SAGE Days 3: A Social and Technical Status Report

William Stein

February 17, 2007, SAGE Days 3

SAGE: Social Status Report

The SAGE Goal: Mission Statement

THE SAGE GOAL: Create a viable completely free open source alternative to Magma, Matlab, Maple, and Mathematica.

Thus SAGE aims to be the analogue for mathematics research of **Firefox** for web browsing and **Linux** for operating systems. *Science and mathematics is supposed to be done openly*.

THE SAGE GOAL is incredibly frickin' hard for a number of reasons. In my opinion nobody has come close yet. When I think about how hard this is, I realize it is basically impossible.
Solution: Do not think about how hard it is.

But we **must** create SAGE. Otherwise a few people and corporations will *continue to exert undue control* over mathematical research, especially as use of computers in math research becomes increasingly important (as it will).

How big is SAGE?

Short answer – 85MB in source form.

- **57 standard packages:** This is close to the limit of what I can maintain. I plan to add a sparse numerical linear algebra package (e.g., SuperLU), pytables, lie, and scipy. People from the numerical community like the SAGE package system, and we are combining forces.
- **27 optional packages:** Lesson learned –
  1. The build-from-source optional packages are a **maintenance nightmare**, since testing that they work on a wide range of systems is incredibly hard.
  2. In the long run all build-from-source optional package should either disappear or become standard. (The ones that don't should be installed via standard tools, e.g., fink, apt-get, Cygwin, etc.)
  3. The goal with SAGE is having a *complete uniform well-tested working system*, and optional packages break this, since often they do **not** just work.
- The **core SAGE library** contains **115,322 unique** lines of code and docstrings. This is still quite manageable.

The Current Active Developers

- **21 people have had patches to the SAGE library accepted during the last two months (many have submitted thousands of lines of code):**

  Martin Albrecht, Nick Alexander, Tom Boothby, Robert Bradshaw, Iftikhar A. Burhanuddin, Alex Clemesha, Didier Deshomme, David Harvey, Josh Kantor, Emily Kirkman, David Kohel, Robert Miller, Joel B. Mohler, Bobby Moretti, Andrey Novoseltsev, Dorian Raymer, Yi Qiang, Steven Sivek, Jaap Spies, William Stein, Carl Witty...

  This is a huge amount of work. Applause!

- Many people have contributed packages, bug fixes to packages, code snippets, etc.

Growing Pains

- I spend a **huge amount of time** just getting patches to **work correctly** and have enough doctests, etc. Soon I will be very busy with teaching, etc., and won't always be able to do this at the currently level.
- So establishing a **patch refereeing system** is crucial for SAGE; otherwise, a lot of great patches simply won't make it into SAGE, which will discourage potential developers.
- This referee system will unfortunately slow down the inclusion of new code into SAGE, but will in the long run **raise quality** and **scales well**.
- It will also help improve the chances work on SAGE will be viewed positively by **hiring and fellowship committees**.

SAGE Patch Referee System

- I would officially like the following people to become the first group of patch referees: **Martin Albrecht, Nick Alexander, Robert Bradshaw**, and **David Harvey.**
- Proposed Workflow (a nag programs is needed for this):
  1. I receive a patch. I **send it** to one of the referees.
  2. The referee applies the patch, reads through all relevant code added in the patch, and adds **comments specifically asking** for more doctests, asking for explanations about how/why things are coded in a certain way, asks for coding conventions to be met, etc.
  3. The **referee records as a new patch** the commented code.
  4. The **patch author answers** in the form of another patch.
  5. The above three steps are iterated until the **referee is happy**.
  6. I receive the patch, read it, and apply it unless there are still problems.
- This process would *not* be anonymous.
- Contributors would be strongly encourage to list refereed patch contributions as such in a section of their CV's.

Funding

Funding for SAGE is **crucial**. **Why??**

- So faculty and graduate student can **work on SAGE more** and teach less (or do teaching that is more related to either SAGE or their research).
- So **undergraduates can work on mathematics-related software** as a job instead of working for other departments or companies. (E.g., the UW undergrads working on SAGE all had jobs in web page design and biology support before working on SAGE). This is **better for their education**.
- To fund **SAGE Days workshops** so developers and users can meet.
- The competition (Magma, Maple, Mathematica, and Matlab) **has many many millions of dollars** per year in funding, and it's difficult to even begin to compete without funding.

Funding SAGE isn't about **me** getting grants, it's about **everyone** getting grants. Undergrads who apply to university research programs for stipends, grad students who apply to local institutes for workshops, etc. These are all happening. Keep it up!

