm fundamentally relied on exactly the computations in rational quaternion algebras that David Kohel had implemented in Magma"

"I had a thesis to finish"

“David [. . .] was able to give me a copy of Magma for my own computer, which had his code in it. [. . .] I was the first person ever to systematically compute Tamagawa numbers of general modular abelian varieties”

“David [. . .] was able to give me a copy of Magma for my own computer, which had his code in it. [. . .] I was the first person ever to systematically compute Tamagawa numbers of general modular abelian varieties”

“So in 1999 David Kohel put me in a situation where I was fundamentally dependent on a closed source non-free program in order to continue my own research”

“David [. . .] was able to give me a copy of Magma for my own computer, which had his code in it. [. . .] I was the first person ever to systematically compute Tamagawa numbers of general modular abelian varieties”

“So in 1999 David Kohel put me in a situation where I was fundamentally dependent on a closed source non-free program in order to continue my own research”

“Magma was not open source [. . .] Magma was not free. And as a language, Magma was significantly behind C++”

"David [...] was able to give me a copy of Magma for my own computer, which had his code in it. [...] I was the first person ever to systematically compute Tamagawa numbers of general modular abelian varieties"

"So in 1999 David Kohel put me in a situation where I was fundamentally dependent on a closed source non-free program in order to continue my own research"

"Magma was not open source [...] Magma was not free. And as a language, Magma was significantly behind C++"

"At that moment I started designing what would eventually become Sage"

“David [. . .] was able to give me a copy of Magma for my own computer, which had his code in it. [. . .] I was the first person ever to systematically compute Tamagawa numbers of general modular abelian varieties”

“So in 1999 David Kohel put me in a situation where I was fundamentally dependent on a closed source non-free program in order to continue my own research”

“Magma was not open source [. . .] Magma was not free. And as a language, Magma was significantly behind C++”

“At that moment I started designing what would eventually become Sage”

“I then realized that if I did this, I would have to do it myself [. . .] I wouldn’t get to do number theory for years. [. . .] I spent the next 5 years writing and using Magma”

2005

“Since Magma didn’t have any parallel capabilities, I stumbled on some language called “Python” (Version 2.3), which looked a lot like Magma, but was designed for general purpose scripting”

2005

“Since Magma didn’t have any parallel capabilities, I stumbled on some language called “Python” (Version 2.3), which looked a lot like Magma, but was designed for general purpose scripting”

“I had by this point developed a large list of issues with Magma. For example the documentation and examples in the Magma reference manual aren’t automatically tested [. . .] Python had solved the dozens of major problems I had with Magma !”

2005

“Since Magma didn’t have any parallel capabilities, I stumbled on some language called “Python” (Version 2.3), which looked a lot like Magma, but was designed for general purpose scripting”

“I had by this point developed a large list of issues with Magma. For example the documentation and examples in the Magma reference manual aren’t automatically tested [. . .] Python had solved the dozens of major problems I had with Magma !”

2005 : développement de `Manin` (précurseur de Sage)

2005

“Since Magma didn’t have any parallel capabilities, I stumbled on some language called “Python” (Version 2.3), which looked a lot like Magma, but was designed for general purpose scripting”

“I had by this point developed a large list of issues with Magma. For example the documentation and examples in the Magma reference manual aren’t automatically tested [. . .] Python had solved the dozens of major problems I had with Magma !”

2005 : développement de `Manin` (précurseur de Sage)

“I had to make this program trivial for him [David Joyner] to install”

2005

“Since Magma didn’t have any parallel capabilities, I stumbled on some language called “Python” (Version 2.3), which looked a lot like Magma, but was designed for general purpose scripting”

“I had by this point developed a large list of issues with Magma. For example the documentation and examples in the Magma reference manual aren’t automatically tested [. . .] Python had solved the dozens of major problems I had with Magma !”

2005 : développement de `Manin` (précurseur de Sage)

“I had to make this program trivial for him [David Joyner] to install”

“So I setup something that would do it all automatically in a self contained way”

2005

“Since Magma didn’t have any parallel capabilities, I stumbled on some language called “Python” (Version 2.3), which looked a lot like Magma, but was designed for general purpose scripting”

“I had by this point developed a large list of issues with Magma. For example the documentation and examples in the Magma reference manual aren’t automatically tested [. . .] Python had solved the dozens of major problems I had with Magma !”

