SAGE: Software for Algebra and Geometry Experimentation

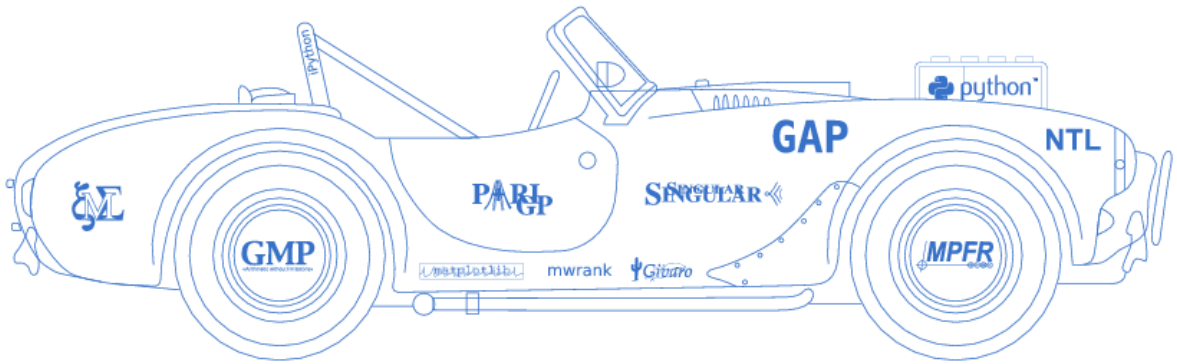
William Stein

December 4, 2006, University of Waterloo

http://modular.math.washington.edu/sage

SAGE

Building »The Car«



»Every free computer algebra system I've tried has reinvented many times the wheel without being able to build the car.«

Background: From HECKE 0.1 to SAGE 1.4

- **1997–1999:** HECKE – my free C++ program for **modular forms** (I wrote an interpreter for it).
- **1999–2004:** I wrote $> 25,000$ lines of Magma code.
- **Feb 2004:** Decide I will not go through my life not knowing how my computations work, and not being allowed to make my software available to students for free. I looked for alternatives to Magma, but Magma is **vastly superior** to everything else for my work.
- **Feb 2005:** I got job offers with **tenure** – **SAGE 0.1**.
- **Feb 2006:** **SAGE Days 1** workshop – **SAGE 1.0**.
- **June 2006:** **High school** workshop – Notebook.
- **August 2006:** **MSRI Grad student** workshop.
- **October 2006:** **SAGE Days 2** workshop.
- **This week:** SAGE 1.5; things are getting exciting.

What is SAGE?

- SAGE is **free open source software** for research in **algebra**, **geometry**, **number theory**, **cryptography**, and **numerical computation**.
- SAGE is an **environment for rigorous mathematical computation** built using Python, GAP, Maxima, Singular, PARI, etc., and provides a **unified interface** to Mathematica, Maple, Magma, MATLAB, etc.
- There have been **several successful SAGE workshops**, and there are many active SAGE developers.
- The **primary goal** of SAGE is to make modern research-level algorithms available in an integrated package with a graphical interface.

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? Moreover: Do we really want to charge colleagues in Moldava several years of their salary for a computer algebra system?"

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

What About MAPLE?

There is a new PDE solver that will be in **Maple**, written for free by a mathematician. My student found out about it at a conference, and wanted to create something similar for SAGE. Someone remarked *"I imagine this would be quite difficult but don't see that "copying" would be an issue."* **This opinion about Maple is common...**
We wrote to Maple to be sure; they said that once anyone includes their routines in Maple it becomes **illegal to use them as a basis for doing anything anywhere else ever**.

```
Reproducing and redistribution of Maple code is a violation of
the license agreement.  this is a direct violation of the EULA
[...] Without the express written permission of Maplesoft,
Licensee shall not, and shall not permit any Third Party to:
(a) reproduce, transmit, modify, adapt, translate or create
any derivative work of, any part of the Software, in whole
or in part ...
(b) reverse engineer, disassemble, or decompile the Software,
create derivative works based on the Software, or otherwise
attempt to gain access to its method of operation or source;
Sincerely, Maplesoft Technical Support
```

Who is Writing SAGE?

