Sage: Unifying Mathematical Software for Scientists, Engineers, and Mathematicians

1 Introduction

The goal of this proposal is to further the development of Sage, which is comprehensive unified open source software for mathematical and scientific computing that builds on high-quality mainstream methodologies and tools. Sage [12] uses Python, one of the world's most popular general-purpose interpreted programming languages, to create a system that scales to modern interdisciplinary problems. Sage is also Internet friendly, featuring a web-based interface (see `http://www.sagenb.org`) that can be used from any computer with a web browser (PC's, Macs, iPhones, Android cell phones, etc.), and has sophisticated interfaces to nearly all other mathematics software, including the commercial programs Mathematica, Maple, MATLAB and Magma. Sage is not tied to any particular commercial mathematics software platform, is open source, and is completely free. With sufficient new work, Sage has the potential to have a transformative impact on the computational sciences, by helping to set a high standard for reproducible computational research and peer reviewed publication of code.

Our vision is that Sage will have a **broad impact** by supporting cutting edge research in a wide range of areas of computational mathematics, ranging from applied numerical computation to the most abstract realms of number theory. Already, Sage combines several hundred thousand lines of new code with over 5 million lines of code from other projects.

Thousands of researchers use Sage in their work to find new conjectures and results (see [13] for a list of over 75 publications that use Sage). Around 200 people from dozens of countries have contributed new code to the core library, and the Sage mailing lists have over 1,500 subscribers and see over 2,000 messages per month. We use Google Analytics to track visitors to the Sage website. During the period October 23 to November 22, 2009, there were 6,271 tracked downloads of Sage and 89,495 visits to the Sage website (session of several page views), with 45% of visitors from Europe, 42% from (North, Central and South) America, and 10% from Asia. This is nearly twice as much traffic as last year during the same time period.

This proposal has **intellectual merit** in that the further development of Sage may be critical to advancing knowledge and understanding in mathematics:

> "I'm amazed! It [Sage Days 18] was beautifully organized! I can't tell you how much fun it was, and also how much I learned in the few days I was there. [...] I think that Sage is just an extraordinary resource. Sage has allowed mathematicians to achieve a level of computation for arithmetic experimentation that would have only been dreamed of just a few years ago. Sage is simply a crucial tool for us mathematicians, akin to the most powerful electron microscope of the biologists."
>
> – Barry Mazur, University Professor of Mathematics at Harvard University, who has written many recent influential papers (e.g., [7, 8, 3, 9]) that rely on Sage.

Though the Sage project is becoming increasingly popular, vast amounts of work remain. Substantial new algorithms and major challenging computations are a constant source of motivation for the project. For example, project members recently computed all (conjectural) congruent numbers up to 1 trillion, a computation that is currently featured on the top of the list on the **Computational Mathematics – News** section of the NSF website [11]

The primary thrust of this proposal is to improve the usability and value of Sage for scientists, engineers, and mathematicians. We intend to produce a series of tutorials, worksheets, screencasts, and videos that make Sage much more accessible, especially to scientists and engineers. Also, we will work with the FEMhub team to create interfaces to a wide range of finite element PDE solvers. We intend to improve on Sage's current state of the art implementations of linear algebra over finite fields, the rational numbers, integers, etc. And, we intend to implement and improve methods for symbolic integration and symbolic solution to differential and difference equations.

Sage Days

Much of the development of Sage has been initiated at Sage Days gatherings, which are events where about 25 undergraduates, graduate students, postdocs, professors, and others come together for about 5 days and *passionately design and code* algorithms that improve Sage.

1.1 Past Support

NSF and the COMPMATH program have funded development of Sage since the PI started the project in early 2005. In particular, NSF funded a previous COMPMATH proposal by the PI in 2007 (DMS-0713225), and also funded the purchase of about $120K in hardware via the SCREMS program (DMS-0821725), on which the above mentioned congruent number computation was done. This NSF-funded hardware will play a central role in many of the activities in this proposal. The NSF FRG grant DMS-0757627 has also indirectly provided substantial support for the development of Sage in number theory. Sage Days 1 was funded by DMS-0555776.

2 Sage Days

The Sage Days gatherings are events where undergraduates, graduate students, postdocs, professors, and others come together for about 5 days and work to improve Sage. *Sage Days are not conferences.* We implement major new algorithms and functionality in areas ranging from algebraic topology to numerical analysis, improve existing algorithms and implementations, and create new ways of interacting with other mathematical software.

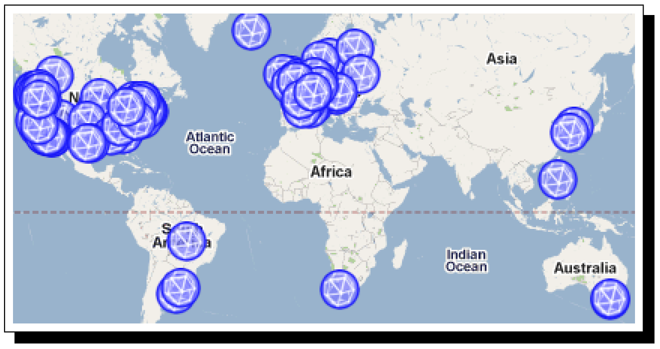
Sage workshops involve much, much more than just the implementation of existing algorithms. There are often several talks by a diverse group of speakers on topics ranging from education, e.g., teaching matrix algebra using Sage, to specialized areas of advanced research mathematics such as the Birch and Swinnerton-Dyer conjecture (Sage Days 18). New algorithms are designed, e.g., David Harvey's 2008 Harvard Ph.D. on computing Monsky-Washnitzer cohomology was initiated at a Sage workshop, and new ideas in computational mathematics are fleshed out, e.g., the Sage coercion model resulted from many intense discussions at Sage workshops. Connections are made between a myriad group of people, because Sage workshops typically combine multiple themes, e.g., there were discussions at Sage Days 14 about the wide range of applications of Hermite normal form in both number theory and algebraic topology; at Sage Days 11, which was held at the headquarters of Enthought—a company that supports numerical computation using Python—there were several deep discussions about the fast evaluation of symbolic expressions.

