SAGEMATH IN EUROPE: MUTUALIZED DEVELOPMENT OF COMPUTER EXPLORATION ENVIRONMENT FOR RESEARCH IN MATHEMATICS

1. Summary

1.1. **Context and scientific theme.** Computer exploration is an instrumental tool for research in mathematics; in some cases, computers have even been used for proving results like the famous four color theorem, or the alternating sign matrices conjecture. The 90's have seen the creation of many specialized open source packages, like GAP for group theory, Maxima for symbolic computation, or Pari/GP for number theory. Yet, specialized packages are insufficient in many areas, like combinatorics and geometry, where one needs to handle simultaneously a wide variety of objects and features. However, not being open source and lacking basic computer science paradigms, general purpose systems like `Maple` or `Mathematica` impede the pooling of the software development efforts.

To tackle this situation, William Stein started in 2005 the `Sage` **project, a free open-source mathematics software system**. It provides a full featured **research environment**, including a rich notebook interface for interactive computations and visualization, collaborative work on the cloud, or intensive batch computations. `Sage` builds on the Python and Scientific Python stack, enriched by dozens of open-source mathematical packages and a native mathematical library.

`Sage` is developed by a large international community of more than 300 researchers and teachers in maths, computer science, and physics, using modern collaborative tools for agile development. A strong emphasis is put on collaborative development and training by regularly organising week-long workshops all around the world. `Sage` fosters inter-disciplinary research by promoting code sharing and reuse in widely different contexts, including **physics and computer science**. It is **particularly beneficial for young researchers**.

1.2. **Challenges and originality of the solution.** So far `Sage` has been running on a tiny budget for its size. Yet, long term critical non mathematical features like portability, modularization, packaging, user interfaces, large data, parallelism, or outreach toward related software, have been lagging behind. Indeed they cannot be implemented as a side product of research projects, and **need to be assigned to full time research engineers**. Regular funding is also needed to better structure the `Sage` community in Europe and support its upcoming major widening through training and development workshops, exchanges, ...

To scale to mainstream and assume its leadership, `Sage` needs direct funding in addition to indirect funding through research projects. Such funding already exists in the United States, e.g. with large NSF Computational Mathematics grants; the boosting effect on the community is already shining (workshops, ICERM semester, many new recruits, ...).

The purpose of this proposal is to trigger a similar boost in Europe.

1.3. **Outreach.** `Sage` already plays a fundamental role in hundreds of scientific publications and triggers cross-fertilization across disciplines. Its usage for `higher education` is spreading. Those benefits will be boosted to the measure of `Sage`. The project will further contribute back specialized and general purpose tools to the Scientific Python ecosystem which is used intensively by both the academic and industrial worlds.

2. Context and scientific theme

Since the 50's computer exploration has been an instrumental tool for research in mathematics; in some cases, computers have even been used for proving results like the famous four color theorem, or the alternating sign matrices conjecture. To support this, much efforts have been spent on the software development. As the sophistication of the required computations increased, supported by the boom of the available computational power, it was progressively realized that single topic/single person software projects, or even projects at the level of a single research team simply did not scale. It was essential to share the software development at the level of a research community, both in term of breath of features and mutualization of development efforts. This lead to the apparition in the 90's of many specialized open source packages, like GAP for group theory, Maxima for symbolic computation, or Pari/GP for number theory.

Yet, this proved insufficient in many areas like combinatorics, as computer exploration often requires simultaneously a wide variety of objects and features which only a general purpose software system can provide and support. At the same time, the then existing general purpose systems like `Maple` or `Mathematica` prevented the scaling of existing packages because they lacked basic computer science paradigms; even more important, they were not open source.

To tackle this situation, William Stein started in 2005 the `Sage` **project, a free open-source mathematics software system**. This project soon crystallized a large international community of contributors; now more than 300 researchers and teachers in maths and computer science. It already played a fundamental role in hundreds of scientific publications, and its usage for `higher education` is spreading.

Building on the very active Python and Scientific Python stack combined with the power of dozens of open-source mathematical packages (GAP, Singular, or FLINT to name but a few), `Sage` provides a full featured research environment including a rich notebook interface for interactive computations and visualization, as well as support for intensive batch computations, for example on the cloud. `Sage` is also the computational workhorse behind sister database projects like LMFDB, the online database of L-functions, modular forms, and related objects, or the EFD, the explicit formulas database, inventoring multiplication formulas for elliptic curve cryptography.