Contributors Include: Martin Albrecht, Tom Boothby, Robert Bradshaw, Iftikhar Burhanuddin, Craig Citro, Alex Clemesha, John Cremona, Didier Deshommes, David Harvey, Naqi Jaffery, David Joyner, Josh Kantor, Kiran Kedlaya, David Kirkby, Emily Kirkman, David Kohel, Jon Hanke, Bill Hart, Robert Miller, Bobby Moretti, Gregg Musiker, Bill Page, Fernando Perez, Yi Qiang, David Roe, Michael Rubinstein, Nathan Ryan, Kyle Schalm, Steven Sivek, Jaap Spies, Gonzalo Tornaria, Justin Walker, Mark Watkins, Joe Weening, Joe Wetherell, ...

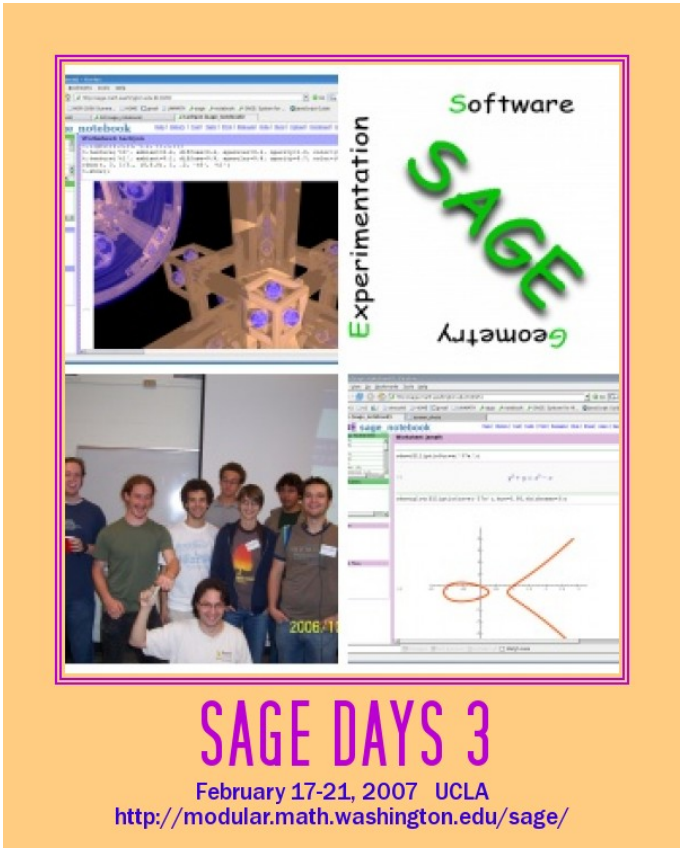
- **7 Undergraduates**: have many **extremely interesting** ideas; superb at researching available free software.
- **Many graduate students**: excellent at implementing optimized code and finding fast algorithms.
- **Faculty and computer professionals**: general direction, great writing, and quality control.

SAGE Days 2: Coding Sprints...

Bobby Moretti (UW undergrad), Robert Miller (UW grad), David Harvey (Harvard grad), Joel Mohler (grad), David Joyner (USNA), Bill page (Axiom).

Upcoming SAGE-related Workshops I'm Organizing

- **Parallel Computation Workshop** at MSRI, Jan 29-Feb 2, 2007.
- **SAGE Days 3** at IPAM (in LA) Feb 17-21, 2007.



- **AIM**, workshop on *L*-functions and modular forms, July 30-Aug 3, 2007; Michael Rubinstein is a co-organizer.

Getting Started with SAGE

1. **Free online** SAGE notebook: `http://sage.math.washington.edu:8100`
2. **Website:** `http://sage.math.washington.edu/sage`
3. **Documentation:** Tutorial, Install Guide, Programming Guide, Reference Manual, Constructions.
4. **Binaries:** For OS X, Windows, and Linux (and building from source is easy). (Windows support needs work.)
5. **Mailing lists:** sage-devel ($>$ 500 messages/month), sage-announce, sage-forum, sage-support.
6. **Wiki:** the SAGE wiki.
7. **Trac:** Organizes development.
8. **IRC Chatroom:** `#sage-dev` on irc.freenode.net