2005 : développement de `Manin` (précurseur de Sage)

“I had to make this program trivial for him [David Joyner] to install”

“So I setup something that would do it all automatically in a self contained way”

“We had Sage Days in early February [2006], and I released Sage version 1.0 during my talk”

```
sage-1.0.0.1.tar 39.27MB  2006-02-05 15:47
...
sage-1.5.2.tar   72.01MB  2007-01-05 00:06
sage-1.5.3.tar   73.74MB  2007-01-05 19:58
...
sage-2.8.11.tar 158.24MB  2007-11-03 00:41
sage-2.8.12.tar 163.28MB  2007-11-07 15:34
sage-2.8.13.tar 164.87MB  2007-11-21 21:56
sage-2.8.14.tar 164.90MB  2007-11-25 07:59
...
sage-4.3.tar    260.62MB  2009-12-24 17:45
sage-4.3.1.tar  263.51MB  2010-01-21 00:18
```

Calculer

```
----------------------------------------------------------------------
| Sage Version 4.3.1, Release Date: 2010-01-20                       |
| Type notebook() for the GUI, and license() for information.        |
----------------------------------------------------------------------
```

Calculer

```
----------------------------------------------------------------------
| Sage Version 4.3.1, Release Date: 2010-01-20                       |
| Type notebook() for the GUI, and license() for information.        |
----------------------------------------------------------------------
```

```
sage: 3 + 5
8
```

Calculer

```
----------------------------------------------------------------------
| Sage Version 4.3.1, Release Date: 2010-01-20                       |
| Type notebook() for the GUI, and license() for information.        |
----------------------------------------------------------------------
```

```
sage: 3 + 5
8
```

```
sage: 57.1 ^ 100
4.60904368661396e175
```

Calculer

```
----------------------------------------------------------------------
| Sage Version 4.3.1, Release Date: 2010-01-20                       |
| Type notebook() for the GUI, and license() for information.        |
----------------------------------------------------------------------
```

```
sage: 3 + 5
8

sage: 57.1 ^ 100
4.60904368661396e175

sage: matrix([[1,2], [3,4]])^(-1)
[  -2    1]
[ 3/2 -1/2]
```

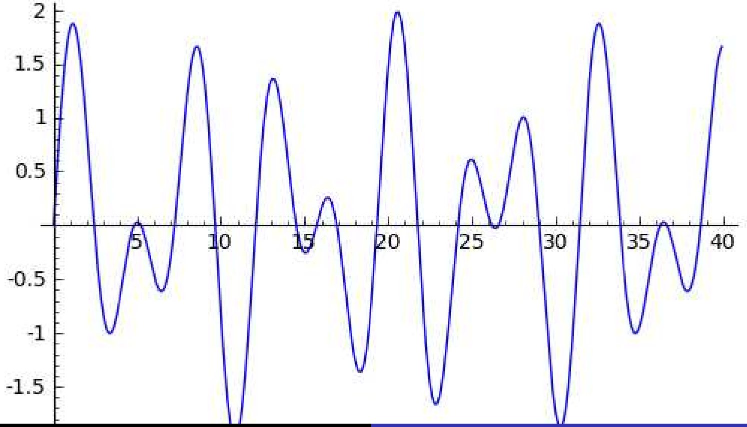
Calculer

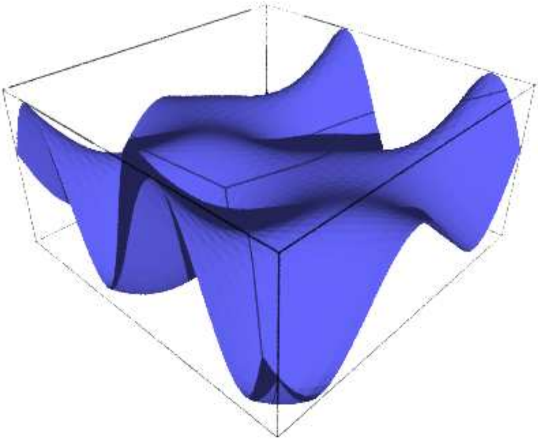
```
----------------------------------------------------------------------
| Sage Version 4.3.1, Release Date: 2010-01-20                       |
| Type notebook() for the GUI, and license() for information.        |
----------------------------------------------------------------------
```

```
sage: 3 + 5
8

sage: 57.1 ^ 100
4.60904368661396e175

sage: matrix([[1,2], [3,4]])^(-1)
[  -2    1]
[ 3/2 -1/2]

sage: x = var('x')
sage: integrate(sqrt(x)*sqrt(1+x), x)
1/4*((x + 1)^(3/2)/x^(3/2) + sqrt(x + 1)/sqrt(x))/((x +
1)^2/x^2 - 2*(x + 1)/x + 1) + 1/8*log(sqrt(x + 1)/
sqrt(x) - 1) - 1/8*log(sqrt(x + 1)/sqrt(x) + 1)
```

```
sage: a = var(’a’)
sage: S = solve(x^2 + x == a, x); S
[x == -1/2*sqrt(4*a + 1) - 1/2,
 x == 1/2*sqrt(4*a + 1) - 1/2]
```

