Message from the SIGSAM Chair

Ilias S. Kotsireas
Wilfrid Laurier University
Waterloo, ON, Canada

Dear SIGSAM Members,

It is a great honor to serve as Chair of ACM SIGSAM, a historical organization with numerous and long-lasting contributions to Computer Algebra and Symbolic Computation. I would like to take this opportunity to thank all candidates at the recent elections. Everyone is a valuable member of the community and I would encourage them to take part in the next election as well and/or to continue to be actively involved in SIGSAM activities and projects. SIGSAM is a volunteer-based organization and therefore it is important to involve as many volunteers as possible. I would also like to welcome the three elected members of the new SIGSAM executive, namely:

- Vice Chair: Jean-Guillaume Dumas (Université Joseph Fourier, France), `VC_SIGSAM@acm.org`
- Secretary: Ziming Li (Institute of Systems Science, China), `Secretary_SIGSAM@acm.org`
- Treasurer: Agnes Szanto (North Carolina State University, USA), `Treasurer_SIGSAM@acm.org`

I am looking forward to working with all three of them. I am also looking forward to work with those of you that would be willing to take some time off your busy schedules to work on some projects that I plan to pursue. To be more specific, I plan to intensify work on the following projects:

- Update and re-organize the SIGSAM webpage `http://www.sigsam.org/`
- Design and implement a campaign to increase SIGSAM membership. This will include reviewing all benefits of SIGSAM membership and discussing ways to increase the value of membership for all of those interested in computer algebra and related fields
- Enlarge and maintain coverage of CCA in on-line publication databases, i.e. DBLP, MathSciNet

The annual SIGSAM business meeting was held during the ISSAC 2013 conference in Boston and it was reported that:

1. SIGSAM is financially healthy with a balance of more than $\$60,000$.
2. SIGSAM sponsorship of ISSAC paper prizes and Jenks award funded by endowed accounts which also remain healthy
3. Item 1 enabled SIGSAM to reduce the fee from ACM for ISSAC sponsorship from 16% to 10% which was the same as when ISSAC was sponsored by INRIA at Grenoble.
4. ACM has revised its publishing policies. This includes authorizer which allows authors to post official copies of their ACM publications. It also allows ACM published proceedings to be available to everyone for one year. I.E. we can post the table of contents of ISSAC proceedings with links to all papers for one year on the ISSAC or SIGSAM website.
5. Past SIGSAM Chair, Prof. Jeremy Johnson, Drexel University, presented the SIGSAM Program Review on March 14, 2013. SIGSAM received renewed viability from ACM SGB for another 3 years. The official ACM statement is:
   SIGSAM
   The SGB EC congratulates SIGSAM on their continuing importance to the community, but has

concerns about submissions and attendance at the ISSAC conference and finds SIGSAM viable to continue its status for the next 3 years.

The preparations for the ISSAC 2014 conference are well underway, under the leadership of general co-chairs Kosaku Nagasaka (Kobe) and Franz Winkler (RISC-Linz). The conference will be held July 23–25 in Kobe, Japan. Please consult the website `http://www.issac-symposium.org/2014/` for more information. ISSAC 2014 is in cooperation with ACM. It is worthwhile to note that ISSAC 2014 is a satellite conference of the International Congress of Mathematicians (ICM 2014) that will be held August 13–21 in Seoul, Korea, `http://www.icm2014.org/` .

The venue for the ISSAC 2015 conference was selected during the business meeting at ISSAC 2013 in Boston. There were four candidate cities and the winning city was Bath, UK. The representative of one of the four candidate cities expressed concerns over the preparation and the logistics of the bidding presentations. I witnessed some aspects of the entire process first hand and I believe the concerns were justified and valid. In order to address these concerns, I believe that SIGSAM and the ISSAC steering committee should investigate the entire bidding process and possibly offer recommendations on how to systematize it and streamline it, so as to hopefully avoid this type of concerns at future bids for ISSAC conferences.

SIGSAM sponsored the following ISSAC 2013 awards:

- Distinguished Student Author Awards: Pierre Lairez for Creative Telescoping for Rational Functions Using the Griffiths-Dwork Method (with Alin Bostan and Bruno Salvy) and Qingdong Guo for Computing Rational Solutions of Linear Matrix Inequalities (with Mohab Safey El Din and Lihong Zhi).
- Distinguished Paper Award: Jingguo Bi, Qi Cheng, and J. Maurice Rojas. Sub-Linear Root Detection and New Hardness Results for Sparse Polynomials Over Finite Fields.
- Distinguished Poster Awards: Jeremy Johnson and Lingchuan Meng. Towards Parallel General-Size Library Generation for Polynomial Multiplication. James Wan. Hypergeometric Generating Functions and Series for $\frac{1}{\pi}$.
- Distinguished Software Presentation Award: Fredrik Johansson. Arb: a C Library for Ball Arithmetic.
- The 2013 Richard Dimick Jenks Memorial Prize for Excellence in Software Engineering applied to Computer Algebra was awarded to the William Stein for the Sage Project.

I would like to thank CCA editor Manuel Kauers (RISC-Linz, Austria) and Information Director William J. Turner (Wabash College, USA) along with the editorial board for making sure that the four yearly issues of CCA are published in a timely manner on-line and in the printed version. Timely publication of CCA is an important aspect of the periodic viability review for SIGSAM. I would also like to thank Clément Pernet (Université Jopseph Fourier, Grenoble, France) for serving as the SIGSAM representative on the Editorial Board of ACM Transactions on Mathematical Software (TOMS) `http://toms.acm.org/` .

Finally I would like to thank the extremely efficient ACM staff for their help and support during my past ten years as Editor of CCA and on other occasions such as conference organization. Special thanks go to Irene Frawley, Program Coordinator, SIG Activities, for her enthusiasm and dedication. I hope to continue to work with all of them and be productive in my new role as SIGSAM Chair.

Ilias S. Kotsireas
SIGSAM Chair `chair_SIGSAM@acm.org`
Wilfrid Laurier University
Waterloo, ON, Canada

ACM Communications in Computer Algebra, Vol. 47, No. 3, Issue 185, September 2013

Asymptotic series of Generalized Lambert W Function

Tony C. Scott[1,2], Greg Fee[3] and Johannes Grotendorst[4]

[1] Institut für Physikalische Chemie, RWTH Aachen University, 52056 Aachen, Germany
email: scott@pc.rwth-aachen.de

[2] Zephyr Health Inc, 589 Howard Street, 3rd floor, San Francisco CA 94105, USA
email: tony@zephyrhealthinc.com

[3] Centre for Experimental and Constructive Mathematics (CECM), Simon-Fraser University, Burnaby, BC V5A 1S6 Canada
email: gjfee@cecm.ca

[4] Institute for Advanced Simulation, Jülich Supercomputing Centre, Forschungszentrum Jülich, 52425 Jülich, Germany
email: j.grotendorst@fz-juelich.de

Abstract

Herein, we present a sequel to earlier work on a generalization of the Lambert W function. In particular, we examine series expansions of the generalized version providing computational means for evaluating this function in various regimes and further confirming the notion that this generalization is a natural extension of the standard Lambert W function.

AMS Numbers: 33E30, 01-01, 01-02
Also related to: 70B05, 81Q05, 83C47, 11A99

1 Introduction

The Lambert W function satisfying $W(t)e^{W(t)} = t$ provides an exact solution to:

$$e^{-cx} = a_o\ (x - r_1) \tag{1}$$

with $x = r_1 + \frac{1}{c}W(c\ e^{-cr_1}/a_o)$. The Lambert W function appears in a myriad number of applications. In particular, it appears in the "lineal" gravity two-body problem [1, 2] as a solution to the Einstein Field equations in $(1 + 1)$ dimensions. The Lambert W function appears as a solution for the case when the two-bodies have exactly the same mass. However, the case of unequal masses required a *Generalization* of Lambert's function [1, eq.(81)].

$$e^{-cx} = a_o\ (x - r_1)(x - r_2) \tag{2}$$

This generalization originally appeared from the (quantum-mechanical) Double Well Dirac Delta Potential model [3], a one-dimensional version of a special case of the quantum-mechanical three-body system known

as the *Hydrogen Molecular Ion* (and also appears in quantum gravity [2]). For this problem, specifically $r_1 = 1$, $r_2 = \lambda$, $c = 2\ R$ where R is the internuclear distance. $a_o = \frac{1}{\lambda}$ and λ was treated formally as real perturbative parameter (the case at $\lambda = 1$ allows eq. (2) to factor into (1) which is solvable in terms of the standard Lambert W function). In its original form, this equation was written in a more complicated form, namely a *pseudo-quadratic*: with two solutions for x [3–6]:

$$x_\pm(\lambda) \ = \ \tfrac{1}{2}(\lambda+1) \pm \tfrac{1}{2}\left\{(1+\lambda)^2 - 4\,\lambda[1 - e^{-c x_\pm(\lambda)}]\right\}^{1/2}$$

where $E_\pm = -x_\pm/2$ are the quantum state energies (for respectively the two distinct solutions $x_\pm$). All these quantities including the energies were real though we do not rule out a generalization to the complex plane.

A difficulty encountered by Byers-Brown and Scott *et al.* is that Physical Chemists followed a conventional practice of starting with the case $\lambda = 0$ whose solution is $x_0 = 1$ as a starting point and considering a series expansion about x_0 of eq. (1) in powers of λ. This was called the "polarization expansion" for the range $0 < \lambda < 1$ and proves very difficult to sum, necessitating the use of Padé-Hermite Approximants [3]. This slow convergence became aggravated for larger but similar molecular systems like the Hydrogen Molecular Ion requiring much discussion (and calculation) to sort out the convergence of the eigenstates and related quantities once and for all [7, 8].

Subsequently, it was realized that eq. (2) could be further generalized to the case of a rational polynomial [9]:

$$e^{-c\,x} \ = \ \frac{P_N(x)}{Q_M(x)} \tag{3}$$

where $c > 0$ is a constant as before and $P_N(x)$ and $Q_M(x)$ are polynomials in x of respectively orders N and M. Eq. (3) expresses the solution for the energy eigenvalues of the three-dimensional (and realistic) version of the Hydrogen molecular ion. These generalizations were found to express solutions to a huge class of fundamental problems and were found to be natural extensions of the standard W function requiring merely a formal nesting of the standard Lambert W function [10] and thus economical conceptually in terms of mathematical resources. Some exact solutions were even found for some special cases for eq. (2) [10].

Herein, we examine the more pragmatic matter of obtaining series expansions for eq. (2) for analytical and computational purposes. In the process, we will show how closely they relate to the series expansions of the standard W function. We will examine three series expansions which apply to three different regimes. Though eq. (2) is not the full generalization in eq (3) it already embodies a link between gravity theory and quantum mechanics albeit in lower dimensions [2] and is therefore instructive as a special case beyond the standard W function. Finally, some concluding remarks are made at the end. Since we are dealing with applications in Physics, the input parameters c, a_o and the polynomial roots r_i where $i = 1, 2, \ldots$ are assumed to be real.

2 Series Expansions

2.1 Taylor series in r_d

By a series of manipulations, eq. (1) can be brought in the familiar standard form:

$$x_0 \ = \ W(x_0)e^{W(x_0)} \quad \text{where} \quad x_0 \ = \ \frac{c\ e^{-c r_1}}{a_o} \tag{4}$$

Using very similar manipulations and defining respectively the mean and difference of the roots r_1 and r_2:

$$r_m \;=\; \frac{r_1 + r_2}{2} \quad \text{and} \quad r_d \;=\; \frac{r_1 - r_2}{2} \;, \tag{5}$$

and by *completing the square* for the quadratic on the right of eq. (2):

$$(x - r_1)(x - r_2) \;=\; (x - r_m)^2 - r_d^2$$

and defining $W(r_d) = x - r_m$, eq. (2) can be rewritten as:

$$e^{-c(W(r_d)+r_m)} \;+\; a_o\, r_d^2 \;=\; a_o\, W(r_d)^2 \;. \tag{6}$$

The above can be viewed as the intersection between an exponential of the form $Ae^{-c\,x}$ and a "simple harmonic oscillator" of the form Bx^2. Potentially, there can be two and as much as three intersections (in the real plane), in some cases, roots of the same sign. To obtain real solutions, we constrain $a_o > 0$. It is very similar to eq. (1) the equation governing the standard Lambert W function with the mean of the roots r_m playing the role of the r_1 in the monomial on the right side of eq. (1), the difference in the roots r_d representing a departure from the form of eq. (1). This makes perfect since because when $r_d = 0$, then $r_1 = r_2$ and eq. (2) can be factored into the form of eq. (1) bringing us back to the standard W function. We define:

$$z_0 = \frac{1}{2}\sqrt{\frac{c^2}{a_o}}e^{-cr_m/2} = \frac{1}{2}\frac{c}{\sqrt{a_o}}e^{-cr_m/2} \tag{7}$$

where it is understood that $W(0)$ is/are the solution(s) when $r_d = 0$:

$$W(0) \;=\; \frac{2}{c}\; W(\pm z_0) \;=\; \frac{2}{c}\; W_0 \tag{8}$$

and where $W(\pm z_0)$ on the right side of eq. (8) is the *standard* Lambert W function. For real results, in particular for the parameters mentioned for the Double Well Dirac potential mentioned just below eq. (2), we are interested in real results and make use of the main branch of the standard W function. In this case, $c > 0$ helps ensure $|z_0| < 1/e$ (although $W(-z_0)$ could have a real result on a different branch for c sufficiently small). Implicit differentiation on both sides of eq. (6) yields:

$$\frac{\partial W(r_d)}{\partial r_d} \;=\; \frac{2r_d}{\frac{ce^{-c(W(r_d)+r_m)}}{a_o} + 2W(r_d)} \;=\; \frac{2r_d}{c\; W(r_d)^2 - c\; r_d^2 + 2\; W(r_d)} \tag{9}$$

Naturally successive derivatives with respect to r_d yields the Taylor series in r_d. Its radius of convergence will be obtained from the disk about the point of expansion $r_d = 0$ (assuming it is regular at the point of expansion) bounded by the closest singularity or branch point in the complex plane namely when the denominator of this derivative and all successive derivatives is zero, with $W(r_d)$ simultaneously satisfying eq. (6). Note that the expression on the right most side of eq. (9), obtained by virtue of eq. (6), does not formally depend on a_o nor r_m but only on c and r_d. Even though this is a quadratic in $W(r_d)$, only one solution satisfies eq. (6), namely:

$$W(r_{d\;crit}) \;=\; \frac{-1 + \sqrt{1 + c^2\; r_{d\;crit}^{\;2}}}{c} \tag{10}$$

The critical radius in the complex plane is:

$$r_{d\;crit} \;=\; \pm\frac{1}{c}\sqrt{2\; W(-2\; z_0^2) + W(-2\; z_0^2)^2} \;. \tag{11}$$

Here $W_0 = W(\pm z_0)$ is the standard W function and the radius is $|r_{d\ crit}|$. Note that when $z_0 = 0$, $W(z_0) = W(-2z_0^2) = 0$ (on the main branch) and the radius of convergence is also zero even though $z_0 = 0$ is analytic on the main branch for the (standard) Lambert W function. The series in r_d is thus:

$$\begin{aligned} W(r_d) &= 2\,\frac{W_0}{c} + \frac{1}{4}\,\frac{c\,r_d^2}{W_0(W_0+1)} + \frac{1}{64}\,\frac{c^3\,r_d^4\,(2\,W_0^2-1)}{W_0^3(W_0+1)^3} \\ &+ \frac{1}{1536}\,\frac{c^5\,r_d^6\,(8W_0^4-12W_0^2+3-4W_0^3)}{W_0^5(W_0+1)^5} \\ &+ \frac{1}{49152}\,\frac{c^7\,r_d^8\,(48W_0^6-132W_0^4+90W_0^2-15-64W_0^5+40W_0^3)}{W_0^7(W_0+1)^7} + O(c^9 r_d^{10}) \end{aligned} \tag{12}$$

which is a series in r_d^2 for $x = W(r_d) + r_m$ with x governed by eq. (2) and the radius of convergence is provided by the magnitude of (11). Within its radius of convergence, it converges rapidly. Note that when argument of z_0 is such that $W_0 = 0$ (which happens when e.g. $z_0 = 0$ on the main branch) or $W_0 + 1 = 0$ (which happens when $z_0 = -e^{-1}$ which is a branch point on the main branch), the individual series coefficients are confronted with divisions by zero, a result consistent, for the case $W_0 = 0$, with a radius of convergence of zero as given by eq.(11).

The validity of this series is demonstrated with some numerical tests. To reiterate the earlier problem, for a relatively high value of $\lambda = 0.8$ and an internuclear distance near the bond length $R = 2$, we have:

$$a_o = \frac{5}{4}, \quad c = 4, \quad r_d = \frac{1}{10}, \quad r_m = \frac{9}{10}$$

The solution of eq. (2) is $x = 1.0485$ obtained to within 4 decimals using the series in eq. (12) to within and including order $O(r_d^{10})$ using $W_0 = W(z_0)$ as the lead term. Similarly, the other solution $x = 0.6248$ is obtained using $W_0 = W(-z_0)$ as the lead term. The convergence of this series is much more rapid than the original "polarization expansion" mentioned in the introduction. Furthermore, this series is not limited to the real plane. For $\lambda = \frac{9}{10} - \frac{1}{10}i$

$$a_o = \frac{45}{41} + \frac{5}{41}i, \quad c = 4, \quad r_d = \frac{1}{20} + \frac{1}{20}i, \quad r_m = \frac{19}{20} - \frac{1}{20}i$$

The series to (and including) order $O(r_d^{14})$ yields $x = 1.0651408 - 0.0281742\,i$ to within 7 decimals for $W_0 = W(z_0)$ and similarly $x = 0.72818558 - 0.0876039\,i$ for $W_0 = W(-z_0)$ This series expansion is valid for small differences in the roots r_d, so clearly an asymptotic expansion valid for large r_d is also needed.

It would seem that in the case of three real roots, that we would only recover at most two out of three solutions. However, when two roots appear for e.g. $x > 0$ and the third root appears for $x < 0$, the latter can be recovered by reflection symmetry on the parameters. Let $x \to -x$, $c \to -c$, $r_i \to -r_i$ and these same formula can be used to recover that third solution.

2.2 Reversion of Power Series

To get an asymptotic series valid for large r_d, we further transform eq. (6) with the following variable transformations:

$$\begin{aligned} W(r_d)^2 &= \left(\frac{2}{c}\right)^2 (U^2 + d^2) \\ d &= c\,r_d/2 \end{aligned} \tag{13}$$

and $x = W(r_d) + r_m$ as before. Following the procedure for the standard W function [11], we start from:

$$z_0 \;=\; f(U) = Ue^{\pm\sqrt{U^2+d^2}} \quad \text{where} \quad z_0 = \frac{1}{2}\frac{c\; e^{-cr_m/2}}{\sqrt{a_o}} \tag{14}$$

where the sign $\pm$ takes into account that the negative square root is also possible. When $d = 0$, eq. (14) reduces to the form of the standard W function. Eq. (14) has the form:

$$z = f(U)$$

and we seek to reverse the power series to obtain:

$$U = g(z);$$

Defining $\phi(U) = U/f(U) = e^{\mp\sqrt{U^2+d^2}}$ and noting that $\phi(0) \neq 0$, we use a specialized version of the Lagrange-Bürmann [12] formula:

$$U(z) \;=\; z\phi(0) \;+\; \sum_{k=1}^{\infty} \frac{z^{k+1}}{(k+1)!}\frac{\partial^k \phi(U)^{k+1}}{\partial U^k}\Big]_{U=0} \tag{15}$$

Implicit differentiation of eq. (14) w.r.t. z yields:

$$\frac{\partial U(z)}{\partial z} \;=\; \frac{\sqrt{U(z)^2+d^2}\; e^{\mp\sqrt{U(z)^2+d^2}}}{U(z)^2+\sqrt{U(z)^2+d^2}} \tag{16}$$

We can see that the square root term dominates the functional form of the derivatives and the branch structure $U(z)$ in the complex plane much in accordance with the findings of Byers-Brown [5,6]. Note that eq. (16) has no explicit dependence on z and thus there is no need to verify its consistency with (14). To get the radius of convergence, we need to consider both the branch structure of the square root term in the denominator of (16) and values of $U(z)$ in the complex plane about the region $z = 0$ which would make this denominator zero. Thus the radius of convergence is limited by either:

$$|U_{crit}| < |d|$$

or

$$|U_{crit}| < \frac{1}{2}|\sqrt{2 \pm 2\sqrt{1+4d^2}}| \tag{17}$$

whichever is smaller. We obtain:

$$U(z) \;=\; ze^{\mp d} \;\mp\; \frac{1}{2}\frac{z^3 e^{\mp 3d}}{d} \;\pm\; \frac{1}{8}\frac{z^5(\pm 5d+1)e^{\mp 5d}}{d^3} \;+\; O(z^7 e^{\pm 7d}) \tag{18}$$

$$\;=\; \frac{1}{2}\frac{c\; e^{-\frac{1}{2}cr_\pm}}{\sqrt{a_o}} \;\mp\; \frac{1}{8}\frac{c^2\; e^{-\frac{3}{2}cr_\pm}}{a_o^{3/2} r_d} \;\pm\; \frac{1}{64}\frac{c^2\; e^{-\frac{5}{2}cr_\pm}(\pm 5\, c\, r_d + 2)}{a_o^{5/2} r_d^3} \;+\; O(e^{-\frac{7}{2}r_\pm c}) \tag{19}$$

where $r_+ = r_1$, $r_- = r_2$, $x = r_m + (2/c)\sqrt{U^2+d^2}$ as x given in eq. (2). This series would have growing exponential terms of the form $\exp(-k * d)$ unless $k > 0$ and consequenly this necessitates the requirement that $c\; r_\pm > 0$. Thus, we obtain a valid asymptotic expansion valid for large d or equivalently large r_d.

As in the previous section, we also get two kinds of solutions, respectively for positive and negative d, but they do not necessarily relate at all to the solutions of the previous section. The first section involved a

series expansion in r_d^2 where $r_d = (2/c)d$ and invariant with respect to the sign of d. Here we are dealing with a situation where the difference between the roots r_d is very large and thus quite possibly only one intersection between the exponential term on the left side of eq. (2) and its right side namely a quadratic in x, and thus only one solution.