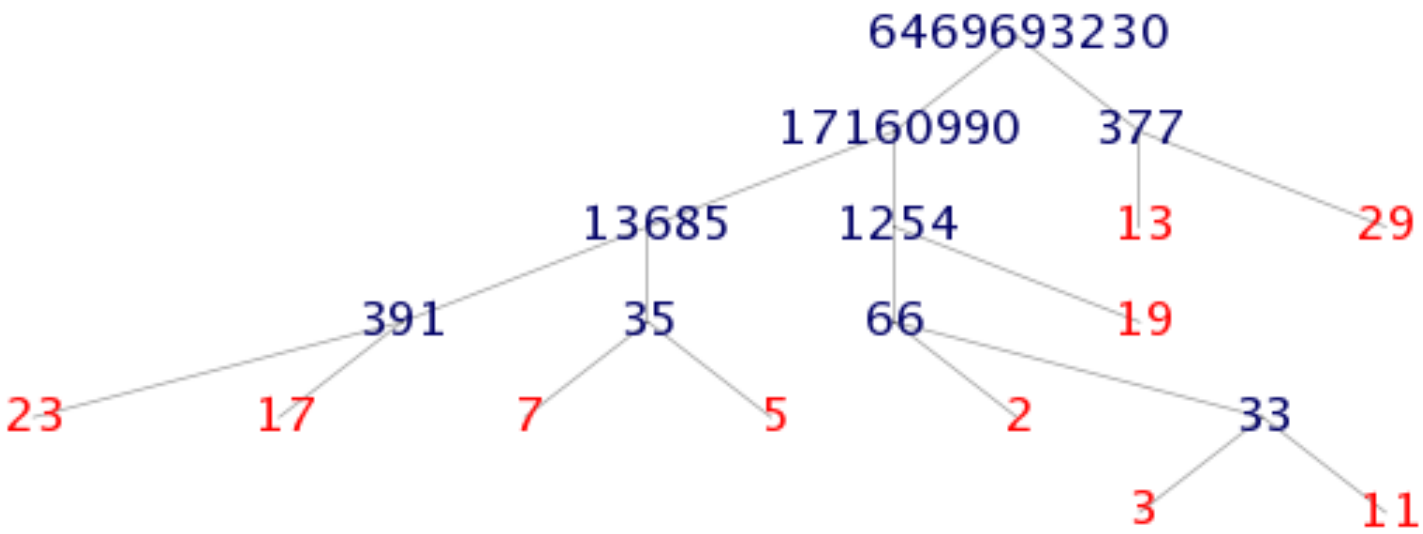
Some Fancy Hardware – A Collaboration Environment

http://sage.math.washington.edu/home

64GB RAM, **16 processor** Opteron server. You can browse all the developer's home directories over the web here!

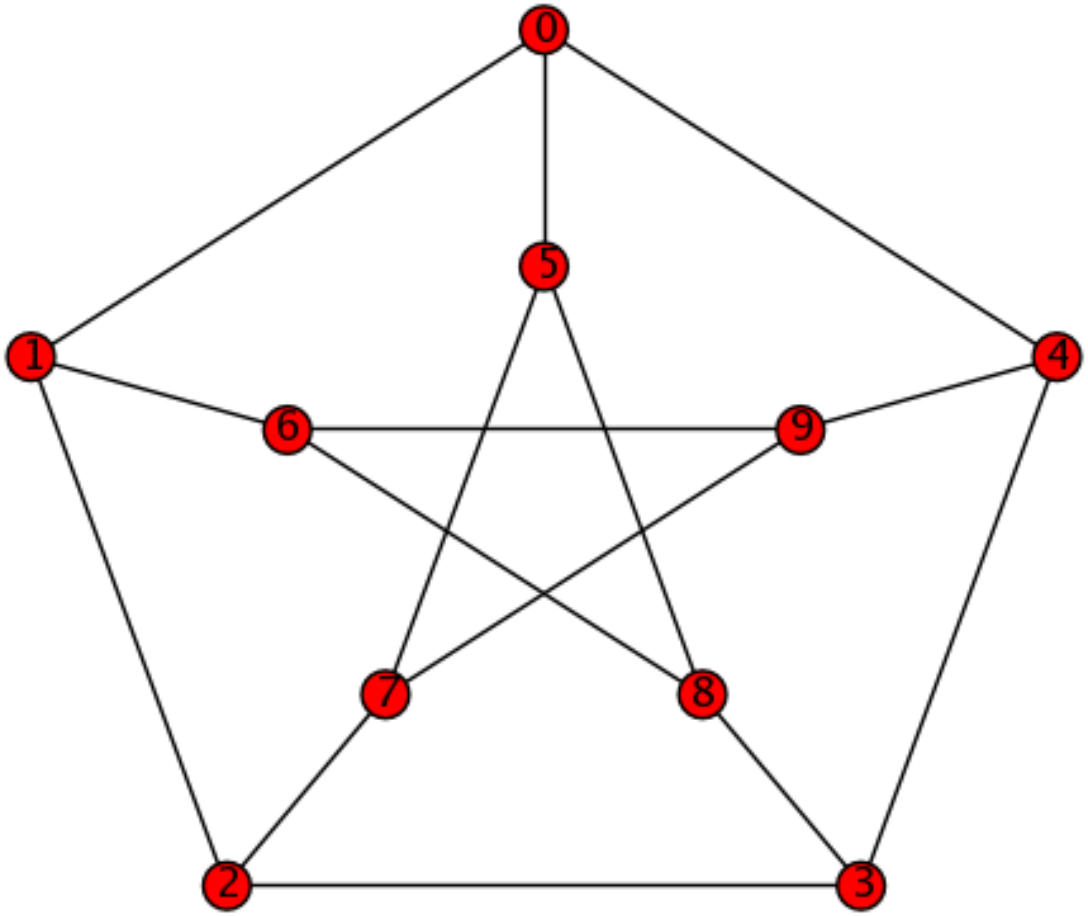
SAGE Demo: Educational Applications

```
sage: notebook()
----
F = factor_tree(prod(primes(30)))
F.show(xmin=-3,xmax=7,ymin=-5,ymax=0.5,
       figsize=[8,2],axes=False)
...
```