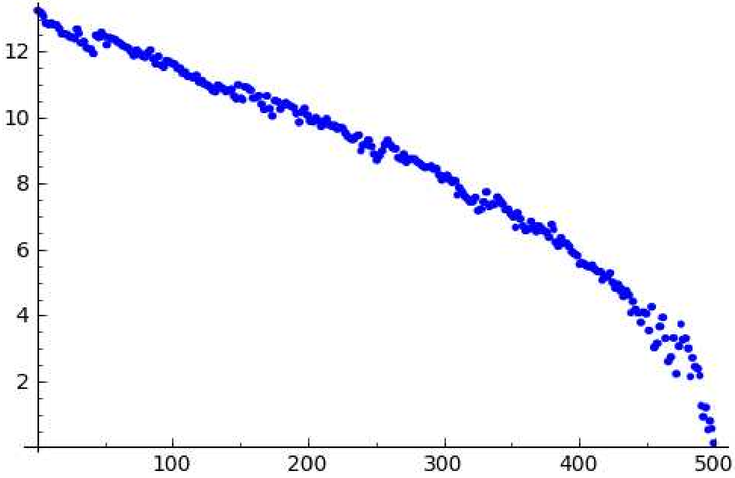
```
sage: a = var(’a’)
sage: S = solve(x^2 + x == a, x); S
[x == -1/2*sqrt(4*a + 1) - 1/2,
 x == 1/2*sqrt(4*a + 1) - 1/2]
sage: S[0].rhs()
-1/2*sqrt(4*a + 1) - 1/2
sage: show(plot(sin(x) + sin(1.6*x), 0, 40))
```





```
sage: var(’y’);
sage: P=plot3d(sin(x-y)*y*cos(x),(x,-3,3),(y,-3,3),
                  mesh=True)
sage: show(P, viewer=’tachyon’)
```

```
m = random_matrix(RDF,500)
sage: e = m.eigenvalues()
sage: w = [(i, abs(e[i])) for i in range(len(e))]
sage: show(points(w))
```



```
sage: factorial(100)
93326215443944152681699238856266700490715968264381621\
46859296389521759999322991560894146397615651828625369\
7920827223758251185210916864000000000000000000000000
sage: time n = factorial(1000000)
CPU times: user 0.81 s, sys: 0.00 s, total: 0.82 s
Wall time: 0.82 s
```

```
sage: factorial(100)
93326215443944152681699238856266700490715968264381621\
46859296389521759999322991560894146397615651828625369\
7920827223758251185210916864000000000000000000000000
sage: time n = factorial(1000000)
CPU times: user 0.81 s, sys: 0.00 s, total: 0.82 s
Wall time: 0.82 s

sage: N(pi, digits=100)
3.1415926535897932384626433832795028841971693993751\
0582097494459230781640628620899862803482534211706 8
```

```
sage: R.<x,y> = QQ[]; F = factor(x^99 + y^99); F
(x + y) * (x^2 - x*y + y^2) * (x^6 - x^3*y^3 + y^6)
* (x^10 - x^9*y + x^8*y^2 - x^7*y^3 + x^6*y^4 -
x^5*y^5 + x^4*y^6 - x^3*y^7 + x^2*y^8 - x*y^9 +
y^10) * (x^20 + x^19*y - x^17*y^3 - x^16*y^4 +
x^14*y^6 + x^13*y^7 - x^11*y^9 - x^10*y^10 -
x^9*y^11 + x^7*y^13 + x^6*y^14 - x^4*y^16 -
x^3*y^17 + x*y^19 + y^20) * (x^60 + x^57*y^3 -
x^51*y^9 - x^48*y^12 + x^42*y^18 + x^39*y^21 -
x^33*y^27 - x^30*y^30 - x^27*y^33 + x^21*y^39 +
x^18*y^42 - x^12*y^48 - x^9*y^51 + x^3*y^57 + y^60)
sage: F.expand()
x^99 + y^99
```

```
sage: R.<x,y> = QQ[]; F = factor(x^99 + y^99); F
(x + y) * (x^2 - x*y + y^2) * (x^6 - x^3*y^3 + y^6)
* (x^10 - x^9*y + x^8*y^2 - x^7*y^3 + x^6*y^4 -
x^5*y^5 + x^4*y^6 - x^3*y^7 + x^2*y^8 - x*y^9 +
y^10) * (x^20 + x^19*y - x^17*y^3 - x^16*y^4 +
x^14*y^6 + x^13*y^7 - x^11*y^9 - x^10*y^10 -
x^9*y^11 + x^7*y^13 + x^6*y^14 - x^4*y^16 -
x^3*y^17 + x*y^19 + y^20) * (x^60 + x^57*y^3 -
x^51*y^9 - x^48*y^12 + x^42*y^18 + x^39*y^21 -
x^33*y^27 - x^30*y^30 - x^27*y^33 + x^21*y^39 +
x^18*y^42 - x^12*y^48 - x^9*y^51 + x^3*y^57 + y^60)
sage: F.expand()
x^99 + y^99

sage: time z = Partitions(10^8).cardinality()
CPU times: user 4.68 s, sys: 0.00 s, total: 4.69 s
Wall time: 4.69 s
sage: str(z)[:40]
’1760517045946249141360373894679135204009’
```

