SAGE: Software for Algebra and Geometry Experimentation

William Stein

July 12, 2006: CNTA

The SAGE Mailing List on Feb 2, 2006

Dear SAGE community.

My name is Tiziano and I'm from Italy. I'm writing this mail first of all because I would like to thank you all for SAGE. It's something the world was really missing.

[Every free computer algebra system I've tried has] **"reinvented many times the wheel without being able to build the car."**

PART I: SAGE – Background: Why bother when we already have Magma? Isn't it enough for everything we'll ever need?

PART II: SAGE – Status Report and Tour

Mathematics Software Timeline

1960s-now	1970s-now	1980s-now	1990s-now	2005-now
Maxima	**Mathematica**	**Magma**	**Singular**	**SAGE**
	Maple	**GAP**	**Macaulay2**	
	Axiom		**PARI**	

What is SAGE?

What is SAGE?

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

1. **Completely free and open source** *distribution* of math software for Windows, OS X, and Linux. (All under GPL compatible licenses.)

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 your favorite (**commercial** or free) mathematics software *together*.

Strong **number-theoretic bias** since I'm a number theorist.

1. 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).

2. 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
```

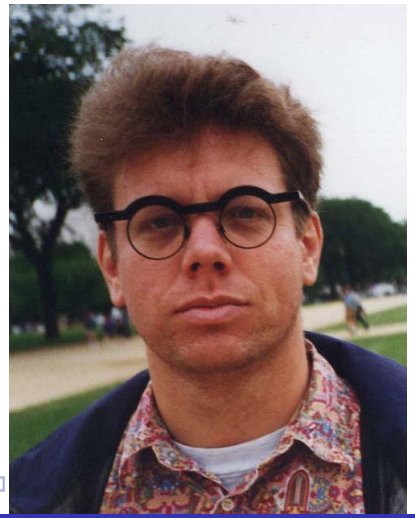
```
$ cat */*.py */*/*.py */*/*/*.py */*.pyx */*/*.pyx |sort|un
73451   # unique lines of human-written source code
```

