SAGEMATH IN EUROPE: MUTUALIZED DEVELOPMENT OF COMPUTER EXPLORATION ENVIRONMENT FOR RESEARCH IN COMBINATORICS, ALGEBRA, AND BEYOND

1. Context and scientific theme

To do: Mettre en avant le fait que Sage a des bugs critiques et qu'on a besoin de travail technique pour en venir a bout.

Since the 50's computer exploration has been an instrumental tool for research in mathematics; in some cases, computers have even used for proving results like for 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.

Originality of the Sage project. 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.

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.

To do: B) Les challenges et l'originalité de la solution

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. Those features are critical to let `Sage` scale to mainstream and assume its leadership. **To do: FINISH THIS SENTENCE For those, 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 (**To do: cite them**), 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, ... **To do: expand to the European scale**), 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.

2. Outreach

To do: C) Les retombées scientifiques et sociétales

To do: Insert this somewhere: Dissemination: Sharing code increases our visibility, spreads our research results to other scientific areas, and benefits education.

2.1. **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`.

To do: Reuse stuff from the NSF grant

3. Consortium

To do: Expand

The consortium will tentatively have participants in Austria (Linz: CS), France (Rouen: CS, Bordeaux: CS, Lyon: math, Marseille: math, Marne: CS, Paris: math, CS, physics, Saclay: CS), Germany (Berlin, Jena: math), Switzerland (Zürich: math), Great Britain (Bristol: math), Ireland (Dublin: physics), Quebec (Montréal, joint lab CRM/CNRS: math).

The PI's have developed a strong community building experience which they gained in particular along the MuPAD-Combinat project (now `Sage-Combinat`) that progressively gathered a community of 42+ contributers since it's foundation in 2000.

On the French side, the consortium will be built around a core team that has been closely collaborating around Sage in France and Québec for a long time. In particular they coorganized and participated to a stream of events that included (international) developer meetings, training sessions, and development of educational material. This core team covers a wide continuum of research themes in discrete mathematics, dynamical systems, algebra, or number theory, with deep connections to neighbor fields like symbolic computations or statistical physics. It has developed an international network of collaborators that will be involved in the project, including a large number of PhD students and postdocs.

To do: Expand

4. Actions

4.1. **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.
- **To do: please expand**

4.2. **Collaborative tools.**

- Development of a database of notebooks **To do: Be clearer: we want to get rid of notebooks over the filesystem and .sws tarball**
- Workflow improvements: continuous integration, ...
- **To do: please expand!**
- foster collaboration with upstream libraries by sharing the development and maintenance of the interface (as a standalone Python bindings)

4.3. **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.

4.4. **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.

4.5. **Other technical.**

- support for SCSCP
- **To do: expand**

4.6. **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).
- indexation of notebooks and tutorials in a database accessible through a web interface
- **To do: please expand!**

4.7. **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
- **To do: please expand!**