Programmer

- soit en Python directement dans l'interprète
- soit via un fichier Python `*.py` avec `load` ou `attach`
- soit en utilisant le compilateur Cython avec un fichier `*.pyx`

Exemple

Référence : exposé de Robert Bradshaw aux Sage Days 6

Calcul de $0 + 1 + 2 + \cdots + N - 1$:

```
def sum1(N):
   s = 0
   for k in range(N):
      s += k
   return s
```

Exemple

Référence : exposé de Robert Bradshaw aux Sage Days 6

Calcul de $0 + 1 + 2 + \cdots + N - 1$:

```
def sum1(N):
   s = 0
   for k in range(N):
      s += k
   return s
```

```
sage: time sum1(10^6)
CPU times: user 1.78 s, sys: 0.01 s, total: 1.78 s
499999500000
```

La même chose avec Cython

```
sage: cat sum2.pyx
def sum2(long N):
    cdef long s = 0, k
    for k in range(N):
        s += k
    return s
```

La même chose avec Cython

```
sage: cat sum2.pyx
def sum2(long N):
    cdef long s = 0, k
    for k in range(N):
        s += k
    return s
```

```
sage: load sum2.pyx
Compiling sum2.pyx...
```

La même chose avec Cython

```
sage: cat sum2.pyx
def sum2(long N):
    cdef long s = 0, k
    for k in range(N):
        s += k
    return s
```

```
sage: load sum2.pyx
Compiling sum2.pyx...
```

```
sage: time sum2(10^6)
CPU times: user 0.00 s, sys: 0.00 s, total: 0.00 s
499999500000
```

La même chose avec Cython

```
sage: cat sum2.pyx
def sum2(long N):
    cdef long s = 0, k
    for k in range(N):
        s += k
    return s

sage: load sum2.pyx
Compiling sum2.pyx...

sage: time sum2(10^6)
CPU times: user 0.00 s, sys: 0.00 s, total: 0.00 s
499999500000

sage: time sum2(10^9)
CPU times: user 1.01 s, sys: 0.00 s, total: 1.01 s
499999999500000000
```

La même chose avec Cython et GMP

```
sage: sum2(5*10^9)
-5946744076209551616
```

La même chose avec Cython et GMP

```
sage: sum2(5*10^9)
-5946744076209551616
```

```
sage: cat sum3.pyx
include "gmp.pxi"
def sum3(long N):
    cdef long k
    cdef mpz_t s
    mpz_init_set_ui (s, 0)
    for k from 0 <= k < N:
        mpz_add_ui (s, s, k)
    res = int(mpz_to_str(s))
    mpz_clear (s)
    return res
```

La même chose avec Cython et GMP

```
sage: sum2(5*10^9)
-5946744076209551616
```

```
sage: cat sum3.pyx
include "gmp.pxi"
def sum3(long N):
    cdef long k
    cdef mpz_t s
    mpz_init_set_ui (s, 0)
    for k from 0 <= k < N:
        mpz_add_ui (s, s, k)
    res = int(mpz_to_str(s))
    mpz_clear (s)
    return res
```

```
sage: load sum3.pyx
Compiling sum3.pyx...
sage: time sum3(5*10^9)
CPU times: user 89.46 s, sys: 0.10 s, total: 89.56
12499999997500000000L
```