- I still have some startup money left – enough to keep things going for at least 1 more full year, and possibly 2 years, even if I get no funding I've applied for.
- GOOD NEWS: Email from Jon Hanke yesterday

```
The good news is that I have [substantial] startup
money,  which I'd like to purchase a reasonably
sized computer to do (numerics and) SAGE development,
hire student programmers, and possibly fund 2 SAGE
Days conferences at UGA (if you'd like).  So if you
have any thoughts about how you'd like this to work,
I'd be happy to try to implement them. =)
```

- SAGE Days 4? Probably in late August at UW??
- Possibilities for SAGE Days 5 – CRM in Vancouver (Nils Bruin); Leiden in Netherlands.

SAGE: Technical Status Report

SAGE 2.1.3.1: Technical Status Report

The Components of SAGE: What and Why

- cddlib-094b – program to generating all vertices and extreme rays of a general convex polyhedron in $\mathbf{R}^d$ given by a system of linear inequalities (needed by gfan, but I wish it's functionality were directly exposed).
- clisp-2.41 – the common lisp interpreter (needed by Maxima; could be replaced by GCL, but clisp is the only *only* lisp that *really* builds everywhere)
- conway_polynomials-0.1 – a database of polynomials that define finite fields (not really used – should be used to implement natural embeddings between finite fields – volunteers?)
- cremona_mini-0.1 – Cremona's database of elliptic curves of conductor up to 10000.
- doc-2.1.3 – install guide (good), tutorial (good, but needs update), reference manual (needs way more explanatory overviews and better organization), and constructions guide (needs to be updated!)

- ecm-6.1.2 – elliptic curve factorization method (fast – good dsage example)
- examples-2.1.3 – examples of use of SAGE (needs to be revamped so all examples can be automatically tested – this would be a good coding sprint project).
- extcode-2.1.3 – code used by SAGE in other languages (e.g., in GP/PARI).
- flintqs-20070102 – Bill Hart's multi-polynomial quadratic sieve.
- freetype-2.1.10 – used by GD
- gap-4.4.9 – Group theory.
- gd-2.0.33.p4 – fast 2d graphics library (render, e.g., fractals much more quickly than with matplotlib).

- gdmodule-0.56.p2 – python bindings for gd library
- genus2reduction-0.3 – conductor and reduction type of genus 2 curves (an important orphaned C program)
- gfan-0.2.2.p1 – compute with Groebner fans; lies at the heart of the popular research area called “tropical goemetry”.
- givaro-3.2.6 – fast finite field arithmetic library
- gmp-4.2.1.p4 – arbitrary precision integers and rationals
- gsl-1.8 – the GNU Scientific Library; a mature self-contained C library for a huge range of numerical computation (discuss how GSL has similar functionality to scipy)
- ipython-20061028 – the interactive IPython command line (provides SAGE’s interactive shell)
- ipython1-20070130 – makes it easy to interactively control execution of dozens of copies of SAGE on a cluster

- lcalc-20070107 – Mike Rubinstein's C++ program for computing with L-functions; it has tons of functionality that hasn't yet been exposed in SAGE.
- libpng-1.2.8.p0 – library for manipulating png images.
- linbox-20070214 – a very powerful C++ library for exact linear algebra
- matplotlib-0.87.7 – 2d Python graphics package; provides Matlab like functionality, and – thanks to Alex Clemesha – SAGE provides Mathematica-like 2d plotting on top of matplotlib.
- maxima-5.11.0 – a very mature sophisticated computer algebra system for symbolic computation. I am honestly worried about efficiency and maintenance issues involved with building SAGE's CAS functionality on top of Maxima, since CAS is very important for the mission of SAGE. We shall see.

- mercurial-0.9.3.p1 – distributed revision control system; every single copy of SAGE (both built-from source or binary) contains 5 separate mercurial repositories! Everyone can check in changes, send patches around, etc. This has been crucial to building the developer community.
- moin-1.5.6 – a mature full-functional wiki is included in SAGE.
- mpfi-1.3.4-rc3.p2 – very fast arbitrary precision interval arithmetic
- mpfr-2.2.1 – we build fast arbitrary precision real and complex number arithmetic on top of MPFR
- mwrank-20061123 – John Cremona's program for computing with elliptic curves. There is some functionality in this program that could be made available in SAGE that isn't currently available.
- networkx-0.33 – a solid graph theory package; Emily Kirkman and Robert Miller have done substantial work to make this much more usable from SAGE.

- ntl-5.4.1 – fast arithmetic in many cases, though it frequently gets beat by MAGMA. Required by both Singular and Linbox. It has a good LLL implementation. The SAGE/NTL interface needs to be optimized (recent work of David Harvey).
- numpy-1.0.1.p1 – excellent library for numerical linear algebra in Python; needs to be better documented and easier to use from SAGE. Very powerful. Is core infrastructure for the Python numerical computation community (which is much bigger than the SAGE community!)
- openssl-0.9.8d.p1 – library for security internet communication; currently used by dsage, and will soon be used by the SAGE notebook.

