Xstra-dev et SAGEMATH

Séminaire Xstra

Pierre Navaro (navaro@math.unistra.fr)

Institut de Recherche Mathématique Avancée, Strasbourg

IRMA, 12 juin 2012

http://www-irma.u-strasbg.fr/~navaro/sagemath.pdf

XSTRA-DEV

- Calcul scientifique et développement : " l’élaboration et l’utilisation de programmes en vue de réaliser des simulations à l’aide de techniques issues des mathématiques appliquées et de l’informatique"
- Toutes disciplines : physique, biologie, chimie, mathématiques, astrophysique, mécanique, etc...
- Langages et de logiciel de calcul ou visualisation.

Les objectifs et les modèles

- Échanger et diffuser le savoir faire
- Constituer une communauté strasbourgeoise du calcul
- Constituer une base de compétences
- Valoriser le travail de chacun
- Rompre l'isolement

- Groupe calcul (`http://calcul.math.cnrs.fr`)
- Devlog (`http://devlog.cnrs.fr/`)
- DevelopR6 (`http://developr6.dr6.cnrs.fr/`)

Les actions du groupe

- Une rencontre annuelle. `http://xstra.u-strasbg.fr/doku.php?id=doc:par_annee#section2012`
- Organisation de formation (Python calcul scientifique).
- Un liste de diffusion `https://listes.u-strasbg.fr/sympa/unistra.fr/info/xstra-dev`
- Prochaine journée autour de Python (Janvier 2013).
- Demande de formation sur l'utilisation des GPU.

Pourquoi Sage ?

- William Stein (University of Washington) était un utilisateur et un contributeur de Magma mais...
- Logiciel cher
- Système de licence très restrictif
- Impossible d'accéder aux algorithmes issus pourtant de la recherche publique.
- Code peu dynamique, il est impossible de créer ses propres types.
- Langage limité (pas de gestion des exceptions)
- Communauté de développeurs très réduite.
- Pas de graphiques... Maple et Matlab possède des outils de visualisation puissants mais sont très inférieurs à Magma pour la théorie des nombres.
- Impossible de compiler ses scripts.

Historique

- William Stein commence en 2005.
- Code une alternative à Magma en Python.
- Collaboration avec David Joyner (Théorie des codes et des groupes)
- GAP et PARI sur cette plateforme Python.
- **S**oftware for **A**lgebraic and **G**eometric **E**xperimentation.

```
sage: E = EllipticCurve('389a'); E
Elliptic Curve defined by y^2 + y = x^3 + x^2 - 2*x over Rational Field
sage: E.gens()
[(-1 : 1 : 1), (0 : -1 : 1)]
sage: G = matrix(GF(5), 4, 7, [1,1,1,0,0,0,0,1,0,1,1,1,0,0,1,0,0,1,1,1,0,0,1,1
sage: C = LinearCode(G); C
Linear code of length 7, dimension 4 over Finite Field of size 5
sage: C.minimum_distance()
2
```

Ajout de fonctions pour le calcul formel

- Ajout de l'interface Maxima en 2006 pour l'enseignement.
- Bobby Moretti le rejoint en 2007, finit l'interface mais Maxima est lent.
- Burcin Erocal en 2008 collabore pour créer Pynac (issu de Ginac)

```
sage: f = 1 /sqrt(x^2+2*x-1)
sage: timeit('f*f')

625 loops, best of 3: 112 µs per loop

sage: g = maxima(f)
sage: timeit('g*g')

25 loops, best of 3: 16 ms per loop
```

Présentation générale

- Toutes les fonctions sont accessibles depuis une plateforme commune codée en Python.
- Tout code Python fonctionne dans sage.
- Tous les composants sont libres.
- Un système de calcul + un notebook accessible via un navigateur.
- Le système dépend le moins possible de la machine hôte. Il contient sa propre distribution Python et tous les packages sont compilés.
- Packages spkg : sources et scripts d'installation.
- Le logiciel est extensible à l'infini.
- Inconvénient : peu pratique sous Windows

Quelques composants de SAGE

- **Algèbre** : GAP, Maxima, Singular.
- **Géométrie algébrique** : Singular, Macaulay.
- **Arithmétique** : GMP, MPFR, MPFI, NTL.
- **Géométrie arithmétique** : PARI, NTL, mwrank, ecm.
- **Calcul formel** : Maxima, Sympy, Pynac.
- **Combinatoire** : Symmetrica, MuPaD-Combinad.
- **Algèbre** : Linbox, IML.
- **Théorie des graphes** : NetworkX.
- **Visualisation graphique** : Matplotlib, Tachyon 3d, Jmol.
- **Théorie des groupes** : GAP.
- **Analyse et algèbre numérique** : GSL, Scipy, Numpy.

