Project Description – SBIR Phase I – Interactive homework problems for advanced computational mathematics

William A. Stein

June 14, 2015

Contents

1 Introduction

We propose to design and implement a new approach to interactive online homework problems called "SageCloud Interactive Problems" (SCIP) that will initially target students in courses involving advanced computational mathematics. Despite enormous computational challenges, we believe that we can make the system cost effective by leveraging open source software, cloud computing, and by relentlessly optimizing our implementation based on real world usage. Moreover, by building on recent work, SCIP will provide realtime feedback, detailed analytics, deeper computational problems than existing systems, and improved realtime interaction between students and instructors. In particular, we would leverage the NSF-funded SageMathCloud (SMC) platform and NSF-funded SageMath software. If successful, our project would greatly increase the ability of students to use open source tools for doing computational mathematics, for writing up and sharing their results, and make courses less work to teach and more effective.

2 Elevator Pitch

SCIP will be a new platform for interactive homework problems built on top of SageMathCloud (SMC).

The customers for SCIP are students and teachers of courses that involve any form of computational mathematics. For example, SMC is a 100% web-based system that was used during Spring 2015 by over 65 courses ranging from traditional Calculus courses to course in Global Health, Numerical Analysis, Elliptic Curves and Mathematical Finance (see https://github.com/sagemathinc/smc/wiki/Teaching). Though SMC does not currently have any interactive homework functionality, many of these courses would be greatly enhanced if they could use the functionality we propose here.

Researchers and their students need easy and affordable collaborative access to mathematical software, software development environments, and tools for writing and sharing their results. SCIP will help in teaching students to use these tools by providing both access to the tools and making it possible for instructors to create, share, and sell interactive problems that help students to learn these tools.

Instead of worrying about installing and updating software, learning complicated revision control systems, and figuring out how to use cloud or local computers, our target instructors want to focus on teaching their students how to compute. Some of these users, especially students and people in developing countries, are extremely price sensitive. Our proposed project would leverage open source software, cheap cloud computing resources, and systematic optimization based on user feedback to make the above affordable.

SMC currently makes math software easily accessible and provides some limited course management functionality, but offers nothing for managing the daunting complexity involved in curating interactive homework problems that involve sophisticated computation and mathematics. Some of the current state of the art systems feel antiquated, mainly addressing freshman and sophomore level material. We instead propose to target college senior (and even graduate level) level material, and support use of sophisticated mathematical software that is potentially computationally demanding.

A key innovation of SMC is that it is built upon a pool of open source software that leverages the combined knowledge and experience of thousands of developers and millions of work hours. We provide value by packaging all of this in a neatly integrated, easy-to-use and fully managed web-based application. We propose to take the same approach with SCIP, building on the platform that we have already developed.

SMC is an efficient and completely open source stack that handles a huge range of mathematical computations, including advanced pure mathematics, numerical analysis, and statistics. Users can write computer code and edit LaTeX documents, and there's extensive support for "interactive computational documents", including IPython notebooks and Sage worksheets. Such rich interactive computational documents are appealing to novice users. We have learned much from the feedback from 100K users and nearly 100 courses that have used SMC.

3 The Commercial Opportunity

3.1 Broader societal need

Many people want to know how to use computation more effectively in their jobs for tasks that involve mathematics. There is a growing need for every-day programmers, operating complicated data-based systems, to implement their domain-specific knowledge. By dramatically reducing the friction to using the computational tools in education, and providing new interactive problem-based tools for learning computation, we will address this need. Moreover, we target free open source tools, so that our users get greater value out of what they learn.

Another social need is easy access to computational resources for projects that benefit society. Companies already provide easy free world-wide access to email (e.g., Gmail), short public broadcasts (e.g., Twitter), and networking with friends (Facebook), but no company provides extremely easy collaborative access to sophisticated mathematical computation at Internet scale. Providing this would potentially enable new innovation worldwide. Our proposed SCIP project would provide tools that would help people to learn to use the broad computational resources that SMC makes available.

3.2 Market and business economics

The initial market for SCIP will be every student who is taking a course that involves advanced computational mathematics.

Our target market values the functionality SMC supports. Education is increasingly collaborative, which leads to the need for better tools to support this shift. Teachers are overworked and greatly value convenience for their IT infrastructure.

