Le logiciel SAGE

François Ducrot

Journées Mathrice - Angers 2009

Différents besoins de logiciels mathématiques

Calcul formel : On manipule des expressions formelles :

```
eq1:= x^2+y^2=R^2;
eq2:=(x-R)^2+y^2 = r^2;
sol:=solve({eq1,eq2},{x,y});
IJ:=allvalues(sol);
```

$$eq1 := x^2 + y^2 = R^2$$

$$eq2 := (x - R)^2 + y^2 = r^2$$

$$sol := \{x = -\frac{-2R^2 + r^2}{2R}, y = \frac{1}{2}\frac{\text{RootOf}(\_Z^2 - 4R^2 + r^2, label = \_L3)\, r}{R}\}$$

$$IJ := \{x = -\frac{-2R^2 + r^2}{2R}, y = \frac{\sqrt{4R^2 - r^2}\, r}{2R}\}, \{x = -\frac{-2R^2 + r^2}{2R}, y = -\frac{\sqrt{4R^2 - r^2}\, r}{2R}\}$$

Différents besoins de logiciels mathématiques

Calcul scientifique : On manipule des tableaux de nombres en virgule flottante. Exemple : vérification de $\sum_{1}^{\infty} \frac{1}{k^2} = \frac{\pi^2}{6}$

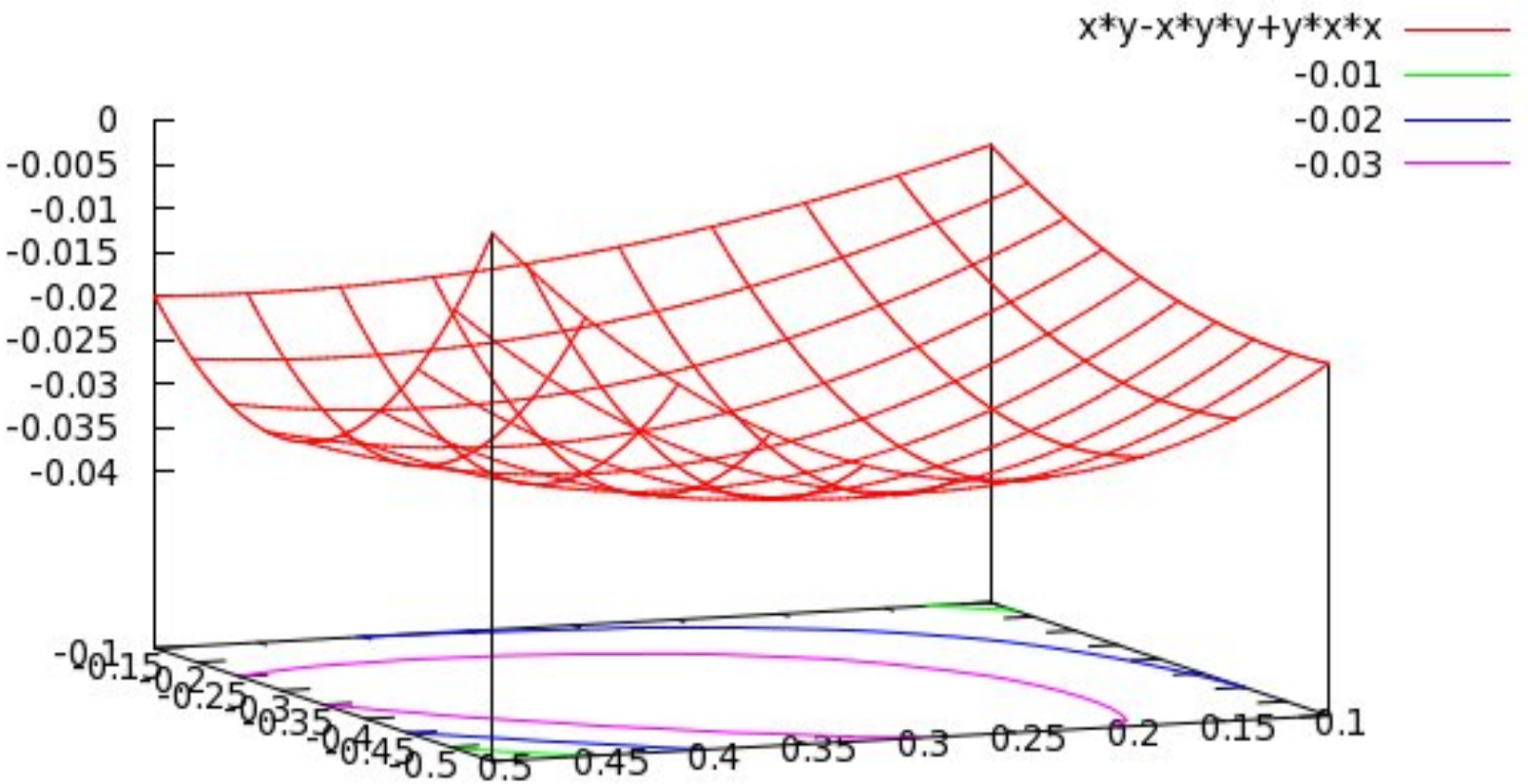
```
-->sum([1:1000000]^(-2))
 ans  =

    1.6449331

-->%pi^2/6
 ans  =

    1.6449341
```

