SAGE: Software for Algebra and Geometry Experimentation

William Stein

May 4, 2006: UW SAGE Seminar

What is SAGE?

I started SAGE in early 2005, and it has since mushroomed. There are now dozens of contributors all over the world and an extremely active mailing list.

1. **Completely free and open source.**

2. **A new computer algebra system.** Uses a *mainstream* language (unlike Magma, Gap, **Mathematica**, **Maple**, etc.)

3. **A new way to use your software.** Use *all* your favorite (**commercial** or free) mathematics software *together*.

Does Open Source Matter for Math Research?

"You can read Sylow's Theorem and its proof in Huppert's book in the library [...] then you can use Sylow's Theorem for the rest of your life free of charge, but for many computer algebra systems license fees have to be paid regularly [...]. You press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.

With this situation **two of the most basic rules of conduct in mathematics are violated**: In mathematics **information is passed on free of charge** and **everything is laid open for checking**. Not applying these rules to computer algebra systems that are made for mathematical research [...] means **moving in a most undesirable direction**. Most important: Can we expect somebody to believe a result of a program that he is not allowed to see? "

– J. Neubüser in **1993** (he started GAP in 1986).

SAGE: A New Computer Algebra System

```
algebras     edu          interfaces  modular   schemes
categories   ext          lfunctions  modules   sets
coding       functions    libs        monoids   structure
crypto       geometry     matrix      plot      tests
databases    groups       misc        rings
```

```
$ cat */*.py */*/*.py */*/*/*.py */*.pyx */*/*.pyx |sort|un
58858
```

```
$ cat */*.py */*/*.py */*.pyx */*/*.pyx |sort|uniq|grep "sa
6845          <-------- EXAMPLE INPUT LINES!
```

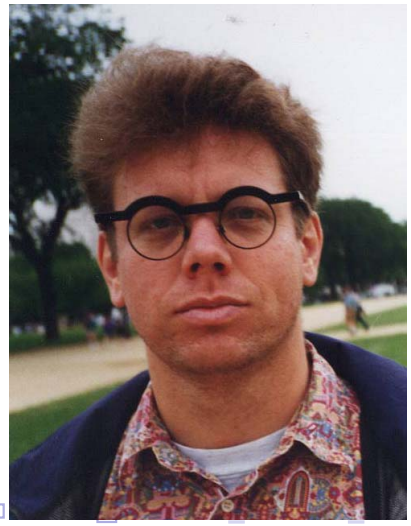
Cooperation: "Everything Under One Roof"

SAGE has many interfaces (**bold included** with SAGE):

- **GAP** (started 1986)– groups, discrete math
- **Singular** (started 1987) – polynomial computation
- **PARI/GP** (started 1987) – number theory
- **Maxima** (started 1967) – symbolic manipulation
- **mwrank, ec, simon, sea** – elliptic curves
- Macaulay2 (started 1993) – (soon to be included)
- KANT/KASH – sophisticated algebraic number theory
- Magma (started 1973) – high-quality research math environment
- Maple – symbolic, educational
- Mathematica – symbolic, numerical, educational
- Octave (started 1992) – numerical analysis
- Specialized: mwrank, gfan, sympow, ntl, genus2reduction, polymake, lcalc (Rubinstein), dokchitser; more to come...!

Python: A Mainstream Programming Language

- Guido van Rossum released first Python in 1991.
- A **"gluing language"**, i.e., design to be easier to use libraries and other programs.
- **VAST range of libraries**: graphics, 3d simulation, web programming, numerical analysis, etc.
- **Easy to read** code
- `function?` gives documentation about function and `function??` gives the **source code**.
- **IPython:** Awesome!



Pyrex: Compiled Python-like language

1. Written by **Greg Ewing** of New Zealand.
2. Code converted to C code that is **compiled by a C compiler**.
3. Easy to use C/C++ code and libraries from Pyrex.
4. **Time-critical** SAGE code gets implemented in Pyrex, which is (as fast as) C code, but easier to read (e.g., since all variables and scopes are explicit).