As a numerical check and departing from the earlier physical chemistry problem in the earlier section, consider these particular values:

$$a_o \;=\; 1, \quad c \;=\; 2, \quad r_1 \;=\; 2, \qquad r_2 \;=\; 1 \quad \Rightarrow \quad d \;=\; r_d \;=\; \frac{1}{2}, \quad r_m \;=\; \frac{3}{2}\,.$$

The asymptotic series in eq. (19) with only the first 3 terms up to and including $O(1/d^3)$ yields the solution $x = 2.01739$ to within 4 decimals. Another test case, this time with some complex values:

$$a_o \;=\; 1, \quad c \;=\; 1, \quad r_1 \;=\; 2-i, \qquad r_2 \;=\; 1+i \quad \Rightarrow \quad d = \frac{1}{4} - \frac{1}{2}i, \quad r_d = \frac{1}{2} - i, \quad r_m = \frac{3}{2}$$

This same series with only 3 terms gives us $x = 1.9703 - 0.9430\,i$ to within 4 decimals. Thus, though initially motivated for the case of real numbers, these expansions can be used in the complex plane within certain restrictions.

2.3 Asymptotic series for large argument

The question arises what happens if we decide the left side z_0 of eq. (14) is large? For the principal branch when $z > 0$, taking logs of both sides of the equation governing the standard Lambert W function i.e. $We^W = z$ yields:

$$\ln[W(z)] \;=\; \ln(z) \;-\; W(z) \tag{20}$$

Recursive substitution yields successively:

$$\begin{array}{l} \ln(z) \\ \ln(z) - \ln(\ln(z)) \\ \ln(z) - \ln(\ln(z) - \ln(\ln(z))) \\ \ldots \end{array}$$

By taking logs on both sides of eq. (14) for the positive square root case only:

$$\ln(z) - \ln(U) = \sqrt{U^2 + d^2} \quad \text{or} \quad (\ln(z) - \ln(U))^2 = U^2 + d^2 \tag{21}$$

Thus, we consider two types of recursion.

$$U \;\rightarrow\; \sqrt{(\ln(z) - \ln(U))^2 - d^2} \tag{22}$$

$$U^2 \;\rightarrow\; \frac{1}{4}(-2\ln(z) + \ln(U^2) - 2d)(-2\ln(z) + \ln(U^2) + 2d) \tag{23}$$

The second recursion avoids the square root (and its messy consequences for recursion) and looks like a factored form involving a combination of asymptotic formulae for the standard W function. By successive substitution, we obtain:

$$U \approx \sqrt{\left(\ln(z) - \ln\left(\sqrt{\left(\ln(z) - \ln\left(\sqrt{\ldots \ln\left(\ln(z) - \ln\left(\sqrt{(\ln(z) - \ln(U))^2 - d^2}\right)\right)^2 \ldots - d^2}\right)\right)^2 - d^2}\right)\right)^2} \tag{24}$$

Table 1: Non-Linear transformations applied to Taylor series of eq.(12) for $r_d = 0.8$

no. of terms	$W(r_d)$ Taylor Series	Shanks	Levin t
1	-0.9999999996	-0.9999999996	-0.9999999996
2	-1.6400000000	-1.6400000000	-2.7777777780
3	-1.4352000000	-1.4848484850	-1.5213977230
4	-1.6099626670	-1.5294964030	-1.5192810810
5	-1.4421905070	-1.5246574640	-1.5243445560
6	-1.6265161880	-1.5271424650	-1.5267037510
7	-1.4108133840	-1.5280997520	-1.5277557490
⋮	⋮	⋮	⋮
	-1.528554071	-1.528554071	-1.528554071

and:

$$\begin{aligned} U^2 &\approx \frac{1}{4}\left(-2\ln(z) + \ln\left(\frac{1}{4}\left(-2\ln(z) + \ln(\ldots\right.\right. \\ &+ \ln\left(\frac{1}{4}\left(-2\ln(z) + \ln(U^2) - 2d)(-2\ln(z) + \ln(U^2) + 2d\right)\right) + \ldots + 2d\right)\Big)\Big) + 2d\Big) \end{aligned} \tag{25}$$

However, we find from experience that the argument z has to be *very large* indeed for these asympotic formulations to converge. This exercise is more to demonstrate the resemblence with the counterpart expansion for the standard W function, namely eq. (21). For computational value, sections 2.1 and 2.2 are more useful. Nonetheless, the very large z_0 argument is tractable.

2.4 Summation techniques

Finally, the series summation can be accelerated even *beyond* the radii of convergence using non-linear transformations as mentioned in the introduction. These transformations are applied to the sequence of partial sums and are capable of accelerating the convergence of a series and even sum divergent series (e.g. see the work of [13, 14]). We take the point of view that a Taylor or asymptotic series has all the desired "information", getting numbers from the series is a matter of a summation technique. For the series in r_d of the first section for both $W(\pm z_0)$, it was found that the series, when oscillating in r_d, could indeed be extended beyond their radius of convergence. This is demonstrated for the test case:

$$a_o \;=\; 1, \quad c \;=\; 1, \quad r_m \;=\; 1\;.$$

Here, the asymptotic solution of eq. (19) matches the extrapolated Taylor series of solution about $W_0(z_0)$ of (12) in 4 decimal places. Here $r_{d\ crit} \;\approx\; 0.64$ and we consider the regime when $r_d > r_{d\ crit}$, the alternating Taylor series is divergent. This Taylor series to order $O(r_d^{12})$ (6 terms in powers of r_d^2) is used for the t transformation of Levin [15] and the Shanks transformation [16]. To demonstrate agreement between the Taylor series and the outcome of the non-linear transformations, tables 1 and 2 compares the Taylor series of eq. (12) and the outcome of the Shanks and Levin t transformations for respectively $r_d = 0.8$ and $r_d = 1.5$. At the bottom of each table is listed what exact solution to the number of digits shown. The Taylor series of eq. (12) diverges violently when $r_d = 1.5$ but the non-linear transformations

Table 2: Non-Linear transformations applied to Taylor series of eq.(12) for $r_d = 1.5$

no. of terms	$W(r_d)$ Taylor Series	Shanks	Levin t
1	0.38889448	0.3888944774	0.3888944774
2	2.81078092	2.8107809190	-0.0743922833
3	-3.02541152	1.0991727410	-5.1438626370
4	24.71693722	1.7964086140	1.7380384290
5	-139.85949420	1.3876539200	1.5296581130
6	953.20098980	1.5894954440	1.5167708910
7	-6823.99405600	1.4791930140	1.5165517370
⋮	⋮	⋮	⋮
	1.516240428	1.516240428	1.516240428

converge nicely. Three terms of the asymptotic expansion in eq. (19) for $r_d = 1.5$, yield $x = 1.516240673$ which agrees with the exact solution starting from $W_0(z_0)$ to within 7 decimals. This demonstrates that the solutions of section 2.2 can match one of the solutions of section 2.1.

3 Conclusions

Previously [10] we had inferred a canonical form for a generalization as expressed by (2) and (3) and given both mathematical and physical justifications for it. Herein, we formulated Taylor series and asymptotic series useful for analysis and computation. We find that the results are similar to those governing the standard W function and represent a natural extension though the branch structure in the complex plane may differ.

This approach could be extended to higher order polynomials fitting the pattern of eq. (3). For example, when the right side of eq. (3) we can *complete the cube* in some special cases, i.e. for

$$x^3 \; + \; a\, x^2 \; + \; b\, x \; + \; c \; = \; \left(x \; + \; \frac{1}{3} \right)^3 \; - \; \left(\frac{1}{27}\, a^3 \; - \; c \right)^3 \quad \text{when} \quad b \; = \; \frac{a^2}{3}$$

which can allow a special case of eq.(3) and create a cubic relation counterpart of eq. (14):

$$\frac{e^{-c r_m}}{a_o} \; = \; Y^3 \; e^{c \; (Y^3 + d_3)^{1/3}} \tag{26}$$

where $(x - r_m)^3 \; = \; Y^3 + d_3$ and $d_3 \; = \; \frac{a^3}{27} - c$ and $r_m = -a/3$. However, for larger order polynomials and rational polynomials, this approach is quickly exhausted and one has to rely on numerical techniques which is very feasible.

Finally, the Taylor series summation can be accelerated even *beyond* the radii of convergence using non-linear transformations known as the Levin or Shanks transformations allowing a matching between the Taylor series and the asymptotic series. The resulting series can be converted into FORTRAN or C code using the interface between Maple and these languages [18].

References

[1] R. B. Mann and T. Ohta, *Exact solution for the metric and the motion of two bodies in* $(1+1)$*-dimensional gravity*, Phys. Rev. D. **55**, (1997), 4723-4747.

[2] P. S. Farrugia, R. B. Mann, and T. C. Scott, *N-body Gravity and the Schrödinger Equation*, Class. Quantum Grav. **24**, (2007), 4647-4659.

[3] T. C. Scott, J. F. Babb, A. Dalgarno, and J. D. Morgan III, *The Calculation of Exchange Forces: General Results and Specific Models*, J. Chem. Phys. **99**, (1993), 2841-2854.

[4] A. A. Frost, *Delta-Function Model. I. Electronic Energies of Hydrogen-Like Atoms and Diatomic Molecules*, J. Chem. Phys. **25**, (1956), 1150-1154.

[5] P. R. Certain and W. Byers Brown, *Branch Point Singularities in the Energy of the Delta-Function Model of One-Electron Diatoms*, Intern. J. Quantum Chem. **6**, (1972), 131-142.

[6] W. N. Whitton and W. Byers Brown, *The Relationship Between the Rayleigh-Schrödinger and Asymptotic Perturbation Theories of Intermolecular Forces*, Int. J. Quantum Chem. **10**, (1976), 71-86.

[7] T. C. Scott, A. Dalgarno, and J. D. Morgan III, *Exchange Energy of* H_2^+ *Calculated from Polarization Perturbation Theory and the Holstein-Herring Method*, Phys. Rev. Lett. **67**, pp. 1419-1422 (1991).

[8] T. C. Scott, J. F. Babb, A. Dalgarno, and J. D. Morgan III, *Resolution of a Paradox in the Calculation of Exchange Forces for* H_2^+, Chem. Phys. Lett. **203**, pp. 175-183 (1993).

[9] T. C. Scott, M. Aubert-Frécon and J. Grotendorst, *New approach for the electronic energies of the hydrogen molecular ion*, Chem. Phys. **324**, (2006), 323-338.

[10] T. C. Scott, R. B. Mann and R. E. Martinez II, *General Relativity and Quantum Mechanics: Towards a Generalization of the Lambert W Function*, AAECC, **17**, (2006) 41-47.

[11] (a) R. Corless, G. Gonnet, D. E. G. Hare and D. Jeffrey, *Lambert's W Function in Maple*, MapleTech **9**, ed. T. C. Scott, (Spring 1993); (b) R. Corless, G. Gonnet, D. E. G. Hare, D. Jeffrey and D. Knuth, *On the Lambert W Function*, Advances in Computational Mathematics, **5**, (1996), 329-359.

[12] A. C. Dixon, *On Burmann's Theorem*, Proc. London Math. Soc. **34**, pp. 151-153, (1902).

[13] J. Grotendorst, *A Maple Package for Transforming Series, Sequences and Functions*, Comput. Phys. Commun. **67**, (1991), 325-342.

[14] E. J. Weniger, *Nonlinear Sequence Transformations for the Acceleration of Convergence and the Summation of Divergent Series*, Comput. Phys. Rep. **10**, (1989), 189-371.

[15] D. Levin, *Development of non-linear transformations of improving convergence of sequences*, Internat. J. Comput. Math. **B 3**, (1973), 371-388.

[16] D. Shanks, *Nonlinear Transformations of Divergent and Slowly Convergent Sequences*, J. Math. and Phys. (Cambridge, Mass.) **34**, (1955), 1-42.

[17] B. W. Char, K. O. Geddes, G. H. Gonnet, B. L. Leong, M. B. Monagan and S. M. Watt, *First Leaves: A Tutorial Introduction to Maple V*, Springer-Verlag, New York, (1992).

[18] C. Gomez and T. C. Scott, *Maple programs for generating efficient FORTRAN code for serial and vectorised machines*, Comput. Phys. Commun., Thematic issue: Computer Algebra in Physics Research, **115**, pp. 548-562 (1998).

NSF Funding Opportunities for Symbolic Computation

Communicated by Erich Kaltofen and Alexey Ovchinnikov

The NSF CCF Core Programs, Algorithmic Foundation, is to fund as one of its areas research symbolic computation. Please, note the change of deadlines of this major funding opportunity this year:

- Medium projects: September 24, 2013 – October 15, 2013
- Large projects: November 4, 2013 – November 19, 2013
- Small projects: January 2, 2014 – January 17, 2014

For further information, please, refer to the program web page:

`http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=503299&org=CISE`

and (new) solicitation:

`http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=503220&org=CISE`

where you can find further details, NSF contact information for further questions, and what has been recently funded under the Core Programs.

ACM Communications in Computer Algebra, Vol. 47, No. 3, Issue 185, September 2013

The Computational Geometry Algorithms Library CGAL*

Efi Fogel† Monique Teillaud‡

Abstract

The Computational Geometry Algorithms Library (CGAL) is an open source software library that provides industrial and academic users with easy access to reliable implementations of efficient geometric algorithms.

Usage. CGAL is used in a diverse range of domains requiring geometric computation such as computer graphics, scientific visualization, computer aided design and modeling, geographic information systems, molecular biology, medical imaging, and many more. Since CGAL provides a wide range of components, we restrict ourselves to mentioning just a few here.

As an example application of CGAL, a series of packages are provided which are useful in robotics and automation: Minkowski sums, offset polygons, Boolean operations on curved regions. The high precision of CGAL allows users to solve geometric problems involving motion in restricted environments, such as those arising in assembly planning.

The robustness and efficiency of components such as the Delaunay triangulation and mesh construction and manipulation packages makes CGAL attractive for simulations, in particular those involving proteins, particle physics, fluid dynamics, medical modeling, biophysics, geophysics, and astronomy. Indeed, the aforementioned components are largely used in these areas.

Some support for manipulations of polynomials and for solving univariate polynomial equations and bivariate polynomial systems is also provided, as well as handling for convex quadratic programs.

History of the CGAL **Open Source Project.** Several European research groups started to develop their own small geometry libraries in the early 90's. In 1996, a consortium of eight sites was created to gather the work of these groups into a single software library, namely CGAL. Their main goal was to promote research in computational geometry and to translate the results into **robust** software suitable for industrial applications.

Around this time the Computational Geometry Impact Task Force Report [C+96, C+99] made a series of recommendations. Amongst these recommendations, the production and distribution of usable (and useful) geometric software, and the need to establish a reward structure for software implementations in academia, were key.

On November 2003, when version 3.0 was released, CGAL officially became an Open Source project, allowing new contributors to join the project.

License. CGAL is distributed under the GPL license (apart from a few basic parts, which are distributed under the LGPL license). In particular, it is publicly and freely available for academic use. Commercial licenses are offered by Geometry Factory, a company founded in 2003 mainly for this purpose.

*`http://www.cgal.org`

†Tel-Aviv University `http://acg.cs.tau.ac.il/people/efifogel`

‡INRIA Sophia Antipolis-Méditerranée `http://www-sop.inria.fr/members/Monique.Teillaud/`

Editorial board. The CGAL editorial board was created in 2001. It currently consists of thirteen members. The main task of the editorial board is to assure the quality of CGAL. It is also responsible for making decisions about technical matters and coordinating communication and promotion of CGAL.

All new packages must be submitted to the Editorial Board to be reviewed before they can be accepted and integrated into the library, in a process that is very similar to the standard review process for papers published in conference proceedings or journals. More information about the submission process is available at `http://www.cgal.org/review_process_rules.html`.

Style and Techniques. CGAL is a unique library both in general and within the field of computational geometry in particular, as it consists of a large number of components with a homogeneous API (Application Programming Interface). Careful choices in design and programming style have made CGAL the *de facto* standard in the field of applied computational geometry. Its development started whilst the standardization process of C++ and the STL (Standard Template Library) was taking place. Indeed, the programming style is very close to the programming style of STL; it rigorously adheres to the generic programming paradigm—a discipline that consists of the gradual lifting of concrete algorithms abstracting over details, while retaining the algorithm semantics and efficiency. The programming style of CGAL also facilitates the process of interfacing with third party software.

Each package comes with header files consisting not only of the interface, but also the generic implementation of the package code, comprehensive and didactic on-line documentation, a set of non-interactive standalone example programs, and an optional interactive demo with a graphical user interface.

Robustness. CGAL follows the exact geometric-computation paradigm, which simply amounts to ensuring that errors in predicate evaluations do not occur; it guarantees robustness of the applied algorithms.

We additionally remark that every package also includes a collection of function and regression test. [1] The tests provided by each package are combined into one place to form the CGAL test suite. This test suite is run daily and its results are automatically assembled, analyzed, and reported.

Impact. Measuring the impact of software is a difficult task, especially in the Open Source software community. Even if some hard numbers can be found, they can be difficult to interpret. The following facts may shed some light on the impact of CGAL:

- There are roughly 1000 downloads per month from `http://gforge.inria.fr/`
- CGAL is included in various software distribution channels, such as Fedora, Debian/Ubuntu, and Macports.
- The range of uses of CGAL is very broad, as shown by the sample list of projects using CGAL, which is available at `http://www.cgal.org/projects.html`. In addition, many projects shown in `http://acg.cs.tau.ac.il/projects` use CGAL or even describe the development of a CGAL component.
- The CGAL triangulation packages were integrated in Matlab 2009a.[2]
- Springer has published a book entitled "CGAL Arrangements and Their Applications" authored by some of the developers of the *2D Arrangements* package and its derivatives.
- Concerning the public mailing lists, there are currently

[1] However these are not distributed as part of the public releases.

[2] Watch the video at `http://www.mathworks.com/products/demos/shipping/matlab/New-MATLAB-Mathematics-Features-in-R2009a.html`

- 4000 subscribers to the announcement list `cgal-announce@lists-sop.inria.fr`
- 1500 to the public discussion list `cgal-discuss@lists-sop.inria.fr`, with high traffic: users are free to ask questions, which are often rapidly answered by the developers or other users.

Acknowledgments
We thank Ross Hemsley for helping us distilling the text.

References

[C+96] Bernard Chazelle et al. Application challenges to computational geometry: CG impact task force report. Technical Report TR-521-96, Princeton Univ., April 1996.

[C+99] Bernard Chazelle et al. Application challenges to computational geometry: CG impact task force report. In B. Chazelle, J. E. Goodman, and R. Pollack, editors, *Advances in Discrete and Computational Geometry*, volume 223 of *Contemporary Mathematics*, pages 407–463. American Mathematical Society, Providence, 1999.

ACM Communications in Computer Algebra, Vol 47, No. 3, Issue 185, September 2013

ISSAC 2013 Poster Abstracts

Communicated by Alin Bostan

Revisiting QRGCD and Comparison with ExQRGCD*

Kosaku Nagasaka† and Takaaki Masui‡
†Kobe University, 3-11 Tsurukabuto, Nada-ku, Kobe 657-8501 JAPAN
‡Kobe-Takatsuka High School, 9-1 Mikatadai, Nishi-ku, Kobe 651-2277 JAPAN
† nagasaka@main.h.kobe-u.ac.jp, ‡ masui.takaaki@gmail.com

1 Introduction

In this poster, we are interested in computing "approximate polynomial GCD": for the input polynomials $f(x), g(x) \in \mathbb{R}[x]$, we call the polynomial $d(x) \in \mathbb{R}[x]$ "approximate polynomial GCD" of tolerance $\varepsilon \in \mathbb{R}_{\geq 0}$ if it satisfies

$$f(x) + \Delta_f(x) = f_1(x)d(x),\ g(x) + \Delta_g(x) = g_1(x)d(x)$$

for some polynomials $\Delta_f(x), \Delta_g(x), f_1(x), g_1(x) \in \mathbb{R}[x]$ such that $\deg(\Delta_f) \leq \deg(f)$, $\deg(\Delta_g) \leq \deg(g)$, $\|\Delta_f\|_2 < \varepsilon \|f\|_2$ and $\|\Delta_g\|_2 < \varepsilon \|g\|_2$ where $\|\cdot\|_2$ denotes the 2-norm. Although there are many studies, we revisit the QRGCD algorithm[2] which is one of algorithms based on matrix decompositions and is also implemented as a part of the SNAP package of Maple. It is notable that the QRGCD algorithm is very simple and has been used as the benchmark algorithm for newly proposed algorithms. The framework of the QRGCD algorithm is as follows. For details, please refer the original paper[2].

1. Compute the QR decomposition of $Syl(f, g)$: $Syl(f, g) = QR$.
2. Find the gap between the k-th and $(k+1)$-th row vectors $\vec{r}_k, \vec{r}_{k+1}$ of R and form the polynomial with coefficients $\vec{r}_k$, which is an approximate polynomial GCD (or its factor).
3. Apply the same procedures to the reversal polynomials of cofactors since R may not have the approximate common divisor whose roots are outside the unit circle in the complex plane.

However, since QRGCD was proposed in the early stage of approximate GCD, its theoretical background is not enough analyzed from the current theoretical point of view and the official implementation is different from the paper. For example, Bini and Boito[1] reported that QRGCD failed to recognize the correct degree of GCD for polynomials with small leading coefficients. This result is caused by the preconditioning routine in the official implementation hence QRGCD works well for such polynomials. Therefore, our aim consists of two parts: 1) verifying the efficiency of QRGCD with much theoretical considerations, and 2) improving the framework and algorithm to be more accurate and able to satisfy the given tolerance.

2 Notable Facts on QRGCD and its Implementation in SNAP

Recently, the concept of "structured perturbation" is important in the theory of approximate GCD. However, at the time of QRGCD proposed, this concept is not widely discussed hence there are unclear statements in the original paper from this point of view. Analyzing their theory from this concept could be interesting. For example, any relationship between the QR factoring and structured perturbation, the reason that the QR factoring can not detect the roots outside the unit circle and so on.

*This work was supported in part by JSPS KAKENHI Grant Number 22700011.

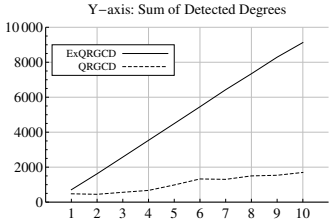
Moreover, we found the 4 significant differences between the original algorithm and the SNAP implementation. According to our personal conversations, some of them are implemented by the original authors and others may be by H. Kai, the person implemented it in the SNAP package. The differences are 1) the preconditioning routine "find non-zero terms", 2) the matrix norm used, 3) the polynomials to be applied to the algorithm "Split", and 4) the fail-safe retry loop. The first one may be the cause that many people think QRGCD is weak for polynomials with small leading coefficients. Without this, QRGCD works well for such polynomials. Other differences seem to be some techniques to make QRGCD working well.