Price matters to our target market. Students are extremely cost sensitive, budgets are tight, and some teachers care about finding solutions for their students that are efficient and affordable.

We have validated the demand for sophisticated web-based open source mathematical software by creating several iterations of SMC, made available for free. We created basic course managment functionality, which makes SMC usable for teaching a course. SMC was used by over 65 courses during Spring 2015 and has had over 100K users. In May 2015, we transitioned SMC to a "freemium" commercial business model and have had 35 paying customers (with nearly $1,500 in revenue so far). Our proposed homework system would dramatically expand the potential ways to use the tools we offer with SMC in education, so we think customers will have an interest.

The customers we will focus on for this proposal are instructors teaching upper division courses (both at universities or online) and the students in those courses. The business model is freemium. This means that the majority of users consider the service fully usable for **free** and (like Gmail or Dropbox) the free version is not just a trial version! However, we also intend to charge many users a monthly fee to run their projects on better members-only computers. We also sell dedicated resources that instructors can use for all students in a course they are teaching, or for research computations. We pay Google for cloud compute resources; in exchange for managing the resources, backing up disks, and providing redundancy, we mark up the cost by a certain percentage. We thus make it easy for many unrelated users to share resources in the cloud and benefit from our experience managing these resources. We provide a web-based interface that makes the underlying virtual machines easy to use for teaching a course, or doing a collaborative research project. Whenever we provide dedicated compute resources we make money, since we are simply marking up their cost by a fixed amount. We also keep the cost down for the free resources by using Google's cheaper "preemptible instances", which randomly get rebooted, and we restrict the memory and speed of each user's project.

The interactive homework system we plan to build would use the freemium business model described above. It would provide SMC with more functionality, which would make SMC valuable for a wider range of courses, hence increase the number of customers. Instructors will be encouraged to make problems they develop freely available; however, they will also have the option to make problems available for a fee to be used in other people's courses. The fee would then be rolled into the cost for the course (and we would take a percent). As part of this proposal we will test the following hypothesis:

Hypothesis: This "app store" style marketplace could motivate the creation of a large number of high quality problems.

Advanced math courses are frequently small, so instructors are free to choose whatever books, tools, etc. they fancy. Though we will initially target advanced mathematics courses, there is a potential for a trickle-down effect. Suppose an instructor likes using SCIP for the advanced problems that other software doesn't support. They may later advocate to their colleagues to use SCIP for larger multi-section courses (e.g., Calculus) since they know and like it. We will test this with surveys of those who use SCIP for one course:

Hypothesis: Instructors that use SCIP for an advanced course will advocate using it later in large introductory courses.

3.3 The competition

There are a many existing online problem systems that each target a specific market segment. SCIP will initially target advanced computational mathematics courses, which is a segment of the market that hasn't received as much attention, perhaps since the computational requirements are so challenging.

A potential competitor to SCIP could be built on Wolfram Research's product *Wolfram Cloud*, which provides web-based access to Wolfram Mathematica. Wolfram Cloud is different than SMC since it is nonfree, is closed source, and has very limited collaboration and course management functionality. Nevertheless, it would be natural to develop interactive problem functionality on top of this platform. Wolfram Research's longterm investment in their Demonstrations functionality would provide an excellent foundation on which to build an online interactive homework problem system. It's thus quite plausible that the Wolfram Cloud could make a major leap in interactive homework support and course management functionality in the next 8 months, exactly when SCIP might enter the market. Similar remarks and analysis apply regarding Mathwork's Matlab system and Maplesoft's Maple software.

There are many automated homework systems that have been on the market for years. Most offer comparably limited sophistication of interactive homework problems, which seems fine for most introductory courses. We believe our proposed product would be more useful for advanced and computational courses than the these systems. It is likely that some of these competitors will add functionality similar to what we propose for SCIP in the next 8 months, since more advanced courses are a natural market for any competitor to expand into.

Leaving aside interactive homework problems, and instead considering interactive document creation, there are many competitors to SMC (not SCIP). For example, https://wakari.io/, provides a way to use IPython notebooks online. But it's more limited in scope, with no course management, realtime synchronization or document editing functionality. However, it's only non-free plan is currently $25/month, which is too expensive for some undergraduate students. Wakari has no collaboration support, and nothing to specifically support teaching or pure mathematics. There are other sites providing similar functionality, e.g., https://www.pythonanywhere.com/ and https://sandstorm.io/. There are also online coding environments such as https://c9.io/, https://codenvy.com/, and https://koding.com/ which resemble SMC, but again are not oriented around mathematics or teaching.