Le concept de spkg

Sage 4.3.1 utilise 96 “packages”, dont `atlas-3.8.3.p10`, `numpy-1.3.0.p2`, `matplotlib-0.99.1.p4`, `scipy-0.7.p3`, `python-2.6.4.p4`, `libfplll-3.0.12.p0`, `linbox-1.1.6.p2`, `maxima-5.20.1`, `pari-2.3.3.p7`.

```
$ cd /usr/local/sage-4.3.1/sage/spkg/standard
$ tar jtvf mpfr-2.4.1.p0.spkg
   0 2009-07-24 08:01 mpfr-2.4.1.p0/
   0 2009-07-17 02:51 mpfr-2.4.1.p0/patches/
6271 2009-07-17 02:25 mpfr-2.4.1.p0/patches/mpn_exp.c
 894 2009-07-17 02:33 mpfr-2.4.1.p0/patches/mpn_exp.c.patch
3963 2009-07-18 00:56 mpfr-2.4.1.p0/SPKG.txt
...
5271 2009-07-18 00:47 mpfr-2.4.1.p0/spkg-install
 106 2007-09-14 22:03 mpfr-2.4.1.p0/spkg-check
   0 2009-07-17 02:54 mpfr-2.4.1.p0/src/
1327 2009-02-20 10:43 mpfr-2.4.1.p0/src/setmin.c
```

```
$ tar jxf mpfr-2.4.1.p0.spkg; cd mpfr-2.4.1.p0
$ cat SPKG.txt
...
== SPKG Maintainers ==
 * Michael Abshoff
 * David Kirkby

== Upstream Contact ==
The MPFR website is located at http://mpfr.org/
The MPFR team can be contact via the MPFR mailing list:

== Dependencies ==
 * GMP

== Changelog ==
=== mpfr-2.4.1p0 (David Kirkby, July 17th 2009) ===
 * Sage TRAC #6453 http://sagetrac.org/sage_trac/ticket/
...
=== mpfr-2.4.1 (Michael Abshoff, March 2nd, 2009) ===
 * update to the official MPFR 2.4.1 release
...
```

Comment installer Sage sur son ordinateur ?

Télécharger `sage-4.3.1.tar` depuis `sagemath.org` (276MB).

```
$ tar xf sage-4.3.1.tar
$ cd sage-4.3.1
$ make
```

Aller prendre un café (ou plusieurs). . .

Quelques commandes utiles pour commencer

```
sage: a=17
sage: a.<tab>
a.abs                          a.kronecker
a.additive_order               a.lcm
a.base_extend                  a.leading_coefficient
...
sage: a.abs?
Return the absolute value of self.  (This just calls
the __abs__ method, so it is equivalent to the abs()
built-in function.)

    EXAMPLES::

    sage: RR(-1).abs()
    1.00000000000000
```

```
sage: search_src("integration",extra1="numerical")
gsl/all.py:14:from integration import numerical_integral
gsl/integration.pyx:117:       that implements numerical
            integration using Maxima.  It is potentially
ext/fast_eval.pyx:4:For many applications such as
            numerical integration, differential
ext/fast_callable.pyx:4:For many applications such as
            numerical integration, differential
functions/piecewise.py:359:        numerical integration
            based on a subdivision into N subintervals.
functions/piecewise.py:394:        numerical integration
            based on a subdivision into N subintervals.
functions/piecewise.py:444:        for numerical integra
            based on a subdivision into N
functions/transcendental.py:27:from sage.gsl.integration
            import numerical_integral
interfaces/maxima.py:2021:        Note that GP also does
            numerical integration, and can do so to very
calculus/desolvers.py:811:    Used to determine bounds f
            numerical integration.
```

Polynômes

```
sage: P.<x> = PolynomialRing(GF(17))
sage: p = P.random_element(); p
16*x^2 + 6*x + 9
sage: p^3
16*x^6 + x^5 + 4*x^4 + 11*x^3 + 15*x^2 + 13*x + 15
sage: p.roots()
[(4, 1), (2, 1)]
```