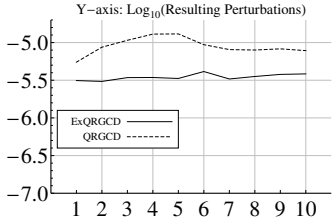
3 Improved QRGCD Algorithm (ExQRGCD)

We refine the framework of QRGCD from the different approach with recent theoretical results of approximate polynomial GCD and propose the improved algorithm called "ExQRGCD". The most notable difference is that our algorithm detects a row vector of R by estimating relative distance from the expected approximate GCD while QRGCD detects by estimating absolute distance. As a result, ExQRGCD works more accurately. For example, it works for the following polynomials (QRGCD does not work well for this kind of polynomials unfortunately). For $i = 1, \ldots, 10$, we have generated 100 pairs of (f, g) such that

$$f(x) = d(x)\prod_{j=1}^{2i}(x-\omega_{f,j})\prod_{j=1}^{2i}(x-\hat{\omega}_{f,j}),\ g(x) = d(x)\prod_{j=1}^{2i}(x-\omega_{g,j})\prod_{j=1}^{2i}(x-\hat{\omega}_{g,j})$$

where $d(x) = \prod_{j=1}^{3i}(x-\omega_{d,j})\prod_{j=1}^{3i}(x-\hat{\omega}_{d,j})$, $\omega_{\cdot,j} = O(10^{-2}), \hat{\omega}_{\cdot,j} = O(10^2)$ is randomly chosen, $f(x), g(x)$ are normalized (i.e. $\|f(x)\|_2 = \|g(x)\|_2 = 1$) and rounded with $Digits := 10$. We computed with tolerance 10^{-5}. Figure 1 shows the result that ExQRGCD is explicitly better than QRGCD though as for computing time, ExQRGCD is 39.8 times slower than QRGCD (note that QRGCD outputs failure for 62% pairs so computing time is very fast for the rest easy cases). The average of resulting perturbations of ExQRGCD is also better. For other random generated examples, ExQRGCD is almost 2 times slower than QRGCD since ExQRGCD is more conservative than QRGCD for detecting approximate GCD hence it computes QR decompositions several times (this is more than that of QRGCD in general).





Number of fail events or wrong Δ_* (QRGCD)											
i=1	2	3	4	5	6	7	8	9	10	Δ_f	Δ_g
29	63	68	71	64	60	66	66	69	68	110	119

Number of fail events or wrong Δ_* (ExQRGCD)											
i=1	2	3	4	5	6	7	8	9	10	Δ_f	Δ_g
0	0	0	0	0	0	0	0	0	0	0	0

Figure 1: Sum of Detected Degrees and Resulting Perturbations (failure is not counted)

We note that our preliminary implementations on Maple and written in C, and generated polynomial data are available: "`http://wwwmain.h.kobe-u.ac.jp/~nagasaka/research/snap/issac2013/`".

References

[1] D. A. Bini and P. Boito. A fast algorithm for approximate polynomial GCD based on structured matrix computations. In *Numerical methods for structured matrices and applications*, volume 199 of *Oper. Theory Adv. Appl.*, pages 155–173. Birkhäuser Verlag, Basel, 2010.

[2] R. M. Corless, S. M. Watt, and L. Zhi. *QR* factoring to compute the GCD of univariate approximate polynomials. *IEEE Trans. Signal Process.*, 52(12):3394–3402, 2004.

Schützenberger's factorization on $q-$stuffle Hopf algebra

C. Bui[♭], G. H. E. Duchamp[♯], Hoang Ngoc Minh[◊,♯]
[♭]Hué University - College of sciences, 77 - Nguyen Hue street - Hué city, Viêt Nam
[♯]Institut Galilée, LIPN - UMR 7030, CNRS - Université Paris 13, F-93430 Villetaneuse, France,
[◊]Université Lille II, 1, Place Déliot, 59024 Lille, France

Schützenberger's monoidal factorization [9] has been introduced and plays a central role in the renormalization [7] of associators which are formal power series in non commutative variables[1]. The coefficients of these power series are polynomial at positive integral multi-indices of Riemann's zêta function[2] [5, 10] and they satisfy quadratic relations [1] which can be explained through Lyndon words. These relations can be obtained by identification of the local coordinates on a bridge equation connecting the Cauchy and Hadamard algebras of polylogarithmic functions and use the factorizations of the non commutative generating series of polylogarithms [6] and of harmonic sums [7]. This equation is mainly a consequence of the double isomorphy between these structures to respectively the shuffle [6] and stuffle [3] algebras both admitting the Lyndon words as a transcendence basis.

Symbolic computation allows us to introduce a formal variable q in order to better understand the mechanisms of the shuffle and to obtain algorithms on stuffles. We will then examine the q-stuffle interpolating between the shuffle [9], stuffle [8] and minus-stuffle [3]. In particular, we will give an effective construction of pair of bases in duality. It uses essentially an adapted version of the Eulerian projector in order to obtain the primitive elements of the q-stuffle Hopf algebra and they are obtained thanks to the computation of the logarithm of the diagonal series. This study completes the treatment for the stuffle [7] and boils down to the shuffle [9].

More precisely, let $Y = \{y_s\}_{s\geq 1}$ be an alphabet with the total order $y_1 > y_2 > \cdots$. Let also $\mathbf{k}$ be a unitary $\mathbb{Q}$-algebra containing q. One defines the q-stuffle, or its dual co-product, as follows, for any $y_s, y_t \in Y$ and $u, v \in Y^*$,

$$u \sqcup\!\sqcup_q 1_{Y^*} = 1_{Y^*} \sqcup\!\sqcup_q u = u \quad \text{and} \quad y_s u \sqcup\!\sqcup_q y_t v = y_s(u \sqcup\!\sqcup_q y_t v) + y_t(y_s u \sqcup\!\sqcup_q v) + q y_{s+t}(u \sqcup\!\sqcup_q v), \tag{1}$$

$$\Delta_{\sqcup\!\sqcup_q}(1_{Y^*}) = 1_{Y^*} \otimes 1_{Y^*} \quad \text{and} \quad \Delta_{\sqcup\!\sqcup_q}(y_s) = y_s \otimes 1_{Y^*} + 1_{Y^*} \otimes y_s + q \sum_{s_1+s_2=s} y_{s_1} \otimes y_{s_2}. \tag{2}$$

This product is commutative, associative and unital. With the co-unit defined by, $\epsilon(P) = \langle P \mid 1_{Y^*}\rangle$, for $P \in \mathbf{k}\langle Y\rangle$, one gets $\mathcal{H}_{\sqcup\!\sqcup_q} = (\mathbf{k}\langle Y\rangle, \mathtt{conc}, 1_{Y^*}, \Delta_{\sqcup\!\sqcup_q}, \epsilon)$ and $\mathcal{H}^{\vee}_{\sqcup\!\sqcup_q} = (\mathbf{k}\langle Y\rangle, \sqcup\!\sqcup_q, 1_{Y^*}, \Delta_{\mathtt{conc}}, \epsilon)$ which are mutually dual bialgebras and, in fact, Hopf algebras because they are $\mathbb{N}$-graded by the weight.

Group-like elements, redefined below, form a group for which the log-exp correspondence is explained by as follows

Lemma 1 (q-extended Friedrichs criterium) *Let $S \in \mathbf{k}\langle\langle Y\rangle\rangle$ (for 2., we suppose in addition that $\langle S \mid 1_{Y^*}\rangle = 1$).*

1. *S is* primitive*, i.e. $\Delta_{\sqcup\!\sqcup_q} S = S \otimes 1_{Y^*} + 1_{Y^*} \otimes S$, if and only if, for any $u, v \in Y^+$, $\langle S \mid u \sqcup\!\sqcup_q v\rangle = 0$.*
2. *S is* group-like*, i.e. $\Delta_{\sqcup\!\sqcup_q} S = S \otimes S$, if and only if, for any $u, v \in Y^+$, $\langle S \mid u \sqcup\!\sqcup_q v\rangle = \langle S \mid u\rangle\langle S \mid v\rangle$.*
3. *S is group-like if and only if $\log S$ is primitive.*

Proposition 1 *Let $\mathcal{D}_Y = \sum_{w\in Y^*} w \otimes w$ be the diagonal series over Y. Then*

1. $\log \mathcal{D}_Y = \sum_{w\in Y^+} w \otimes \pi_1(w)$, *where* $\pi_1(w) = w + \sum_{k\geq 2} \frac{(-1)^{k-1}}{k} \sum_{u_1,\ldots,u_k\in Y^+} \langle w \mid u_1 \sqcup\!\sqcup_q \ldots \sqcup\!\sqcup_q u_k\rangle u_1 \ldots u_k$.
2. *For any $w \in Y^*$, we have* $w = \sum_{k\geq 0} \frac{1}{k!} \sum_{u_1,\ldots,u_k\in Y^+} \langle w \mid u_1 \sqcup\!\sqcup_q \ldots \sqcup\!\sqcup_q u_k\rangle \pi_1(u_1)\ldots\pi_1(u_k)$.

[1]These associators were introduced in quantum field theory by Drinfel'd and the universal associator, *i.e.* Φ_{KZ}, was obtained with explicit coefficients which are polyzêtas and regularized polyzêtas [5].
[2]These values are usually abbreviated MZV's by Zagier [10] and are also called polyzêtas by Cartier [1].

Let $\mathcal{P} = \{P \in \mathbb{Q}\langle Y\rangle \mid \Delta_{\sqcup\!\sqcup_q} P = P \otimes 1 + 1 \otimes P\}$ be the set of primitive polynomials. Since, in virtue of $\Delta_{\sqcup\!\sqcup_q}\pi_1(w) = \pi_1(w) \otimes 1 + 1 \otimes \pi_1(w)$, $\mathrm{Im}(\pi_1) \subseteq \mathcal{P}$, we can state the following

Theorem 1 ([2]) *1. Let $\{\Pi_l\}_{l\in\mathcal{L}ynY}$ be defined by, for any $y_k \in Y, \Pi_{y_k} = \pi_1(y_k)$ and for any $l \in \mathcal{L}ynX$ of standard factorization $l = (s, r), \Pi_l = [\Pi_s, \Pi_r]$. Then $\{\Pi_l\}_{l\in\mathcal{L}ynY}$ forms a basis of $\mathcal{P}$.*

2. Let $\{\Pi_w\}_{w\in Y^*}$ be defined by, for any $w \in Y^*$ such that $w = l_1^{i_1}\dots l_k^{i_k}, l_1 > \dots > l_k, l_1\dots, l_k \in \mathcal{L}ynY$, $\Pi_w = \Pi_{l_1}^{i_1}\dots\Pi_{l_k}^{i_k}$. Then $\{\Pi_w\}_{w\in Y^*}$ forms a basis of $\mathbf{k}\langle Y\rangle$.

3. Let $\{\Sigma_w\}_{w\in Y^*}$ be the family of the quasi-shuffle algebra obtained by duality with $\{\Pi_w\}_{w\in Y^*}$. Then $\{\Sigma_w\}_{w\in Y^*}$ generates freely the quasi-shuffle algebra.

4. The family $\{\Sigma_l\}_{l\in\mathcal{L}ynY}$ forms a transcendence basis of $(\mathbf{k}\langle Y\rangle, \sqcup\!\sqcup_q)$.

We now give formulas which permit to compute the basis $\{\Sigma_w\}_{w\in Y^*}$ without inverting a huge Gram matrix.

Theorem 2 (q-extended Schützenberger's factorization, [2]) *1. For any $y \in Y$, $\Sigma_y = y$.*

2. For any $y_{s_1}\dots y_{s_k} \in \mathcal{L}ynX$, $\displaystyle\Sigma_{y_{s_1}\dots y_{s_k}} = \sum_{\substack{\{s'_1,\cdots,s'_i\}\subset\{s_1,\cdots,s_k\}, l_1\geq\cdots\geq l_n\in\mathcal{L}ynY \\ (y_{s_1}\cdots y_{s_k})\overset{*}{\leftarrow}(y_{s'_1},\cdots,y_{s'_n},l_1,\cdots,l_n)}} \frac{q^{i-1}}{i!} y_{s'_1+\cdots+s'_i}\Sigma_{l_1\cdots l_n}.$

3. For any $w = l_1^{i_1}\dots l_k^{i_k}$, with $l_1, \dots, l_k \in \mathcal{L}ynY$ and $l_1 > \dots > l_k$, $\displaystyle\Sigma_w = \frac{\Sigma_{l_1}^{\sqcup\!\sqcup_q i_1} \sqcup\!\sqcup_q \dots \sqcup\!\sqcup_q \Sigma_{l_k}^{\sqcup\!\sqcup_q i_k}}{i_1!\dots i_k!}.$

4. $\displaystyle\mathcal{D}_Y = \sum_{w\in Y^*} \Sigma_w \otimes \Pi_w = \prod_{l\in\mathcal{L}ynY}^{\searrow} \exp(\Sigma_l \otimes \Pi_l).$

Theorems 1.1 and 2.2 are based mainly on respectively the logarithm of the diagonal series $\mathcal{D}_Y$ and the standard sequences [9, 2] and lead to simplified algorithms getting bases in duality as shown in the following

Example 1

$$
\begin{aligned}
\Pi_{y_2} &= y_2 - \tfrac{q}{2}y_1^2,\\
\Pi_{y_2y_1} &= y_2y_1 - y_1y_2,\\
\Pi_{y_3y_1y_2} &= y_3y_1y_2 - \tfrac{q}{2}y_3y_1^3 - qy_2y_1^2y_2 + \tfrac{q^2}{4}y_2y_1^4 - y_1y_3y_2 + \tfrac{q}{2}y_1y_3y_1^2 + \tfrac{q}{2}y_1^2y_2^2 - \tfrac{q^2}{2}y_1^2y_2y_1^2 - y_2y_3y_1\\
&+ \tfrac{q}{2}y_2^2y_1^2 + y_2y_1y_3 + \tfrac{q}{2}y_1^2y_3y_1 - \tfrac{q}{2}y_1^3y_3 + \tfrac{q^2}{4}y_1^4y_2,\\
\Pi_{y_3y_1y_2y_1} &= y_3y_1y_2y_1 - y_3y_1^2y_2 - \tfrac{q}{2}y_2y_1^2y_2y_1 - y_1y_3y_2y_1 + y_1y_3y_1y_2 + \tfrac{q}{2}y_1^2y_2^2y_1 - \tfrac{q}{2}y_1^2y_2y_1y_2 - y_2y_1y_3y_1\\
&+ \tfrac{q}{2}y_2y_1y_2y_1^2 + y_2y_1^2y_3 + y_1y_2y_3y_1 - \tfrac{q}{2}y_1y_2^2y_1^2 - y_1y_2y_1y_3 + \tfrac{q}{2}y_1y_2y_1^2y_2.\\
\Sigma_{y_2} &= y_2,\\
\Sigma_{y_2y_1} &= y_2y_1 + \tfrac{q}{2}y_3,\\
\Sigma_{y_3y_2y_1} &= y_3y_1y_2 + y_3y_2y_1 + qy_3^2 + \tfrac{q}{2}y_4y_2 + \tfrac{q^2}{3}y_6 + \tfrac{q}{2}y_5y_1,\\
\Sigma_{y_3y_1y_2y_1} &= 2y_3y_2y_1^2 + qy_3y_2^2 + y_3y_1y_2y_1 + \tfrac{3q}{2}y_3^2y_1 + \tfrac{q}{2}y_3y_1y_3 + \tfrac{q^2}{2}y_3y_4 + \tfrac{q}{2}y_4y_2y_1 + \tfrac{q^2}{4}y_4y_3 + qy_5y_1^2 + \tfrac{q^2}{2}y_5y_2 + \tfrac{q^2}{2}y_6y_1 + \tfrac{q^3}{8}y_7.
\end{aligned}
$$

In conclusion, since the pioneering works of Schützenberger and Reutenauer [9], the question of computing bases in duality (maybe at the cost of a more involved procedure, but without inverting a Gram matrix) remained open in the case of cocommutative deformations of the shuffle product. We have given such a procedure allowing a great simplification for an interpolation between shuffle and stuffle. In the next framework, this product will be continuously deformed, in the most general way while remaining commutative [4].

References

[1] P. Cartier.– *Fonctions polylogarithmes, nombres polyzêtas et groupes pro-unipotents*, Sém BOURBAKI, 53ème 2000-2001, n°885.

[2] C. Bui, G. H. E. Duchamp, V. Hoang Ngoc Minh.– *Schützenberger's factorization on the (completed) Hopf algebra of q−stuffle product.* arXiv:1305.4450

[3] C. Costermans and Hoang Ngoc Minh.– *Noncommutative algebra, multiple harmonic sums and applications in discrete probability*, J. of Sym. Comp. (2009), pp. 801-817.

[4] J-Y. Enjalbert, Hoang Ngoc Minh.– *Combinatorial study of Hurwitz colored polyzêtas*, Discrete Mathematics, 1. 24 no. 312 (2012), p. 3489-3497.

[5] T.Q.T. Lê and J. Murakami.– *Kontsevich's integral for Kauffman polynomial*, Nagoya Math., pp 39-65, 1996.

[6] Hoang Ngoc Minh, M. Petitot and J. Van der Hoeven.– *Computation of the monodromy of generalized polylogarithms*, ISSAC'98, Rostock, Allemagne, Aug. 1998.

[7] Hoang Ngoc Minh.– *On a conjecture by Pierre Cartier about a group of associators*, to appear (2013).

[8] M. Hoffman.– *Quasi-shuffle products*, J. of Alg. Cominatorics, 11 (2000), pages 49-68.

[9] C. Reutenauer.– *Free Lie Algebras*, Lon. Math. Soc. Mono, New Series-7, Oxford Sc. Pub., 1993.

[10] D. Zagier.– *Values of zeta functions and their applications*, in "First European Congress of Mathematics", vol. 2, Birkhäuser (1994), pp. 497-512.

Fast parallel GCD algorithm of many integers

Sidi M. SEDJELMACI
LIPN, CNRS UMR 7030
University of Paris-Nord
Av. J.-B. Clément, 93430 Villetaneuse, France
sms@lipn.univ-paris13.fr

Abstract: We present a new parallel algorithm which computes the GCD of n integers of $O(n)$ bits in $O(n/\log n)$ time with $O(n^{2+\epsilon})$ processors, for any $\epsilon > 0$ on CRCW PRAM model.

The computation of the GCD of two integers is not known to be in the NC parallel class, nor it is known to be P-complete [1]. The best parallel performance was first obtained by Chor and Goldreich [2], then by Sorenson [7] and Sedjelmaci [5] since they propose, with different approaches, parallel integer GCD algorithms which can be achieved in $O(n/\log n)$ time with $O(n^{1+\epsilon})$ number of processors, for any $\epsilon > 0$, in PRAM CRCW model. A naive approach, using a binary tree computation to compute the GCD of n integers of $O(n)$ bits would require $O(n)$ parallel time with $O(n^{2+\epsilon})$ processors. One may also use the existing parallel GCD algorithms of two integers and try to adapt them to design a GCD for many integers. However, it is not obvious how to find a parallel GCD for n integers which conserve the same $O(n/\log n)$ time, with $O(n^{2+\epsilon})$ processors, which is roughly the bit-size of all the n input integers. In this paper, we prove that we can compute the GCD of n integers of $O(n)$ bits, in only $O(n/\log n)$ parallel time with $O(n^{2+\epsilon})$ processors, for any $\epsilon > 0$ on CRCW PRAM model, in the worst case. Another probabilistic approach is given in [3]. To our knowledge, it is the first deterministic algorithm which computes the GCD of many integers with this parallel performance and polynomial work. Our algorithm, called Δ-GCD is the following:

Input: A set $A = \{ a_0, a_1, \cdots, a_{n-1} \}$ of n distinct positive integers, $a_i < 2^n$, with $n \geq 4$.
Output: $\gcd(a_0, a_1, \cdots, a_{n-1})$.

$\alpha := a_0\,;\ I := 0\,;\ p := n\,;$
While $(\alpha > 1)$ **Do**
  For $(i = 0)$ **to** $(n-1)$ **ParDo**
    If $(0 < a_i \leq 2^n/p)$ **Then** $\{\ \alpha := a_i\,;\ I := i\,;\ \}$
  Endfor
  If $(\alpha > 2^p/n)$ **Then** /* Compute in parallel I, J and α */
    $\alpha := \min \{\, | a_i - a_j | > 0 \,\} = a_I - a_J$; $a_I := \alpha$;
  Endif
  For $(i = 0)$ **to** $(n-1)$ **ParDo** /* Reduce all the a_i's */
    If $(i \neq I)$ **Then** $a_i := a_i \bmod \alpha$;
  Endfor /* $\forall\, i\,,\ 0 \leq a_i \leq \alpha$ */
  If $(\forall\, i \neq I\,,\ a_i = 0\,)$ **Then Return** α ; /* Here $\alpha = \gcd(a_0, \cdots, a_{n-1})$ */
  $p := np\,;$
Endwhile
Return α.

We use a weak version of the function min based the pigeonhole principle, where only the $O(\log n)$ leading bits of the integers are considered. The integer α is, at each while iteration, $O(\log n)$ bits less. More details for the computations of I, J and α are given in [6], as well as a first C program checking the correctness of the Δ-GCD algorithm.

Theorem : The $\Delta - GCD$ algorithm computes in parallel the GCD of n integers of $O(n)$ bits in length, in $O(n/\log n)$ time using $O(n^{2+\epsilon})$ processors on CRCW PRAM model, with $\epsilon > 0$.

Proof: (Sketch, see [6]). The algorithm terminates after $O(n/\log n)$ loop iterations. Let t_i be the time cost at iteration i, $1 \leq i \leq N$, with $N = O(n/\log n)$. Let k_i be the maximum bit length of all the quotients $q_j = \lfloor a_j/\alpha \rfloor$, with $\sum_{i=1}^{N} k_i \leq n$. We prove that $t_i = O(\min\{\frac{k_i}{\log n}, \log n\})$. The total number of processors is $n \times O(n^{1+\epsilon}) = O(n^{2+\epsilon})$ and the parallel time is then $t(n) = \sum_{i=1}^{N} t_i = \sum_{i=1}^{N} \min(\{\frac{k_i}{\log n}, \log n\}) = \sum_{k_i < \log n} 1 + \sum_{\log n < k_i < \log^2 n} \frac{k_i}{\log n} + \sum_{k_i > \log^2 n} \log n = O(n/\log n)$. □

A Blankinship-like algorithm can be easily designed to compute Extended GCD, and an upper bound of the multipliers [4] could be considered as well. A slightly modified Rosser's algorithm (pivoting with α) can be used to solve linear Diophantine equations. Moreover, a $O(n^2/\log n)$ sequential version of Δ-GCD should be considered with precomputed lookup tables for arithmetic operations on $O(\log n)$ bit integers.