Finally, there are some online LaTeX editing environments, including ShareLaTeX and Overleaf. A core difference between SMC and the other LaTeX editors is that SMC provides a general computational environment, whereas the aforementioned LaTeX editing environments have no direct way to execute arbitrary code or interpret and transform data of computational results into documents. SMC's general approach offers several ways to embed computations and datasets into documents. The relevance to SCIP is that full support for editing LaTeX documents is a useful foundation for pure mathematics courses that involve writing proofs.

3.4 Key risks in bringing the proposed innovation to market

A risk is that the only users of our SCIP problem service will be people who **can't afford to pay anything**. We can mitigate this by making our product add sufficient value.

Another risk is that **our software is open source**, and most users who like SCIP will instead simply take it and run it themselves on the cloud or their own servers, rather than pay us. It's fine if some (even most) users run SCIP themselves, as long as a sufficient percentage don't. The best way to ensure that people use our central service is to implement ways to make SCIP benefit from network effects, so that our single centralized server with a large number of users (and their content) is much more valuable to everybody than a small private server. In particular, our proposed database of refereed and curated interactive problems is critical, as is superb collaboration, communication, and crowdsourcing functionality. Moreover, we need to make it so that SCIP efficiently shares resources between users, so there is much more value in 10 users paying us for a single machine, than 10 users paying Google directly for 10 separate machines–to make this the case, we have to make SCIP extremely efficient, good at sharing resources, and leverage how we know people use the services interactively (often thinking about a question, then running code for a few seconds, then mulling over the results, etc.).

A similar risk is that because **our software is open source** a competitor could simply run exactly the same software on the same cloud as us, and charge a slightly lower margin. We don't know if this will happen or not, and legally there is nothing to prevent it, except that we can require a competitor to call the service by a different name (due to the trademark). Again, maximizing the value of network effects, curated content, and the quality of our user support, reputation and development pace are the best strategy for addressing this risk. Also, from the NSF's perspective, if other companies pick up this technology and run with it and build great businesses that out compete SMC, then NSF's investment will have been a great success, since they will have supported a big advance in technology, even if SMC is not the beneficiary.

Another risk is that **interactive homework is very hard** to get right. Online games are difficult to implement and platforms for writing online games are even harder. There are many attempts (and much research into ways) to create innovative interactive homework systems, methods to describe problems to these systems, strategies for presenting these problems, and techniques for collating, sharing and making available these problems to others. Looking at the results so far suggests this is a very hard problem. Even building on our system using a clean conceptual framework using a powerful and flexible platform (SMC) combined with our experience, users, and powerful open source software for mathematics might not be enough to produce something that works sufficiently better than existing solutions.

3.5 Commercialization Approach

We create open source software, which runs in the cloud on Google Compute Engine. Most users will try or use SMC online for free, but some small percentage will pay. For this to be profitable, it's critical that the software be extremely efficient, since many of our target customers are price sensitive students, and perceive themselves as being overcharged for textbooks and tuition. The primary products we plan to sell are memberships (with a monthly fee) and dedicated virtual machines. Dedicated machines would be rented by teachers (or their students) wanting better resources for running their classes. For the dedicated virtual machines, we will mark up the price by a fixed percentage, so every sale results in profit.

The longterm revenue potential is significant, since the entire market for interactive problems is large. Whether we will create a product good enough to appeal to even a small percentage of that market is unclear. Also, we are initially only targeting advanced computational mathematics, which is a relatively small market.

The resources we need to implement our plan are people to write software, people to talk with and support users, and money to rent cloud compute time until we are cash flow positive. This SBIR would provide enough financial support to hire people to implement a first usable version of the proposed interactive problem platform, get some testing by users, and iterate on this feedback. We would use the funding to hire the PI, who is an experienced teacher, and some programmers with classroom experience. As an academic and founder of the SageMath software project, the PI has the necessary connections. Our hope is that using resources the company already has, we will have people in place for the project at a small percent of time, so if we get funding from SBIR, we can then pay these people to work a larger percent of time, with vastly more ambitious goals.