Students often do research projects using Sage. We have a very rigorous peer review process for including new code in the core Sage library (perhaps this peer review will one day raise the perceived value of contributions to mathematical software during promotion and hiring decisions). Sage Days gatherings provide an efficient environment for helping students learn how to include their work as part of Sage.

> "For summer research, I created code involving the Riemann mapping theorem and complex interpolation. Since finishing it, however I've been bogged down with the publishing requirements, testing, documentation, etc."
>
> – Ethan Van Andel, undergraduate student, Calvin College (Michigan); discussing research supported by NSF grant DMS-0702939.

The Sage Days working model has its roots in the NSF-funded Arizona Winter School workshops `http://math.arizona.edu/~swc/`, which the PI participated in as a graduate student and recently co-organized as a co-PI on DMS-0602287. Just like at the Arizona Winter School, participants work extremely hard on projects. We approach each Sage Days with an ambitious list of goals and project ideas, many of which are described in the rest of this proposal. At a particular gathering, some of the projects will acquire momentum while others will be saved for a later workshop. On the first day of the workshop, we list each project, and people volunteer to work on some of the projects. Usually specific clumps of people get attracted to a subset of the projects, and great progress is made on these projects. On each following day, we spend about an hour reporting on progress for each of the projects so far. These regular progress reports serve as excellent motivation for participants to make herculean efforts to solve problems and write code.

In summary, Sage Days have been wildly successful, helping to grow the Sage developer community so that it is now the world's largest and most diverse open source mathematical software development effort.



There are Sage developers on every continent:
`http://www.sagemath.org/development-map.html`

2.1 Past and Upcoming Sage Gatherings

The following is a list of past and upcoming gatherings that are co-organized by the PI and involve Sage heavily.

- *Sage Days 1,* Feb. 2006 at UC San Diego.
- *Summer Graduate Workshop on Computing with Modular Forms,* July 2006 at MSRI.
- *Sage Days 2,* Oct. 2006 at UW.
- *Interactive Parallel Computation in Support of Research in Algebra, Geometry and Number Theory,* Feb. 2007 at MSRI (Berkeley).

- *Sage Days 3,* Feb. 2007 at IPAM (UCLA).
- *Sage Days 4,* June 2007 at UW.
- *Sage Days 5 – Computational Arithmetic Geometry,* Oct. 2007, Clay Mathematics Institute.
- *Sage Days 6,* Nov. 2007, Bristol, UK.
- *Sage Days 7,* Feb. 2008, IPAM (UCLA).
- *Sage Days 8: Number Theory and High Performance Computation*, March 2008, at UT Austin.
- *Sage Days 8.5: Developer Coding Days*, June 2008, UW.
- *L-functions Summer School and Coding Sprint*, June 2008, UW.
- *Workshop on L-functions and Modular Forms*, June 2008, UW.
- *Sage Days 9: Mathematical graphics and visualization*, Aug. 2009, Vancouver.
- *Sage Days 11: Special functions and computational number theory meet scientific computing*, Nov. 2008, Austin, Texas.
- *Sage Days 12: Bug Smash*, Jan. 2009, San Diego, CA.
- *Sage Days 13: Quadratic Forms and Lattices*, March 2009, Athens, Georgia.
- *Sage Days 14: Sage and Macaulay2 for Algebraic Geometry Exper.*, March 2009, MSRI.
- *Arizona Winter School: Quadratic Forms*, March 2009.
- *Sage Days 15*, May 2009.
- *Sage Days 16: Computational Number Theory*, June 2009, in Barcelona, Spain.
- *Sage Days 17: Computing with Modular forms and L-functions*, Sep. 2009, on Lopez Island.
- *Sage Days 18: Computations related to the Birch and Swinnerton-Dyer Conjecture*, Dec. 2009, at the Clay Mathematics Institute in Cambridge, MA.
- *Sage Days 19: Second Sage Bug Smash*, January 2010 in Seattle.
- *Sage Days 20: Combinatorics*, Feb. 2010, Marseille, France.
- *Sage Days 20.5: Algebraic Combinatorics, Rep. Theory of Algebras*, May 2010, Toronto.
- *Sage Days 21: Function fields*, May 2010 at UW.
- *Sage Days 22: MSRI Graduate Student Workshop on Elliptic Curves*, June 2010 in MSRI.
- *Sage Days 23: Number theory*, July, 2010 in Leiden, Netherlands.
- *Sage Days 24: Symbolic computation*, July, 2010 at RISC in Linz, Austria.
- *Sage Days 25: Numerical computation*, August, 2010, in Mumbai, India.
- *MSRI Program in Arithmetic Statistics*, Spring 2011, at MSRI in Berkeley.

