William A. Stein ***Project Description***

(858) 220-6876 wstein@math.washington.edu http://sage.math.washington.edu

SAGE: Software for Algebra and Geometry Experimentation

Major Points

- 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.

1 Introduction

SAGE is a project that the PI launched in January 2005 to create a software environment for research and experimentation in algebra, geometry, number theory, cryptography, and numerical computation. This involves the creation of free open source software and databases that support advanced mathematical research.

The PI is one of the more sought-after people by the world-wide community of mathematicians, for computational confirmation of conjectures, for algorithms, for data, and for ways of formulating problems so as to make them more accessible to algorithms. He is also successful at bringing together a wide range of people to work on collaborative mathematics projects; for example, he organized the MSRI workshop [30], two NSF-funded SAGE Days workshops [10; 11], and will organize several more workshops during the next year [1; 31; 12; 2]. He has extensive experience with software development, having written HECKE, much of SAGE, and over 25,000 lines of code that are included in Magma.

1.1 Prior Support

The PI has used his startup money at UC San Diego and University of Washington, his NSF grant (DMS-0555776), and the UW's VIGRE grant to support two SAGE workshops, six undergraduate employees, four graduate students, and research visitors. **Obtaining grant funding is critical to the continuation of SAGE development at its present level.**

Other Grant Applications: The PI has also applied this year to the NSF CSUMS program to fund an *undergraduate education project* entitled "CSUMS:

SAGE-Software for Algebra and Geometry Experimentation", and to the NSF ANTC program to support his *computational and theoretical work* on the Birch and Swinnerton-Dyer conjecture.

2 SAGE and the Mathematical Software Landscape

First we discuss the relationship between SAGE and many high quality mathematical software packages. Then we explain precisely what SAGE is and how it unifies existing mathematical software via its flexible interfaces. We also discuss independent verification of research results, free open source mathematics software, and the SAGE development community.

2.1 Mathematical Software

In this section we briefly describe the closed source programs Magma, Maple, Mathematica and MATLAB, and discuss several open source programs.

Magma [13] is a high quality, mature, closed source program developed in Australia. It is a non-free (indeed, quite expensive) software package with a vast range of functionality in number theory, algebraic geometry, arithmetic geometry, graph theory, and group theory. It features a specialized language for mathematics. Magma has an excellent command line interface, though it does not have a graphical user interface. Magma has very limited support for numerical computation and symbolic integration. It is not possible for end users to define their own datatypes, but the Magma developers are often responsive to requests to add new datatypes to the C language core for the next release of Magma. In Magma it is easy to save the complete state of a running Magma session, though there is no way to evaluate arbitrary strings, so it is difficult to store to disk individual objects that one has computed. Magma's huge and highly optimized library is far more powerful and comprehensive for research in certain areas (e.g., arithmetic geometry, group theory, coding theory) than perhaps anything else available. Much of Magma is written in C, so it can link in functionality from some existing C libraries (e.g., GMP), though such linking is only possible by rebuilding Magma and end users are not allowed to build Magma from source. Also, many freely available libraries cannot be legally linked to Magma because of license restrictions.

Maple [14] is a powerful closed source commercial program developed in Canada. It features a sophisticated graphical user interface, and has a wide range of functionality, especially for symbolic computation and plotting relevant to undergraduate education. Its programming language is also highly regarded. Much of the source of Maple is distributed with Maple and can be viewed by users, though unfortunately such interpreted source files can run slowly (see Section 2.4); some arithmetic and many functions related to number theory are notably slower than the corresponding functions in Magma (see Section 3.1.2).

Mathematica [15] is an optimized, closed source, commercial mathematics software package developed in the USA. It has a unique mathematical programming language, a vast library of optimized special functions, powerful symbolic manipulation capabilities, and sophisticated 2 and 3-dimensional plotting. It also features a pioneering graphical user interface. In many areas it has substantial coverage of important algorithms, though in number theory, arithmetic geometry, and cryptography, it has limited functionality.