Représentation graphique :



Différentes solutions logicielles

	Logiciels commerciaux	**Logiciels libres**
Calcul formel généraliste	Maple Mathematica Mupad(†) Magma	Maxima Axiom YaCAS
Calcul formel spécialisé	librairies développées par des universitaires	Macaulay (algèbre) Singular (algèbre) Gap (groupes) Pari/gp
Calcul scientifique	Matlab Splus	Octave Scilab R
Visualisation		Gnuplot

SAGE

- SAGE se veut une alternative libre à l'ensemble des logiciels propriétaires.

SAGE

- SAGE se veut une alternative libre à l'ensemble des logiciels propriétaires.
- Développé initialement par deux mathématiciens spécialistes de théorie des nombres, et gros utilisateurs de magma : William Stein (Univ. Washington) et David Joyner.

SAGE

- SAGE se veut une alternative libre à l'ensemble des logiciels propriétaires.
- Développé initialement par deux mathématiciens spécialistes de théorie des nombres, et gros utilisateurs de magma : William Stein (Univ. Washington) et David Joyner.
- SAGE combine un grand nombre de packages open-source. L'interface, écrite en python, permet d'utiliser conjointement les capacités de calcul de ces différents packages.

SAGE

- SAGE se veut une alternative libre à l'ensemble des logiciels propriétaires.
- Développé initialement par deux mathématiciens spécialistes de théorie des nombres, et gros utilisateurs de magma : William Stein (Univ. Washington) et David Joyner.
- SAGE combine un grand nombre de packages open-source. L'interface, écrite en python, permet d'utiliser conjointement les capacités de calcul de ces différents packages.
- SAGE peut être utilisé comme interface pour différents logiciels commerciaux (maple, mathematica, magma), ou libres.

SAGE

- SAGE se veut une alternative libre à l'ensemble des logiciels propriétaires.
- Développé initialement par deux mathématiciens spécialistes de théorie des nombres, et gros utilisateurs de magma : William Stein (Univ. Washington) et David Joyner.
- SAGE combine un grand nombre de packages open-source. L'interface, écrite en python, permet d'utiliser conjointement les capacités de calcul de ces différents packages.
- SAGE peut être utilisé comme interface pour différents logiciels commerciaux (maple, mathematica, magma), ou libres.
- Taille du répertoire contenant SAGE $\geq 1Go$.
  Temps de compilation de SAGE $\simeq 4h$ (comprenant la compilation de l'ensemble des logiciels utilisés).

Le développement de Sage

Février 2005 : Sage 0.1 : inclut Pari
Octobre 2005 : Sage 0.8 : ajout de GAP et Singular
Janvier 2006 : Sage 0.10 : ajout de Maxima et Clisp
Juin 2006 : Sage 1.3 : interface graphique notebook
Mars 2008 : Sage 2.3

Un exemple de réactivité : calcul du nombre de partitions

Une partition d'un entier est une écriture de cet entier comme somme d'entiers. Le nombre de partitions d'un entier grandit très vite quand l'entier devient grand :

```
sage: list(partitions(5))
[(1, 1, 1, 1, 1), (1, 1, 1, 2), (1, 2, 2), (1, 1, 3), (2, 3), (1, 4), (5,)]
sage: number_of_partitions(5)
7
sage: number_of_partitions(100)
190569292
```

Calculer le nombre de partitions d'un entier est un problème algorithmiquement compliqué !

Un exemple de réactivité : calcul du nombre de partitions

Dans les publi-informations de Mathematica, on trouve l'affirmation :

« **Mathematica computes the number of partitions of 1 billion in a few seconds - a frontier number theory calculation** »

Au moment où cette affirmation a été écrite, Sage (comme Magma !) aurait mis plusieurs années pour faire ce calcul.

