Form 101 Part II: Research Proposal

Synopsis

Provide a concise overview of the scientific and technical objectives, approach, and the new knowledge, expertise, or technology that could be transferred to Canadian industry. Indicate the benefits expected to accrue to Canadian industry, to the academic institution, and to the scientific or engineering discipline

The industrial partner for this project is Waterloo Maple Inc., a Canadian company, whose main commercial product is Maple. Maple is a mathematical software system which contains facilities for exact algebraic computation and numerical computation. It is used by scientists and engineers in many disciplines in Canada, the US, and worldwide, for research and development.

Maple's main algebraic capability is polynomial computation over various rings, most importantly, the integers, finite fields, algebraic number fields, and function fields. Most user level application facilities in Maple, such as solving systems of algebraic and differential equations, make extensive use of polynomial operations. This includes basic polynomial arithmetic, polynomial greatest common divisors (GCDs), polynomial factorization, polynomial resultants, and Gröbner basis computation. However, except for $\mathbb{Z}_n[x]$, Maple does not have any dedicated polynomial data structures and it is not possible to implement many of the best algorithms, in particular the modular algorithms, efficiently in the Maple language because it is interpreted. Consequently, Maple's competitiveness is falling behind that of other commercial systems, for example, Mathematica and Magma. Moreover, Maple is becoming less helpful as a platform for the development of new algorithms.

The main objectives of this project are (i) to design and implement a good data structure for polynomials over the rationals, finite fields and algebraic number fields, that will enable an efficient implementation of modular algorithms, (ii) to implement the best modular algorithms for the critical polynomial operations above so that Maple's core algebraic facilities are the best available for general purpose computation, and (iii) to develop and implement new efficient modular algorithms for polynomial operations over algebraic number and function fields.

Because the focus of this proposal is the "core" algebraic facilities of Maple, this project will be of immediate benefit to most users of Maple and it will provide a solid foundation for future research and development. The research contribution to the field of computational algebra will be the development of efficient algorithms for computing with polynomials over algebraic number and function fields and improved reconstruction techniques.

Background

Relate the proposal to current scientific, technical and commercial developments in the field, referring to the current literature and market conditions. Describe the background research on which the project is built.

The three main problems addressed in this project proposal are the polynomial GCD problem, the polynomial factorization problem and the Gröbner basis computation problem. The scientific literature for each is quite extensive hence our summary is necessarily selective. The texts of Geddes, Czapor, Labahn, [1], Zippel [2] and von zur Gathen and Gerhard [3], contain descriptions of algorithms for the polynomial GCD and factorization problems over $\mathbb{Q}$ and GF(q), and some material on computing Gröbner bases. The Becker, Weisspfenning text [4] contains a deeper treatment of Gröbner basis computation.

1: The polynomial GCD problem.

The first effective algorithm for multivariate polynomial GCD computation over $\mathbb{Z}$ was Brown's algorithm [5]. Subsequent research focussed on improving the time complexity of Brown's algorithm for *sparse* polynomials. Results include Wang's EEZ-GCD algorithm [6], Zippel's sparse modular GCD algorithm [7] and Kaltofen and Trager's black box GCD algorithm [8]. The heuristic algorithm GCDHEU of Char, Geddes, Gonnet in [9] provides an alternative approach in which a polynomial GCD problem is reduced to a single large integer GCD computation. All general purpose computer algebra systems are using some combination of these algorithms.

For GCDs over finite fields, the main difficulty is how to deal with small fields where there may be insufficient evaluation points for the interpolation based algorithms. An obvious solution is to work in an extension field of suitably large size. Kaltofen and Monagan [10] show how to do this more efficiently. This is implemented in Maple for GF$(p)[x, y]$ only.