Many of the above workshops were funded by various NSF grants and the institutes hosting the workshops (e.g., Clay, Heilbronn, IPAM, MSRI, etc.,). The Institute for Defense Analysis also funded many workshops during 2008–2009. However, despite the extreme importance of Sage to their work, funding public workshops is far from their core mission, so they cannot fund Sage workshops further. *It is thus critical that NSF fund this proposal.*

3 Theme: Documentation for Scientists and Engineers

The current default Sage tutorial, reference manual, and other documentation tend to be aimed at people who think in terms of objects such as groups, rings, and vector spaces. However, Sage contains a plethora of powerful packages and capabilities that engineers and scientists find attractive and useful. Sage contains several numerical packages they might want, including R (statistics), NumPy (sophisticated matrix manipulation), SciPy and GSL (numerical libraries), ATLAS (fast matrix arithmetic), CVXOPT (convex optimization), and LAPACK. Sage has extensive 2D and 3D plotting packages. Sage also features extensive numerical integration and symbolic calculus capabilities, building on the GiNaC and venerable Maxima computer algebra libraries. Moreover, Sage has Python as its basis, and Cython (which is much more than just a Python to C compiler) which provides C speed for numerical calculations. Sage also includes image processing support via the included Python Imaging Library (PIL), and SciPy has support for signal processing. There is easy access to web-based information via Python's `urllib`, e.g., the financial package in Sage exploits this to obtain stock price data. NumPy and R also include extensive statistical capabilities, including random variable functions for most distributions used in engineering. The SciPy library has a range of numerical differential equations solvers. Finally, Sage is extremely LaTeX friendly, which helps with both displaying complicated mathematical expressions, and formatting them for inclusion in papers. Engineers also appreciate that Sage is free and open source, so they can easily make copies to run on all of their computer systems and change any of the internals of Sage if necessary.

> "The open source computational tools developed in the Python language and available through Sage represent a major improvement even over the best commercial tools for the kind of research I did on the development of algorithms for fast application of integral operators in multiple dimensions. [...]
>
> – Fernando Perez, Neuroimaging, UC Berkeley

3.1 Specific Aims

The goal of this Sage Days gathering would be to produce a series of videos, worksheets, step-by-step tutorials, and overviews and topical guides aimed at being accessible to engineers and scientists. This would provide an introduction to the tools engineers use. Also, a significant coding sprint project for the workshop would be the addition of a MATLAB-like interface for 3D graphics to Sage. This functionality would build on Sage's current 3D graphics user interface, which is similar to Mathematica's.

The videos, worksheets, and tutorials mentioned above would be based around a sequence of talks at the meeting. The talks would include the following, all with an engineering focus. We would have an overview of Sage consisting of a quick run through many exciting "gee whiz" features. Then a talk about graphics and images in Sage, including 2D plotting, 3D plotting, implicit plotting, and the MATLAB-like interface for 2D graphics. There would be a talk on symbolic calculus followed by one on linear algebra, which would show both symbolic and numerical examples, including computation of eigenvectors, eigenvalues, matrix inversion, etc., for both sparse and dense matrices. Next, there would be a talk about numerical analysis in Sage using NumPy and SciPy to find numerical solutions to differential equations. There would also be a talk on using Cython for high-speed numerical computation, which would include examples of the new fast Cython/NumPy interface. This would lead into a final numerical talk on the various tools for signal processing in Sage. Other talks would discuss how to use the Sage notebook to easily dynamically create interactive sliders and buttons, how to setup network servers and clients, and how to best use, parse, and produce LaTeX and HTML using Sage.

Another activity near the end of the meeting would be to test all the tutorials and other materials that we create. After workshop-level testing, we would then test the materials more widely by advertising them on the Sage mailing list (which has about 1,500 subscribers), and the NumPy and SciPy mailing lists. We would also work with instructors and their graduate students in applied mathematics, the sciences, and engineering at our universities to further test these materials (e.g., with Randy Leveque of Applied Mathematics).

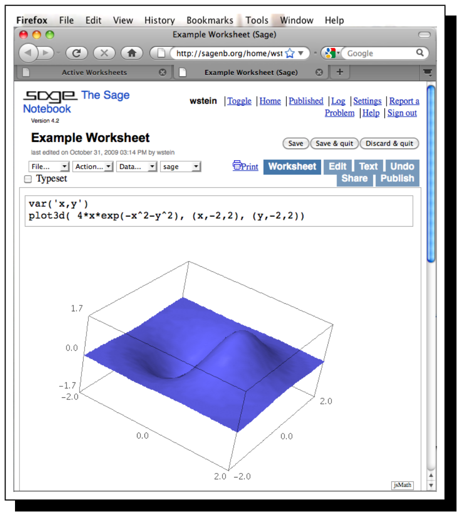
Before the workshop, we will prepare a list of reading materials about good technical writing and also examples (from related projects) of successful tutorials. Participants will be asked to become familiar with these materials.

3.2 Organization Plan

The organization committee would likely include Jarrod Millman and Fernando Perez, both in Neuroimaging at Berkeley, Josh Kantor at MIT Lincoln Laboratory, who wrote the current numerical tutorial in Sage, and Randy Leveque of University of Washington, who teaches a graduate seminar on Python for scientific computing. We would also invite Michael Madison, all of the Enthought employees (since many of the materials we produce would be of use to their customers), Prabhu Ramakrishnan (author of MayaVi), and Stefan van der Walt who is currently leading the charge in organizing the production of quality reference documentation for the NumPy and SciPy projects. The workshop would likely take place in Seattle, and we would also invite people we know in local industries, including Microsoft Research, Boeing, and Google, in addition to advertising on the NumPy, SciPy, and Sage mailing lists. We would also invite younger people, such as Minh Van Nguyen (an undergraduate), who have demonstrated incredible technical skills as *editors* when working on the existing Sage documentation.