```
$ cat */*.py */*/*.py */*/*/*.py */*.pyx */*/*.pyx |wc -l
142402
```

```
$ cat */*.py */*/*.py */*.pyx */*/*.pyx |sort|grep "sage: "
11123          <-------- EXAMPLE INPUT LINES!
```

Python: A Mainstream Language

- Completely free **open source** language.
- Guido van Rossum released first Python in **1991**.
- A **"gluing language"**, i.e., designed to be easy to use other libraries and other programs.
- **VAST range of libraries**: web servers, 2d and 3d graphics, numerical analysis, etc.
- **Code:** Very easy to learn, read and explore.
- **IPython:** Excellent command line.
- SAGE programs can be written in Python. And Python **can be used as a C/C++ library**, so SAGE can also be.



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**. All non-C memory management done automatically.
3. Very easy to read.
4. Easy to use both C/C++ code and libraries and Python code from Pyrex.
5. **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).
6. Analogue of PARI's brilliant **gp2c**.

3. Cooperation: "Everything Under One Roof" (Stoll)

(**Blue** – included with SAGE):

- **GAP** – groups, discrete math
- **Singular** – polynomial computation
- **PARI and GP** – number theory
- **Maxima** – symbolic manipulation
- **mwrank, ec, simon, sea** – elliptic curves
- GMP-ECM, gfan, sympow, NTL, genus2reduction, polymake, lcalc (Rubinstein), Dokchitser (*L*-series); and much more!
- Macaulay2 – commutative algebra
- KANT/KASH – sophisticated algebraic number theory
- Magma – I **really like** Magma!
- Maple – symbolic, educational
- Mathematica – symbolic, numerical, educational
- Octave – numerical analysis

Status Report

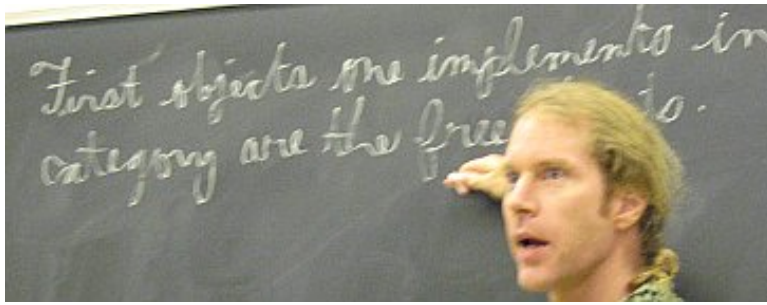
1. SAGE **can do much already**. ICM satellite conference has a list of 15 problems that they invited authors of computer algebra systems to solve. Last I checked, SAGE was the only one that could **solve all of them**.
2. **Growing pains:** SAGE has bugs and annoying issues; these do get steadily resolved over time, especially as more and more people use SAGE.
3. **Gotchas:** Using a mainstraim language (Python) instead of a custom language – The pros far outweight the cons, but there are definite cons.
4. **Installation:** Binary install not easy enough yet; cause is lack of sufficiently stable releases. Next stable release: November.
5. **Optimization:** Some new code is still *slower* than the non-free counterpart. This primarily affects new algorithms implemented for SAGE, e.g., a PARI or GAP program called from SAGE is no slower than in PARI or GAP.

Longterm GOALS for SAGE

- Create **the best** software environment for **all** types of serious mathematics research computation: **BUILD THE CAR**.
  (I personally focus on number theory.)
- Build a large **user base**.
- Build a large **developer base** (with substantial input from **undergraduates**, who are the best programmers).
- Do things **not done well enough** in other free mathematics software before, e.g.:
  - Web browser **graphical user interface**
  - Integrated support for **distributed computation**
  - **Database** support, saving and loading individual objects
  - Exact **linear algebra**
- Start a **SAGE Foundation**.
- Longterm target audience: undergraduates, grad students, both pure and applied mathematicians.

SAGE Workshops

- February 2006 (past): **SAGE Days 1** in San Diego.
- July 2006: **SIMUW** 2-week workshop for *high school* on the BSD conjecture that made extensive use of SAGE.
- August 2006: **MSRI grad student Workshop**
- October 6–10, 2006: **SAGE Days 2** in Seattle.




William Stein (UW) and David Joyner (USNA)
Dept of Mathematics, University of Washington, Seattle
SAGE Days 2
coding sprints October 5,6,9; conference October 7-8, 2006

SAGE Developers

- I've hired a team of **five undergrads** to work on SAGE.
- Some SAGE developers: **David Joyner**, **Tom Boothby**, **Martin Albrecht**, **Alex Clemesha**, **David Kohel**, **Josh Kantor**, **Bobby Moretti**, **Gary Zablackis**, Gonzalo Tornaria, Emily Kirkman, Yi Qiang, Ifti Burhanuddin, John Cremona, Didier Deshommes, Naqi Jaffery, Kiran Kedlaya, David Roe, David Kirkby, Jon Hanke, Gregg Musiker, Fernando Perez, Kyle Schalm, Steven Sivek, Jaap Spies, Justin Walker, Mark Watkins, Joe Wetherell
- **Authorship is very explicit** whenever possible. *Try it* – send me an example for the documentation and your name will be in the SAGE source code and docs.

PART II: SAGE – Status Report and Tour

- **Documentation:** tutorial, install guide, constructions, reference manual, programming guide.
- **Download:** Source code; binaries for Linux, Windows, OS X
- **Mailing lists:** Support, Devel, Announce, Forum, DARCS.
- **Home for Software:** genus2reduction, lcalc, Hanke, sympow, Simon's 2-descent, SEA, etc.; no need to rewrite your package in another language, e.g., if you love C++ or PARI use it!
- **SAGE Notebook:** Flexible Web-browser based graphical user interface for SAGE
- **Databases:** Cremona, Sloane, Jones, Stein-Watkins, etc.
- **Elliptic Curves:** algorithms, graphs, etc.
- **Using PARI, GAP, Magma, etc. from SAGE**
- **Modular Forms:** much written; need optimization and testing

Documentation

- **Tutorial** – 98 pages, written mainly by David Joyner with many contributors. Done. If you want to learn SAGE, sit down and spend 2–3 hours with this.
- **Install guide** – 20 pages. Done.
- **Constructions** – 103 pages, by David Joyner, answers dozens of questions like "How do a make a [[blank]] in SAGE?" Done.
- **Reference manual** – 1258 pages; partly .tex files and partly generated from source code files (literate programming). Needs many more examples and intro text.
- **Programming guide** – 65 pages; needs more, especially regarding style.

Downloading SAGE

- Show the web page.

DARCS

- An excellent mature **distributed** revision control system that Gonzalo Tornario convinced us to use.
- SAGE makes very extensive use of DARCS.
- Makes my life vastly easier.
- I still can't keep up with the volume of quality code I'm getting though!

SAGE Command Line

- SAGE uses IPython, which was written by Fernando Perez, who is a computational physicist at University of Colorado.
- By far the best command line I've ever used for any program.
- Persistent history, completion, introspection, interactive debugger, etc.
- Under active development and moving forward.

SAGE Notebook

- Writing a GUI – why reinvent the wheel? Web browsers do 90% of the work already.
- Alex Clemesha, Tom Boothby, and I wrote this from scratch last month.
- Fills a gap in open source software – until now there was no GUI for most open source math software. Can be started in Magma/PARI/etc. mode.
- Web browser based, which adds flexibility – several people can interact with the same session remotely, etc.
- Very usable right now, though we have many ideas for new features to add.
- Can queue up calculations; use multiple worksheets at once; save/load objects, worksheets, etc.
- Try it any time: `http://sage.math.washington.edu:8100`

Databases

- Cremona's elliptic curves
- Jones's global number fields
- Stein-Watkins (over 100 million elliptic curves)
- Sloane's integer sequences
- Conway Polynomials
- Etc...

Elliptic Curves

- Birch and Swinnerton-Dyer related functions.
- MWRANK – SAGE provides an interpreter interface
- $L(E, s)$
- Plots

Using PARI, GAP, Magma, etc. from SAGE