Matrices

```
sage: m = Matrix(P,2,2)
sage: m.randomize(); m

[13*x^2 + 12*x + 15     11*x^2 + x + 8]
[          4*x + 14  16*x^2 + 11*x + 1]
sage: m.rank() # used to fail (#5014)
2
sage: m.det()
4*x^4 + 2*x^3 + 6*x^2 + 12*x + 5
sage: factor(_)
(4) * (x^4 + 9*x^3 + 10*x^2 + 3*x + 14)
```

```
sage: z = gp(m)
sage: z
[Mod(13, 17)*x^2 + Mod(12, 17)*x + Mod(15, 17),
 Mod(11, 17)*x^2 + Mod(1, 17)*x + Mod(8, 17);
 Mod(4, 17)*x + Mod(14, 17),
 Mod(16, 17)*x^2 + Mod(11, 17)*x + Mod(1, 17)]
sage: type(z)
<class 'sage.interfaces.gp.GpElement'>
sage: z.matrank()
2
```

```
sage: gp.eval(’intnum(x=1,[1],sin(x)/x^2)’)
’0.50720744201747388836088625125948735826’
sage: res = eval(_); res
0.50720744201747392

sage: s = gp.eval(’Mod(8,17)’); s
’Mod(8, 17)’
sage: a = eval(s); a
8
sage: a.parent()
Ring of integers modulo 17
```

Interface sous l’interprète Sage

```
sage: %gp

  --> Switching to GP/PARI interpreter <--

,,

gp: a = factorint(2^128+1)


[59649589127497217 1]

[5704689200685129054721 1]

gp: quit

  --> Exiting back to SAGE <--
```

```
sage: m = gp(’a’); m
[59649589127497217, 1; 5704689200685129054721, 1]
sage: m * m
[3558073483084938890843851471799810,
    59649589127497218;
340282366920938469168063808116897266178,
    5704689200685129054722]
```

Using two different packages

```
sage: %maple
maple: m := linalg[hilbert](5);
m := matrix([[1, 1/2, 1/3, 1/4, 1/5], [1/2, 1/3, 1/4,
1/5, 1/6], [1/3, 1/4, 1/5, 1/6, 1/7], [1/4, 1/5, 1/6,
1/7, 1/8], [1/5, 1/6, 1/7, 1/8, 1/9]])
maple: d := linalg[charpoly](m, x);
d := x^5-563/315*x^4+735781/2116800*x^3-852401/
222264000*x^2+61501/53343360000*x-1/266716800000
maple: quit
sage: d = maple('d')
sage: d = gp(d)
sage: d.polroots()
[0.000003287928772171862957115004760 + 0.E-28*I,
0.0003058980401511917268794978407 + 0.E-28*I,
0.01140749162341980655945145887 + 0.E-28*I,
0.2085342186110133359050025101 + 0.E-28*I,
1.567050691098230795533011006 + 0.E-28*I]~
```

Ressources

A Tour of Sage :
`http://sagemath.org/doc/a_tour_of_sage/`

Ressources

A Tour of Sage :
`http://sagemath.org/doc/a_tour_of_sage/`

Tutoriel :
`http://www.sagemath.org/doc/tutorial/index.html`

Ressources

A Tour of Sage :
`http://sagemath.org/doc/a_tour_of_sage/`

Tutoriel :
`http://www.sagemath.org/doc/tutorial/index.html`

Livre *Sage for newbies* de Ted Kosan.

Ressources

A Tour of Sage :
`http://sagemath.org/doc/a_tour_of_sage/`

Tutoriel :
`http://www.sagemath.org/doc/tutorial/index.html`

Livre *Sage for newbies* de Ted Kosan.

Même des vidéos expliquant comment utiliser Sage !

Ressources

A Tour of Sage :
`http://sagemath.org/doc/a_tour_of_sage/`

Tutoriel :
`http://www.sagemath.org/doc/tutorial/index.html`

Livre *Sage for newbies* de Ted Kosan.

Même des vidéos expliquant comment utiliser Sage !

sage-support :
`http://groups.google.com/group/sage-support`
sage-edu :
`http://groups.google.com/group/sage-edu`

Comment contribuer ?

- envoyer un bug report sur la liste `sage-support`

Comment contribuer ?

- envoyer un bug report sur la liste `sage-support`
- mieux : le poster sur `http://trac.sagemath.org/sage_trac`

Comment contribuer ?

- envoyer un bug report sur la liste `sage-support`
- mieux : le poster sur `http://trac.sagemath.org/sage_trac`
- encore mieux : proposer un *patch* !