4 Theme: The Web-Based Sage Notebook Interface

Imagine an interactive web-based worksheet in which one can enter arbitrary Sage commands, see beautifully typeset output, create 2D and 3D graphics, publish worksheets, and collaborate with other users. The Sage notebook is so far the only math software that has ever offered this capability; it is somewhere between a Maple or Mathematica notebook and the online Google Docs web application.


Firefox File Edit View History Bookmarks Tools Window Help
Example Worksheet (Sage)
http://sagenb.org/home/wst
Active Worksheets
Example Worksheet (Sage)
The Sage Notebook
Version 4.2
wstein | Toggle | Home | Published | Log | Settings | Report a Problem | Help | Sign out
Example Worksheet
last edited on October 31, 2009 03:14 PM by wstein
Save | Save & quit | Discard & quit
File... | Action... | Data... | sage
Typeset
Print | Worksheet | Edit | Text | Undo | Share | Publish
1.7
0.0
−1.7
−2.0
0.0
2.0
0.0
2.0 −2.0
jsMath

```
var('x,y')
plot3d( 4*x*exp(-x^2-y^2), (x,-2,2), (y,-2,2))
```

Many mathematicians in both pure and applied mathematics at dozens of universities around the world are excited about how they can leverage the Sage notebook in their own research and teaching. There is much excitement about this new collaboration tool, which enhances their teaching and research. For example, the FEMhub project `http://www.femhub.org/` at the University of Nevada has adopted and rebranded the Sage notebook for teaching courses and doing research on finite element methods. See `http://wiki.sagemath.org/sagenb` for a list of deployed notebook servers.

> "I am currently working with two students on developing a Sage interface to our Clawpack software for solving conservation laws and hyperbolic PDEs. The Sage notebook provides a powerful user interface, especially for problems that require working with a number of different packages."
>
> – Randy Leveque, applied mathematician at Univ. of Washington

4.1 Specific Aims

We intend to add a first version of a web-based spreadsheet capabilities to the Sage notebook. This would be similar to Microsoft Excel or Google Docs spreadsheets, but the formulas would allow for the full power of Sage. This could be extremely powerful for applications that are data-centric, yet involve sophisticated mathematics. There are about a dozen existing free JavaScript-based spreadsheet applications (all in various stages of development), so we would start by surveying what is available. We would then choose the best of these, and adapt it for inclusion in Sage, then fill in missing functionality.

Another project is to add software development capabilities to the Sage notebook. This would include integrating the code editor CodeMirror into the notebook, provide an interface to the Sage library's revision control system (Mercurial) and the capability of editing any files in the Sage library, compiling code, running and debugging code, checking in changes, creating and submitting patches, etc., all from a web browser. This would, in particular, include creating a web-based interactive debugger (similar to the one provided by the Pylons Python web framework), which would provide access to the Python debugger, the Python profiler, and the GNU debugger (gdb).

We could also implement additional 3D plotting functionality. There are many functions and options for 3D plots that have not yet been implemented, and this could be done. Also, we could improve Sage's ability to render plots using HTML5's new canvas element, including rendering animated 3D output, embedded in a standalone webpage or viewed separately. This would involve improving the options for implicit, parametric and function plotting, and for adding text rendering and axes to HTML5 canvas rendering. There has already been substantial work done on all of the above during Spring and Summer 2009 by UW freshman William Cauchois, supported by UW's NSF VIGRE grant (which has since expired).

Organizational Plan: The organizers would likely include William Stein and John Palmieri (the PI and co-PI, both at UW). Alyson Deines and Tom Boothby (UW graduate students heavily involved in the notebook) and Jason Grout (faculty at Drake University who has also done substantial notebook work) would also likely play a major role in the workshop. We would also invite Tim Dumol (high school student in the Philippines), Mitesh Patel (consultant in California), and Mike Hansen (consultant)—all frequent contributors to the notebook codebase.

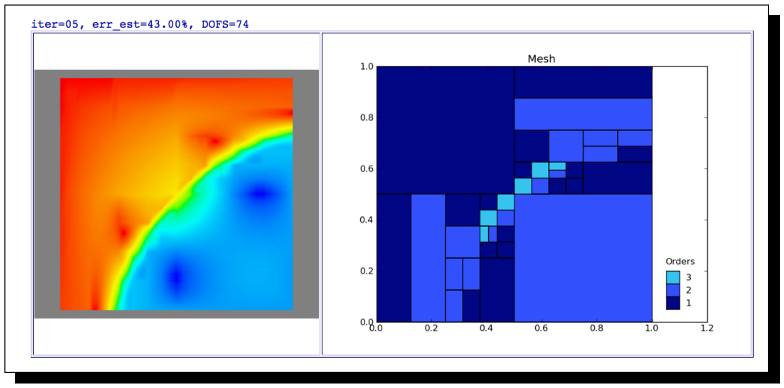
In addition, we would invite Alex Clemesha, Dorian Raymer, and other Codenode `http://www.codenode.org/` developers. We would advertise the workshop on the Sage mailing lists, the Codenode mailing list, and on the general Python mailing lists, since the notebook is of general interest to anybody who does interactive work with Python. We would invite Sage developers Robert Bradshaw and Craig Citro, who will both start working at Google Seattle starting in June 2010, and try to get some people from Google to attend, since several Google technologies overlap with the Sage notebook, and Google has funded substantial work on the Sage project.