`http://www.sagemath.org/packages/standard/`

Objets mathématiques

- Le langage est fortement typé ! Aucun objet n'a de type ambigu
- La complétion automatique permet de guider l'utilisateur vers la bonne méthode.
- On ne peut pas calculer sans savoir à quel ensemble appartiennent les variables.

```
sage: parent(x)
Symbolic Ring
sage: parent(1.2)
Real Field with 53 bits of precision
sage: parent(1/2)
Rational Field
```

Ensembles de nombres

- entiers relatifs : ZZ
- rationnels : QQ

```
sage: 1.2 in QQ
True
sage: sqrt(2) in QQ
False
```

- flottants doubles : *RDF*

```
sage: pi in RDF
True
```

- complexes

```
sage: CC
Complex Field with 53 bits of precision
```

- p-adiques

```
sage: Zp(5)
5-adic Ring with capped relative precision 20
sage: Zp(5,prec=5)(18)
3 + 3*5 + O(5^5)
```

Calcul formel

```
sage: (x+2)^3
(x + 2)^3
sage: type(x)
<type 'sage.symbolic.expression.Expression'>
sage: x+y
NameError: name 'y' is not defined
sage: var('y')
sage: f = (x+y)^3
sage: expand(f)
x^3 + 3*x^2*y + 3*x*y^2 + y^3
sage: f.expand()
x^3 + 3*x^2*y + 3*x*y^2 + y^3
sage: f.roots()
[(-y, 3)]
sage: f.subs(x=y)
8*y^3
sage: latex(f)
{\left(x + y\right)}^{3}
```

Fonctions

- Fonction en calcul symbolique

```
sage: norme(x,y) = sqrt(x^2+y^2)
sage: norme
(x, y) |--> sqrt(x^2 + y^2)
sage: norme(2,3)
sqrt(13)
```

- Fonction python

```
sage: def norme(x, y):
....:         return sqrt(x*x+y*y)
....:
sage: norme(2,3)
sqrt(13)
```

- Lambda fonction (renvoie une valeur unique)

```
sage: norme = lambda x,y: sqrt(x*x+y*y)
sage: norme(2,3)
sqrt(13)
```

Algèbre linéaire

```
sage: A=Matrix(QQ,[[4,8,12],[3,8,13],[2,9,18]])
sage: Y=vector([4,5,11])
sage: A.solve_right(Y)
(1, -3, 2)
sage: A=Matrix(RDF,[[4,8,12],[3,8,13],[2,9,18]])
sage: A \ Y          #   \  == solve_right
(1.0, -3.0, 2.0)
sage: A = Matrix(SR,[[4,x],[3,8]])
sage: A.inverse()
[-3/4*x/(3*x - 32) + 1/4            x/(3*x - 32)]
[           3/(3*x - 32)           -4/(3*x - 32)]
```

- QQ et SR => systèmes résolus avec LinBox .
- RDF => systèmes résolus avec Lapack (scipy).

Polynômes

- Polynôme dans l'espace des rationnels.

```
sage: x = QQ['x'].0
sage: f = x^3 + 1; g = x^2 - 17
sage: h = f/g;  h
(x^3 + 1)/(x^2 - 17)
sage: h.parent()
Fraction Field of Univariate Polynomial Ring in x over Rational Field
```

- p-adiques

```
sage : W=Zp(5, prec=5)
sage : var('X')
sage : A.<X> = PolynomialRing(W)
sage: X
(1 + O(5^5))*X
sage: A
Univariate Polynomial Ring in X over 5-adic Ring with capped relative pre
sage : P = X^2+1
sage : diff(P,X)
(2 + O(5^5))*X
```

Résolution d'équations

- Solution exacte avec un paramètre

```
sage: x, b, c = var('x b c')
sage: solve([x^2 + b*x + c == 0],x)
[x == -1/2*b-1/2*sqrt(b^2-4*c), x == -1/2*b+1/2*sqrt(b^2-4*c)]
```

- Plusieurs variables

```
sage: x, y = var('x, y')
sage: solve([x+y==6, x-y==4], x, y)
[[x == 5, y == 1]]
```

- solve ne renvoie pas forcément une information utile.

```
sage: theta = var('theta')
sage: solve(cos(theta)==sin(theta), theta)
[sin(theta) == cos(theta)]
```

- Si on choisit l'intervalle $]0, \pi/2[$:

```
sage: phi = var('phi')
sage: find_root(cos(phi)==sin(phi),0,pi/2)
0.785398163397448...
```

Dérivées, Intégrales,...