References

[1] A. Borodin, J. von zur Gathen and J. Hopcroft, Fast parallel matrix and GCD computations, *Information and Control,* 52, 3, 1982, 241–256

[2] B. Chor and O. Goldreich, An improved parallel algorithm for integer GCD, Algorithmica, 5, 1990, 1-10

[3] G. Cooperman, S. Feisel, J. von zur Gathen and G. Havas, GCD of many integers, *Lect. Notes in Comp. Sci., Springer-Verlag, Berlin,* 1627 (1999), 310–317

[4] G. Havas, S. Majewski, Extended gcd calculation, *Congressus Numerantium, 111,* 1627 (1998), 104-114

[5] S.M. Sedjelmaci, On A Parallel Lehmer-Euclid GCD Algorithm, in Proc. of the International Symposium on Symbolic and Algebraic Computation (ISSAC'2001), 2001, 303-308

[6] S.M. Sedjelmaci, Fast Parallel GCD algorithm of many integers, lipn.univ-paris13.fr/∼sedjelmaci, Rapport interne, LIPN, April, 2013

[7] J. Sorenson, Two Fast GCD Algorithms, J. of Algorithms, 16, 1994, 110-144

ACM Communications in Computer Algebra, Vol 47, No. 3, Issue 185, September 2013

Gelfand-Kirillov dimensions of differential difference modules via Gröbner bases

Xiangui Zhao
Department of Mathematics, University of Manitoba
Winnipeg, Canada, R3T 2N2
umzha493@cc.umanitoba.ca

Introduction. Differential-difference algebras were defined by Mansfield and Szanto in [5], which arose from the calculation of symmetries of discrete systems (c.f., [2]). Mansfield and Szanto developed the Gröbner basis theory of differential difference algebras over a field by using a special kind of left admissible orderings (which they called differential difference orderings). We generalize the main results of [5] to any left admissible ordering, and apply the generalized results to compute the Gelfand-Kirillov dimensions of cyclic differential difference modules.

Definition of differential difference algebras. Let k be a field, R be a k-algebra and integers $m, n \geq 1$. Suppose that $R[D; \mathrm{id}, \delta] = R[D_1; \mathrm{id}, \delta_1] \cdots [D_n; \mathrm{id}, \delta_n]$ and $R[S; \sigma, 0] = R[S_1; \sigma_1, 0] \cdots [S_m; \sigma_m, 0]$ are two Ore algebras ([5]) such that $\sigma_i \circ \delta_j = \delta_j \circ \sigma_i$ for $1 \leq i \leq m, 1 \leq j \leq n$. Furthermore, suppose that each $\sigma_i : R \to R$, $1 \leq i \leq m$, can be extended to a k-algebra automorphism $\sigma_i : R[D; \mathrm{id}, \delta] \to R[D; \mathrm{id}, \delta]$ such that $\sigma_i(D_j) = \sum_{l=1}^{n} a_{ijl} D_l$, $a_{ijl} \in R$. Let F be the free R-R bi-module with basis $\{S_1, \ldots, S_m, D_1, \ldots, D_n\}$, T be the tensor algebra on F over R, and K be the two-sided ideal in T generated by the set of the following elements of T:
(1) $D_i r - r D_i - \delta_i(r), 1 \leq i \leq n, r \in R$; (2) $S_i r - \sigma_i(r) S_i, 1 \leq i \leq m, r \in R$;
(3) $S_i S_j - S_j S_i, 1 \leq i, j \leq m$; (4) $D_i D_j - D_j D_i, 1 \leq i, j \leq n$;
(5) $D_i S_j - S_j \sigma_j(D_i), 1 \leq i \leq n, 1 \leq j \leq m$.
Then the R-algebra T/K, denoted by $R[D; \mathrm{id}, \delta][S; \sigma, 0]$, is called a *differential difference algebra* of type (m, n), or DD-algebras for short.

DD-algebras are generalizations of commutative polynomial algebras, Ore extensions, skew polynomials of derivation (or automorphism) type, and quantum planes. Since elements in S do not commute with those in D in general, DD-algebras are different from difference-differential rings (see, e.g., [6]). The following example distinguishes DD-algebras from algebras of solvable type [3], or PBW extensions [1], or G-algebras [4].

Example. Let $A = k[D; \mathrm{id}, 0][S; \sigma, 0]$ be a DD-algebra of type $(1, 2)$ with $\sigma_1(D_1) = D_2$ and $\sigma_1(D_2) = D_1$. Then $D_1 S_1 = S_1 D_2$ and $D_2 S_1 = S_1 D_1$. Hence A is not an algebra of solvable type (or a PBW extension, or a G-algebra).

Gröbner bases of DD-algebras. We only consider the special case when $R = k$. From now on, let $A = k[D; \mathrm{id}, \delta][S; \sigma, 0]$ be a DD-algebra. Then, it is easy to see that $\delta = 0$ and $\sigma|_k = \mathrm{id}$. Thus $A = k[D; \mathrm{id}, 0][S; \sigma, 0]$ and $\sigma|_k = \mathrm{id}$. One can prove that the set $\mathcal{M} = \{S^\alpha D^\beta : \alpha \in \mathbb{N}^m, \beta \in \mathbb{N}^n\}$ is a k-basis of A. Let $u = S^\alpha D^\beta \in \mathcal{M}$, $\alpha = (\alpha_1, \ldots, \alpha_m) \in \mathbb{N}^m$ and $\beta = (\beta_1, \ldots, \beta_n) \in \mathbb{N}^n$. Then the *(total) degree* of u is defined as $\deg(u) = \alpha_1 + \cdots + \alpha_m + \beta_1 + \cdots + \beta_n$, and the degree of u with respect to S_i (D_j, respectively) is defined as $\deg_{S_i} = \alpha_i$ ($\deg_{D_j} = \beta_j$, respectively).

For any given well ordering on $\mathcal{M}$ and $f = c_1u_1 + \cdots + c_tu_t \in A$ ($0 \neq c_i \in k$, $u_i \in \mathcal{M}$, $1 \leq i \leq t$) with $u_1 > \cdots > u_t$, the *leading monomial* of f is denoted by $\mathrm{lm}(f) = u_1$. A *DD-monomial ordering* on $\mathcal{M}$ is a well ordering $>$ on $\mathcal{M}$ such that if $S^\alpha D^\beta > S^{\alpha'} D^{\beta'}$ and $f \in A \setminus k$, then $\mathrm{lm}(fS^\alpha D^\beta) > \mathrm{lm}(fS^{\alpha'} D^{\beta'})$. Note that DD-monomial orderings are more general than differential difference orderings defined in [5].

Let $f, g \in A$. If there exists $h \in A$ such that $f = hg$, we say that f is *right divisible* by g.

Let $>$ be a DD-monomial ordering on $\mathcal{M}$ and I be a left ideal of A. A finite set $G \subseteq A$ is called a (finite) *left Gröbner basis* of I with respect to $>$ if G satisfies: (i) G generates I as a left ideal of A; and (ii) For any $0 \neq f \in I$, there exists $g \in G$ such that $\mathrm{lm}(f)$ is right divisible by $\mathrm{lm}(g)$.

Similarly as in [5], we can define reductions and S-polynomials. Then the reduction algorithm and the left Gröbner basis algorithm still work under a DD-monomial ordering. We have

Theorem 1 *Let $G \subseteq A$ be a finite set and I be the left ideal of A generated by G. Then G is a left Gröbner basis of I if and only if* $\mathrm{Spoly}(g_1, g_2) \to_G 0$ *for any $g_1, g_2 \in G$.*

It can be proved that the Hilbert basis theorem is valid for DD-algebras: every left ideal of A is finitely generated. Thus we have

Theorem 2 *Every left ideal of a DD-algebra $k[D; \mathrm{id}, \delta][S; \sigma, 0]$ has a (finite) left Gröbner basis.*

Gelfand-Kirillov dimension of cyclic A-modules. For convenience, let $x_i = S_i, x_{m+j} = D_j$ for $1 \leq i \leq m, 1 \leq j \leq n$ and let $l = m + n$. Denote $X^\alpha = x_1^{\alpha_1} x_2^{\alpha_2} \cdots x_l^{\alpha_l}$ for $\alpha = (\alpha_1, \ldots, \alpha_l) \in \mathbb{N}^l$. Then $\mathcal{M} = \{X^\alpha : \alpha \in \mathbb{N}^l\}$. For $u = X^\alpha \in \mathcal{M}$ and $p \in \mathbb{N}$, define $\mathrm{top}_p(u) = \{i : 1 \leq i \leq l, \alpha_i \geq p\}$ and $\mathrm{sh}_p(u) = X^\beta$, where $\beta_i = \min\{p, \alpha_i\}, 1 \leq i \leq l$.

Then we have the following theorem which computes the Gelfand-Kirillov dimension of a cyclic DD-module.

Theorem 3 *Let I be a left ideal of A and G be a left Gröbner basis of I with respect to a total degree DD-monomial ordering. Set $p = \max\{\deg_{x_i}(\mathrm{lm}(g)) : g \in G, 1 \leq i \leq l\}$. Then*

$$\mathrm{GKdim}(M) = \max\{|\,\mathrm{top}_p(u)| : \mathrm{sh}_p(u) = u\}.$$

References

[1] A. D. Bell and K. R. Goodearl. Uniform rank over differential operator rings and Poincaré-Birkhoff-Witt extensions. *Pacific J. Math*, 131(1):13–37, 1988.

[2] P. E. Hydon. Symmetries and first integrals of ordinary difference equations. *Proceedings of the Royal Society of London (series A)*, 456:2835–2855, 2000.

[3] A. Kandri-Rody and V. Weispfenning. Non-commutative Gröbner bases in algebras of solvable type. *Journal of Symbolic Computation*, 9(1):1–26, 1990.

[4] V. Levandovskyy. *Non-commutative Computer Algebra for polynomial algebras: Gröbner bases, applications and implementation.* PhD thesis, University of Kaiserslautern, 2005.

[5] E. L. Mansfield and A. Szanto. Elimination theory for differential difference polynomials. In *Proceedings of the 2003 international symposium on symbolic and algebraic computation*, pages 191–198. ACM, 2003.

[6] Meng Zhou and Franz Winkler. Gröbner bases in difference-differential modules. In *Proceedings of the 2006 international symposium on Symbolic and algebraic computation*, pages 353–360. ACM, 2006.

Newton-like Iteration for Determinantal Systems and Structured Low Rank Approximation

Éric Schost and Pierre-Jean Spaenlehauer
Western University, Department of Computer Science
London, Ontario, Canada
eschost@uwo.ca, pierre-jean.spaenlehauer@m4x.org

Problem statement. Let $\mathcal{M}_{p,q}(\mathbb{R})$ be the space of $p \times q$ matrices with real entries, $r \in \mathbb{N}$ be an integer, $V_r \subset \mathcal{M}_{p,q}(\mathbb{R})$ be the determinantal variety of matrices of rank at most r and E be a linear (or affine) subspace of $\mathcal{M}_{p,q}(\mathbb{R})$ (*e.g.* Toeplitz, Hankel, Sylvester matrices). Given a matrix $M \in E$, the goal is to compute a close matrix in $E \cap V_r$. More precisely, we want a numerical algorithm computing a function $\varphi : E \longrightarrow E$ such that, if M is close enough to $E \cap V_r$, then the sequence defined by $M_0 = M$, $M_{i+1} = \varphi(M_i)$ converges quadratically towards a matrix $M_\infty \in E \cap V_r$. As shown in [5], this problem which is also known as *Structured Low-Rank Approximation* (SLRA) is central in data fitting or in numerical analysis. It is also underlying classical symbolic-numeric problems.

Main results. We propose a Newton-like algorithm (`NewtonSLRA`) which answers the above specification and appears to converge quadratically. The main principle of this algorithm is close to Cadzow's algorithm [1] which proceeds by a sequence of Singular Value Decompositions (SVD) and orthogonal projections on E. However, we choose a direction of projection which is tangent to the determinantal variety in order to ensure quadratic convergence. Each iteration of the algorithm `NewtonSLRA` computes a function $\varphi(M)$ in three main steps: (1) compute a rank r approximation $\widetilde{M}$ of M; (2) from the left and right kernels of $\widetilde{M}$, compute a set of generators of the tangent space $T_{\widetilde{M}}V_r$; (3) compute the point in $E \cap T_{\widetilde{M}}V_r$ which minimizes the distance to $\widetilde{M}$ (this is achieved by solving a linear least squares problem). Computing the best rank r approximation with respect to the Frobenius norm is achieved by the SVD. It also provides an orthonormal basis (for the scalar product $\langle M_1, M_2 \rangle = \mathtt{tr}(M_1 \cdot {}^T M_2)$) of the normal space $N_{\widetilde{M}}V_r = \mathrm{Ker}_L(\widetilde{M}) \otimes \mathrm{Ker}_R(\widetilde{M})$ which is used for computing the projection on E. The most expensive step is the SVD which is achieved in $O(pq \min(p,q))$ operations in fixed precision. The main theoretical result lies in the following theorem which ensures the local quadratic convergence towards a matrix $M_\infty \in V_r \cap E$ near the optimal solution, under conditions on the dimensions of $\dim(E)$ and $\dim(V_r)$. To the best of our knowledge, this is the first proof of quadratic convergence of an iterative method for the SLRA problem:

Theorem 1. *If* $\dim(E) = \dim(V_r)$ *and* $\dim(E) + \dim(V_r) > pq$*, then the algorithm* `NewtonSLRA` *computes a function* $\varphi : E \to E$ *verifying the following property: for all* $\mu > 1$ *and for all* $\hat{M} \in V_r \cap E$ *such that* V_r *and* E *verify mild transversality conditions at* $\hat{M}$*, there exists* $\epsilon > 0$ *such that for all* M_0 *with* $\|M_0 - \hat{M}\| < \epsilon$*, the sequence* $M_{i+1} = \varphi(M_i)$ *converges towards a matrix* $M_\infty \in V_r \cap E$ *and*

$$\|M_i - M_\infty\| \leq \left(\frac{1}{2}\right)^{2^i - 1} \|M_0 - M_\infty\| \quad \textit{and} \quad \|M_0 - M_\infty\| \leq \mu \|M_0 - \hat{M}\|.$$

The proof relies on tools from Smale's α-theory, slightly modified to take into account the properties of this Newton-like iteration.

Application to univariate approximate GCD. Approximate GCD computation is a symbolic-numeric example of SLRA problem: a degree condition on the GCD of univariate polynomials amounts to

a rank condition on their Sylvester matrix. In this setting, the algorithm takes as input two floating-point polynomials f, g of degrees m and n, and an integer $d \in \mathbb{N}$; it outputs three floating-point polynomials a, b, h of respective degrees $m-d$, $n-d$, d such that $\|f-ah\|^2 + \|g-bh\|^2$ is small. Here, E is the linear space of truncated Sylvester matrices (see *e.g.* [6]) and V_r is the variety of rank deficient matrices of sizes $(m+n-d+1) \times (m+n-2d+2)$. We compare in Table 1 our `Maple` implementation of `NewtonSLRA` with the `Maple` implementation of `GPGCD` [6], which is a state-of-the art algorithm dedicated to the computation of approximate GCDs. Instances are constructed by generating two random polynomials $\widetilde{f}$, $\widetilde{g}$ such that $\deg(\mathrm{GCD}(\widetilde{f},\widetilde{g})) = d$ and by adding a random error polynomial f_ϵ, g_ϵ such that the relative noise $\sqrt{\|f_\epsilon\|^2 + \|g_\epsilon\|^2}/\sqrt{\|\widetilde{f}\|^2 + \|\widetilde{g}\|^2}$ is equal to a fixed parameter ϵ. The column "perturbation" gives the relative distance between the output and the input of the algorithms. Notice that `NewtonSLRA` performs almost as well as `GPGCD`, which relies on optimization techniques to minimize the function $\|f-ah\|^2+\|g-bh\|^2$. In comparison, `NewtonSLRA` does not converge to the minimum of this function, but we see in Table 1 that the distance to the optimum is small. Also, experimental results indicate that `NewtonSLRA` converges quadratically (although $\dim(E)$ and $\dim(V_r)$ do not verify the assumptions of theorem 1), whereas `GPGCD` converges linearly (see the right part of table 1 for an example). We also tried to use directly the `QRGCD` routine from the package `SNAP` in `Maple` [3] but it failed to find an approximate GCD in our examples because of the high level of noise in the coefficients of the input polynomials.

	`NewtonSLRA`		`GPGCD`	
(m,n,d,ϵ)	time	perturbation	time	perturbation
(100, 100, 50, 0.001)	0.803s	4.838e-4	0.806s	4.742e-4
(500, 500, 250, 0.001)	37.5s	5.127e-4	45.4s	4.923e-4
(1000, 1000, 500, 0.001)	282s	5.781e-4	317s	5.155e-4
(2000, 2000, 1000, 0.0001)	1567s	5.104e-5	1161s	5.088e-5

	sizes of iteration steps	
iteration	`NewtonSLRA`	`GPGCD`
1	0.9e-1	0.9e-1
2	0.5e-3	0.5e-3
3	0.6e-8	0.2e-5
4	0.1e-17	0.8e-8
5	0.1e-36	0.4e-10

Table 1: Comparison between `GPGCD` [6] and `NewtonSLRA` for computing approximate GCDs

Other applications of SLRA in symbolic-numeric computations and future work. Several other algebraic problems are characterized by rank conditions on structured matrices, for which these techniques could lead to symbolic-numeric algorithms, *e.g.* solving bilinear systems, computing the minimal polynomial of algebraic power series or computing low degree Pade approximants. Moreover, there is still room for improvement: the most computationally-intensive step of this algorithm is the computation of the SVD, but the algorithm converges quadratically even when less precise rank-approximation techniques are used. Also, we plan to compare our method and implementation with other algorithms for SLRA (see *e.g.* [2] and references therein) and for computing approximate GCDs (see *e.g* [4], which relies on the *Structured Total Least Norm* approach). The main challenge is to extend theorem 1 by relaxing the restrictions on $\dim(E)$ and $\dim(V_r)$.

References

[1] J. A. Cadzow. Signal enhancement-a composite property mapping algorithm. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 36(1):49–62, 1988.

[2] M. T. Chu, R. E. Funderlic, and R. J. Plemmons. Structured Low Rank Approximation. *Linear algebra and its applications*, 366:157–172, 2003.

[3] R. M. Corless, S. M. Watt, and L. Zhi. QR factoring to compute the gcd of univariate approximate polynomials. *Signal Processing, IEEE Transactions on*, 52(12):3394–3402, 2004.

[4] E. Kaltofen, Z. Yang, and L. Zhi. Structured low rank approximation of a Sylvester matrix. In *Symbolic-numeric computation*, pages 69–83. Springer, 2007.

[5] I. Markovsky. Structured low-rank approximation and its applications. *Automatica*, 44(4):891–909, 2008.

[6] A. Terui. An iterative method for calculating approximate gcd of univariate polynomials. In *ISSAC 2009*, pp. 351–358.

A Verification Framework for *MiniMaple* Programs*

Muhammad Taimoor Khan and Wolfgang Schreiner
Dokotratskolleg Computational Mathematics (DK) and
Research Institute for Symbolic Computation (RISC)
Johannes Kepler University, Linz, Austria
muhammad.khan@dk-compmath.jku.at
Wolfgang.Schreiner@risc.jku.at

In this poster, we present an overview of our ongoing work and results on the development of a verification framework for programs written in a (substantial) subset of the language of the computer algebra system Maple, which we call *MiniMaple*. The main goal here is to detect behavioral errors in such programs w.r.t. their specifications by static analysis. However, the task of the formal specification and verification of *MiniMaple* programs is complex as Maple supports various non-standard types of objects such as unevaluated expressions and also requires abstract data types to formalize computer algebra concepts and notions. To approach our goal, we have defined and formalized the syntax, semantics, type system and specification language for *MiniMaple*. For the verification, we translate an annotated *MiniMaple* program into the language Why3ML of the intermediate verification tool Why3 [1] developed at LRI, France. We generate verifi! cation conditions by the corresponding component of Why3 and then prove the correctness of these conditions by various automatic and interactive theorem provers supported by the Why3 back-end. The main test for our verification framework is the Maple package *DifferenceDifferential* [2] developed at our institute to compute bivariate difference-differential polynomials using relative Gröbner bases. All software (lexer, parser, type checker and translator) is open source and freely available from `http://www.risc.jku.at/people/mtkhan/dk10/`.

As a general overview of our verification framework, first any *MiniMaple* program is parsed to generate an abstract syntax tree (AST). Then the AST is type checked and annotated by type information and translated into a (presumably) semantically equivalent Why3ML program. From this program, Why3 generates verification conditions to be proved correct by its various back-end supported provers. All components of the framework may generate errors and information messages.

The syntax of *MiniMaple* [3] covers all the syntactic domains of Maple but supports fewer alternatives in each domain than Maple; in particular, Maple has many built-in expressions which are not supported in our language. We use the type annotations which Maple introduced for runtime checking for the purpose of the static type checking of *MiniMaple* programs; indeed we have defined a formal type system for *MiniMaple* as a decidable logic with various typing judgments. The type system requires that procedure parameters, procedure results and local variables are type annotated. However, global variables in Maple cannot be type annotated, such that values of arbitrary types can be assigned to them. To handle the correct semantics of such variables inside and outside of the body of procedures, we introduced *global* and *local* contexts. In the former, variables can be introduced by assignments and their types can change arbitrarily, while i! n the latter, variables can only be introduced by declarations and their types can only be specialized [3]. Another issue is the handling of dynamic type tests by the *MiniMaple* expression **type**(E,T). The use of a type test in a conditional may result in different type information for the same variable in different branches of the conditional; we use the type information introduced by the corresponding conditional branches to infer

*The research was funded by the Austrian Science Fund (FWF): W1214-N15, project DK10.

the possible type of a variable. We have applied the type checker to the package *DifferenceDifferential*; no crucial errors were found but some bad code parts, e.g. duplicate declaration of variables and global variables that are declared but not used.