5 Theme: Numerical Solution to Partial Differential Equations

In this section we will focus on explaining a Sage workshop involving a specific finite element method project called FEMhub. There is an impressive array of free software for using the finite element method (FEM) to solve certain partial differential equations. Unfortunately, getting this software to interoperate is extremely difficult due to different installation requirements, different input and output formats, and capabilities.

Imagine pulling out your cell phone and doing sophisticated 3D interactive computations using powerful remote FEM codes, then later continuing those same computations on your desktop via another web browser, and finally sharing these results with students in your undergraduate course. Those students will also be able to do everything you just did with any web browser.

The goal of the FEMhub project `http://www.femhub.org/` (partially funded by NIST) is to bring transparency and establish basic standards in the development of open source FEM software. All software included in FEMhub is easy to use together via a unified Python interface and available to anyone for free either as a desktop application or via an interactive Sage notebook. FEMhub itself is built using the Sage package management system along with a subset of the packages in Sage combined with new packages specifically relevant to FEM.



Using FEMhub to Analyze Convergence for the Inner Layer Problem

At this gathering, we would bring together people interested in the finite element method and related techniques, including symbolic derivation. We would make sure the mesh, solution, and other components of FEMhub, etc., work well both on the desktop and in the notebook, and interface a wide range of open source FEM software, including libmesh and SfePy, as well as graphics output programs such as MayaVi and VTK. We would also discuss general issues of how the Sage and FEMhub projects can best work together to create quality software, and discuss what the Sage project can do to better support other groups that want to use the Sage build system as a template for their own Python-based scientific software distributions.

5.1 Specific Aims

We will do work on a system to make it easy to compute weak formulations automatically using symbolic capabilities included in Sage. For more complicated PDE systems or for PDEs in other than Cartesian coordinates, this would take a huge burden off the user. We would also design a suitable protocol for passing the weak formulations to all other code in FEMhub.

We would define geometries interactively via Java applets (or possibly JavaScript or HTML5 canvas or SVG format), and send them to mesh generators. We would then create tools to make it easy for users to visualize and edit the resulting meshes in the notebook.

We can already plot the solutions and meshes from Python using either MayaVi or Matplotlib, but this needs improvements, e.g., to be able to plot stream lines, change plotting parameters easily in the notebook, etc. We have elaborate visualization capabilities in OpenGL and glut in C++, but unfortunately this is not usable in the notebook. Making these capabilities available in the notebook will increase the number of people who can use FEMhub.

We would also generally improve support in Sage for numerically solving differential equations using SciPy and visualizing the solutions. This would mainly involve creating wrapper code and documentation with many examples so that the solver capabilities provided by SciPy work efficiently with native Sage objects (e.g., symbolic expressions). It would also involve writing code so that solutions (and families of solutions) can be efficiently plotted in 2 and 3 dimensions.

Organizational Plan: Organizers would likely include Pavel Solin (Univ. of Nevada) who directs the FEMhub project, Ondrej Certik (Univ. of Nevada), and William Stein. We would advertise the workshop widely on the SciPy, NumPy, and Sage mailing lists, and in both the UW pure and applied mathematics departments (which have a strong interest in PDEs). We would also use contacts we have at Boeing to try to get some involvement by local industry.

6 Theme: Cython and Numerical Computation

Python is a popular choice for scientific computation and visualization. Python is a general purpose scripting language designed without a limited target audience in mind, so it tends to scale well as simple experiments grow to complex applications. From a numerical perspective, Python and associated libraries can be regarded mainly as a convenient shell around computational cores written in natively compiled languages, such as C, C++, and Fortran. For instance, the Python-specific NumPy library contains over 100,000 lines of C and about 80,000 lines of Python; while the SciPy library, which is built on top of NumPy, contains over 200,000 lines of C++, 60,000 lines of C, and 75,000 lines of Fortran, compared to about 70,000 lines of Python code.

Cython is a programming language based on Python, with additional syntax for optional static type declarations. The Cython compiler is able to translate Cython code into C code making use of the Python/C API, which can in turn be compiled into a module loadable into any Python session. Cython is thus a language which allows one to use Python and the power of C interchangeably in the same code. This has two important applications. First, it is useful for creating Python wrappers around natively compiled code, especially when one does not want a 1-to-1 mapping between the library API and the Python API, but rather a higher-level, more intuitive wrapper. Secondly, it allows incrementally (and dramatically) speeding up Python code. One can start out with a simple Python prototype, then proceed to incrementally add type information and C-level optimization strategies in the few locations that really matter. While being a superset of Python is a goal for Cython, there are currently a few incompatibilities and unsupported constructs. The most important of these is inner functions and generators (closures).

Fast array access, added to the Cython language by Dag Sverre Seljebotn and Robert Bradshaw in 2008, was an important improvement in convenience for numerical users. Cython is able to treat most of the NumPy array data types as corresponding native C types.

Cython is used heavily in Sage, accounting for several hundred thousand lines of Sage's code. Moreover, Robert Bradshaw, the director of the Cython project, is also a Sage developer and the PI's Ph.D. student.

6.1 Specific Aims

The main goal of the workshop is to be able to rewrite parts of NumPy using Cython. A direct benefit will be eliminating manual reference counting in as many places as possible. There is a new version of Python, Python 3.x, which is not backwards compatible with the widely used Python

2.x. Code written in Cython works with both versions of Python. Thus by switching to Cython, NumPy will get Python 3.x compatibility almost for free; this is far more than just a powerful "advertising" argument, since almost the entire numerical Python stack of software depends on NumPy—until NumPy supports Python 3.x, nothing else in this numerical suite will.