Examples: Using other systems...

```
sage: mathematica('N[Gamma[Pi + I]]')       # optional
1.030298406982291 6 + 1.60260458767841 4*I
sage: G = mathematica('Plot3D[Sin[x]*Cos[y],{x,2,10},{y,2,10}]')
sage: G.show();

sage: gp.zeta(2)       # number theory
1.64493406684822643647241516 7
sage: gp.factor(2006)
[2, 1; 17, 1; 59, 1]

sage: f = maxima('x*sin(x)^2').integral('x'); f    # calculus
-(2*x*sin(2*x) + cos(2*x) - 2*x^2)/8

sage: maple.eval('solve({ 2*x + 3*y = 1, 3*x + 5*y = 1 })')
{y = -1, x = 2}
```

Linear Algebra

```
sage: A = MatrixSpace(QQ,3)([1,2,3, 4,5,6, 8,10,12]); A
[ 1  2  3]
[ 4  5  6]
[ 8 10 12]
sage: A.charpoly().factor()
x * (x^2 - 18*x - 15)

sage: V = QQ^3
sage: E = End(V); t = E(A); t
Free module morphism defined by the matrix
[ 1  2  3]
[ 4  5  6]
[ 8 10 12]
Domain: Vector space of dimension 3 over Rational Field
Codomain: Vector space of dimension 3 over Rational Field

sage: print latex(A)
\left(\begin{array}{rrr}
1&2&3\\ 4&5&6\\ 8&10&12
\end{array}\right)
```

Algebraic and Combinatorial Geometry

```
sage: P.<x,y,z,w> = ProjectiveSpace(3,QQ)
sage: C = P.subscheme([y^2-x*z, z^2-y*w, x*w-y*z])
sage: len(C.irreducible_components())   # twisted cubic
1
sage: J = C.defining_ideal()
sage: G = J.groebner_fan()
sage: len(G.reduced_groebner_bases())
8
sage: G.fvector()
(1, 8, 8)
sage: f = prod(J.gens())      # \/-- newton polytope
sage: NP = polymake.convex_hull(f.exponents())
sage: NP.facets()
[(3/2, 5/2, -1, 0), (3, 1, -1, 0), (1, 0, 0, 0),
 (-3/2, 2, 1, 0), (3, -1, 4, 0), (-3, 1, 5, 0)]
```

Latex (typesetting)

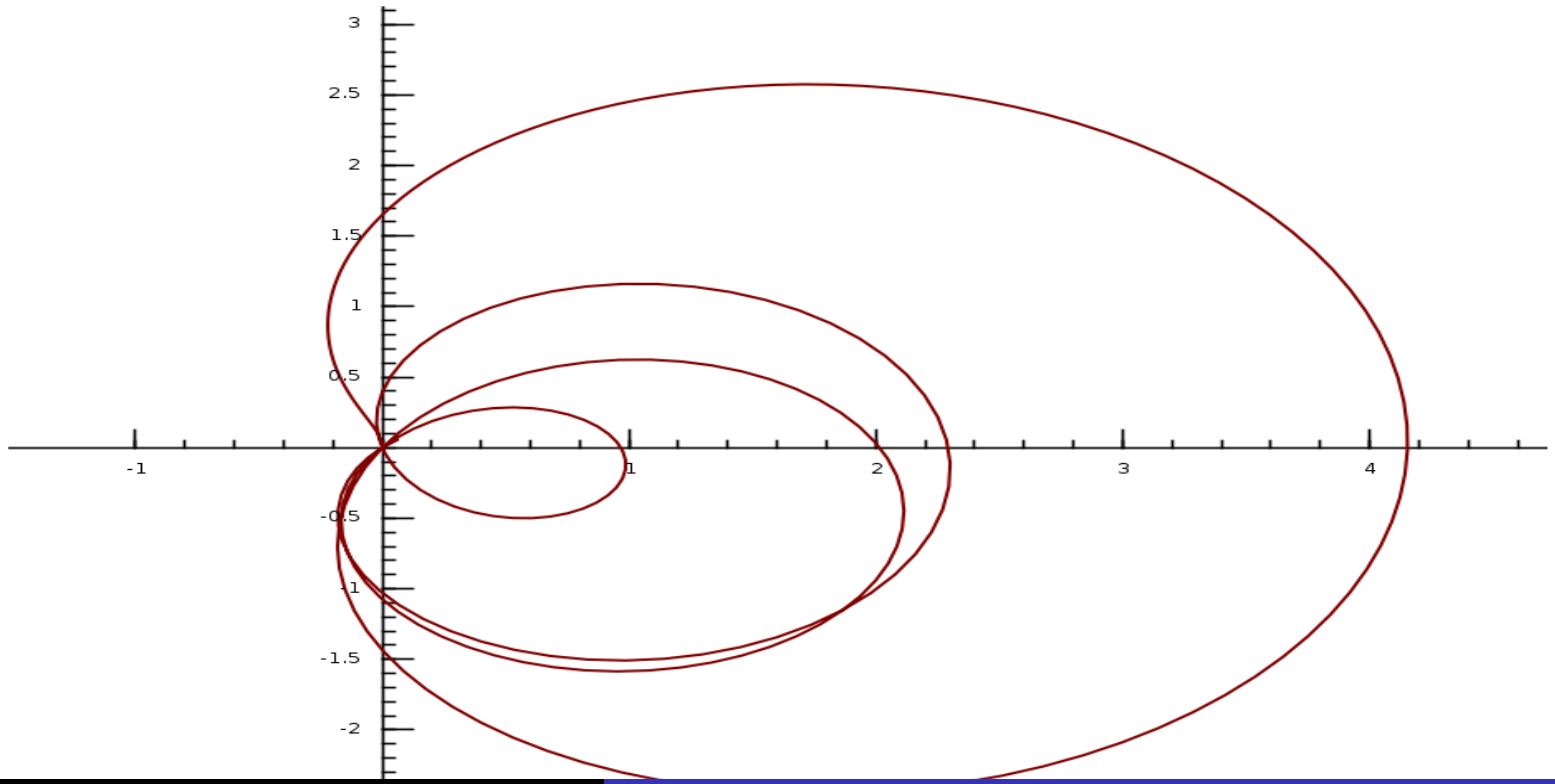
Any object has a latex representation:

```
sage: R.<t> = PowerSeriesRing(QQ, 't')
sage: f = 1/(1-t)
sage: f
1 + t + t^2 + t^3 + t^4 + t^5 + t^6 + t^7 + t^8 + t^9 +
      t^10 + t^11 + t^12 + t^13 + t^14 + t^15 + t^16 +
      t^17 + t^18 + t^19 + O(t^20)
sage: print latex(f)
 1 + t + t^{2} + t^{3} + t^{4} + t^{5} + t^{6} + t^{7}
   + t^{8} + t^{9} + t^{10} + t^{11} + t^{12} + t^{13}
   + t^{14} + t^{15} + t^{16} + t^{17} + t^{18}
   + t^{19} + \cdots
sage: view(f)
```