3.5.1 Product-market fit

The proposed SCIP problem system, which we plan to implement, doesn't exist yet, so it is difficult to address product-market fit. As evidence for product-market fit of SMC, on a typical weekday during Fall 2015, SMC would be used by around 3000 people. There are many sites for interactive homework, which suggests that there is demand for our service.

Based on user feedback, we know of several reasons that college teachers use the SMC platform. SMC lets them get around issues and inefficiencies with campus IT. Permission with campus IT can be hard to get, help is difficult, service can be poor, resources may be insufficient, and it is hard for IT to know *a priori* what user needs will be, hence they will often allocate resources inefficiently. Moreover, many students have slow personal computers on which they are afraid to install software.

We have also found that teachers greatly value even the most atrocious interactive online homework systems. Grading homework is very time consuming, but interactive homework systems are patient and available at all hours, and instructors find that their students are often far more engaged in learning mathematics when they get immediate feedback.

We have seen that with SMC's collaboration capabilities, instructors can easily see and interact with exactly what their students are doing. Also, instructors value that detailed information about what the students type while working on their homework is recorded in SMC, which makes detection of cheating easier, and also provides more detailed feedback on how students are doing in a course. (Unfortunately, clear evidence of cheating, exactly when it happened, and how and by whom, has been periodically reported to us by instructors using SMC in courses.)

3.5.2 Presenting barriers to entry for competition

SCIP is aimed at computational mathematics teachers and students, especially those that care about cost, efficiency, and value open source software. To design SCIP in the first place, and to continue to refine it indefinitely, will require people with extensive experience teaching courses involving computational mathematics. The PI, who is also CEO of SageMath Inc., is the founder of the SageMath open source mathematics software project, which is the largest open source math software project. Sage has had over 500 contributors since when he founded the project

in 2005. He has been passionately involved in the development of mathematical software and computational mathematics research and education since 1998, and has had a substantial impact on the field. For example, he was recognized with the ACM Richard D. Jenks Prize in 2013. Also, there have been 70 Sage Days workshops. Many potential competitors do not have this depth of experience and connection with the community.

Just copying the implementation of SMC wouldn't give a competitor a unique advantage, since the source code of SMC itself is completely open. Instead, we have a head start since we have been building a community around Sage for a decade, and around SMC for several years now (100K registered users). SMC is collaborative, which results in a big network effect, where SMC is more valuable because some of your colleagues already use it. This community is a base of potential users of SCIP.

The pricing model is another barrier to entry. Offering a service with a significant markup over the cost of hosting is a race to the bottom. Instead, SMC shortcuts that race and already starts with a low price (a freemium model with $7/month for some premium features, plus a standard percentage markup on whatever Google happens to be charging for cloud computing resources). This doesn't leave much room for competitors operating on top of the same free tools (Linux, LaTeX, SageMath, R, Python), unless they offer a significantly better service on top, e.g., human support, extra value through having far more users, more software pre-installed, etc. The pricing for SCIP would be identical–we're just adding SCIP as another application that anybody can run on top of the SMC platform.

4 The Innovation

Our proposed online interactive problem system mainly builds on the Sage mathematical software and the SageMathCloud platform.

SageMath is powerful open source mathematical software developed since 2005 with millions in NSF support. SageMath is open source software that provides comparable functionality to Mathematica, Maple, Matlab, and Magma. It is powerful and user friendly, and uses the popular scripting language Python.

SMC is a sophisticated cloud-based collaboration environment for using SageMath, R and other software, which has been under development for about 3 years, with significant feedback from users.

SMC involes only technology that is open source, hence available to all. However, we also have market data due to running the software for years. We have experience and a developer community that cuts vertically through all levels of mathematics. Also, Google, Microsoft, and Amazon are aggressively competing to offer cheap and flexible cloud-based compute, which is key to SMC being viable.

SMC offers realtime sync at all levels of the system, which makes people more efficient at collaborating on computational mathematics. It also makes it easier for an expert to quickly help out with problems that beginners hit. Moreover, the evolution of computational documents (every keystroke!) is recorded, which supports reproducible research, and adds a new dimension.

5 The Company and Team

SageMath, Inc. was founded in February 2015 as a Delaware C Corporation with assistance from Fenwick & West, which is a law firm that advises technology companies and start-ups, such as Facebook, Google, Uber, Dropbox, WhatsApp, and Cisco. William Stein is the owner and CEO of SageMath, Inc., and a modest initial angel investment was provided by William Randolph Hearst, III. William Stein has been working full-time on SageMathCloud and SageMath, Inc. while on a sabbatical leave from the University of Washington. Business operations include a website, bank accounts, and billing through Stripe.