Comment corriger sa propre version

Chaque utilisateur a une version de développement de Sage !

Comment corriger sa propre version

Chaque utilisateur a une version de développement de Sage !

Sage 4.3 installé à partir des sources, dans

```
SAGE=/usr/local/sage-4.3-core2
sage: plot?
...
        EXAMPLES: We plot the sin function::
```

- éditer le fichier `SAGE/devel/sage-main/sage/plot/plot.py`
- lancer `sage -br` (build and run)

```
sage: plot?
...
        EXAMPLES: We plot the sin function:
```

Comment exporter son patch

```
sage: hg_sage.commit()
```

(taper ’q’ et mettre un message de commit)

Comment exporter son patch

```
sage: hg_sage.commit()
```

(taper ’q’ et mettre un message de commit)

```
sage: hg_sage.log()
```

(taper ’q’)

Comment exporter son patch

```
sage: hg_sage.commit()
```

(taper ’q’ et mettre un message de commit)

```
sage: hg_sage.log()
```

(taper ’q’)

Exporter le patch :

```
sage: hg_sage.export(’tip’)
```

Comment exporter son patch

```
sage: hg_sage.commit()
```

(taper 'q' et mettre un message de commit)

```
sage: hg_sage.log()
```

(taper 'q')

Exporter le patch :

```
sage: hg_sage.export('tip')
```

```
tarte% ls -l *patch*
-rw-r--r-- 1 zimmerma caramel 707 2010-02-02
16:47 13535.patch
```

```
tarte% cat 13535.patch
# HG changeset patch
# User Paul Zimmermann <zimmerma@loria.fr>
# Date 1265125578 -3600
# Node ID 40b1293e7fbe4fe22a267103e3e90bf37670f647
# Parent  21efb0b3fc474972b5c7f617d99173536a3d79d0
fixed typo

diff -r 21efb0b3fc47 -r 40b1293e7fbe sage/plot/plot.py
--- a/sage/plot/plot.py Thu Dec 24 09:44:02 2009 -0800
+++ b/sage/plot/plot.py Tue Feb 02 16:46:18 2010 +0100
@@ -2210,7 +2210,7 @@
     possibility of, e.g., sampling sin only at multiples of
     ‘2\pi‘, which would yield a very misleading graph.

-    EXAMPLES: We plot the sin function::
+    EXAMPLES: We plot the sin function:

         sage: P = plot(sin, (0,10)); print P
         Graphics object consisting of 1 graphics primitive
```

Comment soumettre son patch

- aller sur `http://trac.sagemath.org/sage_trac`
- s'enregistrer puis cliquer sur *New Ticket*
- Summary : `[with patch] typo in documentation`
- Type : `enhancement`, Priority : `trivial`
- Milestone : `sage-4.3.2`, Component : `documentation`
- cliquer sur “I have files to attach to this ticket”
- cliquer sur “Preview” puis sur “Create ticket”
- uploader le patch
- cliquer sur “View Tickets” puis sur “needs review”

`http://trac.sagemath.org/sage_trac/ticket/8153`

Revue du code

```
   http://trac.sagemath.org/sage_trac/ticket/7876
symbolic expression displayed wrong [3 weeks ago]
Reported by:  iandrus
sage: f=(1/2-1/2*I )*sqrt(2)
sage: f
-(1/2*I + 1/2)*sqrt(2)
```

Revue du code

```
   http://trac.sagemath.org/sage_trac/ticket/7876
symbolic expression displayed wrong [3 weeks ago]
Reported by:  iandrus
sage: f=(1/2-1/2*I )*sqrt(2)
sage: f
-(1/2*I + 1/2)*sqrt(2)

trac_7876-pynac_print.take2.patch  Download  (2.6 KB)
added by burcin 2 weeks ago
```

Revue du code

```
   http://trac.sagemath.org/sage_trac/ticket/7876
symbolic expression displayed wrong [3 weeks ago]
Reported by:  iandrus
sage: f=(1/2-1/2*I )*sqrt(2)
sage: f
-(1/2*I + 1/2)*sqrt(2)

trac_7876-pynac_print.take2.patch  Download  (2.6 KB)
added by burcin 2 weeks ago

Changed 13 days ago by jason:
I get a single reject from the patch,
in symbolic/random_tests.py on sage.math.
```

Revue du code

```
   http://trac.sagemath.org/sage_trac/ticket/7876
symbolic expression displayed wrong [3 weeks ago]
Reported by:  iandrus
sage: f=(1/2-1/2*I )*sqrt(2)
sage: f
-(1/2*I + 1/2)*sqrt(2)

trac_7876-pynac_print.take2.patch  Download  (2.6 KB)
added by burcin 2 weeks ago

Changed 13 days ago by jason:
I get a single reject from the patch,
in symbolic/random_tests.py on sage.math.

Changed 9 days ago by rossk:
* status changed from needs_review to positive_review
sage: f=(1/2-1/2*I )*sqrt(2); f
(-1/2*I + 1/2)*sqrt(2)
```

Efficacité : entiers

Core 2 à 2.83Ghz, sous Linux.