MATLAB [17] is closed source commercial software developed in the USA, which focuses on numerical computation. It has a wide range of functionality for numerical computation, excellent performance, a graphical user interface, and a custom programming language. It also has symbolic computation capabilities via a link to Maple's symbolic engine.

GAP [6], **PARI** [22], **Singular** [27], **Macaulay2** [7] and **Maxima** [18] are large, mature, high quality, free, open source mathematics software programs. All are currently supported and under active development. There is surprisingly little overlap between their functionality (except Macaulay2 and Singular). For example, GAP has sophisticated functionality for computing with groups, but limited support for commutative algebra or number theory, whereas Singular has a huge range of commutative algebra functionality and PARI has substantial number theory functionality but little group theory or commutative algebra.

Table 1: There is little overlap in functionality among free math software

Function	Singular	Gap	PARI	Maxima
Multivariate polynomial factorization	Yes	No	No	Yes
Subgroup enumeration	No	Yes	No	No
Class groups of number fields	No	No	Yes	No
Symbolical integration	No	No	No	Yes

SAGE incorporates all four programs listed in Table 1 (and many others) into a single unified program so it does all the listed problems well.

Every mathematics software system described above has its own special-purpose programming language. Many of these languages make it easy to express mathematical concepts and get results quickly. Unfortunately, most do not support modern exception handling (like that available in C++, Java, and Python), name spaces and many cannot be extended with new data types (or classes). For example, PARI's programming language is useful for writing short programs but awkward for large projects. In addition, these special purpose mathematics languages can be difficult to use when one needs to do non-mathematical programming (e.g., create a web server to distribute the results of computations, or connect to an online database or web page and parse the results).

Some of the closed source systems, most notably recent versions of Mathematica and MATLAB, make use of multiple processors on a computer when available. None of the open source programs listed above do. Tightly integrated support for parallel computation is a key goal of SAGE.

2.2 SAGE: A Unifying Approach

SAGE is

1. A **distribution** of free open source mathematics software,
2. A **new interpreter, graphical user interface, and library** of implementations of mathematical algorithms,
3. A **better way to use** existing mathematics software together.

2.2.1 A Distribution of Math Software

SAGE is a cohesive distribution of several dozen pieces of open source mathematics software, with almost no external dependencies, which can easily be built completely from source with one command on OS X, Linux, and Windows. Creating this distribution has required the combined work of numerous specialists in the arcane build processes of different operating systems, and requires regular maintenance (which would be funded by this proposal) to remain up to date. The PI has often been told by mathematicians that various open source programs, e.g., PARI and Singular, can easily be built as part of SAGE, but are "impossibly difficult" for them to build without SAGE. The PI also distributes prebuilt binaries of SAGE for some of these platforms.

The PI is requesting funds in order to regularly **purchase new hardware** for testing automated building of SAGE from source on a wide range of platforms, and building binaries. The PI used $38,000 in funding from NSF grant (DMS-0555776) for the purchase of a 16-core computer with 64GB RAM that is the main home for SAGE development. But a much wider range of equipment is needed to test and optimize implementations of algorithms on diverse hardware. The PI must own such hardware in order to have dedicated access to it for intensive testing of new implementations; moreover, regular compilation of SAGE from source is resource intensive. Also, in many cases (e.g., Itaniums) it is nearly impossible to gain sufficient access to hardware without purchasing that hardware. Finally, it is important to test building of SAGE on both a range of hardware and operating system installs, which requires owning the hardware.

2.2.2 A New Interpreter, Graphical Interface, and Library

SAGE improves on the existing systems by using the Python programming language, which is a popular, mainstream, open source, free programming language with support for programming constructs that are useful in mathematics, such as user-defined classes, list comprehensions and regular expressions.

SAGE can be used interactively from the SAGE command line [23], via the SAGE Notebook, and as a library from Python and C/C++ programs.