The first product offering, an individual membership in SageMathCloud at $7/month, was made available in mid-May 2015. Courses with demanding computational needs (such as student exercises in parallel computing) have paid for dedicated resources. Together these customers have generated nearly $1500 in revenue.

The team has an excellent understanding of the needs and requirements of their market. The contributors to SMC are vertically integrated with a balance of contributors from high school, college, graduate school, teaching, research, and business. The core team consists of William Stein (Professor at Univ of Washington), Harald Schilly (applied mathematics graduate student in Vienna, Austria), Rob Beezer (professor at Univ of Puget Sound), and Jon Lee (Undergraduate at Univ of Washington). The team also consults about business matters with Dennis Stein (San Diego businessman), Tim Abbot (co-founder of two successful tech startups, now at Dropbox), and Will Hearst (experienced venture capitalist, investor and philanthropist), and discusses approaches to online homework with high school student Sequoia Lefthand. We have also consulted regularly with numerous faculty at Univ of Washington about the design of SMC, including Randy Leveque and Max Leiblich, and with Craig Citro and Robert Bradshaw at Google.

The PI, William Stein, founded the SageMath open source mathematical software project in 2005. He has managed millions of dollars in funding from NSF, Google, Microsoft, and DOD. He is the 2013 recipient of the ACM/SIGSAM Richard D. Jenks Memorial Prize for excellence in software engineering applied to computer algebra. He has significant market penetration in pure mathematics research: he's been cited in at least 410 articles, theses and books, and is the author of over forty original research articles in mathematics, and three published books. He has twenty years experience teaching undergraduate and graduate courses, and has supervised four completed Ph.D. theses and several undergraduate projects.

Harald Schilly is a Ph.D. student at Universität Wien in Vienna, Austria. Since 2007, he has made numerous contributions to the core SageMath library in numerical mathematics and optimization. He maintains numerous aspects of the Sage infrastructure, such as web sites and mirror sites. He is the design lead of the SageMath website and brand and is responsible for community management and marketing Sage (social media, designing fliers, etc.) He has been heavily involved with SageMathCloud development, and assists William Stein with technical design decisions and implementation. He provides online help for users, and is responsible for marketing SageMathCloud. He owns a consulting business in Austria that helps manage technical aspects of teaching courses at his university, doing data analysis, design, programming, and statistics. He has several years experience teaching programming courses.

Robert Beezer has made numerous contributions to the core Sage library in linear algebra, graph theory and group theory, since 2009. He is an active member of the user and developer community, and has organized four Sage Days workshops specializing in educational uses of Sage. His mathematical research involves Algebraic Graph Theory, and he has published fifteen original

research articles. He has also written three online open source textbooks, which include extensive discussion of Sage. He is the project founder of the open source Mathbook XML authoring system for creating online open textbooks. He has taught courses and designed instructional software at University of Puget sound for over 37 years, and supervised 31 undergraduate research projects.

Jon Lee is an undergraduate computer science major at University of Washington. He became involved with SageMathCloud development when SMC was used for a course he took on scientific computing. In 2014 he spent the summer leading a team of undergraduates who worked fulltime doing SMC development, and is being funded by Google to work fulltime on SMC during Summer 2015.

In addition, the SageMath software is an open source project with over a decade's history of attracting top-flight researchers in pure mathematics and programmers with excellent technical skills. The implementation of sophisticated algorithms in Sage results in a highly reliable and easy-to-use package that has attracted a large user community, who in turn provide testing, discussion, advice, and requests for enhancements. The volunteer contributions of over five hundred mathematician-programmers and thousands of users are a key part of maintaining and improving a critical component of SageMathCloud.

6 Technical Discussion and R&D Plan

6.1 Introduction

We propose to implement, on top of the SageMathCloud (SMC) platform, a next generation system for web-based homework problems called SageCloud Interactive Problems (SCIP).

