Efficient linear algebra on GPUs for Gröbner bases computations

Master internship in Computer Algebra and High-Performance Computing

Équipes PEQUAN and PolSys, LIP6, Sorbonne Université, 4 place Jussieu, 75005 Paris, France

Environment

Advisors: Jérémy Berthomieu[1], Stef Graillat[2], Théo Mary[3], Mohab Safey El Din[4],

This internship will take place in LIP6, a joint lab between Sorbonne Université and CNRS in Paris. The intern will join two dynamical and scientifically ambitious teams which advise and co-advise Ph.D. students and postdoctoral researchers from France and many other countries (currently and recently: China, Germany, The Netherlands, Spain, UK, Ukrainia, USA, Vietnam, etc.). The intern will have access to office space, to all the necessary software, and to computing servers owned by the teams.

This internship is particularly appropriate for students willing to pursue a Ph.D. after obtaining their Master degree.

Context, scientific positioning

Modeling problems from biology, coding theory, combinatorics, robotics or aerospace engineering relies on fundamental problems such as solving polynomial systems $f_1 = \cdots = f_m = 0$ in variables $x_1, \ldots, x_n$ exactly over a finite field or rational numbers. Solving such a system comes down to computing a representation of its solution set. When this solution set has finite size D, we aim to compute a *lexicographic Gröbner basis* of the ideal spanned by $f_1, \ldots, f_m$. In the fashion of Gaussian elimination it returns a triangular system, which generically is of the form $g_n(x_n) = 0, x_{n-1} = g_{n-1}(x_n), \ldots, x_1 = g_1(x_n)$, $\deg g_n = \mathrm{D}$. Gröbner bases are a powerful tool but their computation is in general exponential in n, though some Gröbner bases are easier to compute than others. Indeed, the traditional strategy to obtain a lexicographic Gröbner basis is in two steps. First, compute a Gröbner basis of the ideal, for a *total degree* ordering using Buchberger's (1976) or Faugère's F_4 (1999) and F_5 (2002) algorithms. Then, apply a change of ordering algorithm on it using the FGLM algorithm (Faugère et al. 1993) or its faster variant, the Sparse-FGLM algorithm (Faugère and Mou 2011, 2017). In the generic case, this step yields $g_n, \ldots, g_1$ in essentially $\mathrm{O}(\mathrm{D}^2)$ operations. This latter algorithm guesses recurrence relations of a sequence through dedicated algorithms (Berthomieu et al. 2015, 2017; Berthomieu and Faugère 2018, 2022; Brent et al. 1980; Hyun et al. 2020). Let us notice that all these algorithms rely heavily on linear algebra routines on structured matrices. Still, their complexities are not satisfactory, mostly because the structure of the matrices does not seem to be sufficiently exploited.

Furthermore, to bypass the intrinsic numerical issues generated by the non-linearity of our problem, we use mainly exact arithmetic to compute over rational numbers or in finite fields.

Exact arithmetic historically uses modular arithmetic with integer types. With the rise of Central Processing Units (CPUs) vector extensions and their performances over floating-pointer numbers compared to integers, a new line of research was developed in order to take advantage of these instructions. The idea is to perform modular arithmetic over a 26-bit prime number with double-precision floating-point number (Hoeven et al. 2016; Jean and Graillat 2010). Let us notice that these missing 5 bit matters. Indeed, to reach the same precision over rationals with 26-bit primes as with 31-bit primes, we need to perform 20% more computations. Thus, this can annihilate the gain from using floating-point arithmetic and we shall pay a close attention to this.

Internship Objectives

The main objective of this internship is the conception of linear algebra algorithms for modular integers on a GPU using a floating-point or integer types representation targetting an integration in the `C` library `msolve` for solving exactly polynomial system (Berthomieu et al. 2021a,b). Furthermore, GPUs have also been proven to be efficient for correctly rounded function evaluation (Fortin et al. 2016), reinforcing the interest in using floating-point arithmetic to

[1]Sorbonne Université, jeremy.berthomieu@lip6.fr, https://www-polsys.lip6.fr/~berthomieu
[2]Sorbonne Université, stef.graillat@sorbonne-universite.fr, https://www-pequan.lip6.fr/~graillat
[3]CNRS, theo.mary@lip6.fr, https://www-pequan.lip6.fr/~tmary
[4]Sorbonne Université, mohab.safey@lip6.fr, https://www-polsys.lip6.fr/~safey

simulate a modular one. A first objective will be a thorough study of the most efficient native types to use depending on the size of the primes and that of target the rational numbers obtained after reconstruction.

The F_4 algorithm builds Macaulay matrices, a special kind of structured matrices and sparse matrices. Their rows represent generators, or multiples thereof, of the ideals and their columns monomials. Thus, these rows are not completely dense. Furthermore, since several rows are given as multiples of a common polynomial, they share all their coefficients. Yet, these coefficients are dispatched pretty differently. This makes a dense representation far from optimal but a sparse one, a priori interesting. A second objective of this internship will be the study of a memory representation of these matrices which is suitable for our computations but also not the limited GPU memory. A starting point will be to study the existing algorithms on a GPU (Blanchard et al. 2020; Lopez and Mary 2020) to derive new ones that are more fitting to our structure.