$$1 + t + t^{2} + t^{3} + t^{4} + t^{5} + t^{6} + t^{7} + t^{8} + t^{9} + t^{10} + t^{11} + t^{12} + t^{13} + t^{14} + t^{15} + t^{16} + t^{17} + t^{18} + t^{19} + \cdots$$

Number Theory: Image of a line segment under $L(E, s)$

```
sage: E = EllipticCurve('37a')
sage: v = E.Lseries_values_along_line(1, 1+10*I, 300)
sage: w = [(z[1].real(), z[1].imag()) for z in v]
sage: L = line(w, rgbcolor=(0.5,0,0))
sage: L.save('line.png')
```



Saving and Loading Individual Objects

Most objects in SAGE can easily be loaded and saved in a compressed format. (This is a standard feature of Python!)

```
sage: V = VectorSpace(QQ, 5)
sage: W = V.submodule([[1,2,3,4,5],[2,3,4,5,3]]); W
Vector space of degree 5 and dimension 2 over Rational Fiel
Basis matrix:
[ 1  0 -1 -2 -9]
[ 0  1  2  3  7]
sage: W.save('W')
...quit... and restart
sage: load('W')
Vector space of degree 5 and dimension 2 over Rational Fiel
Basis matrix:
[...]
```

SAGE: The Future

- August 2006: **MSRI grad student Workshop** on Computing with Modular Forms (with SAGE).
- October 7-8, 2006: **SAGE Days 2** HERE!!
- UW *startup money*: Generous financial support for SAGE development (hiring a team of **students**).
- **SAGE users** are coauthors/developers of SAGE.

Alex Clemesha

Graphics Demo

Web site

(give a live tour)

Getting Involved: A specific list of projects

- **HNF over $k[x]$:** Implement Hermite Normal Form (echelon form) for polynomial rings over a field
- **Integer Factorization:** Make a wrapper so can use amazing ECM program easily from SAGE.
- **Auto-tested docs:** a function without examples is broken.
- **Lie Algebras:** Add LIE to SAGE (http://young.sp2mi.univ-poitiers.fr/ marc/LiE/)
- **Randomized automated testing:** Devious code.
- **Linear algebra:** very fast over finite fields / QQ, etc ?
- **Easy:** If f is a maxima-defined function, implement $f(x)$.
- And *much* more!