# Calcul formel avec SageMATH

Nicolas Borie

Laboratoire d'Informatique Gaspard Monge
Université Paris Est à Marne-La-Vallée, France

14 janvier 2016

Le calcul sous toutes ses formes,
du lycée à l'université

# Qu'est ce que le calcul formel ?

Le **calcul formel** manipule des entités mathématiques de manière **exacte**. Tous les objets manipulés DOIVENT ainsi être construits/décrits de manière finie.

**Scilab :**

```
-->%pi
 %pi  = 3.1415927
-->2*%pi
 ans  = 6.2831853
-->exp(-%i*%pi)
 ans  = - 1. - 1.225D-16i
```

**Sage :**

```
sage: pi
pi
sage: 2*pi
2*pi
sage: exp(-i*pi)
-1
sage: pi.n()      #demande explicite d'approximation numérique ici !!!
3.14159265358979
```

# Calcul formel $\neq$ Calcul numérique

- Toujours **juste**. Un algorithme de calcul formel provient (extrêmement souvent) d'une démonstration mathématique **constructive**.
- **Modélisation limitée** : les nombres réels n'existent pas en calcul formel, seule une partie (infime) des réels est considérable, les réels décrits de manière déterministe et constructive en temps fini.
- **Lourdes complexités** : la justesse à un prix. Phénomènes compliqués comme l'explosion arithmétique.

```
sage: QQ.random_element(100,100)
100/43
sage: M=matrix(6,6,[QQ.random_element(100,100) for i in range(36)]); M
[-25/78  83/68  81/71 -78/59 -86/23  23/93]
[ 68/93  -11/3  22/69  -6/59 -59/30 -31/45]
[ 61/45  11/27  44/17   -3/4 -28/39    5/6]
[ -8/21   2/17  32/43 -37/73  49/68    4/3]
[ -5/28       5  -23/9  45/41 -68/75 -16/75]
[  5/23 -31/30  -3/14  52/11       1  19/11]
sage: M.determinant()
1494935005594402758126228891665686657/32905021083980275530810894384000000
```

# Les enjeux du calcul formel

Listes non exhaustives, classification non standard...

- **Enjeux arithmétiques** : théorie des nombres, racines de polynômes, systèmes d'équations algébriques, primalité(s), factorisation(s), cryptographie, théorie de Galois effective, théorie des groupes...

- **Enjeux combinatoires** : modélisation d'objets combinatoires (permutations, partitions, composition, tableaux (de Young), chemins discrets, graphes...), séries formelles, décompositions, énumérations, générations...

- **Enjeux symboliques** : représentation des polynômes, matrices et séries, fonctions spéciales, fonctions D-finie, expressions algébriques, dérivation/intégration formelle, algèbre linéaire exacte, ...

# Systèmes de calcul formel (CAS) : état de l'art

Caractéristiques listées subjectives...

- **Maple** : Propriétaire, langage spécifique, nombreux packages, limité dans la modélisation d'objets, le plus puissant dans la manipulation symbolique, ...
- **Mathematica** : Propriétaire, langage spécifique, généraliste, environnement de développement riche, ...
- **Magma** : Propriétaire, langage spécifique, spécialisé vers l'algèbre, la théorie des nombres et la géométrie, ...
- **Singular** : Libre, langage spécifique, spécialisé vers les polynômes et ses environs, notamment vers la géométrie algébrique effective, ...
- **Gap** : Libre, langage spécifique, spécialiste de la théorie des groupes et ses environs, ...
- **Sage** : Libre, basé sur le langage Python, généraliste, jeune, ...
- ...

# Qu'attends-on d'un système de calcul formel ?

**Point de vu utilisateur classique :**

- Fonctionnalités formelles évoluées
- Prise en main aisée
- Efficacité, stabilité et justesse (tous les CAS ont des bogues...)
- Comprendre les enjeux de ce que l'on demande à la machine

**Point de vu recherche :**

- Un chercheur est tout d'abord un utilisateur classique
- COMPRENDRE LES ENJEUX DE CE QUE L'ON DEMANDE À LA MACHINE
- Être capable de développer son propre code pour faire de l'exploration informatique
- Valoriser sa recherche via le biais d'expériences concrètes sur machine

# Qu'attends-on d'un système de calcul formel ?

**Point de vu étudiant :**

- Savoir résoudre les problèmes

  Facilité de translation entre mathématique et CAS, trouver les choses dont on a besoin rapidement, avoir l'impression d'écrire des maths, ne pas être bridé par syntaxe et environnement de développement, ...