Thanks to a high level programming framework, most of the code can be written generically and reused in widely different contexts, including **physics and computer science**. The induced cross-discipline interactions both between developers and with users are a regular source of cross fertilization, and already lead to several joint research projects.

`Sage` **is particularly beneficial for young researchers**, giving them access to a platform with a wide array of features, and allowing them to focus on those computational aspects directly related to their own research topic. With, at the end, the possibility and incent to contribute back their developments to the community *and get credit for it.*

Collaborative software development involves lots of interactions within the community, around software and user interface design, algorithmic, code review, governance, and maintenance. Those interactions are organized using modern collaborative tools for open source agile development (distributed version control, mailing lists, ticket server, testing tools, ...), and following best practices. A strong emphasis is put on training, in particular at the occasion of numerous week-long workshops like the upcoming *Sage Days 60* in India.

3. Scaling further toward 2020

After 10 years of existence, `Sage` **has proven its viability and leadership**, surviving the many hurdles faced by new open source software projects, reaching a comfortable community size, and proving the adequacy of its "developed by users, for users" model. Yet `Sage` has not yet been systematically adopted, and still **much good research time is wasted developing and redeveloping tools by lack of sharing and coordination**.

By design, the development of `Sage` has been consciously designed to run on a small budget. This was a key feature toward its independence and its early wide adoption across research teams and international frontiers. In particular, it is "developed by users, for users": most if not all of the development is achieved by the researchers themselves and justified w.r.t. their funding institutions by the immediate application to their own research projects. Core development is taken care of by permanent researchers that can afford to invest time over the long run (10 years). Coordination is achieved electronically using modern collaborative development tools (distributed version control, ticket server, mailing lists, ...).

So far, `Sage` has been running viabily on a small budget for its size (300 developers), thanks to its "developed by users for users" model. Yet, long term non mathematical features have been lagging behind because they cannot be split in pieces small enough to be implemented as a side product of research projects: portability (including native Windows support), build infrastructure and modularization, packaging and distribution, improved user interfaces, core support for parallelism, large data support, ... Those features are critical to let `Sage` scale to mainstream and assume its leadership. To this end, **the development model needs to be backed up by full time research ingeneers**.

It is also vital to organize regular physical meetings (missions, invitations, workshops) for fast coordination; this becomes even more critical when it comes to train massively new developers and users. Finally, productive software development requires powerful hardware for compilation, regression testing, continuous integration, benchmarking. Up to now, this has been funded as an aside of research projects (), or through the case by case generosity of labs, taken on their recurrent budget (e.g. developers meeting and missions funded by the LITIS in Rouen, the LRI and LMO in Orsay, ...), but it has reached its limits, with much time wasted on patching together little pieces of funding.

To scale further, `Sage` needs direct funding in addition to indirect funding through research projects. Such funding already exists in the United States, e.g. with large (500k$) NSF Computational Mathematics grants; the boosting effect of those on the community is already shining there (workshops, ICERM semester, many new recruits, ...).

The purpose of this proposal is to trigger a similar boost in Europe.

4. Outreach

4.1. **Teaching.** We plan to integrate `Sage` into the curriculum of the courses at our universities. Many undergraduate and graduate courses in number theory and combinatorics already involve computational components. Moreover, since this year, Sage plays an important role in the Agrégation. With the dwindling resources at the universities, a free and open-source alternative to the currently used systems is most welcome. Since `Sage` is based on Python, one of the top five programming languages, this also gives the students valuable education. Inspired students can furthermore look at the algorithms or even contribute back to `Sage`.

5. Consortium

By design, Sage is developed by a community of indivivally involved researchers, transversely to the institutions. They form a geographically dispersed group of people, whose fields of expertise cover a wide continuum of research themes in discrete mathematics, dynamical systems, geometry, algebra, or number theory, with deep connections to neighbor fields like symbolic computations or statistical physics.

In France, a core team has been closely collaborating around Sage for a long time. In particular, they coorganized and participated to a stream of events that included (international) developer meetings, user group meetings, training sessions (Burkina Faso, France, Québec), and development of educational material.

The Sage-Combinat project, a strong international community of collaborators founded in 2000 that focusses on combinatorics, will be involved in the project, including a large number of PhD students and postdocs.