```
sage: n = -2007
sage: print n.factor()
-1 * 3^2 * 223
sage: print factor(n)
-1 * 3^2 * 223

sage: n.factor(algorithm="kash")
-1 * 3^2 * 223

sage: gap(n).FactorsInt()
[ -3, 3, 223 ]

sage: pari(n).factor()
[-1, 1; 3, 2; 223, 1]

sage: gp(n).factor()
[-1, 1; 3, 2; 223, 1]

sage: maxima(n).factor()
-3^2*223

sage: kash(n).Factorization()
[ <3, 2>, <223, 1> ], extended by:
  ext1 := -1,
  ext2 := Unassign

sage: magma(n).Factorization(nvals = 2)
([ <3, 2>, <223, 1> ], -1)

sage: maple(n).ifactor()
-``(3)^2*``(223)

sage: mathematica(n).FactorInteger()
{{-1, 1}, {3, 2}, {223, 1}}
```

Saving and Loading Objects

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

```
sage: R.<x,y> = PolynomialRing(QQ,2)
sage: f = y^2 + y - x^3 - 17/3*x + 2/3
sage: f.save('f')
sage: load('f')
2/3 + y + y^2 - 17/3*x - x^3

sage: A = MatrixSpace(QQ,50).random_element()
sage: A.save('amat')
sage: load('amat')
[ 2 -1  2  2  2 -2 -1 -2  2  2 -2  1 -1 -1  2 -2  1 -1 -1 -1  1 -1  1 -1  2  1  1 -1 -1  2 -1 -1 -2
1 -2 -1  1  2 -2  2  2 -1 -2  1 -1  1 -2  2  2 -1]
...
sage: M = ModularSymbols(Gamma1(13),2); M
Modular Symbols space of dimension 15 for Gamma_1(13) of weight 2 with
sign 0 and over Rational Field
sage: M == loads(dumps(M))
True
sage: D = M.decomposition(2)
sage: D
[
Modular Symbols subspace of dimension 1 of Modular Symbols space of dimension 15 for Gamma_1(13) of
weight 2 with sign 0 and over Rational Field,
...]
sage: D.save('D')
sage: load('D')
[
Modular Symbols subspace of dimension 1 of Modular Symbols space of dimension 15 for Gamma_1(13) of
weight 2 with sign 0 and over Rational Field, ...
]
```

Some Algebraic Number Theory

This examples makes lots of use of the PARI C library and GAP.

```
sage: V = NumberField(x^2+17).composite_fields(NumberField(x^3-2))
sage: print V
sage: K = V[0]
[Number Field in a with defining polynomial x^6 + 51*x^4 - 4*x^3 + 867*x^2 + 204*x + 4917]
sage: G = K.galois_group()
sage: G
Transitive group number 3 of degree 6
sage: G.order()
12
sage: G.conjugacy_classes_representatives()
[(), (2,6)(3,5), (1,2)(3,6)(4,5), (1,2,3,4,5,6), (1,3,5)(2,4,6), (1,4)(2,5)(3,6)]
sage: gg = gap(G)
sage: gg.NormalSubgroups()
[ Group(()), Group([ (1,4)(2,5)(3,6) ]), Group([ (1,3,5)(2,4,6) ]),
  Group([ (1,3,5)(2,4,6), (1,2)(3,6)(4,5) ]),
  Group([ (1,3,5)(2,4,6), (2,6)(3,5) ]),
  Group([ (1,2,3,4,5,6), (1,3,5)(2,4,6) ]), D(6) = S(3)[x]2 ]
sage: K.class_group()
Multiplicative Abelian Group isomorphic to C4
```

Not Just For Number Theory... E.g., Toric 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)]
```

Thanks