For GCDs over number fields, based on previous work of Langemyr and McCallum [11], Encarnacion shows in [12] how to compute univariate GCDs over a number field presented with one field extension. Encarnacion's algorithm uses *rational reconstruction*, a technique developed by Wang, Guy and Davenport in [13] which has turned out to have wide application. Encarnacion's algorithm is implemented in Axiom, Maple, and Magma. Maple also uses the heuristic algorithm of Gonnet et. al. [15] for one field extension which is implemented for multivariate polynomials. In [16], van Hoeij and Monagan show how to compute GCDs over number fields presented with multiple field extensions. We have completed a multivariate implementation of this algorithm in the Maple language in 2001 and in the Magma language in 2003. Both implementations use a recursive dense polynomial data structure to facilitate an efficient implementation.

The case of polynomial GCDs over algebraic function fields has not, as far as we are aware of, been investigated in the literature. Maple is using the Euclidean algorithm and as a consequence, it grinds to a halt on polynomials of even modest degree. It is clear though that many of the techniques used for the algebraic number field case and multivariate polynomial GCDs over $\mathbb{Z}$ will apply.

2: The polynomial Factorization problem.

The polynomial factorization problem can be divided into four main cases, namely, univariate factorization over (i) finite fields, (ii) the integers, (iii) number fields, and (iv) multivariate factorization over the preceding cases. The first polynomial time algorithm for (i) was given by Berlekamp in [17] and for (ii) was given by Lenstra, Lenstra, and Lovasz, in [18]. Despite the latter result, all computer algebra systems have continued to use the Berlekamp-Hensel procedure for (ii) which, though not polynomial time, has a much better average case performance. The non-polynomial step in the Berlekamp-Hensel procedure was recently eliminated by van Hoeij in [19]. This improvement is already implemented in Maple and Magma. As a consequence Trager's algorithm in [20] for (iii), which previously was not polynomial time (even in the average case), is now also polynomial time.

Multivariate factorization is polynomial time reducible to the univariate case for (ii) and (iii) and to the bivariate case for (i). Several polynomial time factorization algorithms for $\mathrm{GF}(q)[x, y]$ are known though they are not implemented in Axiom, Maple, Magma, and Mathematica. We mention those of von zur Gathen and Kaltofen in [21] and Noro and Yokoyama in [22].

3: The Gröbner basis computation problem.

We do not mention here the research done in parallel Gröbner basis computation and Gröbner basis computation over non-standard rings because our focus is to build a solid algorithm for the most important cases in practice which can be plugged into Maple.

Many of the algorithmic developments in Gröbner basis computation have been modifications of Buchberger's original algorithm to reduce the number of unnecessary S-polynomials that are processed. We mention Buchberger's "normal" selection strategy [23], the "sugar" selection strategy of Traverso et. al. [24], and Faugere's new "F5" algorithm [25]. We also mention two results of important practical significance. The first is Faugere's F4 algorithm [26], which exploits sparse linear algebra to allow multiple pairs to be processed simultaneously. The second is the "FGLM" algorithm of [27], which allows one to rapidly convert a Gröbner basis for a zero dimensional ideal from one term ordering to another.

A number of special purpose systems have been developed whose primary computational facility is Gröbner basis computation. We mention Macaulay, Singular, CoCoa, and GB, each of which represents a substantial research project. Here we only give some details about the Gröbner basis implementations in Maple, Mathematica, and Magma. Maple's implementation is using the the sugar selection strategy. Mathematica's and Magma's are automatically using FGLM. Magma is also using the F4 algorithm. Maple's implementation compares poorly with Mathematica's and very poorly with Magma's and the other special purpose systems mentioned above. One reason for this is that the general purpose data structure that Maple is using to represent expressions is not suitable for implementing Buchberger's algorithm. In contrast, all other systems mentioned are using dedicated polynomial data structures.

Modular methods can also be applied to Gröbner basis computation. The main theoretical difficulty is that we do not yet have an efficient termination test. In [28] Arnold has provided some progress towards such a test. In de Kleine and Monagan [29] we have developed a modular implementation using the sugar selection strategy and an appropriate polynomial data structure. Our algorithm is probabilistic; we stop when we have five 26 bit primes (40 decimal digits) of agreement. The implementation computes modulo b primes simultaneously, in order to amortize the cost of the monomial arithmetic across several primes and it uses IEEE double precision floating point arithmetic to speed up computation in $\mathbb{Z}_p$.