Cython was not around when NumPy was first created. Currently few people feel comfortable working with the C-level core of NumPy, which is a huge amount of C code written directly against the Python/C API. Our hope is that having the core in Cython may make NumPy accessible enough so that new developers will dive in—ultimately, that will be the greatest victory.

Rewriting NumPy in Cython would likely require that we make a number of improvements to Cython. At a Sage Days workshop on Cython and NumPy, we would thus look at features needed in Cython to make it easy to port parts of NumPy, such as templates and generic programming, the creation of "compiled ufuncs" (user functions) from Cython, making it easier to mix C and Cython code in the same Python extension, and speeding up Cython compilation. We would also continue to improve Cython's support for Fortran and C++. Better C++ support in Cython would make the whole C++ STL available directly in Cython, and which would help with the use of Cython in SciPy. Also, C++ support would help wrapping SciPy libraries partially written in C++. Regarding Fortran, the current de facto standard Fortran wrapper, f2py, does not support features in Fortran 90 or later, makes certain assumptions which are not compatible with standards (so does not necessarily work cross-platform or with future compilers), and provides no good way of calling Fortran code from C or Cython using the same wrapper. Current development work on Cython's Fortran support ("fwrap") is in a good position to solve all these problems. If we can rely on Cython for the Fortran support required by SciPy, we can remove f2py from NumPy. f2py is no longer actively maintained as the original author has started a new project.

At a workshop we would also start actually porting NumPy to Cython, and think about extending NumPy arrays to efficiently support arbitrary precision exact data types (integers, rationals, high-precision floating point, etc.) of interest to mathematicians, scientists, and engineers. In general, we would thus take advantage of the mix of people present to take further action toward making Sage better at numerical computation.

Organizational Plan: The workshop organization committee would likely include Jarrod Millman, Robert Bradshaw, and Travis Oliphant. We would also love to have Stefan Behnel, Dag Sverre Seljebotn, Kurt Smith, Danilo Freitas, Stefan van der Walt, David Cournapeau, and Chuck Harris participate. We would advertise the workshop on the SciPy, NumPy, Cython, and Sage mailing lists, and contact people in industry that have an interest in both NumPy and Cython (e.g., at the last Cython/Sage workshop, we had an employee of a hedge fund talk about their extensive use of NumPy and Cython). We would also encourage as many interested graduate students and young developers as possible to attend.

7 Theme: Linear Algebra

Many areas of Sage rely on high-performance routines for solving equations, computing canonical bases, inverting matrices, and computing eigenvalues. There are efficient routines for many of these computations, over a variety of rings and fields, and because so much of mathematics depends on linear algebra, improvements in these routines will strengthen much in Sage. We have implemented or wrapped many specialized algorithms. In these cases, Sage is often the fastest in the world. Unfortunately, numerous algorithms have not yet received proper attention.

7.1 Specific Aims

Many linear algebra routines have generic implementations, which could be made much faster in some situations if they exploit knowledge or specialized libraries for specific coefficient fields. We would also add and improve the matrix decomposition algorithms in Sage, including LU, Cholesky, and Jordan decomposition. Some of these could be improved to work more efficiently and accurately over a wider range of base fields (e.g., higher precision, finite fields, etc.).

Sage's high precision floating point arithmetic is built on top of Paul Zimmermann's rigorously justified MPFR library (see `http://www.mpfr.org/`). Sage thus currently has the capability to do numerical linear algebra using real and complex arithmetic and with arbitrary precision real or complex floating point numbers. However, the implementations of the underlying algorithms for these fields is in many cases ill-suited for inexact rings. In consultation with experts in numerical analysis, we would implement some well-documented and easy-to-read numerically stable algorithms for linear system solving, singular value decomposition, LU, QR and Cholesky decomposition, and matrix inversion for generic dense matrices, sparse matrices, and matrices with extra structure. We would also consult with Fredrik Stromberg, who has created a similar library for his own applications to computation with Maass forms.

Organizational Plan: Organizers would likely include Jason Grout (Drake University), Martin Albrecht (Royal Holloway), Rob Beezer (University of Puget Sound), William Stein, and Robert Bradshaw (UW), all of whom have done substantial work on Sage's linear algebra. We would consult with Fredrik Stromberg (Darmstadt), Jim Demel (Berkeley), Clint Whaley (Texas), Erich Kaltofen (NCSU), David Saunders (Deleware), Clement Pernet (Grenoble), and other experts.

8 Theme: Symbolic Computation

Symbolic computation is the study of exact methods to solve mathematical problems on the computer, for example algorithms to solve algebraic, differential or difference equations. These methods have wide ranging applications from experiments in pure mathematics to practical uses in the engineering design and analysis process.

Focusing on symbolic integration and summation problems, we divide the relevant topics naturally into several areas. The first main area is differential (and difference) algebra for addressing indefinite integration (and summation) and differential (and difference) solvers. The second area is transforms and definite integration, mostly done with table lookup and pattern matching approaches that are used to evaluate and transform integrals and special functions. The final area is fundamental algorithms in computer algebra required to implement the algorithms above efficiently; these include multivariate polynomial factorization, sparse multivariate polynomial interpolation, and exact linear algebra over multivariate polynomial rings.