**Point de vu enseignant :**

- Écrire/Trouver/Reprendre des problèmes dont la résolution est intéressante sur le CAS

  Limiter les coûts de prise en main, limiter les coûts d'apprentissage de la syntaxe, limiter les coûts de recherche d'information, limiter les coûts relatif aux déboguages, ...

# Les CAS aujourd'hui...

- **changent trop souvent.**
  (fin d'instruction : - ; - ;; - : - (rien) - ... )
  (exit - quit - quit();; - exit(1): - ... )
  (help func - ?func - ??func - help(func) - func? - ... )
- **sont possiblement chers.**
  Les élèves/étudiants ne peuvent pas raisonnablement le posséder chez eux.
- **ne présente pas de plus value à leur apprentissage.**
  (J'ai tapé du code XXXX durant un certains nombre d'année mais XXXX n'existe plus...)

D'autres petits problèmes comme la documentation en langue anglaise, le manque d'exemples, la difficulté pour savoir quels algorithmes sont utilisés en interne, ...

# Le système Sage

- logiciel **libre** (gratuit, sources consultable, bogues documentés, contribution ouverte, ...)
- **développé par** des enseignants/étudiants/chercheurs **pour** des enseignants/étudiants/chercheurs
- propriété intellectuelle du code disséminée sur toute la planète
- utilise un langage de programmation généraliste existant : **Python**
- **review en peer to peer** des correctifs/améliorations/nouvelles fonctionnalités
- philosophie de ré-utiliser les meilleurs programmes existants sans les réécrire (nombreux interfaces vers d'autres CAS)
- utilise un maximum de standards existants (documentation Sphnix à la **Python**, système de gestion de version **git**, gestionnaire de tickets **trac**)

Sage : building the car, not reinventing the wheel.

# Historique

- 2005 : une demi-douzaine de développeurs lassés des gros CAS lancent Sage, basé sur le langage Python
- 2006 : Gap, Singular, Clisp, Maxima sont dans Sage
- Février 2006 : Premières journées Sage (UCSD)
- Octobre 2008 : Journées Sage 10 à Nancy
- Février 2010 : Journées Sage 20 et Journées Sage pour l'enseignement 2 à Marseille
- 2014-2015 : Sage langage possible pour l'agrégation de mathématique
- Aujourd'hui : Journées Sage 70: (9-13 Novembre, 2015, Berkeley)
  - environs 800 k lignes de codes
  - plus de 5300 classes d'objets
  - plus de 54 k fonctions
  - contenu mathématique riche
  - de nombreux algorithmes exclusifs
  - plus de 500 contributeurs sur la planète
  - plus de 19800 tickets sur la plateforme de gestion de développement

# Ce qui ne fonctionne toujours pas...

- Sage est difficile à installer complètement (6 heures de compilation depuis les sources sous Linux).
- L'installation de Sage sous Windows n'est pas aisée.
- Sage contient de nombreux bogues.
- De nombreuses fonctionnalités disponibles dans les logiciels tiers ne sont pas encore interfacées correctement.
- La jeunesse de Sage force parfois de casser la compatibilité lors d'une sortie de nouvelle version.
- Sage manque de performance dans certains domaines.
- ...

# Ce qui fonctionne...

- Utilisation de Sage gratuitement en ligne sans installation (juste un navigateur web).
- La mayonnaise a pris (le nombre d'utilisateurs et de contributeurs croient très fortement).
- Les décisions de design sont longues à prendre (discussion sur internet) mais sont des délibérations fines pour le futur.
- De nombreuses publications scientifiques utilisent Sage pour illustration (expériences scientifiques reproductibles gratuitement).
- Une grande maîtrise de ce que l'on exécute en machine.
  ( ?? : double point d'interrogation pour voir le code exécuté)
- Python : Premier langage en licence math-info à Marne-la-vallée
  Langage utilisé en lycées
  Langage informatique des CPGE
  Langage fort utilisé pour des cours de math-info et cryptographie
  Apprendre le CAS Sage, c'est aussi apprendre un langage de programmation généraliste.

# Contributeurs à Sage