- palp-1.1 – a program for computing with lattice polytopes, which are objects of great importance in algebraic geometry, combinatorics, number theory, etc.
- pari-2.3.1.cvs-20061215 – free number theory library and interactive calculator; much of SAGE's number theory functionality is built on top of PARI. Also, the PARI library provides arbitrary precision numerical integration, evaluation of special functions to high precision, etc.
- pexpect-2.0 – used so SAGE can communicate with almost any other program via a pseudo-tty. (There is a pexpect-2.1, but it doesn't work for crap with SAGE. Why?!)

- pycrypto-2.0.1 – package that implements cryptographic algorithms and protocols. E.g., Hash functions (MD2, MD4, RIPEMD, SHA256), block encryption algorithms (AES, ARC2, Blowfish, CAST, DES, Triple-DES, IDEA, RC5), stream encryption algorithms (ARC4, simple XOR), and public-key algorithms: RSA, DSA, ElGamal, qNEW. YES – quality implementations of all these are included standard in SAGE. That this is the case, and how to use them should be documented much better.
- pyopenssl-0.6 – needed by Python programs that use openSSL (e.g., Twisted).
- pyrexembed-0.1.1-20060517 – currently not used much; slated for possible removal (?)
- pysqlite-2.3.2 – allows one to use the sqlite relational database from Python.
- python-2.5.p4 – the Python interpreter, which is the interpreter for SAGE.

- readline-5.2 – distributed with SAGE since the readlines on a lot of systems are often totally screwy, and often people don't have the readline devel headers.
- sage-2.1.3 – the core SAGE library; implementation of new algorithms and defines the "unified interface" to all these packages that SAGE presents to the world.
- sage_c_lib-2.1.3 – a range of code used by the SAGE libraries that is written in C. E.g., our interrupt handler is here, along with a number of cool optimization tricks that came out of SAGE Days 2.
- sage_scripts-2.1.3 – these are mostly shell scripts that implement the SAGE package manager.
- sagex-20070126 – SageX is a sort of "compiled variant of Python". It's utterly crucial for writing certain types of code for SAGE. Martin Albrecht will give a great talk about SageX. This is our (unfortunately necessary) fork of Pyrex.

- singular-3-0-2-20070105 – polynomial computation. Has a huge range of functionality for commutative algebra, and is also supposed to have powerful non-commutative algebra functionality as well. None of the non-commutative functionality has been made available directly in SAGE (coding sprint idea). Martin Albrecht has also embarked on a fascinating project to make Singular directly usable by SAGE at the C-library level – which promises to be "frickin' insanely fast" in some ways.
- sqlite-3.3.11 – SAGE's relational database: "SQLite is a small C library that implements a self-contained, embeddable, zero-configuration SQL database engine." For certain types of mathematical tables, SQLite will be amazingly useful.
- sympow-1.018.1.p1 – Mark Watkins's program for computing symmetric power L-functions. Fairly specialized.

- tachyon-0.97.p1 – a very lightweight ray tracing program. Fun. Has motivated development of some useful 3d graphics and plotting algorithms by Tom Boothby and Josh Kantor.
- termcap-1.3.1 – used by many other programs to determine terminal capabilities (friend of readline).
- twisted-2.5.0.p2 – "the ultimate" event driven networking framework; support TCP, UDP, SSL/TLS, multicast, Unix sockets, a large number of protocols (including HTTP, NNTP, IMAP, SSH, IRC, FTP, and others), and much more. This is a typical example of the sort of functionality SAGE has that sets it apart from Maple/Mathematica/Magma/Matlab.
- twistedweb2-0.2.0 – relevant for Twisted and the SAGE Notebook, but not used by SAGE yet.

- weave-0.4.9 – mature system from inlining pure C code inside Python programs. Very powerful. I've never used it, but Josh Kantor has and swears by it. It's especially good for numerical computation.
- zlib-1.2.3.p1 – compression.
- zodb3-3.6.0.p1 – a robust mature Python object-oriented database; basically this is a way to have a potentially huge Python dictionary that is mostly stored on disk.

Main Current Development Directions

For SAGE, I think this is the year of *optimization*.

- **Parallel computation** – **an optimization technique**:
  1. Crucially requires robust **object serialization** (pickling).
  2. **DSAGE** – Goal is super-easy task farming.
  3. **IPython** – Goal is easy control of a bunch of SAGE's at once (and support for using MPI).
  4. **pthreads** – Limited use for low level libraries
- **Fast numerical and exact linear algebra**: use and improve linbox. Implement new advanced optimized algorithms in the context of SAGE (in progress). More optimized classes.
- **Fast basic arithmetic:** FLINT, optimize all basic arithmetic (e.g., SageX more of polynomials, power series, p-adics, etc), fast multivariate polynomials via Martin's new Singular library interface, speed up object creation and memory management.

Discussion

Discussion