As mentioned before, the other major mathematics software packages have their own custom programming languages. This can be particularly costly to occasional and beginning users of mathematical software, as well as those whose work straddles several mathematical subdisciplines. The use of Python for SAGE avoids this problem. Moreover, because Python has long been used for database and web programming applications, it provides excellent support for saving and loading of individual objects and network programming. Also Python has a massive standard library and active development community that is constantly optimizing the language and moving it forward.

The SAGE Notebook is a sophisticated web-browser based graphical user interface, which the PI created in response to student demand. Graphical output can be easily embedded in the notebook, and the notebook can be used as a front end for many other mathematical software systems that do not have a graphical interface, including Magma, PARI, and Singular. The PI intends to use funds to improve the robustness and security model of the notebook using Twisted [33].

The PI hopes to include support in SAGE for practical parallel computation at several different levels. For example, the PI is chairing the organizing committee of the upcoming MSRI workshop "Interactive Parallel Computation in Support of Research in Algebra, Geometry and Number Theory" [1], which he funded using a grant from the National Geospatial Agency. Also he has employed Yi Qiang for several months to design and implement functionality for coarse grained parallelism that is tightly integrated with SAGE.

The PI also hopes to use this grant to fund work by SAGE developers to create highly optimized new code for SAGE for basic arithmetic, number theory, algebraic geometry, arithmetic geometry, cryptography, graph theory, numerical computation, and group theory (see Sections 3.1–3.2).

2.2.3 Using Existing Mathematical Software from SAGE

The PI invented and implemented for SAGE a new way to create and work with numbers, matrices, vectors, polynomials, graphs, functions, and all other mathematical objects defined in other mathematical software, including Axiom, Maple, Mathematica, Macaulay2, Magma, MATLAB, Octave, Maxima, Singular and PARI. These interfaces are mostly implemented using pseudo-ttys and files, so they can be created for any program with a command line interface, and it is not necessary to make any changes to that program. A running instance of SAGE can thus simultaneously orchestrate dozens of other mathematics programs. This gives SAGE a rich range of features, and makes it possible for researchers to use

SAGE while still using the other systems they are accustomed to.

The PI would use support from this grant to hire graduate students to create new SAGE interfaces to PHCpack, Bertini, 4ti2, REDUCE, CoCoA, and many other significant pieces of mathematical software (both free and commercial). Though each interface is initially fairly straightforward to write, it is crucial that the author have intimate knowledge of the functionality and mathematics behind the program being interfaced with, and students are often experts in them.

2.2.4 Support for Specialized Programs

An exciting aspect of SAGE is how it makes it easy to make use of small special purpose programs for mathematical research as part of a larger project. For instance, in low-dimensional topology, algebraic combinatorics, and number theory (see [16]), there are numerous specialized programs which compute one or two things. Often, research projects involve using several of these in conjunction with larger software packages.

> "A recent project used:
>
> · Snap, a version of the 3-manifold SnapPea library with PARI,
> · KhoHo, for computing Khovanov homology of links in S^3,
> · PHCpack, the polynomial equation solver, and
> · Mathematica.
>
> In the past, I've always done this with Python scripts, but there was always a lot of overhead in translating back and forth between the various programs, and this was limiting in a lot of ways because *Python understands so little about mathematics*. (Using e.g. Magma or Mathematica as the "core" program never seemed to work very well because the file input/output functionality is poor, as well as the weakness of custom data-structures.) *SAGE makes this so much easier.*"
>
> — Nathan Dunfield, Caltech

The PI would use funds from this proposal to invite authors or major users of some of these specialized programs as month-long visitors, during which time they would turn the programs into integrated components of SAGE.

2.3 Independent Verification of Research Results

> "One should always have at least two proofs of any result."
>
> – J-P. Serre (personal communication)

Many algorithms in number theory and cryptography, e.g., 3-descent on elliptic curves, modular forms computation, quaternion algebras arithmetic, and fast arithmetic on Jacobians of curves are implemented only in Magma. Having an independent implementation of these algorithms will help increase our confidence in computational results. SAGE addresses Serre's remark both by providing new implementations of algorithms currently available in only one place, and by making it easier to run a computation in more than one system and compare the results.