Detailed Proposal

Discuss the scientific issues, research problems or technical complexities, and describe the research methodology and experimental design proposed to explain or resolve them. Provide a workplan and relate it to the milestone schedule.

The detailed proposal is divided into four sub-projects.

1: Polynomial data structure sub-project.

The first sub-project is to build a data structure for polynomials in one or more variables over the integers, finite fields, and algebraic number fields, and secondly to implement a good modular algorithm for the polynomial GCD operation for all cases, including both sparse and dense polynomials. This sub-project will be undertaken jointly with Waterloo Maple personnel. Our goal is to complete it in the first year of the project.

In [16] we experimented with a recursive data structure for implementing GCDs over algebraic number fields and finite fields which we plan to adopt. The recursive dense data structure makes the implementation efficient and facilitates operations on field extensions. Our approach will be to implement polynomial arithmetic and support operations, such as polynomial evaluation and interpolation, in the C language for efficiency, but to write most cases of the GCD algorithm in Maple for flexibility. This implementation approach was effective for $\mathbb{Z}[x]$ where the facility developed by Monagan in [30] for $\mathbb{Z}_n[x]$ enabled us to implement modular algorithms for polynomial GCD, resultant and factorization in $\mathbb{Z}[x]$ efficiently in Maple.

In ongoing work, de Kleine and Monagan have just completed an implementation of Zippel's sparse modular GCD algorithm for multivariate GCDs over number fields and finite fields for the monic case. We propose to investigate different approaches for handling the non-monic case, including developing a sparse rational function reconstruction technique.

2: Polynomial factorization sub-project.

The second sub-project is to implement known algorithms for polynomial factorization over the integers, finite fields and algebraic number fields using the new data structure. We will begin this sub-project in the second year, after the polynomial data structure is implemented, and work on it through the end of the project. Recent asymptotic improvements in polynomial factorization and absolute factorization give us many opportunities to make improvements on existing implementations. As such, this sub-project will provide excellent training opportunities for graduate students.

For example, we mention Kaltofen and Shoup's subquadratic time algorithm [32] for factorization in $\mathrm{GF}(q)[x]$ and the practical recommendations made therein. Another place for improvement is factorization in $\mathrm{GF}(q)[x, y]$. The Maple implementation of Bernardin in [33] uses Hensel lifting to find the factors modulo $m(y)^k$. This is not polynomial time in the worst case because of a final combinatorial step. We propose to modify the solution of van Hoeij [19] for $\mathbb{Z}[x]$ to work in $\mathrm{GF}(q)[y][x]$. This should yield an asymptotically faster algorithm than that of Noro and Yokoyama in [22].

3: Gröbner basis sub-project.

The third sub-project is to develop an efficient data structure and implementation for computing Gröbner bases of ideals in $F[x_1, ..., x_n]$ using modular methods. In [29], we implemented a modular algorithm for $F = \mathbb{Q}$ which exploits fast floating point arithmetic. In the first year we will extend our implementation to include Faugere's F4 algorithm, the FGLM algorithm [27], and to integrate it into the Maple library. For the latter two years of the project we propose the following:

(i) To allow for parameters $a, b, c, ...$ in F we will use evaluation and rational function reconstruction. When there is more than one parameter, we will develop a sparse rational reconstruction technique to improve the performance in the sparse case. The technical difficulties here are dealing with the many possible failures that can occur so that we can guarantee termination of the algorithm.

(ii) To investigate the application of a modular Gröbner basis algorithm to the polynomial GCD problem. Let F be a field, possibly with parameters. Let $f_1, f_2 \in F[x_1, x_2, ..., x_n]$ and let $g = \mathrm{GCD}(f_1, f_2)$. The following approach to computing g is a refinement of an idea given to us by Gianni and Trager. Let $\alpha_1, ..., \alpha_n \in F$ satisfy $\mathrm{lc}_{x_1}(f_1)(\alpha_2, ..., \alpha_n) \neq 0$ and $\mathrm{lc}_{x_1}(f_2)(\alpha_2, ..., \alpha_n) \neq 0$. Then a multiple of the the primitive part of g will appear in a Gröbner basis for the ideal $I = \langle f_1, f_2, (x_2-\alpha_2)^{1+b_2}, ..., (x_n-\alpha_n)^{1+b_n} \rangle$ provided $b_i \geq \deg_{x_i}(g)$. Now since we can bound $\deg_{x_i}(g)$ quickly and accurately with one univariate GCD in $F[x_i]$ and, assuming we have a good *modular* Gröbner basis implementation for F, this approach to compute g may prove competitive. An advantage is that it is easy to extend it to treat algebraic numbers and algebraic functions.