Furthermore, we have defined a specification language for *MiniMaple* to formally describe mathematical theories (types, functions, axioms), behavior of procedures (pre- and post-conditions and other constraints), loops (invariants and termination terms) and commands (assertions). In addition to basic formulas, our specification language supports various forms of quantifiers, i.e. logical quantifiers (**forall** and **exists**), numerical quantifiers (**add**, **mul**, **max** and **min**) and sequential quantifiers (**seq**) to represent truth values, numeric values and sequences of values respectively. The language slightly extends the Maple syntax, e.g. logical quantifiers use typed variables and numerical quantifiers use logical conditions that filter values from the specified range of a variable. The language supports abstract data types to specify abstract mathematical concepts, e.g. polynomial rings. As an example, we have f! ormally specified a substantial part of the package *DifferenceDifferential*, e.g. difference-differential operators are formalized by a corresponding abstract data type.

To verify a *MiniMaple* program annotated with types and specifications, we translate this program to the language Why3ML of the intermediate verification tool Why3. We use the Why3 verification conditions generator to produce a set of verification conditions: the pre-conditions of called procedures, the post-conditions of defined procedures, the initial establishing of loop invariants, the preservation of loop invariants after every iteration and the decreasing of termination terms. We then prove their correctness by automated provers (e.g. Z3 and CVC3) and proof assistants (e.g. Coq) supported by the Why3 back-end. The wide range of proof support was one the reasons why we chose Why3, as we are, e.g., dealing with non-linear arithmetic which requires in general an interactive prover. For verification, we have defined the translation of *MiniMaple* into semantically equivalent constructs of Why3ML, e.g. the *MiniMaple* return statement is translated using t! he Why3 exception-handling mechanism, union types are translated to algebraic types and the corresponding type tests are translated using pattern matching. Using this approach, we have already verified most of the low level procedures of the package *DifferenceDifferential*, e.g. “gleicheterme” (comparing two difference-differential terms), “sigmamax” (computing a differential term with given constraints) and “ddsub” (subtraction of differential operators).

Currently, we are in the process of verifying higher level procedures with abstract (data type based) specifications: based on an example we experiment with appropriate proof strategies for such specifications using our verification framework. As a next step, a proof of the soundness of translation for selected *MiniMaple* constructs is planned. One of the reason for choosing Why3 was that it provides a formal (originally weakest precondition based, later also operational) semantics. We have correspondingly defined a formal denotational semantics of *MiniMaple* programs [3], e.g. the semantics of command execution is defined as a state relationship between pre- and post-states, i.e. the *MiniMaple* command semantics $[[C]](e)(s, s')$ states that in an environment e the execution of a command C in a pre-state s may result in a post-state s'. Based on these definitions, we plan to prove that our translation preserves the programs' semantics.

References

[1] F. Bobot and et al. Why3: Shepherd Your Herd of Provers. In *Boogie 2011: First International Workshop on Intermediate Verification Languages*, Wrocław, Poland, August 2011.

[2] Christian Dönch. Bivariate Difference-Differential Dimension Polynomials and Their Computation in Maple. Technical report, Research Institute for Symbolic Computation, University of Linz, 2009.

[3] M. T. Khan and W. Schreiner. Towards the Formal Specification and Verification of Maple Programs. In J. Jeuring and et al., editors, *Intelligent Computer Mathematics*, volume 7362 of *LNCS*, pages 231–247. Springer, 2012.

Relaxing Order Basis Computation

Pascal Giorgi and Romain Lebreton
LIRMM, CNRS-UM2 France
pascal.giorgi@lirmm.fr, romain.lebreton@lirmm.fr

The computation of an order basis (also called sigma basis in [3]) is a fundamental tool for linear algebra with polynomial coefficients. Such a computation is one of the key ingredients to provide algorithms which reduce to polynomial matrices multiplication. This has been the case for column reduction [3] or minimal nullspace basis [11] of polynomial matrix over a field. In this poster, we are interested in the application of order basis to compute minimal matrix generators of a linear matrix sequence (see [9]). In particular, we focus on the linear matrix sequence used in the Block Wiedemann algorithm [1].

As of today, the fast order basis algorithm PM-Basis from [3] suffers from two issues. In our applications, the bound σ on its degree may be pessimistic and therefore we need to use early termination. However the recursive aspect of PM-Basis is unhelpful to implement such an early termination. Also PM-Basis may require to know more coefficients of F than necessary. This can hinder the complexity when the cost of computing coefficients of the entry is dominant. This is the case for instance for the block Wiedemann algorithm which motivates this work.

Main results In this work we propose a relaxed variant of the PM-Basis algorithm. The property of relaxed algorithms is that they do not require more knowledge on the input than necessary while keeping a quasi-optimal complexity in the order σ.
We first propose an iterative variant Iterative-PM-Basis of PM-Basis which is more suited to the relaxed model and also to early termination. Then we show how to relax Iterative-PM-Basis *via* the use of a relaxed polynomial matrix multiplication algorithm. Thus we obtain our relaxed order basis computation within the complexity of PM-Basis with only an extra logarithmic factor in σ. Finally, we show the benefit of this algorithm to gain a constant factor on average on the block Wiedemann algorithm.

Order basis algorithms Let $\mathbb{K}$ be a field, $F = \sum_{i \geqslant 0} F_i x^i \in \mathbb{K}[[x]]^{m \times n}$ a matrix of power series, σ a positive integer and (F, σ) be the $\mathbb{K}[x]$-module defined by the set of $v \in \mathbb{K}[x]^{1 \times m}$ such that $vF \equiv 0 \bmod x^\sigma$. A polynomial matrix P is a (left) order basis of F of order σ and shift $\vec{s}$ if the rows of P form a basis of (F, σ) and P is $\vec{s}$-row reduced (see [10] for details). Without loss of generality we only consider in this poster the case $n = O(m)$ with a balanced shift $\vec{s}$ as in [3]. Indeed the techniques of [10] allow to reduce the general case to our particular case.

Two different algorithms presented in [3] compute an order basis P of F. The M-Basis algorithm works iteratively on the order σ to compute the order basis P. It is a lazy algorithm that costs $O(m^\omega \sigma^2)$ arithmetic operations in $\mathbb{K}$,

Algorithm 1: Iterative-PM-Basis

Input: $F \in \mathbb{K}[[x]]^{m \times n}$, $\sigma > 0$, $\vec{s} \in \mathbb{N}^m$
Output: $P \in \mathbb{K}[x]^{m \times n}$ such that P is a $\vec{s}$-row reduced order basis of (F, σ)

1: $P_0, \vec{u} :=$ M-Basis$(F \bmod x,\ 1,\ \vec{s}\ )$; $P := [P_0]$; $S := [0, \ldots, 0, F]$ with $\lceil \log_2(\sigma) \rceil$ zeros
2: **for** $k = 1$ **to** $\sigma - 1$ **do**
3: $\quad \ell := \nu_2(k)$; $\ell' := \begin{cases} \lceil \log_2(\sigma) \rceil & \text{if } k = 2^\ell \\ \nu_2(k - 2^\ell) & \text{otherwise} \end{cases}$
4: $\quad$ Update P by merging its first $\ell + 1$ elements by multiplication //Product tree of step 4)
5: $\quad S[\ell + 1] :=$ MiddleProduct$(P[1], S[\ell' + 1], 2^\ell)$ //Update of the series of step 2)
6: $\quad P_k, \vec{u} :=$ M-Basis$(S[\ell + 1] \bmod x, 1, \vec{u}\ )$ //Recursive calls on leafs of steps 1) and 3)
7: $\quad$ Insert P_k at the beginning of P
8: **return** $\prod_i P[i]$

i.e. it only requires the coefficients F_j of F for $0 \leqslant j \leqslant (i-1)$ for computing the intermediate order basis of order i. The PM-Basis algorithm uses a divide-and-conquer approach on the order σ to reduce the arithmetic complexity to $O(m^\omega \mathsf{M}(\sigma)\log(\sigma)) = O\tilde{}(m^\omega\sigma)$, where M denotes the arithmetic complexity of polynomial multiplication. Roughly speaking, the algorithm is made of four steps: 1) a recursive call to compute an order basis P_{low} of F of order $\sigma/2$, 2) an update of the problem *via* the middle product $F' := (x^{-\sigma/2}P_{\text{low}}F) \bmod x^{\sigma/2}$, 3) a recursive call to compute an order basis P_{high} of F' of order $\sigma/2$ and 4) return the order basis $P_{\text{high}}P_{\text{low}}$ of F of order σ. Step 2) implies that one may need at most twice as much coefficients of the input series than necessary to go from an intermediate order basis of order i to $i+1$.

Fast iterative order basis Let us give an iterative version of PM-Basis. Our algorithm performs exactly the same operations on matrices as PM-Basis when σ is a power of two. This iterative presentation of PM-Basis is original. Let us denote $\nu_2(k)$ the valuation in 2 of any integer k and index our lists from 1.

Relaxing order basis algorithm In algorithm PM-Basis, we have noticed that only the middle product of step 5 reads more entries of F than necessary at step k. Let us perform this step differently so that it reads at most the coefficients $F_0, \ldots, F_{k-1}$ of F at step k. This property is called a *relaxed* (or on-line) algorithm w.r.t. F.

A naive approach would be to compute a full $2n \times n$ product using a relaxed multiplication algorithm on polynomial of matrices ([2, 5, 6, 4, 8]) in time $\mathsf{R}(n) = O(\mathsf{M}(n)\log(n))$ [2]. We propose another relaxed algorithm that gains asymptotically a factor 2 compared to the full $2n \times n$ relaxed product. We decompose the relaxed middle product in a normal high product (in black) followed by a multiplication (in white and gray) relaxed w.r.t. only b using [4] in this example (see Figure 1).

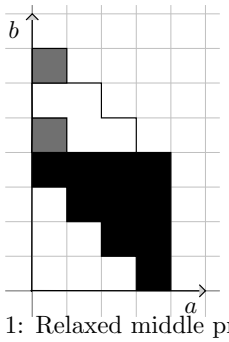


Figure 1: Relaxed middle product

Using this relaxed middle product algorithm within Iterative-PM-Basis we obtain an order basis algorithm Relaxed-PM-Basis relaxed w.r.t. F. This relaxed order basis algorithm costs $O(k^\omega \mathsf{R}(\sigma)\log(\sigma)) = O(k^\omega \mathsf{M}(\sigma)\log^2(\sigma))$.

Application to block Wiedemann algorithm Let $A \in \mathrm{GL}_N(\mathbb{K})$ with $O(N)$ non-zero elements. Block Wiedemann approach uses a minimal matrix generator of the matrix series $S = \sum_{i\in\mathbb{N}} UA^iVx^i$ for any random $U, V^T \in \mathbb{K}^{m\times N}$ in order to solve a linear system $Ax = b \in \mathbb{K}^N$. As described in [9], this matrix generator can be obtained from an order basis of $F = [S \mid I_m]^T \in \mathbb{K}[[x]]^{2m\times m}$. We can derive a bound on the maximal degree δ of this order basis using the stopping criteria of [7, Th. 4.19]. Since this bound may be loose, a constant factor in the complexity can often be saved using an early termination in the order basis algorithm.

We compare the complexity of Iterative-PM-Basis and Relaxed-PM-Basis in this setting. Computing S at precision σ costs $O(k^{\omega-1}N\sigma)$. In practice $k \ll N$ so that the cost of computing S always dominates the cost of (relaxed) order basis algorithm.

Assume that δ is uniformely distributed between 2^p+1 and 2^{p+1} for $p \in \mathbb{N}$. Iterative-PM-Basis requires the coefficients $F_0, \ldots, F_{2^{p+1}-1}$ whereas Relaxed-PM-Basis only asks for $F_0, \ldots, F_{\delta-1}$. Therefore our relaxed approach improves the dominant cost of computing F in block Wiedemann by a factor 2 at most and 4/3 on average.

References

[1] D. Coppersmith. Solving Homogeneous Linear Equation Over GF(2) via Block Wiedemann Algorithm. *Mathematics of Computation*, 62(205):333–350, 1994.

[2] M. J. Fischer and L. J. Stockmeyer. Fast on-line integer multiplication. *J. Comput. System Sci.*, 9:317–331, 1974.

[3] P. Giorgi, C.-P. Jeannerod, and G. Villard. On the complexity of polynomial matrix computations. In *ISSAC'03*, p. 135–142. ACM, 2003.

[4] J. v. d. Hoeven. Relaxed multiplication using the middle product. In *ISSAC'03*, p. 143–147, New York, 2003. ACM.

[5] J. v. d. Hoeven. New algorithms for relaxed multiplication. *J. Symbolic Comput.*, 42(8):792–802, 2007.

[6] J. v. d. Hoeven. Faster relaxed multiplication. Technical report, HAL-00687479, 2012.

[7] E. Kaltofen and G. Yuhasz. On the matrix berlekamp-massey algorithm. *ACM Trans. on Algorithms*, 2013. To appear.

[8] R. Lebreton and É. Schost. Relaxed power series multiplication using middle and short product. In preparation.

[9] W. J. Turner. *Black Box Linear Algebra with the LinBox Library.* PhD thesis, North Carolina State University, 2002.

[10] W. Zhou and G. Labahn. Efficient algorithms for order basis computation. *J. Symbolic Comput.*, 47(7):793 – 819, 2012.

[11] W. Zhou, G. Labahn, and A. Storjohann. Computing minimal nullspace bases. In *ISSAC '12*, p. 366–373. ACM, 2012.

Block Wiedemann algorithm on multicore architectures

Bastien Vialla
LIRMM, CNRS-UM2 France
bastien.vialla@lirmm.fr

Solving a linear system with large sparse matrices is a computational kernel used in a wide range of applications, *e.g.* cryptography, Gröbner basis ... Classical methods such as Gaussian elimination are not well suited because they tend to fill the matrix. In [7] Wiedemann proposed a blackbox algorithm which takes advantage of the sparsity to reduce the complexity. The main operations of this approach are sparse matrix vector products and the computation of the minimal generator of a scalar sequence. Despite a better complexity than classical methods, this algorithm is not efficient in the context of parallel computation as it needs a good repartition of the non-zero elements in the matrix. The block version of Wiedemann's algorithm proposed in [2] avoids this problem by using blocks instead of vectors. Therefore it offers parallelism outside the scope of the matrix.

Let $\mathbb{K}$ be a field, $A \in \mathcal{M}_{n\times n}(\mathbb{K})$, $U, V \in \mathcal{M}_{n\times k}(\mathbb{K})$ random matrices with $k \leq n$. We denote by γ the number of non-zero elements in A and we assume that $\gamma = O(n \log n)$. Block Wiedemann algorithm follows three steps:

1. Compute the first $O(\frac{2n}{k})$ elements of $S = [U^T A^i V]_{i\in\mathbb{N}}$ using $O(n\gamma + n^2 k^{\omega-2})$ operations in $\mathbb{K}$.
2. Find the minimal matrix polynomial generator of the sequence S using $\tilde{O}(k^{\omega-1}n)$ operations in $\mathbb{K}$.
3. Compute the solution using the polynomial found in step 2 using $O(n\gamma + n^2)$ operations in $\mathbb{K}$.

In practice the cost of the first step is dominant, therefore its parallelization is crucial. The capacity to parallelize the first step heavily relies on the dimension k of the blocks.

A classical approach is to take a block size equal to the number of cores. The parallel complexity of the first step becomes $O(\frac{n\gamma}{k} + n^2)$ operations in $\mathbb{K}$. We notice that the $O(n^2)$ part does not benefit from parallelism. In order to take advantage of parallelism everywhere in step 1, we must proceed otherwise.

Our approach We naturally extend the use of sparse blocks proposed by Eberly et al. in [3] to our context of block Wiedemann algorithm. Hence, instead of using random dense block for U, we use blocks of the form

$$U = \begin{bmatrix}\delta_1 I_k & \cdots & \delta_s I_k & \delta_{s+1} I'\end{bmatrix}^T \in \mathcal{M}_{n\times k}(\mathbb{K})$$

where $s = \lfloor n/k \rfloor$, $\delta_1, \cdots, \delta_{s+1} \in \mathbb{K}$ chosen at random, I_k the identity matrix of size k, and $I' = Diag(1, \cdots, 1) \in \mathcal{M}_{k\times r}(\mathbb{K})$ the matrix with only ones on the diagonal with $r = n \bmod k$. Using these new block projections, the sequential complexity of step 1 drops down to $O(n\gamma + n^2)$ operations in $\mathbb{K}$, eliminating the influence of the block size. In this work we study how these new block projections perform in practice and we show that they improve the performance of the first step of block Wiedemann algorithm.

Implementation and Benchmarks For the benchmarks, we have in mind matrices arising from NFS algorithm [6], which are very sparse. As γ is cheaper, the part of step 1 of complexity $O(n^2)$ has more importance. Therefore the block size has more influence on the sequence computation as γ is cheaper. In this case, we use a sparse matrix of size $10^5 \times 10^5$ over $\mathbb{F}_{65537}$ with ~ 15 non-zero elements per row uniformly dispatched.

The parallel complexity of step 1 using sparse blocks with k cores becomes $O(\frac{n\gamma+n^2}{k})$, hence offering perfect parallelism. So we want to see the influence of the block size on the computation of step 1.

First, we determine the most efficient block size depending on the number of cores. Let c be the number of cores, we benchmark the computation of the sequence starting with blocks of size c to $128c$ on 12 cores. As expected by the complexity analysis, a block size of c offers better performance for dense blocks. For sparse projections the theoretical study shows no influence of the blocks size. In practice we observe that a block size of $\simeq 32c$ is better,

which could be caused by memory management issues. However, this sparse block size is related to the number of non-zero elements of the matrix, so these values stand just for our test matrix. Despite their good complexity, sparse blocks have two flaws impacting the performance. The choice of matrix representation is important: first we choose to store blocks in column major representation to avoid concurrent writing, as suggested in [1]. Secondly, sparse blocks induce cache defaults as their size increase with the number of cores. To circumvent this problem, we permute block elements to obtain a cache friendly sparse blocks following ideas from [5]. For our tests we use an NUMA with four intel XEON E4620 with 8 cores at 2.2Ghz and 384GB of RAM. To obtain good performance on an NUMA, we design an hybrid MPI/tbb implementation that create one MPI process by node which use tbb to compute a part of the sequence. Each nodes own a copy of the sparse matrix and the block is split by column over the nodes, the results are gather at the end of the computation. All the libraries used are in the latest version from their svn directory. In table 1 we compare dense block which used LinBox's dense blocks implementation and our implementation using sparse blocks. For computing of dense block, LinBox relies on a BLAS library, in this case we use OpenBLAS wich is well optimized for intel XEON. The timings are in seconds and in parenthesis we indicate the block size used.

	Dense blocks (LinBox)		Sparse blocks	
	time in s	speed-up	time in s	speed-up
1 core	2205(1)	1	2165(32)	1
8 cores	540(8)	4	308(256)	7
16 cores	623(16)	3,5	154(512)	14
24 cores	798(24)	2,7	102(768)	21,2
32 cores	960(32)	2,2	77(1024)	28,1

Table 1: Times of sequence computation.

The time for dense blocks on one core is just for benchmark purposes. As predicted, sparse blocks perform better than dense blocks. However, the reasons that LinBox implementation does not perform well are that LinBox use an external library to compute dense block which as to create and destroy is own pool of threads for each computed element. Secondly, the LinBox implementation is not designed for a NUMA architecture as all the data is store in the first node memory.

This is a first step in an efficient implementation of block Wiedemann algorithm on multicore architectures. The next step will be an efficient implementation of $\sigma-$basis [4].

References

[1] Brice Boyer, Jean-Guillaume Dumas, and Pascal Giorgi. Exact Sparse Matrix-Vector Multiplication on GPUs and Multicore Architectures. *Proc. PASCO'10: Parallel Symbolic Computation*, 2010.

[2] Don Coppersmith. Solving Homogeneous Linear Equation Over GF(2) via Block Wiedemann Algorithm. *Mathematics of Computation*, 62(205):333–350, 1994.

[3] Wayne Eberly, Mark Giesbrecht, Pascal Giorgi, Arne Storjohann, and Gilles Villard. Faster inversion and other black box matrix computations using efficient block projections. *Proceedings of the 2007 international symposium on Symbolic and algebraic computation - ISSAC '07*, 3(1):143, 2007.

[4] Pascal Giorgi, Claude-Pierre Jeannerod, and Gilles Villard. On the complexity of polynomial matrix computations. *Proceedings of the 2003 international symposium on Symbolic and algebraic computation - ISSAC '03*, pages 135–142, 2003.

[5] Sardar A. Haque, Shahadat Hossain, and M. Moreno Maza. Cache friendly sparse matrix-vector multiplication. *Proceedings of the 4th International Workshop on Parallel and Symbolic Computation (PASCO'10)*, pages 175–176, 2010.

[6] A. K. Lenstra and H. W. Lenstra. *The development of the number field sieve*. Springer-Verlag, 1993.

[7] D. Wiedemann. Solving sparse linear equations over finite fields. *IEEE Transactions on Information Theory*, 32(1):54–62, January 1986.

Recursive Sparse Interpolation

Andrew Arnold Mark Giesbrecht
Symbolic Computation Group
University of Waterloo, Canada
{a4arnold,mwg}@uwaterloo.ca

Dan Roche
United States Naval Academy, USA
roche@usna.edu

We consider the problem of interpolating a sparse univariate polynomial f over an arbitrary ring, given by a straight-line program. In this problem we are given a straight-line program that computes f, as well as bounds D and T on the degree and sparsity (i.e., the number of nonzero terms) of f respectively. We build on ideas developed in Garg and Schost (2009) and Giesbrecht and Roche (2011) towards algorithms for this specific problem. We present a Monte Carlo algorithm that improves on the best previously-known algorithm for this specific problem by a factor (softly) on the order of $T/\log D$. Thus this new algorithm is favourable for "moderate" values of T.

Our algorithm is recursive. At a recursive step of the algorithm we have a straight-line program for f, an approximation f^* of f, and respective bounds T and D on the sparsity and degree of the difference $g = f - f^*$. We initialize f^* to zero. We will construct an approximation f^{**} to g such that, with high probability, $g - f^{**}$ has at most $T/2$ terms. We then recurse with $f^* + f^{**}$ as our refined approximation for f.

The algorithms in Garg and Schost (2009) and Giesbrecht and Roche (2011), as well as the algorithm we will present, interpolate f by using its straight-line program to evaluate f at a symbolic k-th root of unity, for appropriate choices of k. This effectively gives the image $f \bmod (z^k - 1)$. We call such an evaluation a *probe* of degree k. The cost of a degree-k probe to a length-L straight-line program is quasi-linear in kL. We use the number of probes, multiplied by a bound on the probe degree, as a rough measure of the cost of an interpolation algorithm.