2.4 Free Open Source Software

> "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 [20]

Though the *development of SAGE is very expensive*, SAGE itself is freely available, so researchers that have limited software budgets are guaranteed access to SAGE. The source of SAGE and all dependencies are "laid open for checking"; any part of SAGE may be inspected, modified, and redistributed.

Some non-free programs are partly implemented in an interpreted language, and include this interpreted source code. Unfortunately, due to their commercial nature, there are restrictions when viewing such source files. It may be illegal to read the implementation of a function and use what is learned in ones research:

> "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 code.*"
>
> — The Maple end user license agreement

2.5 The Development Community

SAGE development is driven by a strong spirit of altruism, openness, community, cooperation, and collaboration. SAGE is a large project with over 30 developers: five are undergraduate student employees, one is a full-time employee, many graduate students work on the project (three supported by the PI's startup money), and there are numerous volunteers in the USA, Europe, and Australia. There were 571 posts on the sage-devel mailing list during October 2006.

SAGE developers include Martin Albrecht (Bremen, Germany, grad student), Tom Boothby (UW undergrad), Robert Bradshaw (UW grad student), Iftikhar Burhanuddin (USC grad student), Craig Citro (UCLA grad student), Alex Clemesha (UW employee), John Cremona (Nottingham), Didier Deshommes (North Carolina student), Jon Hanke (Duke University), William Hart (Warwick, UK), David Harvey (Harvard grad student), David Joyner (US Naval Academy), Josh Kantor (UW grad student), Emily Kirkman (UW undergrad), David Kohel (Univ. of Sydney), Kevin McGown (UC San Diego), Bobby Moretti (UW undergrad), Yi Qiang (UW undergrad), Naqi Jaffery (UCSD undergrad), Kiran Kedlaya (MIT prof), Bill Page (top Axiom developer), David Roe (Harvard grad student), Steven Sivek (Princeton grad student), Jaap Spies (The Netherlands), Gonzalo Tornaria (Uruguay), Justin Walker (retired Apple OS X developer), Mark Watkins (Bristol), Joe Wetherell (San Diego), and Gary Zablackis (high school teacher).

The wide range of backgrounds of the contributors helps ensure that SAGE is relevant to a broad constituency of students and researchers.

2.5.1 Advisory Board

These people have agreed to serve on an advisory board, which will provide feedback before the PI allocates any funding from this grant:

- Tom Boothby (undergraduate, University of Washington)
- David Harvey (graduate student, Harvard University)
- Kiran Kedlaya (assistant professor, MIT)
- David Joyner (professor, US Naval Academy)

All have contributed substantial code to SAGE, and their range of backgrounds and positions ensure that a wide range of SAGE users are represented.

2.5.2 Workshops

The PI organized *SAGE Days 1* [10], which was a workshop at UC San Diego in February 2006, which had about 40 participants and was funded by NSF grant DMS-0555776. There were talks on a wide range of computer algebra systems and software, ranging over pure and applied mathematics, along with coding sprints.

In June 2006, the PI ran a workshop [26] at UW for high school students, in which they used SAGE to investigate the Birch and Swinnerton-Dyer conjecture.

In August 2006, the PI ran a two-week workshop at MSRI [30] in which 30 graduate students, and 6 research mathematicians worked together to design and implement a wide range of algorithms in SAGE for computing with modular curves, modular forms, elliptic curves and related objects.

In October 2006, the PI ran *SAGE Days 2* [11] at UW, which had about 40 participants, and was funded by NSF grant DMS-0555776. The main topic of the workshop was improving the speed and robustness of SAGE. There were 2 days of talks and 3 days of coding sprints, which initiated numerous projects.