The European consortium, in building, aims at scaling this dynamics and experience in both geographical and thematic directions. We already got positive feedback and engagement of researchers from Austria (Linz, Wien: computer science), France (Rouen, Bordeaux, Marne, Saclay: computer science; Lyon, Marseille: math ; Paris: math, computer science, physics), Germany (Hannover, Berlin, Jena: math), Switzerland (Zürich: math), Great Britain (Oxford, Bristol: math), Ireland (Dublin: physics), Québec (Montréal, joint lab CRM/CNRS: computer science, math).

6. Actions

6.1. **Collaborative tools.**

- Development of an online repository for indexing and sharing notebooks and other Sage resources.
- Workflow improvements: continuous integration, ...
- Collaborative interactive tools: e.g. embedding live shared Sage sessions (and more generally IPython) in voice-over-IP or teleconference calls.

6.2. **Modularization of the Sage distribution.**

Separation of the different components of Sage (communication with third-party softwares, build system, Sage native code code). This is a prerequisite for easier packaging and integration in standard Linux distributions and lmonade, native integration within the IPython notebook and other interfaces (larchenv, Spyder, ...) and collaboration with sister projects.

6.3. **Deployment.**

- Better Windows support
- Live USB keys
- Creation, deployment, and distribution of preconfigured virtual machines for Sage as a cloud service, in particular within the StratusLab infrastructure.

6.4. **Interfaces with other computation systems.**

- Support for SCSCP
- foster collaboration with upstream libraries by sharing the development and maintenance of the interfaces (typically as standalone Python bindings)

6.5. **Dissemination and teaching.**

- Documentation improvements: overview, cross links, overview of recent improvements
- Thematic tutorials
- Collections of pedagogical documents
  E.g. a complete collection of interactive class notes with computer lab projects for the "Algèbre et Calcul formel" option of the French math aggregation (starting from 2015, only open-source systems will be supported, and Sage is a major player).
- Localization of the `Sage` user interface and key documents in various European languages.
- Distribution of the documents either in the main distribution of Sage or through the online repository (see collaborative tools).
- Massive online introduction course to Sage, drawing on the sage tutorial/notebooks. Could be "First year `Sage` course in a box".
- Taking the opportunity of Python courses to propose `Sage` as a natural extension for mathematics; an example is French's "Classes préparatoires"[1], where Python has been recently selected as the language to learn programming[2].

6.6. **Community animation and training.**

- Sage Days (training workshops and developers meetings)
  –
- Exchange visits for students, researchers, ... in between partner institutions, and with developers worldwide
- Strengthening of collaborations with sister projects: GAP, Linbox, Singular, PARI/GP, IPython, Scientific Python, R, to name but a few.
-

6.7. **Math features.**

- Foundation work: categories, morphisms, coercion, ...
- Generic parallelism tools for combinatorics
- Representation theory of monoids and algebras
- Ore algebras and applications to combinatorics
- Further development of combinatorial species
- Automatons and combinatorics on words
- Symbolic dynamical systems (substitutive systems, Bratelli diagrams, sofic shifts, tilings, interval exchange transformations)
- Manifolds (symbolic computations in charts, differential equations, etc)
- Intersection theory in algebraic geometry
- Complexes of modules, resolutions.
- Graded commutative differential algebras and applications to rational homotopy.
- ...

6.8. **Parallel Computing.**

- Transparent integration of Ipython capabilities for cluster computing.
- Customization of blas/lapack implementation to take advantage of GPU computing.
- Implementation of a transparent abstraction over mpi.
- Develop or integrate existing solutions for MapReduce operations over big data.

[1] `http://en.wikipedia.org/wiki/Classe_préparatoire_aux_grandes_écoles`

[2] See the "Annexe" at `http://www.education.gouv.fr/pid25535/bulletin_officiel.html?cid_bo=71586`

7. Tentative budget

Research engineer (5 pers for 5 years)	1250k€
Postdoc (2 pers for 5 years)	400k€
Invitations and missions (national / international), student internships	100 k€
Sage Days and developers meetings (1+2 per year)	100k€
Subcontracting, participation to the Systematic cluster (GT Logiciel Libre)	50 k€
Hardware	100 k€
Total (including 4% overhead)	2000 k€