The image $f \bmod (z^k - 1)$ in practise gives a large amount of useful information about the polynomial f. Namely, a term cz^e of f will appear as $cz^{e \bmod k}$ in the image $f \bmod (z^k - 1)$, so the image should give us f's vector of exponents modulo k. However, there are potential obstacles. We may not be able to match images of the same term in multiple images of f. In addition, terms can *collide* modulo $z^k - 1$ if they have the same degree modulo k. Collisions are problematic because it is difficult to detect if a term in an image $f \bmod (z^k - 1)$ is in fact the image of a sum of colliding terms. Alternatively, colliding terms may sum to zero modulo $z^k - 1$, which also may be difficult to detect.

Previous Las Vegas interpolation algorithms require a "good" prime, a prime p for which the terms of f remain distinct modulo $z^p - 1$. If p is a good prime, $f \bmod (z^p - 1)$ has the same number of terms as f. Thus, once we have a good prime with high probability, we can detect the presence of collisions in other images of f. In order to guarantee one can find such a prime with high probability, one chooses primes at random on the order of $T^2 \log D$ as probe degrees.

In order to reduce this probe degree, we relax the condition that p separates all the terms of the difference g. We instead look for an "ok" prime: a prime which separates *most* of the terms of g. This allows instead to search over primes p of size $\mathcal{O}(T \log D)$.

Once we have an "ok" prime, we make probes of degree pq_i for a set of co-prime q_i, each of size $\mathcal{O}(\log D)$. Our probe degree thus becomes $\mathcal{O}(T \log^2 D)$. If a term of g does not collide with another term modulo $z^p - 1$ then it will not collide modulo $(z^{pq_i} - 1)$. These probes will allow us to construct a polynomial f^{**} containing the non-colliding terms of g, plus potentially a small proportion of deceptive terms: terms

constructed from garbage information due to collisions in the images $f \bmod (z^{pq_i} - 1)$. Fortunately, if p is an ok prime we can give an upper bound on the number of such deceptive terms that can appear in f^{**}.

After we construct f^{**} we then recursively interpolate the new difference $g - f^{**}$, with a new sparsity bound $T/2$. We continue in this fashion $\lfloor \log T \rfloor + 1$ times until the sparsity bound reaches 0. An advantage of the recursive nature of the algorithm is that, when we reach a threshold where $\log D$ begins to dominate T, we can plug in a better-suited algorithm to interpolate what remains.

References

Sanchit Garg and Éric Schost. Interpolation of polynomials given by straight-line programs. *Theor. Comput. Sci.*, 410(27-29):2659–2662, June 2009. ISSN 0304-3975. doi: 10.1016/j.tcs.2009.03.030. URL `http://dx.doi.org/10.1016/j.tcs.2009.03.030`.

M. Giesbrecht and D.S. Roche. Diversification improves interpolation. *ISSAC '11*, pages 123–130, 2011. doi: 10.1145/1993886.1993909. URL `http://doi.acm.org/10.1145/1993886.1993909`.

Towards Parallel General-Size Library Generation for Polynomial Multiplication

Lingchuan Meng and Jeremy Johnson
Department of Computer Science
Drexel University
Philadelphia, PA 19104
lm433@drexel.edu,jjohnson@drexel.edu

Fast Fourier Transforms (FFTs) are at the core of many operations in scientific computing. In computer algebra, FFTs are used for *fast polynomial and integer arithmetic and other modular methods*. FFT-based polynomial multiplication outperforms multiplication based on classical and Karatsuba-based algorithms. Computer algebra libraries, such as `modpn` [3], provide hand-optimized low-level routines implementing fast algorithms for multivariate polynomial computations over finite fields, in support of higher-level code. Such libraries do not fully utilize the underlying hardware, and in order to take advantage of platform-dependent optimizations, automated performance tuning that supports general input sizes should be incorporated.

Recently, we extended [2] the use of Spiral from fixed-size code generation to general input size library generation to produce a modular FFT [1] library. By incorporating and extending the new library generation mechanism in Spiral [4], the generated library provides similar speedup as the fixed-size code, which is an order of magnitude faster over the original implementations in `modpn`, and allows arbitrary input sizes. Additional parallelism exploiting multi-core architecture leading to further speedup also has been implemented. This addition required adding new rules and a new transform definition and parameterization in the library generation framework in order to generate *recursive function closure* in the resulting library. The backend was also extended to enable the generation of scalar and vectorized code for modular arithmetic.

Let the n-point modular DFT matrix be $\mathbf{ModDFT}_{n,p,\omega} = \left[\omega_n^{k\ell}\right]_{0\le k,\ell<n}$, where ω_n is a primitive nth root of unity in $\mathbb{Z}_p$. Let $n = rs$, then the divide and conquer step in the Cooley-Tukey algorithm can be represented as the parameterized matrix factorization:

$$\mathbf{ModDFT}_{n,p,\omega} = (\mathbf{ModDFT}_{r,p,\omega_r} \otimes \mathrm{I}_s)\, \mathrm{T}^n_s (\mathrm{I}_r \otimes \mathbf{ModDFT}_{s,p,\omega_s})\, \mathrm{L}^n_r,$$

where T^n_s is a diagonal matrix containing *twiddle factors*; the *stride permutation* matrix L^n_r permutes the input vector as $is + j \mapsto jr + i, 0 \le i < r, 0 \le j < s$; I_s is the $s \times s$ identity matrix; and the *tensor product* is defined as $A \otimes B = [a_{k,l}B]\,, A = [a_{k,l}]\,.$

The tensor product serves as the key construct in Spiral and its many fast algorithms, in that it captures loops, data independence, and parallelism concisely. For instance, Fig. 1 shows that it produces substructures that can be interpreted as vector and parallel operations. Furthermore, the formulae can be transformed to adapt to a given vector length and number of cores; and permutations can be manipulated to obtain desired data access patterns. *Rewriting systems* and *hardware tags* have been developed in Spiral to fully exploit two levels of parallelism: *vector parallelism* and *thread parallelism*.

We report experimental data comparing the performance of hand-optimized FFTs from the `modpn` library, fixed-size FFTs and general size parallel FFT library generated by Spiral. Performance is reported in Gops (giga-ops) or billions of operations per second (higher is better). As shown in Fig. 2, all Spiral generated codes are faster than the hand-optimized implementation in `modpn` by an order of magnitude. The

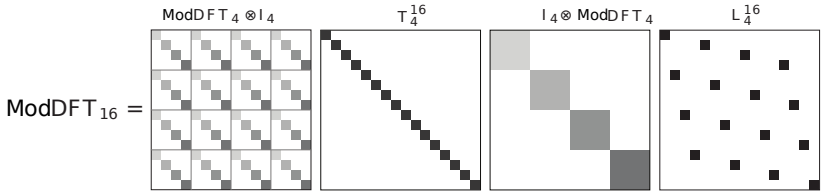


Figure 1: Representation of the matrix factorization based on the Cooley-Tukey algorithm. Shades of gray represent values that belong to the same tensor substructure

performance of general size library's scalar and vector codes are within 81% to 91% of that of corresponding fixed-size codes. For large sizes, the library code is up to 1.5 time faster than the fixed-size code, due to the use of thread level parallelism.

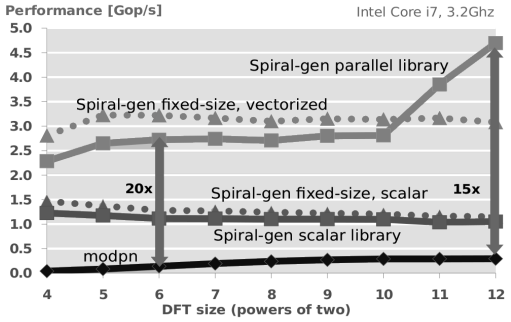


Figure 2: Performance comparison

To eventually generate an optimized parallel library for polynomial multiplication, we are developing additional algorithms for modular FFT, including Prime-factor algorithm and Rader's algorithm. We have also developed a Cooley-Tukey type algorithm for the Truncated Fourier Transform and its inverse (TFT/ITFT) for non-contiguous and non-power-of-two input/output. We have proved block symmetry of the ITFT matrices and derived direct generation formulae that can be used during library generation. Convolutions by definition and the Convolution Theorem have also been implemented in Spiral, whose performance relies on the auto-tuned underlying transforms, such as modular DFT and TFT, and the exploration of hybrid algorithms and automatic tuning of threshold parameters.

References

[1] L. Meng, J. Johnson. Automatic Parallel Library Generation for General-Size Modular FFT Algorithms. To appear in *Proc. of the CASC 2013*, 2013

[2] L. Meng, J. Johnson, F. Franchetti, Y. Voronenko,M. Moreno Maza and Y. Xie. Spiral-Generated Modular FFT Algorithms. In *Proc. of PASCO 2010*, p. 169–170, 2010.

[3] X. Li and M. Moreno Maza: Efficient implementation of polynomial arithmetic in a multiple-level programming environment. In *Proc. Intl. Congress of Mathematical Software*, p. 12–23, Springer, 2006.

[4] Y. Voronenko. Library Generation for Linear Transforms. PhD. thesis, Electrical and Computer Engineering, Carnegie Mellon University, 2008

A Parallel Algorithm to Compute the Greatest Common Divisor of Sparse Multivariate Polynomials

Jiaxiong Hu and Michael Monagan
Department of Mathematics, Simon Fraser University
Burnaby, Canada, V5A 1S6
jha107@sfu.ca and mmonagan@cecm.sfu.ca

Extended Abstract

Efficient algorithms for computing greatest common divisors (GCD) of multivariate polynomials have been developed over the last 40 years. Many of the general purpose computer algebra systems are using either Zippel's GCD Algorithm [5] or the EEZ-GCD [4] Algorithm or both. Both algorithms sequentially interpolate variables one at a time which limits parallel speedup. Since multi-core processors are now widely available, parallel algorithms are desirable. In this poster, we present a first multivariate GCD computation algorithm over $\mathbb{Z}$ which is based on the Ben-Or/Tiwari interpolation [1]. By using Ben-Or/Tiwari interpolation, we reduce the number of points needed to interpolate the GCD and improve parallelism.

Our algorithm considers multivariate GCD problems with at least three variables. The structure of the algorithm is similar to Zippel's GCD Algorithm except the way we determine the first modular image which determines all the monomials. Once this correct *form* is obtained with those monomials, we use Zippel's sparse interpolation with this form to compute more modular images and apply Chinese remaindering to reconstruct the true GCD over $\mathbb{Z}$.

Our algorithm determines the first modular image as follows: Suppose $a,\ b \in \mathbb{Z}[x_1, \ldots, x_n]$ are the input polynomials and let

$$g = \gcd(a, b) = \sum_{i=1}^{l} c_i M_i(x_1, x_2)$$

where l is the number of terms of $g(x_1, x_2)$ and M_i is the ith monomial of $g(x_1, x_2)$ and $c_i \in \mathbb{Z}[x_3, ..., x_n]$ is the ith coefficient of $g(x_1, x_2)$. The algorithm projects a and b down to bivariate polynomials by evaluating $\{x_3, \ldots, x_n\}$ at specific point $\{e_3^k, \ldots, e_n^k\}$ which satisfies the requirement of the Ben-Or/Tiwari interpolation. Then we compute bivariate

$$g_k = \gcd(a(x_1, x_2, e_3^k, \ldots, e_n^k), b(x_1, x_2, e_3^k, \ldots, e_n^k)) \in \mathbb{Z}_p[x_1, x_2],$$

where p is a carefully chosen prime. We redo this for $k = 0, 1, 2, \ldots, m$ until m is large enough. Now all bivariate GCDs should have the same monomials but different coefficients. For each monomial $M_i(x_1, x_2)$ in the g_k, we form an integer sequence by collecting M_i's coefficient in g_k $(0 \le k \le m)$. Then the Ben-Or/Tiwari algorithm is applied to this sequence to interpolate the coefficient $c_i \in \mathbb{Z}_p[x_3, \ldots, x_n]$. For this to work we require $m \ge 2t$ where $t = \max_{i=1}^{l}(\#\text{ terms } c_i)$. Obviously all polynomial coefficients $c_i(x_3, ..., x_n)$ can be recovered in parallel. Moreover, the bivariate GCDs can be computed in parallel as well. In general, this approach is easy to parallelize.

Compared with Zippel's algorithm, our algorithm uses fewer evaluation points – $O(t)$ instead of $O((n-2)dt)$ and fewer trial divisions – $O(1)$ instead of $O(n)$. One disadvantage of our algorithm is that we do not know t. We must try $t = 2, 4, 8, 16, \ldots$ stopping when we have redundancy.

A problem with the original Ben-Or/Tiwari algorithm is the intermediate expression swell that occurs using $e_3^k, e_3^k, e_4^k, ... = 2^k, 3^k, 5^k, ...$ and computing over $\mathbb{Q}$. A modular version of the algorithm was first developed by Kaltofen, Lakshman and Wiley in [3]. Their algorithm uses a small prime q with a lifting technique to determine the monomials in the c_i. One lifts until $q^k > p_{n-2}^d$ where p_n denotes the n'th prime and $d \geq \max(\deg c_i)$. Instead, we adapt Giesbrecht, Labahn and Lee's method in [2]. We construct a smooth prime p so that we can efficiently compute discrete logarithms in $\mathbb{Z}_p$. The prime p is slightly larger than $\prod_{i=3}^n d_i$ where $d_i = \deg_{x_i} g$ thus of size $O(n \log d)$. To determine the d_i accurately we compute one univariate image of g in each variable (in parallel).

A further problem is that all underlying bivariate GCDs are monic over $\mathbb{Z}_p$. The leading coefficient of the true GCD is required to scale all bivariate GCDs consistently. We use Wang's leading coefficient algorithm [4] to solve this problem. We compute and factor the gcd h of the leading coefficients of $a, b \in \mathbb{Z}[x_3, \ldots, x_n][x_1, x_2]$. This creates another sequential step in our algorithm. This is the main reason why we reduce to bivariate GCDs instead of univariate – we likely reduce the size of h. We also likely reduce t and hence the number of bivariate GCDs needed. If $a(x_1, x_2)$ and $b(x_1, x_2)$ are dense (which they often are in practice) we lose nothing by doing this.

We have implemented our algorithm in Maple. For most large problems, it outperforms Maple's default multivariate GCD procedure, which is a Zippel based algorithm and almost entirely coded in C. For example, for input polynomials having 5 variables and 500 terms, our algorithm is almost 2 times faster than Maple's default procedure; with input polynomials having 40 variables and 4000 terms, our algorithm is almost 20 times faster. We have not yet attempted a parallel implmentation but plan to do so using Cilk. We expect that such an implementation will be much faster.

References

[1] M. BEN-Or, P. Tiwari: A deterministic algorithm for sparse multivariate polynomial interpolate. *Proc. 20th annual ACM Symp Theory Comp*, 1988, 301–309.

[2] M. Giesbrecht, G. Labahn, W-S. Lee: Symbolic-numeric sparse interpolation of multivariate polynomials. *ISSAC'06*, 2006.

[3] E. Kaltofen, Y.N. Lakshman, J-M. Wiley: Modular rational sparse multivariate polynomial interpolation. *Watanabe and Nagata*, 1990, 135–139.

[4] P. Wang: The EEZ-GCD Algorithm. *SIGSAM Bulletin*, 14, 1980, 50–60.

[5] R. E. Zippel: Probabilistic algorithms for sparse polynomials. *EUROSAM '79*, Springer-Verlag LNCS, 2, 1979, 216–226.

Calculating Approximate GCD of Multiple Univariate Polynomials using Approximate Syzygies

Akira Terui
Faculty of Pure and Applied Sciences, University of Tsukuba
Tsukuba, 305-8571, Japan
terui@math.tsukuba.ac.jp, http://researchmap.jp/aterui/

For given n univariate polynomials with $n \geq 3$, we present a Symbolic-Numeric method for calculating approximate greatest common divisor (GCD) of them by calculating approximate Syzygies. This kind of GCD calculation can be used in application such as blind image deconvolution [1]. In such a case, it is especially effective when we try to restore the original image from several number of blurred images.

In our previous research, we have developed a method for calculating approximate GCD, called *GPGCD* [4]. Furthermore, we have extended the original method for n polynomial inputs [3] based on Rupprecht's first algorithm [2, Sect. 4]. However, this method is inefficient for large number and/or degree of input polynomials because, in such cases, the dimension of a generalized Sylvester matrix becomes large and sparse. While Rupprecht's second algorithm [2, Sect. 5] seems more efficient by using Syzygies with another generalization of Sylvester matrix whose dimension is much smaller than those used in the first algorithm, as for our GPGCD method, we have difficulty applying the method directly (we will explain the reason in detail below). We present a method to overcome the difficulty.

For $i = 1, \ldots, n$, let $P_i(x)$ be real univariate polynomial of degree $d_1 \leq \cdots \leq d_n$, respectively, with $d_1 > 0$, given as $P_i(x) = p_{d_i}^{(i)} x^{d_i} + \cdots p_1^{(i)} x + p_0^{(i)}$. At first we assume that $P_1, \ldots, P_n$ have a GCD. Let $H = \gcd(P_1, \ldots, P_n)$ and $d = \deg(H)$ with $d \leq d_1$.

For a real univariate polynomial $P(x)$ represented as $P(x) = p_n x^n + \cdots + p_0 x^0$, let $C_k(P)$ be a real $(n+k, k+1)$ matrix (called "convolution matrix") defined as $C_k(P) = \left({}^t(p_n, \ldots, p_0, 0, \ldots, 0), {}^t(0, p_n, \ldots, p_0, 0, \ldots, 0), \ldots, {}^t(0, \ldots, 0, p_n, \ldots, p_0)\right)$ and let $\boldsymbol{p}$ be the coefficient vector of $P(x)$ defined as $\boldsymbol{p} = (p_n, \ldots, p_0)$, and vice versa.

As a generalized Sylvester matrix, we use the second definition by Rupprecht [2, Sect. 5]. For $k > d_1$, define the k-th Sylvester matrix of $P_1, \ldots, P_n$ as $N_k(P_1, \ldots, P_n) = \begin{pmatrix} C_{k-d_1}(P_1) & C_{k-d_2}(P_2) & \cdots & C_{k-d_n}(P_n) \end{pmatrix}$, where $C_{k-d_i}(P_i)$ has empty element for $k < d_i$.

If a vector $\boldsymbol{v} = {}^t\begin{pmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \cdots & \boldsymbol{r}_n \end{pmatrix}$ with $\dim(\boldsymbol{r}_i) = k - d_i + 1$ satisfies $N_k \boldsymbol{v} = \boldsymbol{0}$, then we see that the polynomials $R_1, \ldots, R_n$ whose coefficient vectors are $\boldsymbol{r}_1, \boldsymbol{r}_2, \ldots, \boldsymbol{r}_n$, respectively, satisfy $R_1 P_1 + \cdots + R_n P_n = 0$. In such a case, we call a tuple of polynomials $(R_1, \ldots, R_n)$ a Syzygy of $P_1, \ldots, P_n$ of degree k.

In Rupprecht's second method, we first calculate Syzygies of $P_1, \ldots, P_n$, then calculate cofactors of $P_1, \ldots, P_n$ by using calculated Syzygies, as follows.

1. Calculate $n-1$ "independent" (as elements in a module over polynomial ring R[x]) Syzygies which satisfy the following condition on the degrees. For $j = 1, \ldots, n-1$, let $R_j = (U_1^{(j)}, U_2^{(j)}, \ldots, U_n^{(j)})$ be a Syzygy of $P_1, \ldots, P_n$ of degree r_j. Then, we have

$$d = d_1 + \cdots + d_n - (r_1 + \cdots + r_{n-1}) \tag{1}$$

   [2, Lemma 5.3]. With numerical computation on coefficients, we calculate a Syzygy by the Singular Value Decomposition (SVD) on Sylvester matrix N_k by increasing the degree k by 1 from the initial value d_1, until we obtain $n-1$ Syzygies satisfying condition (1).

2. For calculated Syzygies $R_j = (U_1^{(j)}, U_2^{(j)}, \ldots, U_n^{(j)})$, $j = 1, \ldots, n-1$, define a matrix $U = (u_{ij})$ as $u_{ij} = U_j^{(i)}$, and let Δ_i be the minor of U by deleting the i-th column (note that we must define U satisfying that $\Delta_i \neq 0$ for all i). Then, Δ_i is the cofactor of P_i satisfying $P_i = H \cdot \Delta_i$ [2, Lemma 5.2]. Thus, by calculating U and Δ_i, we obtain desired GCD H.

In our GPGCD method, we accept polynomials $P_1, \ldots, P_n$ that are pairwise relatively prime in general, then find "perturbation terms" ΔP_i, $i = 1, \ldots, n$, satisfying that "perturbed polynomials" $\tilde{P}_i = P_i + \Delta P_i$ have a nontrivial GCD H. With the original method by Rupprecht, we may encounter the following issue in Step 1. In our GPGCD method, we set up constrained optimization problem with constraints on the coefficients in input polynomials and their Syzygies that need coefficients in *all* input polynomials at once. On the other hand, Step 1 in the above may not involve *all* input polynomials from the beginning step(s), thus it is not clear if we can calculate appropriate perturbed terms incrementally to make all the perturbed polynomials satisfy the Syzygy relations in the final phase. Therefore, we modify the method so that we use Syzygy relations that involve *all* input polynomials from the beginning step, as follows.

1. Let l be greater than or equal to d_n satisfying (1). For given polynomials $P_1, \ldots, P_n$, calculate perturbed polynomials $\tilde{P}_1, \ldots, \tilde{P}_n$ along with Syzygies $R_j = (U_1^{(j)}, U_2^{(j)}, \ldots, U_n^{(j)})$ of degree l satisfying $U_1^{(j)}\tilde{P}_1 + \cdots + U_n^{(j)}\tilde{P}_n = 0$, as follows. Let $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$ be the right singular vectors of $N_l(P_1, \ldots, P_n)$ calculated with the SVD. By optimization method (in our case we use so-called the modified Newton method; see our literature [4] for reference), we obtain $\tilde{P}_1, \ldots, \tilde{P}_n$ by perturbing coefficients in $P_1, \ldots, P_n$, and $\tilde{\boldsymbol{v}}_1, \ldots, \tilde{\boldsymbol{v}}_m$ by perturbing $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_m$, respectively, satisfying $N_l(\tilde{P}_1, \ldots, \tilde{P}_n)\tilde{\boldsymbol{v}}_j = \boldsymbol{0}$. From vector $\tilde{\boldsymbol{v}}_j = \begin{pmatrix} \boldsymbol{r}_1^{(j)} & \boldsymbol{r}_2^{(j)} & \cdots & \boldsymbol{r}_n^{(j)} \end{pmatrix}$, we extract coefficients of a Syzygy $R_j = (U_1^{(j)}, U_2^{(j)}, \ldots, U_n^{(j)})$.