The PI has secured funding for four upcoming SAGE-related workshops: the parallel computation workshop [1] at MSRI in January 2007, Sage Days 3 [31] at IPAM in February 2007, a Banff workshop on computing modular forms in June 2007 [2], and an AIM workshop on databases of modular forms in July 2007 [12].

These workshops have been pivotal in the development of SAGE, so securing funding for them is crucial. The PI hopes to run four SAGE Days workshops per year, and is requesting funding from the NSF for one of these workshops each year.

3 Specific Goals

In this section we outline some specific projects that this proposal would fund. These include designing and coding fast core algorithms for exact arithmetic, creating and implementing a wide range of algorithms for number theory and cryptography research, and improving SAGE's visualization capabilities.

3.1 Fast Core Arithmetic Algorithms

3.1.1 Support for Optimized Compiled Code

To implement a state-of-the-art mathematics software system, it is crucial to have an easy-to-use language for writing optimized compiled code, which integrates well with the high-level interpreter. SAGE accomplishes this using SageX, which is a customized version of Pyrex [5]. Using SageX, one can write compiled programs, which are easy to use from SAGE. Programs written in SageX are converted to C, compiled, and loaded at runtime. The SageX language is similar to Python, and provides easy access to both Python objects and arbitrary C and C++ functions. The PI has done much to make SageX easier to use in the context of SAGE, and intends to write more documentation and implement new features. This proposal would fund work by one computer science graduate student who is an expert in compilers for one quarter to greatly improve the quality of SageX.

SageX supports compilation during runtime, which can provide huge speed gains to interactive users. For example, Tom Boothby implemented a program so that when a polynomial will be evaluated repeatedly, it can be converted to an optimized SageX representation that is compiled and made available to the user, all from the interpreter—this can speed up evaluation by a factor of over 1000.

3.1.2 A New Fast C Library for Number Theory

Much of the arithmetic in SAGE for number theory is currently based on the NTL and PARI libraries. However, neither library is fast enough to satisfy the PI's goals for SAGE. To address this problem, two SAGE developers, William Hart and David Harvey, are creating FLINT, which is a library written in C for number theory applications. The PI is requesting funds to support work on FLINT, and for William Hart to visit UW for at least one quarter to work full time on FLINT.

Currently available libraries for arithmetic are not fast enough for SAGE. In some cases the algorithms are not asymptotically fast or their implementation is not sufficiently optimized. PARI's integer factorization is only slightly better than Magma's on some platforms and well behind the state of the art on all platforms. It also does not have the latest algorithms in many cases. For example it does not have an asymptotically fast Schoenhage-Strassen implementation for multiplying polynomials. NTL is a well written package, having set numerous world records. However its implementation strategy makes it difficult to get maximum performance from the GMP library, and in some cases Magma contains much faster implementations of core algorithms (see Table 3).

Neither PARI nor NTL support fine-grained parallel computation, which is a severe limitation as multicore multi-processor machines are becoming common.

FLINT includes routines for integer factorization. See Table 2 for how long it takes a **1.8GHz Opteron** to factor the 61-digit number pq, where p is the next prime after 10^{29} and q the next prime after 10^{31}. (These programs implement roughly the same algorithms; but some implementations are clearly much more optimized than others.) FLINT also contains an optimized Schoenhage-Strassen polynomial multiplication implementation, whose speed is illustrated in Table 3.