Many of the innovations of existing interactive homework systems involve integrating with publisher content, using statistical techniques to provide more relevant sequences of problems, checking answers and providing hints very quickly by working entirely in the browser, integration with video, and leading the user step-by-step to solutions using hints. Our plan is to implement a system that not only leverages many of the above innovations, but also supports sophisticated compute on the backend servers, and records everything the student does in detail, synchronized in such a way that instructors and tutors can see what is happening as students work and provide feedback in realtime. This would make the experience of walking around a computer lab available online. Moreover, all this will be optimized to support application in pure mathematics, statistics, and numerical analysis. Problems will be shared among users, and data about how the problems have been used (basically every keystroke by every single student) will be mined to provide a better experience. We also plan to implement improved functionality for computer-assisted manual grading of advanced problems and for enabling peer grading of problems by students in a course. We will focus mainly on theoretical mathematics, and courses that involve programming in Python and R, and computational mathematics.

6.2 Difficulties

At its core, SCIP involves designing and implementing a way to create interactive problems, a way to share those problems, and a way to systematically combine problems into assignments. There are many daunting problems that arise in addressing these challenges.

6.2.1 Computational documents are hard

One technical difficulty is that making "computational documents" (Sage worksheets, IPython notebooks, etc.) available online efficiently at scale is very hard, because these documents are more expensive to host. The realtime synchronization problem for computational documents is challenging and SMC demonstrates that it can be solved (of course, building on our implementation of similar algorithms that are used in Google Docs). We envision our interactive problems as being a special type of computational document with realtime sync.

6.2.2 Checking answers is hard

The full power of Sage and other software will be available to problem authors to check answers. E.g., a problem author might want the strings 1/3, 0.333, 0.33331 to be considered correct answers for $f(3)$ when $f(x) = 1/x$. Nontrivial numerical and computer algebra algorithms in Sage can be adapted to address some of these problems, with the solutions made available in a reusable library. Many current online problem systems simply sample functions at a range of values and presume that multiple numerical agreement is function equality; in addition to that approach, Sage can do much more by using sophisticated symbolic simplification, interval arithmetic, and other techniques. This is a place where Sage is potentially superior to some existing systems's answer checkers, which though quite good, are limited in scope to "basic" algebra, functions and matrices. Much more sophisticated functionality from Sage will be required to address more advanced mathematical problems, e.g., involving abstract algebra, number theory, algebraic geometry, combinatorics, algebraic topology and so on (all areas where Sage is highly developed).

6.2.3 Maintaining problem quality is difficult

You only have to admit in class that one or two problems are busted and then every evening you get a flood of 10pm emails, "I *know* I have number 7 right; it must be a busted problem?" Well, no.

Instead of trying to amass the largest number of problems possible, or to try to copy problems from existing textbooks, we will aim initially for high quality problems for more advanced courses. We will carry over ideas from the Sage software development process, which is a carefully refined peer review process inspired by a mix of how the peer review process works for journals and how code review works in other open source projects (see http://trac.sagemath.org/). We believe homework problems need to meet the same high standards of review and automated testing that we apply to Sage's codebase. This is expensive, frustrating and difficult, but is needed to ensure problems are useful to instructors and stay reliable several years later as Sage and other software evolves.

6.2.4 Authoring problems is hard

We do *not* expect an instructor to single-handedly author 30 problem sets of 10 interactive problems each, on-the-fly during a semester. We estimate that a good problem takes at least an hour to write, test, debug, refine, and often much more time, depending on whether the problem involves adapting an existing problem or not. Creating a good problem is much like creating a puzzle or game, with a range of input states, hints along the way to mastery, and so on.

Making the authoring tools as easy as possible to use is very important. We have some experience in this direction with Sage worksheet and the interact functionality, which automatically turns functions into interactive graphical interfaces. However, much additional innovation, leveraging the latest web development techniques (e.g., reusable UI components), will be needed. Also, we will design and implement social mechanisms to encourage the growth of an ecosystem of peer reviewed and automatically tested problems, including giving SMC credits in exchange for making good problems available, and even creating a marketplace for problems. In addition to our own peer review and testing processes, we can also analyze student behavior (and simply ask for feedback from students) to ascertain how good problems are.

6.2.5 Discovering useful problems is tricky

It is nice when an interactive problem is designed to exactly match the flavor of a problem in a popular mathematics textbook. But the chain rule is the chain rule, no matter what book you use. So curation, cataloging, indexing, keywords, etc., are extremely important, and not done well in some existing interactive homework software. A rigorous review and testing procedure, plus simply making it very easy for problem authors to classify their problems (e.g., automatically suggesting tags based on heuristics), would ensure that the problems in SCIP are more discoverable by instructors.