Un exemple de réactivité : calcul du nombre de partitions

William Stein pose une question sur le forum sage-devel. De nombreuses réponses ont suivi, proposant des algorithmes.
L'algorithme est désormais intégré dans les versions récentes de Sage.
Maintenant Sage bat mathematica :

```
sage: time len(str(number_of_partitions(10^9)))
CPU times: user 44.83 s, sys: 0.02 s, total: 44.85 s
Wall time: 44.98 s
35219
```

Le nombre de partitions de 10^9 s'écrit avec 35219 chiffres !
Mathematica 6.0 a besoin de 83s et Mathematica 6.1 a besoin de 77s.

SAGE est programmé en python

- Langage interprété
- Librairies dédiées au calcul scientifique : Numpy, Scipy
- Librairie d'affichage 2D : MatPlotLib
- Interface avec des langages compilés : Cython, Weave. Les scripts python sont convertis en C et compilés.
- Interface avec des logiciels scientifiques : R, gnuplot,...

Les briques de SAGE

- Outils python : Cython, Ipython, MatPlotLib, Rpy, Scypy, Sympy
- Calcul scientifique : Atlas, Blas, Lapack, Gnuplot, Octave, R
- Calcul en nombres entiers : GMP, Pari/GP
- Calcul formel généraliste : Maxima
- Calcul formel spécialisé : Gap, Macaulay2, Singular
- JsMath : Implémentation de LaTeX en javascript
- Jmol : visualisation 3D (initialement pour la chimie !)
- Tachyon : visualisation 3D par lancé de rayon
- Interface avec des logiciel commerciaux (si ils sont installés !) : Magma, Maple, Mathematica, Matlab
- ...

Quelques expérimentations

Lancement, aspect objet, complétion automatique

```
$ sage
----------------------------------------------------------------------
| Sage Version 3.2.2, Release Date: 2008-12-18                       |
| Type notebook() for the GUI, and license() for information.        |
----------------------------------------------------------------------
sage: a=2^256+1;
sage: type(a)
<type 'sage.rings.integer.Integer'>
sage: a.
a.abs                          a.jacobi
a.additive_order               a.kronecker
a.base_extend                  a.lcm
a.base_ring                    a.leading_coefficient
a.binary                       a.list
a.bits                         a.log
a.category                     a.mod
a.ceil                         a.multifactorial
a.coprime_integers             a.multiplicative_order
a.crt                          a.n
(...)
```

Quelques expérimentations

Différents aspects d'une même commande

```
sage: a.factor()
1238926361552897 * 934616397153579777691635581996068965840512375416381885802803
sage: gp.factor(a)
[1238926361552897, 1; 934616397153579777691635581996068965840512375416381885802
sage: %gp
  --> Switching to GP/PARI interpreter <--
''
gp: factor(2^256+1)

[1238926361552897 1]

[934616397153579777691635581996068965840512375416381885802803 21

gp:  exit
  --> Exiting back to SAGE <--
sage:
```