Table 2: Time to factor a 61-digit number with two large factors

Software	**Time (seconds)**
SAGE (FLINT)	16s
PARI 2.3.1	54s
MAGMA 2.13	63s
Mathematica 5.2 (FactorInteger)	299s
Maple 10 (ifactor)	715s

3.1.3 Groebner Bases

Groebner bases are central in commutative algebra, just as echelon forms are central in exact linear algebra. SAGE currently computes Groebner bases using

Table 3: 10000 multiplies of two degree 255 polynomials with 1000 bit coefficients

Software	**Time (seconds)**
FLINT	23s
MAGMA 2.13	28s
NTL-5.4	136s
PARI 2.3.1	295s
SINGULAR 3-0-2	839s
Maple 10	2100s
Mathematica 5.2	2075s
GAP 4.4.8	2846s

either Singular or (optionally) Macaulay2 or Magma (if available). Magma is substantially faster at computing Groebner bases in many cases than anything else available (in particular, Singular, Macaulay2 and CoCoA). One of the SAGE developers, Martin Albrecht, has implemented Faugere's F4 algorithm for SAGE in some cases, and achieved impressive results. The PI is requesting student support for Martin Albrecht to refine his F4 implementation into what might become the first *complete free* optimized implementation of F4, since the other two complete implementations, Faugere's and Magma's, are closed source and non-free.

3.1.4 Asymptotically Fast Exact Linear Algebra

Sparse and dense exact linear algebra is the bottleneck in many algorithms in graph theory, group theory, combinatorics, number theory, and other areas. Unfortunately, no existing free libraries meet the quality, speed and functionality requirements for SAGE (none approach the speed or functionality of Magma overall), so we propose to put substantial effort into optimized asymptotically fast sparse and dense exact linear algebra. The PI, David Harvey and Robert Bradshaw have designed and implemented asymptotically fast block echelon and matrix multiplication algorithms, but much optimization work remains to be done. The PI is requesting funding for graduate students to further optimize the implementation, add sparse and dense support for many rings and fields, and design and implement new multimodular linear algebra algorithms over cyclotomic fields.

3.1.5 Numerical Computation

Support in SAGE for numerical computation is crucial. Computation of special functions, Fourier transforms, and numerical linear algebra all play a role in number theoretic computations. Efficient algorithms for computing exact algebraic objects, e.g., Bernoulli numbers and models for curves, often proceed by computing numerical approximations, and algebraic problems can sometimes be solved (with proof) more efficiently via numerical methods. Support for numerical computation also expands the range of people who can benefit from SAGE.

SAGE includes numpy [21], which is a mature and well supported Python library for numerical matrix computations. SAGE also includes the GNU Scientific Library [8], which is a C library that provides a wide range of numerical algorithms. The PI is requesting funding to hire a graduate student to design and implement substantial new algorithms and code for SAGE to make it easier to use numpy, scipy [25], and GSL to support algebraic computation.

3.2 Advanced Number Theory Algorithms

3.2.1 Modular Forms and Modular Abelian Varieties

The PI has done much work on modular forms computations in his thesis and for Magma and published a book on computing with modular forms [29]. The PI proposes to create an optimized, flexible package in SAGE for computing with modular forms for general congruence subgroups of $\mathrm{SL}_2(\mathbf{Z})$. He would also like to create a package for computing with half-integral weight and weight 1 modular forms that uses a combination of Gabor Weise's new algorithm and Tate's classical algorithm. Jointly with Lassina Dembele and David Kohel, the PI intends to implement algorithms of John Voight and others for computing with quaternion algebras that will be used to compute both classical and Hilbert modular forms. With Robert Pollack, the PI intends to implement algorithms for computing with p-adic modular forms and p-adic L-functions, especially algorithms building on Glenn Steven's work on p-adic modular symbols. The PI would also like to implement a range of trace formula algorithms for investigating p-adic modular forms. The PI would use funds from this proposal to invite John Voight, Lassine Dembele, David Kohel, and others to UW to help with implementations.

The PI has done work on explicit computations with modular abelian varieties, including creating a package for computing with them that is included with Magma. Working with Jordi Quer (of Barcelona), he intends to implement in SAGE a package for computing with modular abelian varieties over $\mathbf{Q}$ and over number fields. This involves decomposition into simple factors, computation of endomorphism rings, isomorphism testing, computation of period lattices and defining equations (Jacobians, low genus), and special values of L-functions.