SAGE: Excellent Graph Theory

```
sage: g = graphs.PetersenGraph()
sage: show(g)
```



What is SAGE?

SAGE is:

1. **A Distribution** of free open source math software. 64MB source tarball that builds self-contained.
2. **New Readable Code** that fill in gaps in functionality; implement new algorithms.
3. **A Unified Mainstream Interface** to math software: to **Magma**, **Macaulay2**, Singular, **Maple**, MATLAB, Mathematica, Axiom, etc.

SAGE runs on **Linux, OS X, and Windows**.

1. A Distribution

Basic Arithmetic	**GMP, NTL, MPFR, PARI**
Command Line	**IPython**
Commutative algebra	**Singular** (libcf, libfactory)
Database	**ZODB**, Python Pickles
Graphical Interface	**SAGE Notebook, jsmath**
Graphics	**Matplotlib, Tachyon, GD**
Group theory and combinatorics	**GAP**
Graph theory	**Networkx**
Interactive programming language	**Python** (mainstream !!!)
Networking	**Twisted**
Numerical computation	**GSL, Numpy, etc.**
Symbolic computation, calculus	**Maxima**

All core components are **free and open source** (mostly GPL'd). You may **read the code** and **change anything** in SAGE or any of the core libraries it includes, and redistribute the result.

SAGE Demo: A Distribution

```
$ wget http://sage.math.washington.edu/sage/dist/
src/sage-1.4.1.2.tar
$ tar xvf sage-1.4.1.2.tar
...
$ cd sage-1.4.1.2
$ make      # completely automatic on OS X and Linux (!)
...
$ ./sage
-------------------------------------------------------
| SAGE Version 1.4.1.2, Build Date: 2006-10-19        |
| Distributed under the GNU General Public License V2. |
-------------------------------------------------------
sage: install_scripts('/home/was/bin/')
 ... (installs gap, gp, singular, etc. scripts).
$ /home/was/bin/gap
GAP4, Version: 4.4.8 of 18-Sep-2006, i686-apple-darwin8.7.1-gc
gap>
```

2. New Code

Python and Pyrex code — **designed to be readable**:

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

```
UNIQUE Source Code Lines (including docstrings):
$ cat */*.py */*/*.py */*/*/*.py */*.pyx \
                */*/*.pyx */*/*.pyx |sort |uniq | wc -l
95706
UNIQUE Input Documentation Examples:
$ cat */*.py */*/*.py */*/*/*.py */*.pyx \
          */*/*.pyx */*/*.pyx |sort|uniq|grep "sage:" | wc -l
11105
```

SAGE Demo: New Code (interactive help)

```
sage: bernoulli?   # one ? for help
    Return the n-th Bernoulli number, as a rational number.
    INPUT:
        n -- an integer
        algorithm:
            'pari' -- (default) use the PARI C library, which
                      by *far* the fastest.
            'gap'  -- use GAP
            'gp'   -- use PARI/GP interpreter
            'magma' -- use MAGMA
            'python' -- use pure Python implementation

    EXAMPLES:
        sage: bernoulli(12)
        -691/2730
        sage: bernoulli(50)
        4950572052410796482124775 25/66
    ...
    AUTHORS: David Joyner and William Stein
```