Quelques expérimentations

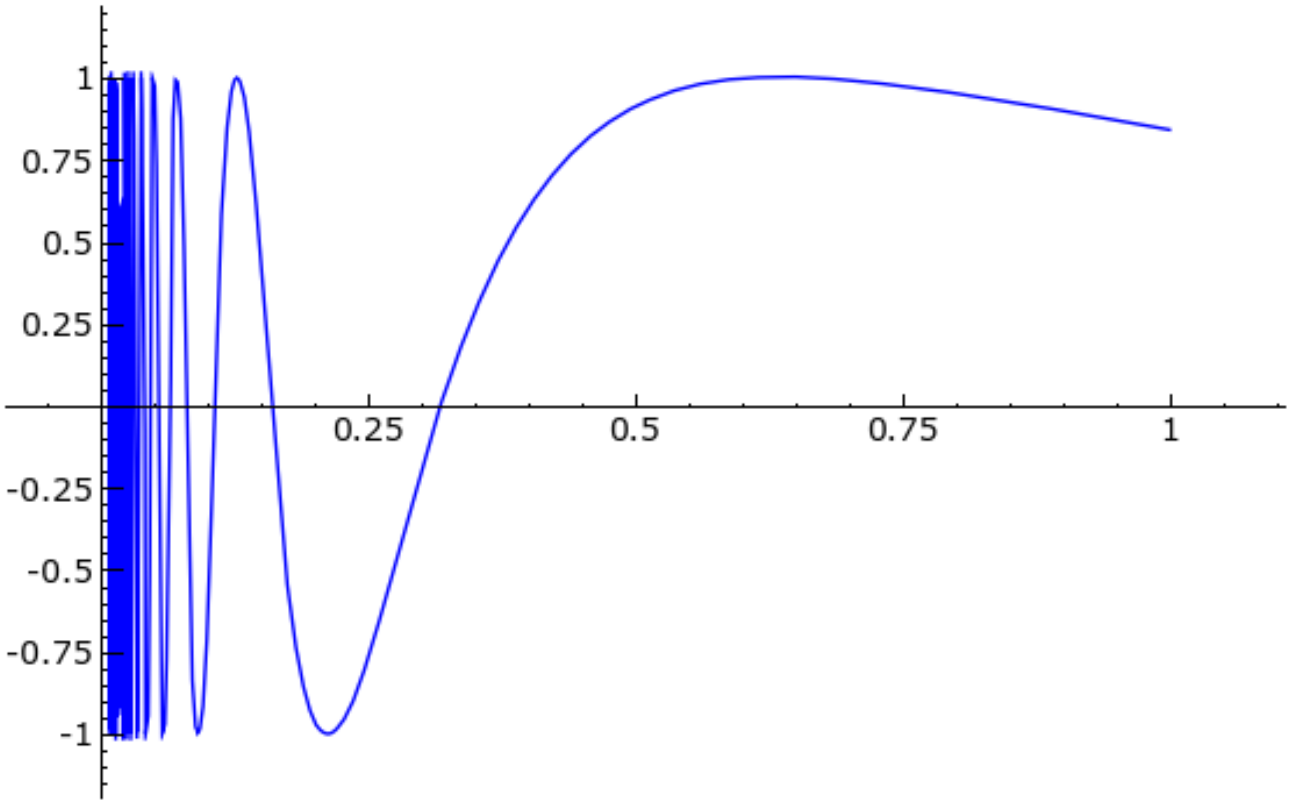
Chaque objet a des méthodes qui lui sont attachées

```
sage: R=QQ['x']
sage: type(R)
<class 'sage.rings.polynomial.polynomial_ring.PolynomialRing_field'>
sage: x=R.gen()
sage: type(x)
<class 'sage.rings.polynomial.polynomial_element_generic.Polynomial_rational_d
sage: P=x^2+1
sage: P.
P.abs                          P.is_unit
P.additive_order               P.is_zero
P.args                         P.lcm
P.base_extend                  P.leading_coefficient
P.base_ring                    P.list
P.category                     P.mod
P.change_ring                  P.monic
P.change_variable_name         P.multiplicative_order
P.coefficients                 P.n
P.coeffs                       P.name
P.complex_roots                P.newton_raphson
P.constant_coefficient         P.newton_slopes
```

Quelques expérimentations

```
sage: plot(sin(1/x), (x,0,1))
```

Sage fabrique une image png et lance une fenêtre séparée pour l'afficher :



Quelques expérimentations

Le langage de programmation est python. Sage utilise Ipython comme interpréteur.

```
def isprime(p):
    i = 2
    while i*i <= p:
        if p % i == 0:
            return False
        i = i + 1
    return True
l=[]
for p in range(40):
    if isprime(p):
        l.append(p)
print l
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37]
```

L'interface utilisateur

Deux possibilités :

- Interface en mode texte, lancé par la commande **sage**.

L'interface utilisateur

Deux possibilités :

- Interface en mode texte, lancé par la commande **sage**.
- Interface de type notebook (analogue des feuilles de travail de Maple).
  - Un serveur http, basé sur le module python « Twisted web2 », écoute sur le port 8000.
  - Chaque notebook est une page html, fabriquée par Sage, et contenant les calculs et leurs résultats
  - L'utilisateur accède aux notebooks par un navigateur web.

  On peut utiliser cette possibilité de deux façons :
  - L'utilisateur lance, depuis une session sage, la commande **notebook()**. Pas adapté à un système multiutilisateur.
  - Tous les utilisateurs se connectent à un unique serveur de notebook. Questions de sécurité...

Le serveur de Notebook

- Le serveur est lancé sous l'identité d'un utilisateur dédié (pas root !) :

  ```
  sage -c "notebook(address='',accounts=true,
                                    open_viewer=false)"
  ```

- La documentation recommande un environnement chrooté.
- Chacun peut ouvrir un compte (login/mot de passe) et créer des notebooks.
- Un notebook peut être publié ou non.
- Tous les fichiers sont stockés dans le répertoire de l'utilisateur qui a lancé le serveur http de sage.
- Mauvaise protection des fichiers : un utilisateur peut accéder assez facilement aux fichiers créés par un autre utilisateur.

Une petite démonstration vaut mieux qu'un long discours !