4: Algebraic function fields sub-project.

The fourth sub-project is to develop a modular algorithm for computing a polynomial GCD over an algebraic function field, and to make a prototype implementation in Maple. This is a necessary first step for the development of an efficient algorithm for factorization over algebraic function fields.

Let $m_t(z) \in \mathbb{Q}(t)[z]$ be the minimal polynomial for A, an algebraic function. For example, $m_t(z) = z^2 - t$ is the minimal polynomial for $A = \sqrt{t}$. Given $f_1, f_2 \in F[x, y]$ where $F = \mathbb{Q}(t, A)$, the problem is to compute the monic GCD of f_1 and f_2. We also need to consider the case where the algebraic function depends on more than one parameter, for example $A = \sqrt{a/b - c^2}$. Such problems arise naturally in many science and engineering applications. If there are many parameters, we will again need to develop a sparse reconstruction technique. We propose to develop and prototype an algorithm for one field extension with multiple parameters in the first year and to develop the polynomial data structure from the first sub-project to support also (multiple) function field extensions in the latter two years.

Remark on rational reconstruction techniques: In current work we have improved on the rational number reconstruction algorithm of Wang, Guy and Davenport [6] so that when reconstructing a rational number n/d from it's image u modulo m, the size of the modulus need only be a few bits longer than the size of nd, irrespective of whether the n is larger then d

or d is larger then n. This means that we need fewer primes in general to reconstruct the n/d. The idea should be applicable to rational function reconstruction case. An asymptotically fast algorithm also seems possible. In [14] Pan described an asymptotically fast algorithm for rational number reconstruction but remarks that the algorithm may not be practical. However, Steel, in unpublished work, had already implemented a simpler asymptotically fast rational number reconstruction algorithm in Magma and believes our algorithm can similarly be accelerated.

Remark about the data structure approach: The computer algebra systems Axiom and Magma, and our own experimental system Gauss [31] provide generic facilities for building polynomials and rational functions over *any* field whereas the data structure approach we are taking in this project restricts the field to the cases that we decide to support. Surely the generic approach is the better? For example, in Magma, one can build the ring $F[x, y]$ mentioned above as follows.

```
> Q := RationalField();
> K<t> := FunctionField(Q);     // Q(t)
> P[z] := PolynomialRing(K);    // Q(t)[z]
> m := z^2-t;
> F<A> := quo<P|m>;             // F = Q(t,A)
> S<y> := PolynomialRing(F);    // F[y]
> R<x> := PolynomialRing(S);    // F[y][x]
```

The advantage is that an efficient representation, at least for basic polynomial arithmetic, is immediately available for any coefficient ring, in this example for Q, F, K and also S. Having implemented modular algorithms using both approaches it is our experience that a special purpose data structure will provide us with enough opportunities for greater efficiency that it is worthwhile having such a data structure for the common coefficient fields.

Team Expertise

Explain how the knowledge and experience of each team member relates to the expertise needed to accomplish the project objectives, and how the contributions of the team members will be integrated.

The research area of each team member below is computer algebra. Dr. Michael Monagan, the principal investigator, has expertise in polynomial GCD computation and the other areas of this proposal. He has also worked on the Axiom and Magma projects. Dr. Laurent Bernardin is head of the Mathematics group at Waterloo Maple. His Ph.D. was on polynomial factorization over finite fields and he will contribute to sub-projects 1 and 2. Dr. Mark van Hoeij of Florida State University, whose expertise in polynomial factorization and algebraic functions, will be contributing to sub-projects 2 and 4. Mrs. Jennifer de Kleine, who has recently graduated with a Master's in computer algebra, will be paid under the project to work on sub-projects 1 and 3.