2. Using Syzygies R_j calculated in the above step, select and/or calculate Syzygies of appropriate degree satisfying (1) to make up matrix U with the following strategies.

   (a) If we need to calculate Syzygies of degree k smaller than l, make appropriate linear combination of the right singular vectors $\tilde{\boldsymbol{v}}_1, \ldots, \tilde{\boldsymbol{v}}_m$ to eliminate coefficients of degrees greater than k in the corresponding Syzygy, as follows. Let M be a submatrix of $\begin{pmatrix} \tilde{\boldsymbol{v}}_1 & \cdots & \tilde{\boldsymbol{v}}_m \end{pmatrix}$ consisting of the rows corresponding the coefficients of $U_i^{(j)}$ of degree greater than k. Then, calculate the SVD on M to find basis of the null space of M. Repeat this step until we find appropriate Syzygies satisfying (1) along with $\Delta_i \neq 0$ for all i.

   (b) If we could not find all of $n-1$ independent Syzygies satisfying (1) with the above procedure, then, for degree $k \neq l$ satisfying (1), calculate new Syzygies from $N_k(\tilde{P}_1, \ldots, \tilde{P}_n)$ until we find all of $n-1$ independent Syzygies satisfying (1) along with $\Delta_i \neq 0$.

References

[1] Z. Li, Z. Yang, and L. Zhi. Blind image deconvolution via fast approximate GCD. In *Proceedings of ISSAC '10*, pages 155–162, 2010.

[2] D. Rupprecht. An algorithm for computing certified approximate GCD of n univariate polynomials. *J. Pure Appl. Algebra*, 139:255–284, 1999.

[3] A. Terui. GPGCD, an iterative method for calculating approximate GCD, for multiple univariate polynomials. In *Lecture Notes in Computer Science* 6244, pages 238–249. Springer, 2010.

[4] A. Terui. GPGCD: An iterative method for calculating approximate GCD of univariate polynomials. *Theoretical Computer Science*, 479:127–149, 2013. Symbolic-Numerical Algorithms.

Incremental PSLQ with Application to Algebraic Number Reconstruction *

Yong Feng, Jingwei Chen, Wenyuan Wu
Automated Reasoning and Cognition Key Laboratory of Chongqing, CIGIT, CAS

{yongfeng, chenjingwei, wuwenyuan}@cigit.ac.cn

1. Introduction. A vector $\boldsymbol{m} = (m_1, \ldots, m_n) \in \mathbb{Z}^n \setminus \{\mathbf{0}\}$ is called an *integer relation* for $\boldsymbol{x} = (x_1, \ldots, x_n) \in \mathbb{R}^n$ if $\sum_{i=1}^n m_i x_i = 0$. In literature, the HJLS algorithm [3, Sec. 3] and the PSLQ algorithm [2] solve the problem of finding integer relation polynomially. Although it has been theoretically proved that PSLQ is to some extent equivalent to HJLS, under the exact real arithmetic computational model (see, e.g., [7, 1]), the PSLQ algorithm seems more practical.

The problem of finding the *minimal polynomial* from an approximation $\overline{\alpha}$ of a d_0 degree *algebraic number* α, equivalent to finding an integer relation for the vector $(1, \alpha, \ldots, \alpha^{d_0})$, was first solved in [5] by using the celebrated LLL algorithm [6]. This routine has been recently improved in [4]. Naturally, the PSLQ algorithm applies to the algebraic number reconstruction problem as well [8].

Given an approximation to an unknown algebraic number α, a degree bound d and an upper bound M on its height, if the exact degree d_0 of α is also unknown, then no matter whether one uses PSLQ or LLL, one has to search an integer relation for the vector $(1, \alpha, \ldots, \alpha^i)$ from $i = 2, 3, \ldots$ until d_0 ($\leq d$). Hence, if the complexity of a polynomial algorithm for finding an integer relation is $\mathcal{O}(P(n, M))$ for an n-dimensional vector, then the complexity of the minimal polynomial algorithm, based on the integer relation finding algorithm, is $\mathcal{O}(d_0 \cdot P(d_0, M))$. Our main contribution in the present work is to give an *incremental* version of PSLQ, which leads to an efficient algebraic number reconstruction algorithm with complexity only $\mathcal{O}(P(d_0, M))$, even though the exact degree of the algebraic number is unknown.

Algorithm 1 (IPSLQ).

Input: A vector $\boldsymbol{x} = (x_1, \cdots, x_n) \in \mathbb{R}^n$ with $x_i \neq 0$ for $i = 1, \ldots, n$ and a positive number M.
Output: Either return an integer relation for $\boldsymbol{x}$, or return "no relation with length smaller than M".
1. Construct $H_x \in \mathbb{R}^{n \times (n-1)}$. Set $H := H_x$, $A := I_n$ and $B := I_n$. Size-reduce H and update A and B.
2. For k from $n-1$ to 1 do
   (a) While $h_{n-1,n-1} \neq 0$ do
      i. Choose r such that $\gamma^r |h_{r,r}| = \max_{j \in \{k, \cdots, n-1\}} \{\gamma^j |h_{j,j}|\}$.
      ii. Swap the r-th and the $(r+1)$-th rows of H and update A and B.
      iii. If $r < n-1$ then update H to L-factor of H.
      iv. Size-reduce H and update A and B.
      v. If $\max_{j \in \{k, \cdots, n-1\}} |h_{j,j}| < 1/M$ then do the following: If $k > 1$ then go to Step 2; Else return "no relation with length smaller than M".
   (b) Return the last column of B.

2. The Incremental PSLQ Algorithm. The main difference between IPSLQ (Algorithm 1) and PSLQ is that PSLQ considers $x_1, \ldots, x_n$ directly, while IPSLQ considers $x_i, \ldots, x_n$ gradually, i.e., if the vector $(x_i, \ldots, x_n)$ has no relation with 2-norm less than M then add x_{i-1} to the left; see Step 2(a)v.

The application of IPSLQ to efficient reconstructing minimal polynomial depends on the following two key points: (1) We use $(x_1, \ldots, x_n) = (\alpha^{n-1}, \ldots, \alpha, 1)$, which is the reverse order of the traditional version, to construct the matrix H_x (see [2, Def. 2] for the construction). (2) The important observation is that

*This work was partially supported by NKBRPC (2011CB302400) and NSFC (11001040, 11171053, 91118001).

the matrix H_x for $(x_i, \ldots, x_n)$ is exactly the right-bottom most submatrix of H_x for $(x_{i-1}, x_i, \ldots, x_n)$. Thus, the results produced by the previous iterations are still valid for the new matrix H. However, the traditional methods can not reuse those previous information. Therefore, the complexity of IPSLQ for minimal polynomial without knowing degree is only $\mathcal{O}(P(d_0, M))$, which is the same as PSLQ for minimal polynomial with knowing degree.

3. Experiments. The following experiments are preliminary and to compare the performance between traditional PSLQ and IPSLQ for minimal polynomial reconstruction. Consider approximations of $\alpha = 3^{1/s} + 2^{1/t}$ with 500 decimal digits. Running these experiments in Maple 15 with `Digits :=500` gives a preliminary experimental results in Table 1. Note that here `Digits :=500` may not be necessary (see [8] for the a detailed error control). In Table 1, the input degree bound and height bound in these tests are d and $M+1$; the exact degree and height of α are $d-1$ and M, respectively. All these experimental results are obtained by using a Windows 7 (32 bits mode) PC with AMD Athlon II X4 645 processor (3.10 GHz) and 4 GB memory. Note that there exists a built-in function `IntegerRelations:-PSLQ` in Maple 15, but for the comparison in Table 1, we implement the PSLQ algorithm by ourselves. The reasons we do not use the built-in function is that there does not exist a height parameter in the built-in function. This may cause that the built-in function will go on the iterations even if the height has been greater than M. In our implementations of PSLQ and IPSLQ, the same function uses the same technique for fairness. According to Table 1, the IPSLQ algorithm is faster than the PSLQ algorithm. Meanwhile the ratio between T_{PSLQ} and T_{IPSLQ} seems to get larger and larger with increasing d, but always smaller than d.

No.	s	t	d	M	T_{IPSLQ}	T_{PSLQ}	$\frac{T_{PSLQ}}{T_{IPSLQ}}$
1	2	2	5	10	0.08	0.16	2.00
2	2	3	7	36	0.16	0.64	4.00
3	3	3	10	125	0.89	5.34	6.00
4	3	4	13	540	3.14	21.34	6.79
5	2	7	15	5103	6.91	45.91	6.64
6	3	6	19	10278	23.37	144.11	6.17
7	4	5	21	11160	32.73	249.54	7.62
8	5	5	26	57500	78.95	838.99	10.63
9	5	6	31	538380	186.28	2089.87	11.22
10	6	6	37	4281690	421.94	4313.99	10.22

Table 1: IPSLQ VS PSLQ for minimal polynomial

References

[1] J. Chen, D. Stehlé, and G. Villard. A new view on HJLS and PSLQ: Sums and projections of lattices. In *Proc. ISSAC '13*, Boston, USA, 2013. To appear.

[2] H. R. P. Ferguson, D. H. Bailey, and S. Arno. Analysis of PSLQ, an integer relation finding algorithm. *Math. Comput.*, 68(225):351–369, 1999.

[3] J. Håstad, B. Just, J. Lagarias, and C. Schnorr. Polynomial time algorithms for finding integer relations among real numbers. *SIAM J. Comput.*, 18(5):859–881, 1989.

[4] M. van Hoeij and A. Novocin. Gradual sub-lattice reduction and a new complexity for factoring polynomials. *Algorithmica*, 63(3):616–633, 2012.

[5] R. Kannan, A. Lenstra, and L. Lovász. Polynomial factorization and nonrandomness of bits of algebraic and some transcendental numbers. *Math. Comput.*, 50(181):235–250, 1988.

[6] A. Lenstra, H. Lenstra, and L. Lovász. Factoring polynomials with rational coefficients. *Math. Ann.*, 261(4):515–534, 1982.

[7] A. Meichsner. Integer Relation Algorithms and the Recognition of Numerical Constants. Master's thesis, Simon Fraser University, 2001.

[8] X. Qin, Y. Feng, J. Chen, and J. Zhang. A complete algorithm to find exact minimal polynomial by approximations. *Int. J. Comput. Math.*, 89(17):2333–2344, 2012.

Hypergeometric generating functions and series for $1/\pi$

James G. Wan
Singapore University of Technology and Design
james_wan@sutd.edu.sg

Introduction

Some truly innovative series for $1/\pi$, first discovered by Ramanujan and elucidated in [1], take the form

$$\sum_{n=0}^{\infty} \frac{(s)_n(\frac{1}{2})_n(1-s)_n}{n!^3}(a+bn)z_0^n = \frac{1}{\pi}. \tag{1}$$

In other words, the constant $1/\pi$ can be written as a suitable linear combination of a hypergeometric function (in this case a ${}_3F_2$) and its derivative at some z_0. Such series have both theoretical and practical applications. In a recent preprint [3], some double sums are conjectured to also evaluate to $1/\pi$; we aim to prove them using the theory behind (1). Examples of these sums include

$$\sum_{n=0}^{\infty}\sum_{k=0}^{n} \binom{n}{k}\binom{2n-2k}{n-k}\binom{2k}{k}\binom{2n}{n}\frac{140n+19}{2^{6k}}\Big(\frac{2}{17}\Big)^{2n} = \frac{289}{3\pi}, \tag{2}$$

$$\sum_{n=0}^{\infty}\sum_{k=0}^{n} \binom{2n-2k}{n-k}\binom{2k}{k}^2\binom{2n}{n}\frac{12n+1}{6^{2k}}\Big(\frac{3}{20}\Big)^{2n} = \frac{75}{8\pi}. \tag{3}$$

Method

Instead of ${}_3F_2$'s, these conjectural series in [3] all contain functions of the form

$$G(x,z) = \sum_{n=0}^{\infty}\sum_{k=0}^{n} F(n,k)x^n z^k,$$

where F is a product of four or more binomial coefficients. It is routine to find a differential equation in x satisfied by G; however such ODEs have degrees ≥ 4 and current CAS struggle to find or rule out hypergeometric solutions implicitly required in (1). Our approach is to guess, based on numerical evidence, that x and z are connected by a simple algebraic relation r. For instance, we may guess that

$$G(x, r_{a,b}(x)) = \sum_{n=0}^{\infty}\sum_{k=0}^{n} F(n,k)\frac{x^{k+n}}{(a+bx)^{2n+1}} \text{ or } \sum_{n=0}^{\infty}\sum_{k=0}^{n} F(n,k)\frac{(-1)^n x^{k+n}}{(a+bx)^{n+1/2}}, \tag{4}$$

for some a and b. We compute sufficiently many coefficients in the x-expansion of (4), and attempt to find a, b such that they satisfy a *three-term* recurrence (with polynomials of bounded degrees as coefficients). Such a recurrence corresponds to a degree 3 ODE satisfied by G. The key step then comes down to an easy problem in linear algebra of checking if a certain determinant is zero for some a and b.

Once suitable a and b are found, we need to solve the 3rd order ODE satisfied by $G(x, r_{a,b}(x))$; for this we have a more complete theory. E. g. in the case corresponding to (2), *Maple 13* is able to give the solution which can be rearranged into a ${}_3F_2$. In the case of (3), the ODE is of Heun type, and can be solved using [2, eqn. (3.5b)] followed by a transform due to E. Goursat; we obtain

$$\sum_{n=0}^{\infty}\sum_{k=0}^{n}\binom{2n-2k}{n-k}\binom{2k}{k}^2\binom{2n}{n}\frac{x^{k+n}}{(1+4x)^{2n+1}} = \sum_{n=0}^{\infty}\frac{(\frac{1}{3})_n(\frac{1}{2})_n(\frac{2}{3})_n}{n!^3}\big(108x^2(1-4x)\big)^n. \tag{5}$$

In either case the ${}_3F_2$ is of the type in (1), and the extensive theory for producing formulas of this type can be used to prove equations (2) and (3).

Some details

When we take the x-derivative of (5) (as is required in (1)), linear dependence on k appears on the left hand side, which is not found in (3). To cancel this k term, a vanishing, k-dependent identify (known as a 'satellite identity', coined in [4]) is required. For (5), the satellite identity is

$$\sum_{n=0}^{\infty}\sum_{k=0}^{n}\binom{2n-2k}{n-k}\binom{2k}{k}^2\binom{2n}{n}\frac{x^{k+n}}{(1+4x)^{2n}}\big(4x+2k(4x+1)+n(4x-1)\big) = 0. \tag{6}$$

Identity (6) was *guessed* as follows: pick a small, irrational x and compute $a_0 = \sum_{n,k} A(n,k,x)$, $a_1 = \sum_{n,k} A(n,k,x)k$, and $a_2 = \sum_{n,k} A(n,k,x)n$ (A being the summand), then use PSLQ to find a null integer linear combination among the elements of $\{a_0, a_1, a_2, a_0x, a_1x, a_2x, a_0x^2, a_1x^2, a_2x^2, \ldots\}$. Once found, the satellite identity can be proven by the multiple WZ algorithm. Similarly, (5) itself can be rigorously proven (as the 3rd order recursion was only a guess): write the coefficients of x on the LHS as a double sum, apply the multiple WZ algorithm to obtain a recursion, convert it to an ODE for the LHS, and finally check that the ODE annihilates the RHS. Many conjectures from [3] have been settled using our method, via the discovery of generating functions like (5).

Future work

Some conjectures in [3] do not fall into the type (4); perhaps more elaborate algebraic relations are needed – this could also anticipate more exotic generating functions. It would be illuminating to be able to find suitable a and b in (4) analytically (without extensive computer searches), and also to prove the existence of satellite identities whenever F is a hypergeometric term.

References

[1] J. M. Borwein and P. B. Borwein, *Pi and the AGM: A study in analytic number theory and computational complexity* (Wiley, New York, 1987).

[2] R. Maier, Transforming the Heun equation to the hypergeometric equation: I. Polynomial transformations, *preprint* (2002).

[3] Z.-W. Sun, List of conjectural series for powers of π and other constants, *preprint* `arXiv: 1102.5649` (Jan 2012).

[4] W. Zudilin, A generating function of the squares of Legendre polynomials, *Bull. Austral. Math. Soc. to appear* (2013).

Annihilating monomials with the integro-differential Weyl algebra

Johannes Middeke
Department of Applied Mathematics
University of Western Ontario
London (ON), Canada, N6A 5B7
jmiddeke@uwo.ca

Abstract

The integro-differential Weyl algebra provides an algebraic model for differential and integral operators with polynomial coefficients. It has a natural action on the ring of polynomials. We are interested in computing the annihilator of a given polynomial with respect to this action. This contribution contains a first step towards that goal—namely we give a description of the annihilator of a monomial.

1 The integro-differential Weyl algebra

The (univariate) integro-differential Weyl was introduced for the first time in [1]. It arose from the algebra of integro-differential operators first discussed in [2] and refined in [3, 4] with the goal of solving boundary value problems using purely algebraic methods. While the aforementioned papers construct integro-differential operators with arbitrary coefficients using a Gröbner basis approach, in [1] the authors chose to model operators with polynomial coefficients using Ore polynomials.

A way to construct the integro-differential Weyl algebra as a generalised Weyl algebra has been discussed in [5].

In this abstract we briefly recall the basic properties of the integro-differential Weyl algebra and refer to [1] for details. Let $\mathbb{K}$ be a field of characteristic 0. The *integro-differential Weyl algebra*—denoted $A_1(\partial, \int)$—is the $\mathbb{K}$-algebra generated by the symbols x, ∂ and $\int$ and defined by the equations

$$\partial x = x\partial + 1, \qquad \textstyle\int\int = x\int - \int x, \qquad \text{and} \qquad \partial\int = 1. \tag{1}$$

One can prove that $A_1(D, \int)$ is neither simple nor left or right Noetherian; it even contains zero divisors. Also, the integro-differential operators from [2, 3, 4] are isomorphic to $A_1(\partial, \int)/(\mathrm{E}x - c\mathrm{E})$ where $\mathrm{E} = 1 - \int\partial$ and c is a constant depending on the integral operator—cf again [1].

Interpreting ∂ as derivation and $\int$ as an integral, the integro-differential Weyl algebra has a natural action on the polynomial ring $\mathbb{K}[x]$. More precisely, the action $*\colon A_1(\partial, \int) \times \mathbb{K}[x] \to \mathbb{K}[x]$ is defined by

$$x * x^n = x^{n+1}, \qquad \partial * x^n = \frac{dx^n}{dx} = nx^{n-1}, \qquad \text{and} \qquad \textstyle\int * x^n = \int_0^x x^n dx = \frac{1}{n+1}x^{n+1}$$

where $n \geq 0$. With this action, the relations in (1) model the Leibniz rule, partial integration and the fundamental theorem of calculus, respectively. Moreover, E corresponds to the evaluation at 0.

2 Annihilators

We want to compute annihilators of polynomials, i. e., the set of all operators in $A_1(\partial, \int)$ whose action sends a given polynomial to zero. It is well-known that annihilators are left ideals in $A_1(\partial, \int)$. As a first step towards computing them, we give the following theorem:

Theorem 1 *For any $n \geq 1$, the annihilator of x^n within the integro-differential Weyl algebra $A_1(\partial, \int)$ is generated as a left ideal by*

$$\partial^{n+1}, \quad (x - \textstyle\int)\partial^n, \quad \mathrm{E}\partial^{n-1}, \quad \ldots, \quad \mathrm{E}\partial, \quad \mathrm{E}$$

for $n \geq 0$ and the annihilator of 1 is generated by ∂ and $x - \int$. In particular, all these annihilators are finitely generated.

As a next step, we intend to generalise this result to annihilate arbitrary polynomials. Also, we shall add more evaluation operators to our algebra in order to model boundary conditions instead of only initial value problems. Moreover, a theorem in [6] states that all finitely generated left ideals in $A_1(\partial, \int)$ can be generated by only two elements. Therefore, another path of research is to attempt to compute these two generators for the annihilators.

References

[1] Regensburger, G., Rosenkranz, M. & Middeke, J. A skew polynomial approach to integro-differential operators. In May, J. P. (ed.) *Proceedings of ISSAC 2009* (Association for Computing Machinery, 2009).

[2] Rosenkranz, M. A new symbolic method for solving linear two-point boundary value problems on the level of operators. *Journal of Symbolic Computation* **39**, 171–199 (2005).

[3] Rosenkranz, M. & Regensburger, G. Solving and facroring boundary problems for linear ordinary differential equations in differential algebras. *Journal of Symbolic Computation* **43**, 515–544 (2008).

[4] Regensburger, G. & Rosenkranz, M. An algebraic foundation for factoring linear boundary problems. *Annali di Matematica Pura ed Applicata* **188**, 123–151 (2009).

[5] Bavula, V. V. The algebra of integro-differential operators on a polynomial algebra. *Journal of the London Mathematical Society* **83**, 517–543 (2011).

[6] Bavula, V. V. The algebra of integro-differential operators on an affine line and its modules. *Journal of Symbolic Computation* 495–529 (2013).

Probabilistic Analysis of Wiedemann's Algorithm for Minimal Polynomial Computation

Gavin Harrison, Jeremy Johnson, B. David Saunders
Department of Computer Science
Drexel University, University of Delaware
gmh33@drexel.edu,jjohnson@cs.drexel.edu,saunders@udel.edu

Blackbox algorithms for linear algebra problems start with one sided (Lanczos) or two sided (Wiedemann) projection of the sequence of powers of a matrix to a sequence of scalars or a sequence of smaller matrices. Such algorithms usually require that the minimal polynomial of the resulting sequence should be that of the given matrix. Exact formulas are given for the probability that this occurs based on the Jordan structure of a matrix, and from these formulas sharp bounds follow. The bounds are valid for all finite field sizes and show that a small blocking factor can give high probability of success for all cardinalities and matrix dimensions.