SAGE Demo: New Code (interactive help)

```
sage: bernoulli??          # two question marks for source code
File: ... python2.5/site-packages/sage/rings/arith.py
...
    if algorithm == 'pari':
        x = pari(n).bernfrac()     # Use the PARI C library
        return Rational(x)
    elif algorithm == 'gap':
        x = sage.interfaces.gap.gap('Bernoulli(%s)'%n)
        return Rational(x)
    elif algorithm == 'magma':
        x = sage.interfaces.magma.magma('Bernoulli(%s)'%n)
        return Rational(x)
    elif algorithm == 'gp':
        x = sage.interfaces.gp.gp('bernfrac(%s)'%n)
        return Rational(x)
    elif algorithm == 'python':
        return sage.rings.bernoulli.bernoulli_python(n)
    else:
        raise ValueError, "invalid choice of algorithm"
```

SAGE Demo: Unique New Code

```
sage: bernoulli_mod_p?
    Computes bernoulli numbers B_0, B_2, ... B_{p-3}
    modulo p.
    PERFORMANCE: Should be complexity O(p log p).
    INPUT: p -- integer, a prime
    OUTPUT: list -- the bernoulli numbers modulo p.
    EXAMPLES:
        sage: bernoulli_mod_p(37)
         [1, 31, 16, 15, 16, 4, 17, 32, 22, 31, 15,
          15, 17, 12, 29, 2, 0, 2]
    AUTHOR: David Harvey (2006-08-06)
```

This implements a famous algorithm of Buhler et al.

And there is much much more that is unique in SAGE.

3. A Unified Interface

- SAGE **interfaces to**: Axiom, GAP, GP/PARI, Kash, Macaulay2, Magma, Maple, Mathematica, MATLAB, Maxima, Octave, Singular, etc.
- Wide range of **functionality**.
- Unified **command completion and help**.

SAGE Demo: Interfaces

HOW IT WORKS: Use buffered psuedo-tty's and Python objects that wrap native objects. This makes it possible to wrap **all** math software that has a command line interface using very similar code.

```
sage: x = singular('2+3')
```

This fires up one copy of Singular (if it wasn't already started) and sends the line `'2+3'` to Singular. It also creates a Python class R with a field set to `"sage0"`.

```
sha:~ was$ ps ax |grep Singular
21664  pe  Ss+    0:00.01 /bin/sh /Volumes/HOME/s/local/bin/Si
21666  pe  S+     0:00.06 Singular-3-0-2 -t --ticks-per-sec 10
sage: type(x)
<class 'sage.interfaces.singular.SingularElement'>
sage: x
5
sage: R.name()
'sage0'
```

The Overall Structure of SAGE

- **Custom package management system** – 46 standard packages, and 32 optional ones. Automated upgrades.
- **Awesome interactive command-line** interface – IPython.
- **Graphical user interface** – via your web browser (AJAX app).
- **Fast underlying arithmetic** – built on mature robust C libraries (GMP, NTL, PARI, GSL). New code in C, Pyrex and Python.
- **Interfaces with other software** use buffered **psuedo-tty**'s.
- **Special purpose components** – e.g., Rubinstein's **Lcalc**, **GMP-ECM and FlintQS** (for integer factorization), etc.
- **Mercurial revision control system** – included standard; encourages users to be developers.

The SAGE Notebook: GUI For Mathematics Software

1. The SAGE Notebook – an **"AJAX application"** like Google maps or Gmail.
2. **Written from scratch** by me, Alex C. and Tom B.
3. Uses Python's built-in **BaseHTTPServer** web server (we will switch to Twisted for robustness).
4. Works well with Firefox, Safari, Opera, and Konqueror.
5. Client/server model which works **over network** or locally.
6. Current version is **stable and in use by many people**.
7. Try it: `http://sage.math.washington.edu:8101`

Goals for SAGE 2.0

Planning and bug tracking is **done in the open**:

`http://sage.math.washington.edu/trac`

Main Goals for SAGE 2.0 (January 31, 2007):

1. Optimize **basic arithmetic**, e.g., finite fields, exact linear algebra, etc.; this involves moving classes from interpreted Python to compiled code.
2. Improve the **SAGE Notebook**: easier to edit, better security and robustness.
3. Improve **graphics**: 3d graphics, a java applet for the SAGE notebook.

SAGE in 2007: Parallelism

1. Support for **parallelism** and use of it for algorithms, e.g., multimodular matrix multiply over $\mathbb{Q}$.

2. I'm co-organizing a **workshop January 29–Feb 2, 2007** at MSRI on parallel computation, which will help get the ball rolling.

Questions?