Training of highly qualified personnel

Describe the opportunities the project offers for advanced training, or other relevant experience, for students, post doctoral fellows, or supporting organizations' R & D staff.

There are many good and difficult research and development problems to work on within the scope of this proposal. For the graduate students, company personnel, and other research personnel, this project will provide excellent training in computational mathematics with a good balance between theoretical algorithmic development and practical implementation. It involves writing software for polynomial computations over different rings from scratch. Few people will ever have the opportunity to do this for a system that will be widely used.

Intellectual property

Discuss the plans for protecting and disposing of intellectual property arising from the grant. Outline the broad terms of the agreement between the company and the university on the rights to exploit the technology being transferred.

Waterloo Maple has established an intellectual property agreement with another research project between Waterloo Maple and Simon Fraser University that we intend to adopt. That agreement gives Waterloo Maple ownership rights to own and use any code (and documentation) developed by the researchers. In the case of our project, this is in our interest as we do want our software to be made available to the Maple user community worldwide. The best way to facilitate that is to have it integrated into Maple rather than reside in an applications library. The agreement gives graduate students and research personnel the right to publish all scientific results without delay.

Value of the results

Describe the anticipated value of the project results, highlighting the relevance of the scientific or technical advances, or the innovative techniques, processes or products that will be developed. Show how the outcome will address a current or future industrial or market need.

The main expected results of this project:

- New and improved modular algorithms for polynomial GCD computation.
- New and improved rational reconstruction algorithms.
- A polynomial data structure for Maple for multivariate polynomials over the integers, finite fields, and number fields, and modular algorithms for polynomial GCDs and factorization will improve Maple's performance across a range of applications. In particular, the improved support for finite fields will open up new applications of Maple in coding theory and cryptography.
- A software module for Maple which supports Gröbner basis computation over the rationals (with parameters) using modular methods will make Maple competitive at an important computation with many applications.

This is an ambitious project and I am excited about it. The project will help make Maple the best general purpose system for algebraic computation and, important for the research community, give Maple a strong foundation for future development.

References

[1] K. Geddes, C. Czapor, and G. Labahn, *Algorithms for Computer Algebra*, Kluwer Academic, 1996.

[2] R. Zippel, *Effective Polynomial Computation*, Kluwer Academic, 1993.

[3] J. von zur Gathen and J. Gerhard, *Modern Computer Algebra.* U. of Cambridge Press, 1999.

[4] T. Becker and V. Weisspfenning, *Gröbner bases – A Computational Approach to Commutative Algebra.* Springer-Verlag, New York, 1993.

[5] W. S. Brown, On Euclid's Algorithm and the Computation of Polynomial Greatest Common Divisors, *J. ACM* **18** (1971), pp. 476–504.

[6] P. Wang, The EEZ-GCD Algorithm. *SIGSAM Bulletin*, **14** (2) pp. 50–60, 1980.

[7] R. Zippel, Probabilistic algorithms for sparse polynomials, *Proc. of EUROSAM '79*, Springer-Verlag LNCS, **2** (1979), pp. 216–226.

[8] E. Kaltofen, B. Trager, Computing with polynomials given by black boxes for their evaluations: Greatest common divisors, factorization, separation of numerators and denominators. *J. Symb. Comp.*, **9**, pp. 301–320, 1990.

[9] B. Char, K. Geddes, G. Gonnet, Gcdheu: Heuristic polynomial GCD algorithm based on an integer GCD computation. *J. Symb. Comp.*, **7**, pp. 31–48, 1989.

[10] E. Kaltofen, M.B. Monagan, On the Genericity of the Modular Polynomial GCD Algorithm. *Proc. of ISSAC '99*, ACM Press, pp. 59–66, 1999.

[11] L. Langemyr, S. McCallum, The Computation of Polynomial GCD's over an Algebraic Number Field, *J. Symbolic Computation* **8** (1989), pp. 429–448.