Maple 13 :

```
> a := 3^20959032: b := 7^11832946:
> st:=time(): c:=a*b: time()-st;
                          1.523
> st:=time(): d:=c/a: time()-st;
                         1419.373
```

Efficacité : entiers

Core 2 à 2.83Ghz, sous Linux.

Maple 13 :

```
> a := 3^20959032: b := 7^11832946:
> st:=time(): c:=a*b: time()-st;
                          1.523
> st:=time(): d:=c/a: time()-st;
                         1419.373
```

Sage 4.3 :

```
sage: a = 3^20959032; b = 7^11832946
sage: time c=a*b
CPU times: user 0.73 s, sys: 0.06 s, total: 0.79 s
sage: time d=c/a
CPU times: user 7.77 s, sys: 0.12 s, total: 7.89 s
```

Efficacité : flottants

Maple 13 :

```
> Digits:=5000000: a:=3.^10479516: b:=7.^5916473:
> st:=time(): c:=a*b: time()-st;
                                     5.964
> st:=time(): d:=c/a: time()-st;
                                     5.945
```

Efficacité : flottants

Maple 13 :

```
> Digits:=5000000: a:=3.^10479516: b:=7.^5916473:
> st:=time(): c:=a*b: time()-st;
                                          5.964
> st:=time(): d:=c/a: time()-st;
                                          5.945
```

Sage 4.3 :

```
sage: R = RealField(ceil(5000000*log(10)/log(2)))
sage: a = R(3)^10479516; b = R(7)^5916473
sage: time c=a*b
CPU times: user 0.32 s, sys: 0.00 s, total: 0.32 s
sage: time d=c/a
CPU times: user 1.62 s, sys: 0.00 s, total: 1.62 s
```

Efficacité : polynômes

Maple 13 :

```
> st:=time(): d:=expand((a+b+c+1)^100): time()-st;
                                     0.748

> st:=time(): factor(d), time()-st;
                        100
         (a + c + b + 1)   , 8.857
```

Efficacité : polynômes

Maple 13 :

```
> st:=time(): d:=expand((a+b+c+1)^100): time()-st;
                                     0.748


> st:=time(): factor(d), time()-st;
                        100
         (a + c + b + 1)   , 8.857
```

Sage 4.3 :

```
sage: var('a,b,c'); time d=expand((a+b+c+1)^100)
CPU times: user 4.06 s, sys: 0.17 s, total: 4.23 s
sage: time e=factor(d)
<does not answer in reasonable time>
sage: P.<a,b,c> = PolynomialRing(QQ)
sage: time d=(a+b+c+1)^100
CPU times: user 10.28 s, sys: 0.07 s, total: 10.35
sage: time e=d.factor()
CPU times: user 28.87 s, sys: 0.36 s, total: 29.23
```

Efficacité : matrices

Maple 13 :

```
> m:=LinearAlgebra:-HilbertMatrix(100):
> st:=time(): i:=LinearAlgebra:-MatrixInverse(m): t
                                     6.581
> st:=time(): evalm(i &* m): time()-st;
                                     4.200
```

Efficacité : matrices

Maple 13 :

```
> m:=LinearAlgebra:-HilbertMatrix(100):
> st:=time(): i:=LinearAlgebra:-MatrixInverse(m): t
                                     6.581
> st:=time(): evalm(i &* m): time()-st;
                                     4.200
```

Sage 4.3 :

```
sage: n=100; m=matrix(QQ,n)
sage: for i in range(n):
         for j in range(n):
            m[i,j]=1/(1+i+j)
sage: time i=m^(-1)
CPU times: user 8.11 s, sys: 0.05 s, total: 8.16 s
sage: time j=i*m
CPU times: user 0.16 s, sys: 0.00 s, total: 0.16 s
```

Sage Days 20

CIRM, Luminy, 22-26 février 2010