3.2.2 Elliptic and Hyperelliptic Curves

The arithmetic of elliptic curves is a rich central area of number theory. SAGE currently includes a wide range of functionality, and we hope to add more, and improve the quality of what is there now. SAGE includes Denis Simon's program for algebraic 2-descents, but does not take full advantage of its functionality yet. Cremona's mwrank program [3] is also included in SAGE, but some of its functionality has not been exposed to the SAGE interpreter.

There has been a major thrust to use 3, 4, and even 12 descents to compute Mordell-Weil groups of elliptic curves. All current implementations of these methods are part of Magma—there are no independent or free implementations, though the algorithms are published in recent Ph.D. theses and papers. The PI would use money from this proposal to fund SAGE implementations of these algorithms.

For cryptographic applications, SAGE requires optimized implementations of algorithms for computing with Jacobians of hyperelliptic curves. David Kohel has implemented preliminary algorithms in this direction for SAGE, and the PI intends to optimize these so that they are useful for serious research.

3.2.3 Databases

Though SAGE already has many databases, including Sloane's tables [28] of integer sequences and Cremona's tables [4] of elliptic curves, the PI intends to greatly expand the range of databases available in SAGE. This will mainly involve creating new databases related to the PI's research, including databases of modular forms, elliptic curves, modular abelian varieties, and special values of L-functions.

3.3 The SAGE Notebook: Graphics and Visualization

The SAGE notebook is implemented as a Python-based web server that interacts with numerous SAGE instances (one for each running worksheet). The user interface to the notebook is implemented using Javascript in a web browser.

There are three categories of graphics and visualization that must be addressed for SAGE: static graphics embedded in the notebook, dynamic graphics in the notebook, and dynamic local graphics that do not use the notebook.

Matplotlib [9], which is included with SAGE, provides 2d graphics with an interface very similar to MATLAB's. Alex Clemesha, a full time employee doing SAGE development, has used matplotlib to implement static 2d graphics with a Mathematica-like interface, which embed nicely in the notebook. For example, the PI used this package to illustrate Mazur's recent Nature article [19].

There is some support for 3d rendering in matplotlib, and Clemesha intends to create a similar Mathematica-like interface for 3d graphics. For high quality static rendering of 3d images, SAGE includes a multi-threaded 3d ray tracer [32], which produces beautiful 3d images with shadows, transparency, and reflections.

For dynamic graphics in the notebook, the PI intends to either find or hire a student to write a free open source Java applet that can display a scene involving mathematical primitives, and allows for dynamic rotation and scaling. For interactive local graphics not using the notebook, Mayavi [24] is an excellent solution.

4 Justification for Funding

The primary goal of SAGE is to make a wide range of modern research-level algorithms available in a high quality, easy-to-use, integrated package with a graphical interface. Tables 2–3 and the discussion of advanced research algorithms above illustrate that this has not yet been achieved by any free or commercial software. The commercial programs have several million dollar per year revenues, but do not address the research needs of our community; for SAGE to achieve similar quality while serving the needs of researchers requires significant funding, and until SAGE is much more established, grant funding is the primary source.

The PI hopes that in three years SAGE will be sufficiently pervasive that SAGE development can be funded by a foundation that receives donations and support contracts. Until then **NSF funding is critical**.

References

[1] I. Burhanuddin, J. Demmel, E. Goins, E. Kaltofen, F. Perez, W. Stein, H. Verrill, and J. Weening, *Workshop: Interactive Parallel Computation in Support of Research in Algebra, Geometry and Number Theory*, (Jan. 29–Feb. 2, 2007), `http://modular.math.washington.edu/msri07/`.

[2] J. Cremona, H. Darmon, K. Ribet, R. Sharifi, and W. Stein, *Banff workshop on Modular Forms*, (June 3–June 8, 2007).

[3] J. E. Cremona, `mwrank` *(computer software)*, `http://www.maths.nott.ac.uk/personal/jec/ftp/progs/`.