[12] M. J. Encarnacion, Computing GCDs of Polynomials over Algebraic Number Fields, *J. Symbolic Computation* **20** (1995), pp. 299–313.

[13] P. Wang, M. J. T. Guy, J. H. Davenport, *p-adic Reconstruction of Rational Numbers*, in SIGSAM Bulletin, **16**, No 2 (1982).

[14] V. Pan, Acceleration of Euclidean Algorithm and Extensions. *Proc. of ISSAC '2002*, ACM Press, pp. 207–213, 2002.

[15] K. O. Geddes, G. H. Gonnet, T. J. Smedley, Heuristic Methods for Operations with Algebraic Numbers. *Proc. of ISSAC'88*, Springer-Verlag LNCS **358**, pp. 475–480, 1988.

[16] M. van Hoeij, M. B. Monagan, A Modular GCD Algorithm over Number Fields Presented with Multiple Field Extensions. *Proc. of ISSAC '2002*, ACM Press, pp. 109–116, 2002.

[17] E. Berlekamp, Factoring polynomials over finite fields, *Bell Systems Technical Journal*, **46**, pp. 1853–1859, 1967.

[18] A. Lenstra, H. Lenstra, L. Lovasz, Factoring Polynomials with Rational Coefficients, *Mathematische Annalen*, **261** pp. 515–534, 1982.

[19] M. van Hoeij, Factoring Polynomials and the Knapsack Problem. *J. Number Theory*, **95**, pp. 167–189, 2002.

[20] B. Trager, Algebraic Factoring and Rational Function Integration. *Proc. of ISSAC '76*, ACM Press, pp 219–226, 1976.

[21] J. von zur Gathen, and E. Kaltofen, Factorization of Multivariate Polynomials over Finite Fields. *Math. Comp.* **45** pp. 251–261, 1985.

[22] M. Noro, K. Yokoyama, Yet another practical implementation of Polynomial Factorization over Finite Fields. *Proc. of ISSAC '02*, ACM Press, pp. 200–206, 2002.

[23] B. Buchberger, *Groebner bases: an algorithmic method in polynomial ideal theory.* Multidimensional Systems Theory, edited by N. K. Bose, D. Reidel Publishing Company, Dordrecht, pp. 184-232, 1985.

[24] A. Giovini, T. Mora, G. Niesi, L. Robbiano, and C. Traverso, "One sugar cube, please" OR Selection strategies in the Buchberger algorithm. *Proc. of ISSAC '91*, ACM Press, pp. 41–29, 1991.

[25] J. Faugere, A New Efficient Algorithm for Computing Gröbner Bases Without Reduction to Zero (*F5*). *Proc. of ISSAC '02*, ACM Press, pp. 75–83, 2002.

[26] J. Faugere, A New Efficient Algorithm for Computing Gröbner Bases (*F4*). *J. Pure and Applied Algebra*, **139**, pp. 61–83, 1999.

[27] J. Faugere, P. Gianni, D. Lazard, and T. Mora, Efficient computation of zero-dimensional Gröbner bases by change of ordering. *J. Symb. Comp.*, **16** pp. 329–244, 1993.

[28] E. Arnold, Modular Methods for Computing Groebner Bases. ISSAC '01 Poster Session, 2001.

[29] J. de Kleine and M. B. Monagan, A Modular Design and Implementation of Buchberger's Algorithm. *Proc. of RWCA '02*, 2002.

[30] M. Monagan, In-place arithmetic for polynomials over $\mathbf{Z}_n$. *Proc. of DISCO '92*, Springer-Verlag LNCS, **721**, pp. 22–34, 1993.

[31] M. Monagan, Gauss: a Parameterized Domain of Computation System with Support for Signature Functions. *Proc. of DISCO '93*, Springer-Verlag LNCS, **722**, pp. 81–94, 1993.

[32] E. Kaltofen and V. Shoup, Subquadratic-Time Factoring of Polynomials over Finite Fields. *Math. Comp.* **67**, pp. 1179–1197, 1998.

[33] L. Bernardin, *Factorization of Multivariate Polynomials over Finite Fields*, Ph.D. Thesis, ETH Zürich, 1999.