GSoC Project Proposal 2025

Nicholas Bell

Personal Information

Email: nicholasbell972@gmail.com
University: Sorbonne Université, 75005 Paris, France
Location during project: Paris, France (UTC+2)

Background

Currently, I am a first year student pursing a Master of Computer Science at Sorbonne Université, France, and hold a BSc in Mathematics and Computer Science from Victoria University of Wellington, New Zealand.

Experience in software development. From 2022–2024, I was a full time graduate analyst developer at FNZ Wellington, where I worked primarily on bug fixes for a financial management platform, with a focus on backend maintenance of the fees and charges engine. I am most proficient in C and Java and am familiar with Python, as my bachelor's was mainly in Java with some C, and my master's studies so far have been mainly in C and Python. Given my professional experience, I am also very used to using Git. I have some prior open-source experience developing a plugin for the game client Runelite in Java.

Experience with SageMath and with algebraic algorithms. I have been using SageMath for multiple courses this year, in both cryptology and linear algebra algorithms, and have in preparation for this project installed the developer version and attempted a couple of bug fixes (#39774 and #39852) to familiarise myself with the codebase and the contribution standards. My master's project for this semester has also been focussed on Oil and Vinegar signature schemes and a certain cryptographic attack by Kipnis and Shamir which makes use of the minimal polynomial of a matrix. As such, I have a direct interest in improving the complexity or practical efficiency of these methods, as well as some recent experience with them.

Project

Project (suggested by SageMath at https://wiki.sagemath.org/GSoC/2025):
Functionalities for Krylov methods over exact fields
Mentors: Vincent Neiger and Xavier Caruso
Project length: 350 hours

Description
This project aims to add functionalities for Krylov methods in exact linear algebra, notably in order to provide algorithmic solutions for the following two questions. A first task will be to split known algorithms for these problems into consistent sub-components that would be relevant by themselves for integration in SageMath.

1. Let $\mathbb{K}$ be a field, and let n and m be two integers with $n \ll m$. We are given an $m \times m$ matrix which brings a structure of $\mathbb{K}[x]$-module to $\mathbb{K}^m$, and we are given a $\mathbb{K}[x]$-linear map $f : \mathbb{K}[x]^n \to \mathbb{K}^m$, represented by an $m \times n$ matrix over $\mathbb{K}$. From this data, compute a representation of the kernel of f as a $\mathbb{K}[x]$-module; and similarly for the co-kernel of f; and compute associated invariant factors. This body of problems is discussed in the book (Kailath, Linear Systems, 1980), and recent algorithms allow one to exploit fast matrix multiplication via ideas from Keller-Gehrig's characteristic polynomial algorithm.
2. Given a square matrix over a ring of univariate polynomials, compute its characteristic polynomial following the block-Krylov method described in Kaltofen and Villard's determinant algorithm.

Preparation

To prepare for this project, I have begun reading up the classical mathematical theory on Krylov subspaces, minimal polynomials of a vector, and the rational canonical form. On the side of algorithms, I am getting familiar with Keller-Gehrig's computation of the characteristic polynomial, as well as with Wiedemann's algorithm for the minimal polynomial of a sparse matrix.

Although not mentioned in the project description provided by SageMath, Wiedemann's algorithm seems highly relevant to me in this context, since it precisely works with some Krylov subspace and allows one to find the minimal polynomial of a matrix, which is related to the characteristic polynomial of a matrix. Furthermore, the block-Wiedemann algorithm is a block-Krylov version of it, and is cited as an important related work in the paper by Kaltofen and Villard. What is less clear to me at this point, but which I intend to clarify by the start of the project, is that in other works the block-Wiedemann algorithm is mostly cited for dealing with sparse matrices, whereas here in this project and in Kaltofen and Villard's paper it seems that one is interested in “usual” (not necessarily sparse) matrices.

Identified key points of the implementation work

The above-described project seems sufficiently detailed, in terms of the broad aim about adding two related sets of functionalities. However, as soon as one wishes to concretely implement these functionalities, one has to consider several additional aspects, especially if one wants the additions to SageMath to be as versatile and useful as possible and at the same time to reach a good level of performance.