6.2.6 Problem management is a rats nest

Problem management is a real rats nest: rosters, scores, partial credit for late assignments, extended deadlines for sick students, multiple or unlimited attempts to solve problems, and automated peer grading of advanced problems (e.g., proofs or writing assignments). We all have our quirks when it comes to student work, and making these issues easy to manage is a place where a lot of effort is required to get it right.

6.2.7 Problems are resource intensive

Unlike most online homework systems, SCIP aims to support problems that may be very computationally intensive for the students to solve. For example, imagine a problem in a scientific computing course that requires interactively running parallel code on a 32-core computer: the student would click a button to start working interactively on the problem, and a dedicated Google compute engine pre-empt instance would start running at a cost to us of \$0.352/hour (see https://cloud.google.com/compute/pricing). The student would have a fixed maximum amount of time in which to solve that problem, to ensure that the amount of resources they use aren't too expensive.

6.3 Development Plan (6-Month Timeline)

1. 2 weeks – Implement an initial minimal interactive problem system.
2. 2 weeks – Make the problem system usable via SMC and collect feedback as we encourage people to test it.
3. 1 month – Implement a sharing marketplace
4. 1 month – Test the system on students via homework assignments
5. 3 months – Iterate

6.3.1 Implement a first simple interactive problem system (2 weeks)

Our design is inspired by well-known game programming and user interface design patterns.

The first step is to create a simple version of an interactive problem system built on top of the SMC platform. We have access to many problems created by instructors as Sage worksheets, IPython notebooks, and LaTeX documents, which we can draw on for inspiration regarding design requirements.

An interactive problem will be specified by four components:

- **Description:** A short title, a longer description, and tags.
- **Initialization:** A problem author would write code, written in *any* programming language, that initializes the problem state (this is like randomly populating a game world). The might run and output a string that can be parsed as JSON. For example, it could create a random 3x3 matrix with integer entries that can be easily reduced to echelon form (it could leverage sophisticated code in Sage for constructing example of such matrices!). The state will also include information about what the student has attempted so far, which would be initialized to empty.
- **Render:** The author would create code (or a template) that, given the problem state object, renders a view of it for the student to look at and interact with. There are many ways to do this, and we may eventually support several. For the first version, we would create a small Python library of code that generates user interface widgets, which are then rendered in a browser (possibly using Facebook's React.js library), and eventually also rendered in a mobile app (possibly using React native). Widgets would include blocks of text that can include LaTeX (or markdown, etc.), 2d and 3d plots (easily produced using Sage), and standard controls such as input boxes, sliders, buttons, checkboxes, point selectors on plots, and graphical formula input. This will be inspired by the interact functionality we have had in Sage since 2007, but redesigned for interactive problems. As much as possible of this rendering logic will live in the client, and what can't will be asynchronously evaluated on the back end. Also, multiple users (e.g., the student and the instructor) could both view this rendered problem on separate computers, and see the same thing.

A key point is that the above UI widgets could leverage everything Sage and SMC have to offer.

- **Respond:** In realtime as the controls change state (e.g., drag a slider, click a button, entering a formula in a box), the controls state is synced back to the server and updated for all viewers. In addition, a program running on the server is given the chance to respond to certain state changes by updating the state. These state updates could provide additional user interface guidance (e.g., display a sequence of hints, additional reading material, etc.). They could generate new random parameters for the problem, in case the hints have been too excessive. If the student answers too slowly, it could generate another problem. Each interactive problem is a game, which the student has to master. A single interactive problem consists of potentially many randomly generated math problems. We will also add an option so that the response actually punts to a human to do the grading in some cases, e.g., grading a proof written in LaTeX of a theoretical mathematics problem. When this state is reached the problem would indicate it and the user would close the problem

and do something else until being notified that the problem was graded, and the proposed solutions would be pre-processed (e.g., check that programs run or latex compiles before sending it to an instructor for manual grading). Graders would also be notified that there are a batch of proposed answers to a given question ready to be graded. Graders could optionally in some cases be other students in the same course, in similar courses, or outsourced students in India.

The problem author would start out with a very simple template for each component, and make each part more sophisticated. We imagine they might see the problem template on the left side of the screen and a realtime rendering of the problem running on the right hand side, along with buttons to reset or manually change the problem's state to better test it.