8.1 Workshops on Differential Algebra and Integral Transforms

Burcin Erocal, a graduate student at the Research Institute for Symbolic Computation (RISC) in Austria, has attended numerous Sage workshops. He is responsible for much of the symbolic functionality in Sage. As part of his thesis work, he has implemented special classes of difference fields, called Pi-Sigma fields (by Michael Karr), which are required for other work he is doing on evaluation of formal infinite sums. Erocal is planning to implement generic difference/differential rings/fields and operators in the near future.

A theoretical framework for investigating solutions of differential/difference equations is provided by differential/difference algebra. At the moment, Sage does not have any facilities for working with differential/difference fields/rings/ideals, etc. An implementation of the basic structures required here would pave the way to a proper implementation of the Risch and Karr algorithms for indefinite integration and summation respectively, as well as providing a basis for research in Galois theory of differential and difference equations, which forms the foundation for algorithms to deal with higher order equations.

Our first goal would be to provide support for working with differential/difference rings/fields and their operator domains. Singular (one of the components of Sage) does this for polynomial rings. In order to handle denominators, we hope to use GiNaC, passing the harder problems, such as computing normal forms with respect to an ideal, to Singular after clearing denominators.

Another goal is to implement the Risch-Norman (parallel integration) algorithm (see [4]). Finally, we would attempt to implement recent algorithms by Abramov, van Hoeij, and Hendriks-Singer to find Liouvillian solutions of difference/differential equations.

Important cases of definite integration in computer algebra systems are handled through applying some integral transforms (such as the Mellin transform) to convert the integral to a contour integral which can also be expressed as a well-known special function (hypergeometric or Meijer-G), then use properties of this special function to arrive at an evaluation. While these methods have been implemented, and used in practice by closed source systems such as Mathematica and Maple, literature on how to correctly apply these methods to get practical results is still scarce. Sage has an efficient framework based on the symbolic computation library GiNaC to represent these functions, and perform pattern matching operations. This provides the basic building block to implement the required transforms, both integral and hypergeometric.

Some goals for a workshop would include implementing a class for hypergeometric functions, code for recognition of hypergeometric functions, and code for conversion to a standard representation (${}_pF_q$). We would also build lookup tables for well-known transforms, including hypergeometric and integral transforms, but also Mellin, Hilbert, Laplace, and Fourier transforms.

Organizational Plan: Organizers would likely include Burcin Erocal and William Stein, and involve strong consultation with Michael Singer, Mark van Hoeij, Sergei Abramov, Peter Paule, Victor Moll, and Victor Adamchik.

9 Theme: Special Functions

Special functions are an essential part of the toolbox of the applied mathematician, scientist or engineer. Since their use in the solutions of differential equations by Newton and Leibniz, the subject has been a center of attention for scientists from different fields.

A possible way of defining the set of special functions is to choose those mathematical functions which are widely used in scientific and technical applications, and of which many useful properties are known. A familiar classification of special functions is by increasing complexity, starting with polynomials and algebraic functions and progressing through the elementary or lower transcendental functions (logarithms, exponentials, trigonometric, etc.) to the higher transcendental functions (Bessel, parabolic cylinder, etc.). Compilations like the Handbook of Mathematical Functions [2] or its modern electronic version, the Digital Library of Mathematical Functions [1], assist the process of standardization much as a dictionary enshrines the words in common use at a given time.

With the advances in computer technology and availability of new algorithmic methods, possibilities for experimentation with special functions have expanded tremendously. Even though these advances have the potential to change the daily work of anyone who uses special functions for research or applications, making them available through a coherent interface still remains a problem. Established environments such as Mathematica, Maple or MATLAB have implemented some of these methods and use them in their *simplification* routines. Unfortunately, most of the time their use is opaque to the user, which leaves little possibility for verification of results or user-driven experimentation. In some cases, even numerical evaluation can be a problem [5].

Sage already includes implementations of many special functions through the different mathematical software packages it contains as well as novel implementations of some number theoretic functions within the Sage library. For example, the MPFR and mpmath libraries (included in Sage) provide arbitrary precision evaluation of various functions, and Pari has extensive support for number theoretic special functions ranging from the j-function to general motivic L-functions.

Algorithmic methods for special functions and orthogonal polynomials rely heavily on differential and difference algebra functionality, which will be provided by the symbolic computation workshop. Coupling advanced methods for solving difference and differential equations with powerful special functions manipulations will allow extensive experiments by researchers and provide an ideal platform for applications where simplifying certain expressions is the main goal.

9.1 Specific aims

The main goal of the workshop will be to provide the basic tools for research and experimentation with special functions and orthogonal polynomials in Sage. To this effect, we will implement constructs to manipulate symbolic expressions involving special functions, with the necessary framework to support arbitrary precision numeric evaluations, asymptotic expansions, efficient derivations and shifts (differences).

The symbolics module of Sage, backed by the GiNaC library, already provides much of the needed functionality. We will interface state of the art libraries included in the Sage distribution, such as MPFR and mpmath, for arbitrary precision evaluations, and implement the necessary code for functionality not available elsewhere.

We will also connect these constructs to the symbolic summation and integration tools developed within the symbolic computation workshop, i.e., algorithms working with hypergeometric or hyperexponential functions as well as elements of certain difference or differential fields. To our knowledge, this will give Sage the most sophisticated tools available to work with special functions in any computer algebra system, extending the results of a previous project to *implement* the Askey-Wilson scheme [6].

A concrete outcome of this effort will be to get Sage listed in the software page of the Digital Library of Mathematical Functions. We have already contacted the computer algebra editor of the DLMF, Peter Paule, to find out the requirements for this listing.