Let K be a finite field with cardinality q. Given $A \in K^{n\times n}$, and $\bar{A}$ be the linearly generated sequence $\{I, A, A^2, \ldots\}$. Given $U, V \in K^{n\times b}$ whose elements are selected uniformly randomly from K, $U^T\bar{A}V$ is a linearly generated sequence of smaller matrices, and with high probability, the minimal generating polynomial of $U^T\bar{A}V = \{U^TV, U^TAV, U^TA^2V, \ldots\}$ is the minimal polynomial of the matrix A. All square matrices are similar to a generalized Jordan form matrix, $A = PJP^{-1}$, where $J, P \in K^{n\times n}$. If U and V are selected uniformly randomly, then $X = P^TU$ and $Y = P^{-1}V$ are also uniformly random. $U\bar{A}V = X^T\bar{J}Y = \{X^TY, X^TJY, X^TJ^2Y, \ldots\}$, and $X\bar{J}Y$ has the same probability as $U\bar{A}V$ of having its minimal generating polynomial match the minimal polynomial of A and J. We call this probability $Prob_{q,b}(A)$.

Let $C_f \in K^{d\times d}$ represent the companion matrix for the polynomial $f(x) = f_0 + f_1x + \ldots + f_{d-1}x^{d-1} + x^d$ with coefficients in K. Let J_{f^e} be the generalized Jordan block of an irreducible f occurring with multiplicity e. Since $Prob_{q,b}(J_f \oplus J_g) = Prob_{q,b}(J_f)Prob_{q,b}(J_g)$ when $\gcd(f, g) = 1$, unique irreducibles can be treated separately, and for each irreducible only its highest multiplicity affects the probability that the projection preserves the minimal polynomial. Furthermore we show $Prob_{q,b}(J_f) = Prob_{q,b}(J_{f^e})$ for any e. Therefore, letting $T = \{(f_1, e_1, t_1), (f_2, e_2, t_2), \ldots\}$, where the polynomials f_i are the irreducibles occuring in the invariant factors of A, e_i is the highest multiplicity of f_i, and t_i is the number of occurrences of $f_i^{e_i}$, it follows that

$$Prob_{q,b}(J) = \prod_{k=1}^{|T|} Prob_{q,b}\left(\bigoplus_{t_k} J_{f_k}\right).$$

For an irreducible polynomial f of degree d, the probability that $U^T\bar{C}_fV$ has minimal polynomial f is easy to determine. The minimal polynomial of the projection is always a factor of f, which for irreducible f is 1 or f. It is 1 only if the sequence is a sequence of zero matrices, which is to say that one of U, V is zero. Thus

$$Prob_{q,b}(C_f) = (1 - q^{-db})^2.$$

If $J = \bigoplus_t C_f$, then for $U, V \in K^{dt\times b}$, with blocking conformal to the diagonal blocks of J, we have $U^T\bar{J}V = \sum_{k=1}^{t} U_k^T\bar{C}_fV_k$. We show $Prob_{q,b}(C_f) \leq Prob_{q,b}\left(\bigoplus_t J\right)$.

	field cardinality			
block size	2	3	10007	$2^{31}-1$
1	0.000467	0.00112	0.0499	0.911
2	0.25	0.444	$1-2\times10^{-4}$	$1-4.3\times10^{-11}$
4	0.766	0.927	$1-2\times10^{-12}$	$1-9.4\times10^{-30}$
8	0.984	$1-9.1\times10^{-4}$	$1-2\times10^{-28}$	$1-4.4\times10^{-67}$
16	$1-6\times10^{-5}$	$1-1.4\times10^{-7}$	$1-2\times10^{-60}$	$1-9.8\times10^{-142}$
32	$1-9.3\times10^{-10}$	$1-3.2\times10^{-15}$	$1-2\times10^{-124}$	$1-4.8\times10^{-291}$

Table 1: Bounds for worst case probability of success to preserve minimum polynomial, matrix size $10^8 \times 10^8$

It is evident that the probability of success increases with d as well as with b. The worst case is a matrix whose minimal polynomial is a distinct product of the smallest possible irreducibles. This yields an exact lower bound formula for the probability that a projection $U^T\bar{A}V$ of A has the same minimal polynomial. Let $L_q(d,n)$ be the number of degree d irreducible factors over the finite field of cardinality q that fit in a matrix of dimension n after all smaller degree irreducibles have been inserted. Then, for an $n \times n$ matrix A,

$$Prob_{q,b}(A) \geq \prod_{d=1}^{\infty} \left(1 - \frac{2q^{db}-1}{q^{2db}}\right)^{L_q(d,n)}$$

We compare this bound to previously given lower bounds in the case when field cardinality and matrix dimension are of similar size. For small primes, Wiedemann (proposition 3) treats the case $b = 1$ and he fixes the projection on one side because he is interested in linear system solving and thus in the sequence $\bar{A}b$ [2] . For small q, his formula, $1/(6\log_q(N))$, computed with some approximation, is nonetheless quite close to our exact formula. However as q approaches N the discrepancy with our exact formula increases. At the large/small crossover, $q = N$, Kaltofen/Pan's lower bound is 0, Wiedemann's is 1/6, and ours is $1/e$. The Kaltofen/Pan probability bound improves as q grows larger from N [1] . The Wiedemann bound becomes more accurate as q goes down from N. But the area $q \approx N$ is of some practical importance. In integer matrix algorithms where the finite field used is a choice of the algorithm, sometimes practical considerations of efficient field arithmetic encourages the use of primes in the vicinity of N. For instance, exact arithmetic in double precision and using BLAS works well with $q \in 10^6..10^7$. Sparse matrices of order N in that range are tractable. Our bound may help justify the use of such primes.

But the primary value we see in our analysis here is the understanding it gives of the value of blocking, $b > 1$. Table 1 shows the bounds for the worst case probability that a random projection will preserve the minimal polynomial of a matrix $A \in K^{10^8 \times 10^8}$ for various fields and projection block sizes. It shows that the probability of finding the minimal polynomial correctly under projection converges rapidly to 1 as the projected block size increases. Even over $GF(2)$, with block size $b = 16$ the probability is very good.

References

[1] Erich Kaltofen and B. David Saunders. On wiedemann's method of solving sparse linear systems. In *Proceedings of the 9th International Symposium, on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes*, AAECC-9, pages 29–38, London, UK, UK, 1991. Springer-Verlag.

[2] D. Wiedemann. Solving sparse linear equations over finite fields. *Information Theory, IEEE Transactions on*, 32(1):54–62, 1986.

Implementation of a Solution to the Conjugacy Problem in Thompson's Group F

James Belk, Nabil Hossain, Francesco Matucci, and Robert McGrail
Bard College, Annandale-on-Hudson, NY, USA, 12504
Université Paris-Sud 11, Bâtiment 425, Bureau 21, F-91405 Orsay Cedex, France
belk@bard.edu, nh1682@bard.edu, francesco.matucci@math.u-psud.fr, mcgrail@bard.edu

Abstract

We present an efficient implementation of the solution to the conjugacy problem in Thompson's group F. This algorithm checks for conjugacy by constructing and comparing directed graphs called strand diagrams. We provide a description of our solution algorithm, including the data structure that represents strand diagrams and supports simplifications.

1 Thompson's Group F and Strand Diagrams

The elements of Thompson's Group F [3] are piecewise, linear homeomorphisms of the interval $[0, 1]$ such that each piece has slope that is a power of 2 and, furthermore, the breakpoints between pieces take place at dyadic rational coordinates. The group operation is simply function composition. In a group, the **conjugacy problem** is the problem of determining whether any two elements are conjugate. The conjugacy problem is not solvable in general [5], but is solvable in certain cases.

A **strand diagram** [2] is a finite acyclic digraph embedded on the unit square. The digraph has a **source** along the top edge of the square and a **sink** along the bottom edge. Any internal vertex is either a **merge** or a **split** (Figure 1). Elements of Thompson's Group F can be translated to strand diagrams. Each element in a generating set corresponds to a particular strand diagram. A composition of such elements is represented by a concatenation of the associated strand diagrams.

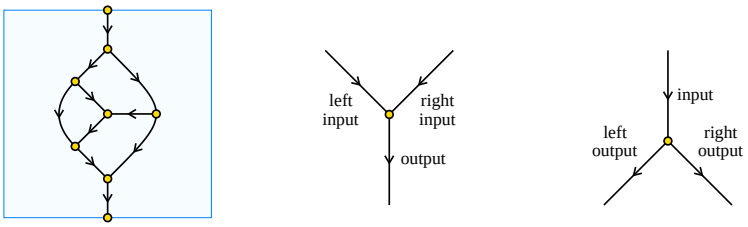


Figure 1: A strand diagram, a merge, and a split (image taken from [2]).

2 Algorithm for the Conjugacy Problem in F

The algorithm to determine whether two strand diagrams inhabit the same conjugacy class proceeds as follows. First, we convert the strand diagrams to **annular strand diagrams**. This is achieved by a process called **closing**, in which sources are identified with sinks. Next, the annular strand diagrams are

reduced using a graphical rewriting system that is both confluent, terminating, and respects conjugacy [1]. Furthermore, any two connected and reduced annular strand diagrams s_1 and s_2 can be encoded into two planar graphs g_1 and g_2 respectively such that s_1 and s_2 represent conjugate elements if and only if g_1 and g_2 are isomorphic. Hence the problem reduces to checking whether two simplified planar graphs are isomorphic. Moreover, this enterprise can be carried out in linear time given a linear time planar-graph-isomorphism checker [4].

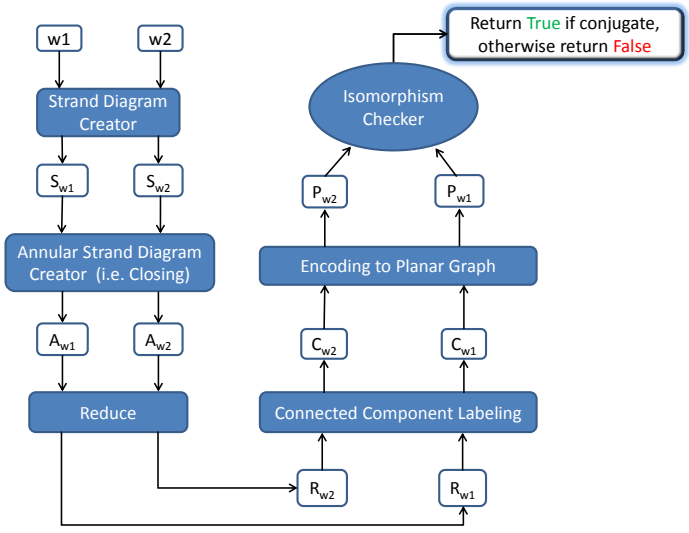


Figure 2: Algorithm Flowchart

References

[1] F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1999.

[2] J. Belk and F. Matucci. Conjugacy and dynamics in Thompson's groups. Preprint, 2013.

[3] J. W. Canon, W. J. Floyd, and W. R. Parry. Introductory notes on Richard Thompson's groups. *Enseignement Mathématique*, 42: 215–256, 1996.

[4] J. E. Hopcroft and J. K. Wong. Linear time algorithm for isomorphism of planar graphs (preliminary report). *Proceedings of the Sixth Annual ACM symposium on Theory of Computing*, 172–184, 1974.

[5] P. S. Novikov. Unsolvability of the conjugacy problem in the theory of groups. *Izv. Akad. Nauk SSSR. Ser. Mat*, 18: 485–524,1954.

A Linear Sparse Systems Solver (LSSS) applied to the Classification of Integrable non-abelian Laurent ODEs

Thomas Wolf
Brock University
Ontario, Canada
twolf@brocku.ca

Eberhard Schrüfer
Fraunhofer Institut
Bonn, Germany
eschruefer@ca-musings.de

Kenneth Webster
University of
Waterloo, Canada

Sparse linear algebraic systems can be of different nature. A well known class are systems, we call them "numerical", that result in the discretization of partial differential equations (PDEs). A very different class of systems, we call them "selective", arises from integrability investigations of differential equations. Both types of systems behave very differently during the solution process and give results of different nature and therefore are also solved most efficiently with different methods.

The Linear Selective Systems Solver LSSS, described on the poster, was developed to solve "selective" systems that result when the aim is to find discrete mathematical objects of symbolic nature like Lie symmetries or first integrals if they exist. Table 1 compares both types of problems.

type	"numerical" systems	"selective" systems
examples	systems resulting from a discretization of PDEs	systems resulting from a symmetry investigation of PDEs
value of free parameters when applying the solution of the linear system	any floating point numbers (boundary values of PDE)	0 or 1 (to isolate the individual symmetries)
number of zero-valued variables in solution	essentially none	most variables
initial sparsity	yes	yes
sparsity throughout exact solution	yes	yes
overdetermination	no	yes
usability of iteration schemes for large problems of that type	useful	not useful

Table 1: Characterization of two different types of sparse linear systems

The special nature of selective systems allows a dedicated computer program LSSS (Linear Sparse Systems Solver) running in the computer algebra system REDUCE to be much more efficient than conventional computer programs for solving these systems [1]. Reasons are:

- Because of the existence of many variables that take the value of zero in the solution the systems involve 1-term equations which are utilized first to simplify the remaining system and generate more 1-term equations.
- The simplification of a system due to the vanishing of variables can be accomplished much faster than the simplification due to other substitutions.
- From the mathematical problem it is clear whether the type of a system is numerical or selective and thus to apply the most suitable technique from the start.
- Selective linear systems are typically formulated by separation of larger expressions. The complete separation and up-front formulation of the whole linear system can be avoided through a repeatedly selective splitting and thus formulation and solution of 1-term equations.
- Increasing the complexity of the mathematical problem (e.g. by a increased degree of the ansatz for symmetries or first integrals) the overdetermination and sparseness increases compensating partially the exploding size of the initial linear system if 1-term equations are used rigorously.

The package LSSS was developed in the course of investigating the integrability of the Kontsevich system ([3])

$$u_t = uv - uv^{-1} - v^{-1},\ v_t = -vu + vu^{-1} + u^{-1} \tag{1}$$

where u, v are non-commutative variables (in particular, square matrices of arbitrary size). This is the first non-abelian system with non-polynomial right hand sides for which integrability could be shown, in this case by computing a Lax pair with spectral parameter [2]. Essential ingredients were the computation of Lie-symmetries, first integrals, a pre-Hamiltonian operator and a recursion operator of (1), all of them requiring the solution of selective linear systems. With the program LSSS it was possible to compute Lie-symmtries of degree up to 16. The complete linear system that had been solved includes over 10^9 equations for 172 Mio variables. Despite of the majority of them being zero, the general solution is not trivial as it has 32 free parameters and its formulation requires already several mega byte.

Current applications of LSSS include the integrability investigation of generalizations of the form

$$u_t = uv + P(u, v, u^{-1}, v^{-1}),\ v_t = -vu + Q(u, v, u^{-1}, v^{-1}). \tag{2}$$

References

[1] Wolf, T., Schrüfer, E., Webster, K. Solving large linear algebraic systems in the context of integrable non-abelian Laurent ODEs, *Programming and Computer Software*, (2012) DOI:10.1134/S0361768812020065, also arXiv:1109.2785 (nlin.SI).

[2] Wolf, T., Efimovskaya, O. On integrability of the Kontsevich non-abelian ODE system, *Lett. in Math. Phys.*, vol 100, no 2 (2012), p 161-170 DOI:10.1007/s11005-011-0527-4, also arXiv:1108.4208v1 (nlin.SI).

[3] Kontsevich, M., private communication.

Abstracts of Recent Doctoral Dissertations in Computer Algebra

Each month we are pleased to present abstracts of recent doctoral dissertations in Computer Algebra and Symbolic Computation. We encourage all recent Ph.D. graduates (and their supervisors), who have defended in the past two years, to submit their abstracts for publication in CCA.
Please send abstracts to the CCA editors <editors_SIGSAM@acm.org> *for consideration.*

Author: Brice Boyer
Title: Efficient matrix multiplication and design for the exact linear algebra library LinBox
Institution: Laboratoire Jean Kuntzman, Université de Grenoble.
Thesis Advisor: Jean-Guillaume Dumas
Defended: June 2012
Keywords: exact linear algebra, sparse matrix, SpMV, dense matrix, fast matrix multiplication, pebble game, schedulings, design patterns, generic mathematics library, LinBox.

Matrix multiplication is a major cornerstone in exact linear algebra: its study can concern algorithmic, complexity, design, reduction, *etc.* problems. We are interested in the few following aspects.

We first expose, in this thesis, efficient exact matrix multiplication techniques, developed for both multiplication ($A = B \times C$) and product with accumulation ($A = A + B \times C$). We set up new schedules that allow us to minimize the extra memory requirements during a Strassen–style matrix multiplication, while keeping the complexity competitive with Winograd's multiplication algorithm. In order to obtain them, we develop external tools (pebble games), tight complexity computations and new hybrid algorithms.

We then use parallel technologies (multicore CPU and GPU) in order to efficiently accelerate the sparse matrix–dense vector multiplication (SpMV) or sparse–matrix dense matrix multiplication (SpMM), crucial to *blackbox* (block) algorithms. We also set up new hybrid, environment dependant, sparse matrix formats that help yield large speed-ups. We exemplify these results by speeding up the block Winograd rank algorithm in the LinBox library.

Finally, we establish generic design methods focusing on efficiency, especially via *building block* conceptions or self-optimization. We also propose tools for improving and standardizing code quality in order to make it more sustainable and more robust. This is applied in particular to the LinBox computer algebra library.

Author: Ibrahim Adamou
Title: Curve and Surface Bisectors, and Voronoi Diagram of a family of parallel half-lines in $\mathbb{R}^3$
Institution: Universidad de Cantabria

Thesis Advisor: Laureano Gonzalez-Vega and Mario Fioravanti
Committee Members: Tomas Recio, Marie-Françoise Roy, Bert Jüttler
Defended: September 10th, 2013 *Keywords:* Bisectors, Rational Curves and Surfaces, Voronoi Diagram, Spatial Subdivision, Meshing.

This thesis has three main parts: computation of the bisectors of two curves or a point and a curve in the plane, of the bisector of two surfaces in $\mathbb{R}^3$, and of the Voronoi diagram of a finite family of parallel half lines in $\mathbb{R}^3$, with the same orientation. These subjects are closely related, and have applications in CAD/CAGD and Computational Geometry. In each of the three parts, we present algorithmic methods for computing certain representations of the geometric object of interest: the bisector curve, the bisector surface, or the Voronoi diagram.

We present a new approach to determine an algebraic parametrization (rational or non rational) of the bisector curve of two given planar rational curves. The method uses Cramer's rule and algebraic elimination steps. The method is applied, in particular, to obtain parametrizations of the bisector of two rational plane curves, when one of them is a circle or a straight line. Then, this approach is generalized to determine an algebraic parametrization of the bisector surface of two low degree rational surfaces. We show how to easily obtain parametrizations of the bisector of the following pairs of surfaces: plane-quadric, plane-torus, circular cylinder-non developable quadric, circular cylinder-torus, cylinder-cylinder, cylinder-cone and cone-cone. These parametrizations are rational in most cases. In the remaining cases, the parametrization involves one square root which is well-suited to determine a good approximation of the bisector.

In addition, we present a different approach for the bisector curve problem. This new method uses dynamic color in GeoGebra (a dynamical geometry software) for the geometric and numerical characterizations of the bisector of two curves, or a curve and a point, in the plane. Even if it does not provide an algebraic representation, the method could lead to the computation of an approximate representation of the bisector curve.

The Voronoï diagram (VD) is a fundamental data structure in computational geometry with various applications in theoretical and practical areas. We consider the VD of a set of parallel half-lines, with the same orientation, constrained to a compact domain $\mathcal{D}_0 \subset \mathbb{R}^3$, with respect to the Euclidean distance. This new kind of VD can be used to provide an efficient solution to some problems in the drilling industry. We present an efficient algorithm for computing an approximate VD, using a box subdivision process, which produces a mesh representing the topology of the VD in $\mathcal{D}_0$. The concept of minimization diagram plays an important role in the method.

Recent and Upcoming Events

October 23–25, 2013
2nd International Seminar on Program Verification, Automated Debugging and Symbolic Computation
Beijing, China

Organizers: Tudor Jebelean, Wei Li, Dongming Wang
Dates: Submission deadline: September 10, 2013
Website: `http://pas2013.cc4cm.org/`

December 11–13, 2013
5th International Conference on Mathematical Aspects of Computer and Information Sciences
Nanning, China

Organizers: Dongming Wang, Jinzhao Wu
Dates: Submission deadline: October 12, 2013
Website: `http://www.mpi-inf.mpg.de/conference/macis2013/`

January 29–February 1, 2014
9th International Conference on Applied Informatics (ICAI 2014)
Eger, Hungary

Organizers: Attila Pethő, Franz Winkler, Roland Kunkli, Gabor Kusper
Dates: Submission deadline: January 6, 2014
Website: `http://icai.ektf.hu/`

March 31–April 4, 2014
Latin American Theoretical INformatics (LATIN 2014)
Montevideo, Uruguay

Organizers: A. Viola (PC chair), A. Pardo (Local chair)
Dates: Abstract submission: September 17, 2014; Full submission: September 22, 2014
Website: `http://www.fing.edu.uy/eventos/latin2014/`

June 16–20, 2014
25th International Conference on Probabilistic, Combinatorial, and Asymptotic Methods for the Analysis of Algorithms (AofA'14)
Paris, France

Organizers: M. Bousquet-Melou, M. Soria
Dates: Submission deadline: January 27, 2014
Website: `http://www.aofa14.upmc.fr/`

June 29–July 3, 2014
26th International Conference on Formal Power Series and Algebraic Combinatorics (FPSAC 2014)
Chicago, USA

Organizers: L. Billera, I. Novik (PC Chairs), B. Tenner (Local Chair)
Dates: Submission deadline: November 18
Website: `https://sites.google.com/site/fpsac2014/`

July 9–12, 2014
20th Conference on Applications of Computer Algebra
New York, USA

Organizers: R.H. Lewis (general chair), T. Shaska, I. Kotsireas (PC Chairs)
Dates: session proposal: February 28, 2014; talk submissions: May 15, 2014
Website: `http://faculty.fordham.edu/rlewis/aca2014/`

July 9–24, 2014
Vienna Summer of Logic
Vienna, Austria

Organizers: M. Baaz, A. Ciabattoni, Th. Eiter, A. Leitsch, G. Gottlob, T. Henzinger, V. Sabljakovic-Fritz, S. Szeider, H. Veith, S. Woltran and others
Website: `http://vsl2014.at/`