- **Tests:** There are at least two types of automated tests we would initially implement: **Logical tests** would be 3-tuples of states: an input state for the problem, a new state caused by user feedback, and the resulting state update caused by the Respond component. When authoring a problem, the sequence of states would be available graphically in the preview pane, and the author could drag and drop states from the preview pane to the tests pane. **UI tests** would involve rendering the UI for random (or prescribed) valid states of the problem and ensuring that the rendering code doesn't give any errors, and also satisfies some conditions.

6.3.2 Make the interactive problem system available (2 weeks)

After implementing a first *minimal usable version* of 1, which would take about 2 weeks of fulltime work by our team (only 2 weeks, since this is similar to the `@interact` functionality already in Sage), we would make it available from SMC for people to test out. It would take about a week to make this available to other people, since we would need to include feedback functionality. This would not at all be production ready, e.g., the Respond component above would run as the same Linux user as the person using the problem, hence cheating would be easy. Also, this stage is focused entirely on specifying and interacting with a single interactive problem, rather than creating homework assignments.

We would solicit feedback from our community (e.g., by notifying the 100K account holders on SMC about this feature, by posting on social media where we have thousands of follows, etc.). People would then create interactive problems using what we've written, and provide us with feedback. We would also try converting some of our own problems and encourage people to try other people's problems. The result will be a lot of people playing little math games, while we collect data about how things work. In particular, we record the state sequence as users interact with problems, along with any errors that occur. We also have a text box next to every interactive problem, in which the user is encouraged to type anything that comes to mind.

Based on about a week of feedback from users, we would then iterate on Step 1 again.

6.3.3 Implement a sharing marketplace (1 month)

Next we would implement a way for authors to share the interactive problems they create. We'll add an additional pane to the problem authoring tool from Step 1 through which the instructor can publish a problem to a central database. The database would then get a snapshot

of the complete problem description, including testing code, etc. The instructor would specify the copyright license under which other people are allowed to use their problem, which would include both free (creative commons) and commercial with a set price.

Once a problem is pushed to the sharing database, we (the developers, initially - later other people) would be notified and would peer review the problem. The author would then get a notification about our peer review conclusion, and they could update their problem accordingly. Sage developers are very used to doing a huge amount of this, and to having every contribution extensively peer reviewed (see for yourself at http://trac.sagemath.org/).

We anticipate it will take us about 1 month to get a basic usable version of this up and running.

6.3.4 Test the system on students via homework assignments (1 month)

Instructors will browse the marketplace and build a homework assignment from the interactive problems they find there, and also problems they craft themselves. Again, there will be a graphical browser, with drag and drop, which allows the instructor to put together the problem set, assign points to each question, the order of problems, etc. Instructors will have extensive access to information about how the problem has been used already by other students–how many people have done the problem? do they like it? do they hate it? do they feel it helps them learn? do they think it is broken? has it been peer reviewed? is it hard or easy?

Once a problem set is constructed, the instructor will assign it to the students in their course using an updated version of the course management functionality built into SMC. Once assigned, the instructor (and the teaching assistants) will have full access to the exact state of all problems as they are worked on by students. They will see in a dashboard an overview of how students are doing, who might be falling behind, etc. Students will be able to request help and the instructors will then be able to see exactly what the student is doing, (video/text) chat with them, make changes, highlight things, etc. The instructor can also use a slider to quickly see exactly what the student has tried to do while working on the problem before offering advice (rather than only asking the student "what did you try").

We will spend about 1 month implementing a usable minimal version of the above. This takes much longer than the previous 3 steps, since the quality requirements are much higher, as student grades are potentially at stake (instructors are way more likely to provide quality feedback when you're actually saving them time!). We would also spend two weeks systematically gathering usage data.

6.3.5 Iterate (3 months)

We would spend the remaining 3 months iterating the above process and collecting data. We would in particular focus much of our effort on refining the tools for creating problems, since there are many difficult UI components that problem authors will demand, and we will also see a need for a library of sophisticated code tools for implementing the Respond part of the interactive problems, possibly going as far as integrating some natural language processing functionality.

Based on our data and experience we will learn whether or not there is a market for the sort of sophisticated automated homework problem system that we propose above. If so, we will then apply for a second stage SBIR grant.