Besides research into new algorithmic advances to handle special function-based solutions of difference, differential and mixed equations, this workshop could provide a starting point for an interface between the DLMF and Sage. Previous efforts in this direction include [10], an online version of the database is available at `http://algo.inria.fr/esf/`.

Organizational Plan: Organizers would likely include Diego Dominici (SUNY) and Flavia Stan (RISC). We will work in consultation with Annie Cuyt (Universiteit Antwerpen), Walter Gautschi (Purdue University), Amparo Gil (Universidad de Valencia), Peter Paule (RISC), Veronika Pillwein (RISC), Javier Segura (Universidad de Cantabria), Nico M. Temme (Centrum voor Wiskunde en

Informatica) and Raimundas Vidunas (Kobe University), and Paul Zimmermann (INRIA Nancy).

10 Theme: Topology

Sage currently has fairly sophisticated support for topology, and we would like to push it further. Besides being central to much of pure mathematics, topology has many applications, e.g., in facial recognition. This section assumes the reader has a fairly sophisticated background in topology.

Sage can do Steenrod algebra calculations (at both $p = 2$ and odd primes), Sage can do simplicial homology, group cohomology (via the optional package by Simon King and David Green), and Sage has good support for free modules over commutative rings.

We intend to improve simplicial homology: we should be able to compute the ring structure for cohomology, and we should be able to keep track of the generators. We intend to add an interface to CHomP, which computes cubical homology. We would add some capabilities to compute with formal group laws (homotopy theorists mainly care about 1-dimensional formal group laws, but we do not need to limit ourselves), and the related computations with the spectrum BP. We also propose to implement differential graded algebras.

For graded connected algebras, we intend to implement Bob Bruner's algorithm, or develop an interface between his C programs and Sage. Christian Nassau has done some related work, but it is focused on the Steenrod algebra, not on general graded connected algebras. For group algebras, we will check to see if parts of the group cohomology package should be made standard, and possibly extended to other algebras.

In general, we intend to implement free modules, projective modules, and injective modules over arbitrary rings, and we intend to implement resolutions and derived functors. Where possible, we also intend to implement methods of computing resolutions (as in the case of graded connected algebra and group algebra). We also hope to implement some interesting noncommutative rings over which we can compute examples.

> "Studying 3-dimensional topology requires a broad range of computational tools in areas like hyperbolic geometry, combinatorial group theory, number theory, etc. Over the years, many specialized programs of varying sizes have been written to study various of these facets, but most problems require several, or even many, different points of view. Sage excels at stitching together such disparate pieces into a seamless whole, and so has been a fundamental tool for my recent work. Indeed, I think the shear breadth and flexibility of Sage effectively make it a new kind of mathematical software [...]. As a result, Sage has become my first-choice tool for all my computational work."
>
> – Nathan Dunfield, low-dimensional topologist at UIUC.

Organizational Plan: The organizers would likely include William Stein and John Palmieri. We would encourage Bob Bruner and Christian Nassau to attend (they attended Sage Days 15).

11 Time Line

We plan to have three workshops each year funded by this proposal. They would likely take place in mid-September (before the UW quarter starts), in January, in March around Spring Break, and in mid-June (right after the UW quarter ends).

If funded, during **year 1**, we would have the first *Cython and Numerical Computation* workshop, which would have the dual goal of building a solid foundation for later work and improving interactions between Sage, Cython, and NumPy developers. Next we would have a workshop on *Numerical Solution to Partial Differential Equations*, which would lay important foundations for future work and collaboration between the Sage and computational PDE communities. After these two workshops, we will be primed for a workshop on either *Documentation for Scientists and Engineers* or *Symbolic Computation*, which would lay the foundation for future work on special functions.

We would start **year 2** with a second workshop on *Cython and Numerical Computation*, which would polish and finalize projects started at the previous workshops. We would next have our first workshop on linear algebra, where we implement new algorithms, benefiting from our experience with Cython and NumPy from the previous workshops. After that, we would have a workshop on the Sage notebook, taking into account the lessons learned from the documentation workshop in year 1.

We would start **year 3** with a second workshop on documentation. Then we would have a workshop on topology, which would build on the linear algebra workshop from the previous year, and also suggest new linear algebra work. Finally, we would finish year 3 with a second workshop on special functions with a more numerical flavor, which would nicely tie together several previous workshops.

12 Evaluation Plan

Each year we will conduct an annual survey of Sage users and developers in which we ask about 50 questions to better understand our users, find out what we are doing right and wrong as a project, and how we can improve. We will add specific questions to address the applied topics of this proposal. We have done this survey in 2008 and 2009. In 2008 we received 186 valid answers, and 235 people filled out the 2009 survey.

We use Google Analytics on the main Sage website to measure the number of visitors to `http://www.sagemath.org` and have maps that show how many people visit from different geographic locations, etc. We also have active and friendly mailing lists, which provide immediate feedback and evaluation. We can also monitor how many people are using the public Sage notebook server `http://www.sagenb.org`, and what commands they are typing, and mistakes they are making.

After each workshop, we will have an anonymous questionnaire asking participants to share their ideas about how we could better structure the format and content of the workshop. A good example of the sort of feedback we get can be found at `http://wiki.sagemath.org/HowToHostASageDays`, which was written by an undergraduate and a graduate student at Sage Days 7 (at the Institute for Pure and Applied Mathematics). The PI has organized the anonymous feedback forms for the NSF-funded Arizona Winter School on several occasions, and will use similar technology.