- Dérivées

```
sage: u = var('u')
sage: diff(sin(u), u)
cos(u)
sage: diff(sin(u),u,u)
-sin(u)
sage: diff(sin(u),u,2)
-sin(u)
sage: x, y = var('x,y')
sage: f = x^2 + 17*y^2
sage: f.diff(x),f.diff(y)
(2*x, 34*y)
```

- Intégrales

```
sage: integral(x*sin(x^2), x)
-1/2*cos(x^2)
sage: integral(x/(x^2+1), x, 0, 1)
1/2*log(2)
```

Equations différentielles

- $x' + x - 1 = 0$

```
sage: t = var('t')    # la variable t
sage: x = function('x',t)   # x est une fonction de t
sage: DE = diff(x, t) + x - 1
sage: desolve(DE, [x,t])
(c + e^t)*e^(-t)
sage:  DE = diff(x, t) + x - 1 == t
sage: desolve(DE, [x,t])
((t - 1)*e^t + c + e^t)*e^(-t)
```

- Transformée de Laplace

```
sage: s, t = var("s t")
sage: f = t^2*exp(t) - sin(t)
sage: f.laplace(t,s)
2/(s - 1)^3 - 1/(s^2 + 1)
```

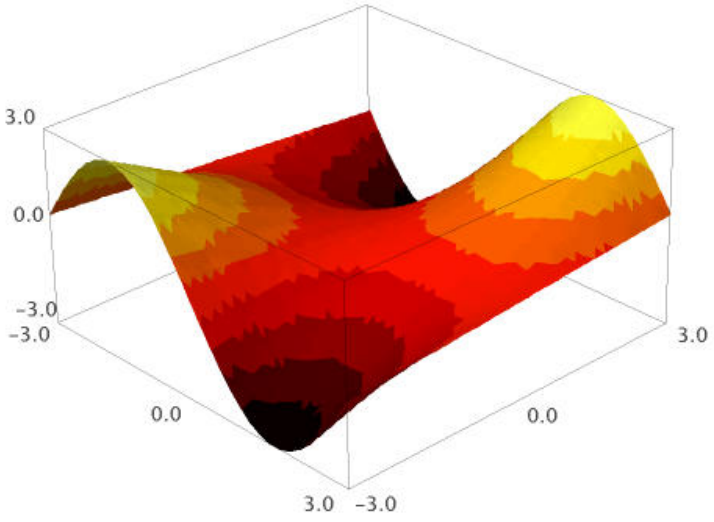
Graphe 3D

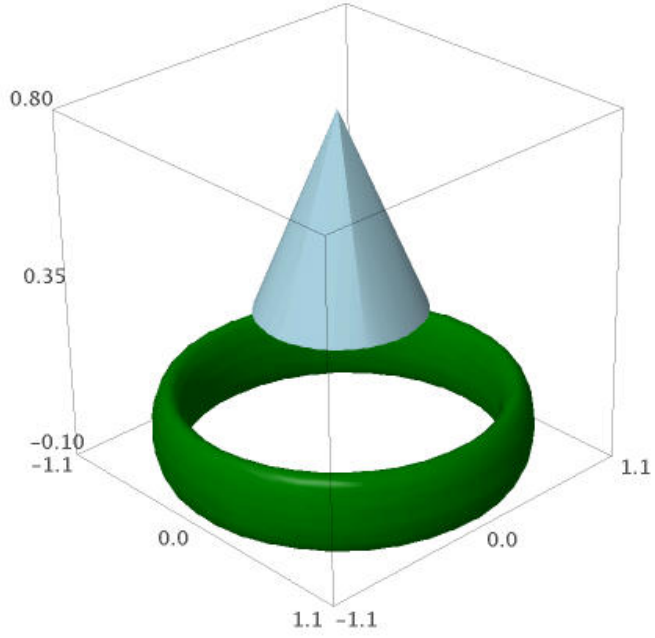
- Tracé d'une fonction

```
import matplotlib.cm
hotmap = map(tuple,matplotlib.cm.hot(srange(0,1,.1)))
plot3d(sin(x)*y,(x,-3,3),(y,-3,3),color=hotmap, adaptive=True)
```

- Bibliothèque d'objets

```
from sage.plot.plot3d.shapes import Cone, Torus
Cone(.5,.5,color='lightblue').translate(0,0,.3)+Torus(1,.1,color='green')
```





Quelques démos

- Calcul formel
- Graphes 3D
- Courbe de Béziers quadratique rationnelle
- Cython-f2py-PyOpenCL
- Animation
- Interaction

Liens et références

Documentation officielle

`http://www.sagemath.org/doc/`

Wiki

`http://wiki.sagemath.org/`

Notebook public

`http://www.sagenb.org/`

GDS Mathrice

`http://sage.math.cnrs.fr/`

Pages de Franco Saliola

`http://thales.math.uqam.ca/~saliola/sage/`

Paul Zimmermann *et al.*

Calcul mathématique avec Sage

`http://sagebook.gforge.inria.fr/`.