The sequence used in the Sparse-FGLM algorithm is built through a Krylov subspace method by picking at random a vector v_0 and computing $v_1 = Mv_0, v_2 = Mv_1, \ldots$, with M a $D \times D$ matrix with a special structure: mainly either a column is dense, or it is a column of the identity matrix. A first naive implementation on a GPU of matrix-vector products already gave us a speedup with a factor 24 of this costly step compared to our efficient CPU code. It is very promising and another objective is an optimized dedicated GPU implementation in order to greater factor such as 100.

Requirements

In order to carry out this project, the student is expected to have good knowledge in high-performance computing, linear algebra and `C` implementation.

References

J. Berthomieu, B. Boyer, and J.-C. Faugère (2015), "Linear Algebra for Computing Gröbner Bases of Linear Recursive Multidimensional Sequences", in *Proceedings ISSAC '15*, Bath, United Kingdom: ACM, pp. 61–68, DOI: 10.1145/2755996.2756673.

— (2017), "Linear Algebra for Computing Gröbner Bases of Linear Recursive Multidimensional Sequences", in *Journal of Symbolic Computation* 83, pp. 36–67, DOI: 10.1016/j.jsc.2016.11.005.

J. Berthomieu, C. Eder, and M. Safey El Din (2021a), *msolve: A Library for Solving Polynomial Systems*, https://msolve.lip6.fr/.

J. Berthomieu, C. Eder, and M. Safey El Din (2021b), "Msolve: A Library for Solving Polynomial Systems", in *Proceedings of the 2021 on International Symposium on Symbolic and Algebraic Computation*, ISSAC '21, Virtual Event, Russian Federation: Association for Computing Machinery, pp. 51–58, DOI: 10/gk8549.

J. Berthomieu and J.-C. Faugère (2018), "A Polynomial-Division-Based Algorithm for Computing Linear Recurrence Relations", in *Proceedings ISSAC '18*, New York, NY, USA: ACM, pp. 79–86, DOI: 10.1145/3208976.3209017.

— (2022), "Polynomial-division-based algorithms for computing linear recurrence relations", in *Journal of Symbolic Computation* 109, pp. 1–30, DOI: https://doi.org/10.1016/j.jsc.2021.07.002.

P. Blanchard, N. J. Higham, F. Lopez, T. Mary, and S. Pranesh (2020), "Mixed Precision Block Fused Multiply-Add: Error Analysis and Application to GPU Tensor Cores", in *SIAM Journal on Scientific Computing* 42.3, pp. C124–C141, DOI: 10.1137/19M1289546, eprint: https://doi.org/10.1137/19M1289546.

R. P. Brent, F. G. Gustavson, and D. Y. Yun (1980), "Fast solution of Toeplitz systems of equations and computation of Padé approximants", in *Journal of Algorithms* 1.3, pp. 259–295, DOI: https://doi.org/10.1016/0196-6774(80)90013-9.

B. Buchberger (1976), "A Theoretical Basis for the Reduction of Polynomials to Canonical Forms", in *SIGSAM Bull.* 10.3, pp. 19–29, DOI: 10.1145/1088216.1088219.

J.-C. Faugère (1999), "A New Efficient Algorithm for Computing Gröbner bases (F4)", in *Journal of Pure and Applied Algebra* 139.1, pp. 61–88, DOI: https://doi.org/10.1016/S0022-4049(99)00005-5.

— (2002), "A New Efficient Algorithm for Computing Gröbner Bases Without Reduction to Zero (F5)", in *Proceedings ISSAC '02*.

J.-C. Faugère, P. Gianni, D. Lazard, and T. Mora (1993), "Efficient Computation of Zero-dimensional Gröbner Bases by Change of Ordering", in *J. Symbolic Comput.* 16.4, pp. 329–344, DOI: http://dx.doi.org/10.1006/jsco.1993.1051.

J.-C. Faugère and C. Mou (2011), "Fast Algorithm for Change of Ordering of Zero-dimensional Gröbner Bases with Sparse Multiplication Matrices", in *Proceedings of the 36th International Symposium on Symbolic and Algebraic Computation*, ISSAC '11, San Jose, California, USA: ACM, pp. 115–122, DOI: 10.1145/1993886.1993908.

— (2017), "Sparse FGLM algorithms", in *Journal of Symbolic Computation* 80.3, pp. 538–569.

P. Fortin, M. Gouicem, and S. Graillat (2016), "GPU-Accelerated Generation of Correctly Rounded Elementary Functions", in *ACM Trans. Math. Softw.* 43.3, DOI: 10.1145/2935746.

J. van der Hoeven, G. Lecerf, and G. Quintin (2016), "Modular SIMD Arithmetic in Mathemagix", in *ACM Trans. Math. Softw.* 43.1, DOI: 10.1145/2876503.

S. G. Hyun, V. Neiger, H. Rahkooy, and É. Schost (2020), "Block-Krylov techniques in the context of sparse-FGLM algorithms", in *Journal of Symbolic Computation* 98, Special Issue on Symb. and Alg. Comp.: ISSAC 2017, pp. 163–191, DOI: 10/gkx9.

J. Jean and S. Graillat (2010), "A Parallel Algorithm for Dot Product over Word-Size Finite Field Using Floating-Point Arithmetic", in *2010 12th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*, pp. 80–87, DOI: 10.1109/SYNASC.2010.10.

F. Lopez and T. Mary (2020), "Mixed Precision LU Factorization on GPU Tensor Cores: Reducing Data Movement and Memory Footprint", working paper or preprint.