[4] ———, *Tables of Elliptic Curves,* `http://www.maths.nott.ac.uk/personal/jec/ftp/data/`.

[5] G. Ewing, *A Language for Writing Python Extension Modules*, `http://www.cosc.canterbury.ac.nz/greg.ewing/python/Pyrex/`.

[6] GAP, *Groups, algorithms, programming—a system for computational discrete algebra*, `http://www-gap.mcs.st-and.ac.uk/`.

[7] D. Grayson and M. Stillman, *Macaulay2: A software system devoted to supporting research in algebraic geometry and commutative algebra*, `http://www.math.uiuc.edu/Macaulay2/`.

[8] GSL, *GNU Scientific Library,* `http://www.gnu.org/software/gsl/`.

[9] John Hunter, *Matplotlib: Python software for drawing graphs*, `http://matplotlib.sourceforge.net/` (2005).

[10] D. Joyner and W. Stein, *Workshop: SAGE Days 1*, UC San Diego (February 2006), `http://sage.math.washington.edu/sage/days1/`.

[11] ———, *Workshop: SAGE Days 2*, University of Washington (October 2006), `http://sage.math.washington.edu/sage/days2/`.

[12] K. Kedlaya, M. Rubinstein, N. Ryan, N.P. Skoruppa, and W. Stein, *Work-*

shop: L-functions and modular forms, (July 30–Aug 3, 2007), `http://aimath.org/ARCC/workshops/lfunctionsandmf.html`.

[13] Magma, *High performance software for Algebra, Number Theory, and Geometry*, `http://magma.maths.usyd.edu.au/`.

[14] Maple, `http://www.maplesoft.com/`.

[15] Mathematica, `http://www.wolfram.com/`.

[16] K. Mathews, *Number theory ftp sites/calculator programs/archives*, `http://www.numbertheory.org/ntw/N1.html`.

[17] MATLAB, `http://www.mathworks.com/`.

[18] Maxima, *A GPL CAS based on DOE-MACSYMA*, `http://maxima.sourceforge.net/`.

[19] B. Mazur, *Controlling our errors*, Nature **443** (7 September 2006).

[20] J. Neubüser, *An Invitation to Computational Group Theory*, London Math. Soc. Lecture Notes Ser. **212** (1995), 457–475, Cambridge University Press.

[21] T. Oliphant, *The fundamental package for scientific computing with Python*, `http://numpy.scipy.org/`.

[22] PARI, *A computer algebra system designed for fast computations in number theory*, `http://pari.math.u-bordeaux.fr/`.

[23] F. Perez, *An Enhanced Python Shell*, `http://ipython.scipy.org/moin/`.

[24] P. Ramachandran, *Mayavi*, `http://mayavi.sourceforge.net/`.

[25] SciPy, *Scientific Tools for Python*, `http://www.scipy.org/`.

[26] SIMUW, *Workshop: Summer Institute of Mathematics at University of Washington (the PI led 1 of 6 high-school student workshops)*, (June 27—July 8, 2006), `http://sage.math.washington.edu/simuw/`.

[27] Singular, *A computer algebra system for polynomial computations*, `http://www.singular.uni-kl.de/`.

[28] N. J. Sloane, *The Online Encyclopedia of Integer Sequences*, `http://www.research.att.com/~njas/sequences/index.html`.

[29] W. Stein, *Explicitly Computing Modular Forms*, Graduate Studies in Mathematics, vol. 79, American Math. Society, 2007, With an appendix by Paul Gunnells.

[30] W Stein, *Workshop: Computing with modular forms*, (August, 2006), `http://modular.math.washington.edu/msri06`.

[31] W. Stein, *Workshop: SAGE Days 3*, IPAM (February 2007).

[32] J.E. Stone, *The Tachyon Multiprocessor Ray Tracer,* `http://jedi.ks.uiuc.edu/~johns/raytracer/`.

[33] Twisted, *A framework for networked applications*, `http://twistedmatrix.com/trac/`.