In this regard, here are some software implementation details I can carve out of the project description, and from my readings related to this project:

- *Identifying / optimising linear algebra subroutines.* The algorithms involved in this project are based on a small number of subcomponents, which can serve as basic building blocks. Their performance will be decisive for the overall performance, so they must be optimised and called whenever possible:

  - matrix-vector product Mv for $M \in \mathbb{K}^{m\times m}$ and $v \in \mathbb{K}^{m}$, which is e.g. a core component of the Wiedemann algorithm;
  - matrix-matrix product MN for $M, N \in \mathbb{K}^{m\times m}$, which is e.g. a core component of the Keller-Gehrig algorithm;
  - matrix-"few vectors" product MV for $M \in \mathbb{K}^{m\times m}$ and $V \in \mathbb{K}^{m\times k}$ for k typically small (relative to m), which is e.g. a core component of block-Krylov methods;
  - Gaussian elimination to compute the row rank profile, or nullspace, or LU decomposition of some rectangular matrix $M \in \mathbb{K}^{m\times n}$.

  Among these operations, the last one is the most diverse and broad, and it will be important to specify more clearly the few particular sub-instances of it that arise in the developed algorithms.

- *Exploiting fast linear algebra implementations.* In many places the code will contain operations that boil down to a combination of the above operations. Therefore, for best efficiency, the code for these subcomponents (and for possible other basic linear algebra operations) should rely on existing SageMath methods, that call optimised linear algebra routines. When getting familiar with SageMath's matrices, I have observed that it calls different libraries depending on the field $\mathbb{K}$, for example FFLAS-FFPACK/LinBox for a prime field $\mathbb{K} = \mathbb{Z}/p\mathbb{Z}$ with $p < 2^{26}$, and FLINT for the rationals $\mathbb{K} = \mathbb{Q}$. It will be crucial to make sure that the produced code relies on these specialised, highly optimised libraries whenever available, instead of SageMath's generic matrix code for an abstract field $\mathbb{K}$ which is much slower.

- *Accelerating polynomial matrix operations.* This project is related to operations on matrices over univariate polynomials. This is transparent in the second item in the project, but this may also be the case for the first item, as I found out when reading about the block-Wiedemann approach (its first step involves matrix-"few vectors" products, whereas its second step involves polynomial matrices). While working on the pull request #39852 for SageMath, I discovered that SageMath already incorporates many functionalities for such matrices, but at the same time that these are basic python implementations which do not seem to rely on efficient linear algebra routines (cf. the point just above) and do not always exploit fast polynomial operations. At least, I compared the speed of polynomial matrix multiplication, which is one of the most basic operations, between SageMath and FLINT (function `nmod_poly_mat_mul`), and it occurred that FLINT is faster than SageMath by orders of magnitude. Since there are already strong ties between FLINT and SageMath, could they be extended to provide better performance for these matrices? Or should I consider turning the existing SageMath polynomial matrix code into Cython? These are questions that I think should be studied during the project. The motivation is that any improvement on the performance of these computations would automatically accelerate the functionalities targeted by this project.

Rough Timeline

Week	Content
Week 1	Study the fundamentals of Krylov subspaces, $\mathbb{K}[x]$-modules, and minimal/characteristic polynomials. Study Wiedemann's and Keller-Gehrig's algorithms in detail, with a focus on how their structures relate to SageMath's existing framework. Identify appropriate invariants for addition to SageMath.
Week 2	Look in depth at core methods needed to implement Krylov invariants. Choosing the most efficient method for each application, begin implementation of kernels, co-kernels, invariants and doctests.
Week 3	Continue implementation from previous week. Verify performance with bench tests. Start work on implementing characteristic polynomial computation via Keller-Gehrig's algorithm, choosing appropriate methods.
Week 4	Continue implementation of Keller-Gehrig algorithm. Add doctests and measure performance across various base fields.
Week 5	Study and implement the block-Krylov method (Kaltofen–Villard) for computing characteristic polynomials over polynomial rings.
Week 6-7	Testing, bug fixing and performance optimisation. Expand on thoroughness of doctests, fix any bugs if they appear and optimise performance (computation time and memory usage).
Week 8	Final polish and clean-up. Ensure code and documentation meet SageMath's contribution standards.

Risk Management

Early Finish

In the case that the work is completed sooner than expected, it will be augmented with faster algorithms specialised for specific cases or related to specific applications, such as when dealing with structured matrices (Karpman, Pernet, Signargout, Villard, *Computing the Characteristic Polynomial of Generic Toeplitz-like and Hankel-like Matrices*, Proceedings ISSAC 2021).

Delays in Implementation

I will regularly communicate with my mentors to avoid delays as best as possible. In the case that there are significant delays with regards to the initial plans due to some unexpected difficulty, I will talk with my mentors to adapt the timeline to revise the goals, targeting ones that are achievable in the given time frame and still useful to SageMath.

Other Applications

I do not intend to apply to any other GSoC mentoring organisations.