

Commenced Publication in 1973

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Editorial Board

David Hutchison

Lancaster University, UK

Takeo Kanade

Carnegie Mellon University, Pittsburgh, PA, USA

Josef Kittler

University of Surrey, Guildford, UK

Jon M. Kleinberg

Cornell University, Ithaca, NY, USA

Friedemann Mattern

ETH Zurich, Switzerland

John C. Mitchell

Stanford University, CA, USA

Moni Naor

Weizmann Institute of Science, Rehovot, Israel

Oscar Nierstrasz

University of Bern, Switzerland

C. Pandu Rangan

Indian Institute of Technology, Madras, India

Bernhard Steffen

University of Dortmund, Germany

Madhu Sudan

Massachusetts Institute of Technology, MA, USA

Demetri Terzopoulos

New York University, NY, USA

Doug Tygar

University of California, Berkeley, CA, USA

Moshe Y. Vardi

Rice University, Houston, TX, USA

Gerhard Weikum

Max-Planck Institute of Computer Science, Saarbruecken, Germany

Jürgen Gerhard

Modular Algorithms in Symbolic Summation and Symbolic Integration



Springer

Author

Jürgen Gerhard
Maplesoft
615 Kumpf Drive, Waterloo, ON, N2V 1K8, Canada
E-mail: Gerhard.Juergen@web.de

This work was accepted as PhD thesis on July 13, 2001, at

Fachbereich Mathematik und Informatik
Universität Paderborn
33095 Paderborn, Germany

Library of Congress Control Number: 2004115730

CR Subject Classification (1998): F.2.1, G.1, I.1

ISSN 0302-9743

ISBN 3-540-24061-6 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media

springeronline.com

© Springer-Verlag Berlin Heidelberg 2004
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Boller Mediendesign
Printed on acid-free paper SPIN: 11362159 06/3142 5 4 3 2 1 0

To Herbert Gerhard (1921–1999)

Foreword

This work brings together two streams in computer algebra: symbolic integration and summation on the one hand, and fast algorithmics on the other hand.

In many algorithmically oriented areas of computer science, the *analysis of algorithms* – placed into the limelight by Don Knuth’s talk at the 1970 ICM – provides a crystal-clear criterion for success. The researcher who designs an algorithm that is faster (asymptotically, in the worst case) than any previous method receives instant gratification: her result will be recognized as valuable. Alas, the downside is that such results come along quite infrequently, despite our best efforts.

An alternative evaluation method is to run a new algorithm on examples; this has its obvious problems, but is sometimes the best we can do. George Collins, one of the fathers of computer algebra and a great experimenter, wrote in 1969: “I think this demonstrates again that a simple analysis is often more revealing than a ream of empirical data (although both are important).”

Within computer algebra, some areas have traditionally followed the former methodology, notably some parts of polynomial algebra and linear algebra. Other areas, such as polynomial system solving, have not yet been amenable to this approach. The usual “input size” parameters of computer science seem inadequate, and although some natural “geometric” parameters have been identified (solution dimension, regularity), not all (potential) major progress can be expressed in this framework.

Symbolic integration and summation have been in a similar state. There are some algorithms with analyzed run time, but basically the mathematically oriented world of integration and summation and the computer science world of algorithm analysis did not have much to say to each other.

Gerhard’s progress, presented in this work, is threefold:

- a clear framework for algorithm analysis with the appropriate parameters,
- the introduction of modular techniques into this area,
- almost optimal algorithms for the basic problems.

One might say that the first two steps are not new. Indeed, the basic algorithms and their parameters – in particular, the one called dispersion in Gerhard’s work – have been around for a while, and modular algorithms are a staple of computer algebra. But their combination is novel and leads to new perspectives, the almost optimal methods among them.

A fundamental requirement in modular algorithms is that the (solution modulo p) of the problem equal the solution of the (problem modulo p). This is generally not valid for all p , and a first task is to find a nonzero integer “resultant” r so that the requirement is satisfied for all primes p not dividing r . Furthermore, r has to be “small”, and one needs a bound on potential solutions, in order to limit the size and number of the primes p required. These tasks tend to be the major technical obstacles; the development of a modular algorithm is then usually straightforward. However, in order to achieve the truly efficient results of this work, one needs a thorough understanding of the relevant algorithmics, plus a lot of tricks and shortcuts.

The integration task is naturally defined via a limiting process, but the Old Masters like Leibniz, Bernoulli, Hermite, and Liouville already knew when to treat it as a symbolic problem. However, its formalization – mainly by Risch – in a purely algebraic setting successfully opened up perspectives for further progress. Now, modular differential calculus is useful in some contexts, and computer algebra researchers are aware of modular algorithms. But maybe the systematic approach as developed by Gerhard will also result in a paradigm shift in this field. If at all, this effect will not be visible at the “high end”, where new problem areas are being tamed by algorithmic approaches, but rather at the “low end” of reasonably domesticated questions, where new efficient methods will bring larger and larger problems to their knees.

It was a pleasure to supervise Jürgen’s Ph.D. thesis, presented here, and I am looking forward to the influence it may have on our science.

Paderborn, 9th June 2004

Joachim von zur Gathen

Preface

What fascinated me most about my research in symbolic integration and symbolic summation were not only the strong parallels between the two areas, but also the differences. The most notable non-analogy is the existence of a polynomial-time algorithm for rational integration, but not for rational summation, manifested by such simple examples as $1/(x^2 + mx)$, whose indefinite sum with respect to x has the denominator $x(x + 1)(x + 2) \cdots (x + m - 1)$ of exponential degree m , for all positive integers m . The fact that Moenck's (1977) straightforward adaption of Hermite's integration algorithm to rational summation is flawed, as discussed by Paule (1995), illustrates that the differences are intricate.

The idea for this research was born when Joachim von zur Gathen and I started the work on our textbook *Modern Computer Algebra* in 1997. Our goal was to give rigorous proofs and cost analyses for the fundamental algorithms in computer algebra. When we came to Chaps. 22 and 23, about symbolic integration and symbolic summation, we realized that although there is no shortage of algorithms, only few authors had given cost analyses for their methods or tried to tune them using standard techniques such as modular computation or asymptotically fast arithmetic. The pioneers in this respect are Horowitz (1971), who analyzed a modular Hermite integration algorithm in terms of word operations, and Yun (1977a), who gave an asymptotically fast algorithm in terms of arithmetic operations for the same problem. Chap. 6 in this book unites Horowitz's and Yun's approaches, resulting in two asymptotically fast and optimal modular Hermite integration algorithms. For modular hyperexponential integration and modular hypergeometric summation, this work gives the first complete cost analysis in terms of word operations.

Acknowledgements. I would like to thank:

My thesis advisor, Joachim von zur Gathen.

Katja Daubert, Michaela Huhn, Volker Strehl, and Luise Unger for their encouragement, without which this work probably would not have been finished.

My parents Johanna and Herbert, my sister Gisela, my brother Thomas, and their families for their love and their support.

My colleagues at Paderborn: the Research Group Algorithmic Mathematics, in particular Marianne Wehry, the MUPAD group,

in particular Benno Fuchssteiner, and SciFace Software, in particular Oliver Kluge.

My scientific colleagues all over the world for advice and inspiring discussions: Peter Bürgisser, Frédéric Chyzak, Winfried Fakler, Mark Giesbrecht, Karl-Heinz Kiyek, Dirk Kussin, Uwe Nagel, Christian Neliuss, Michael Nüsken, Walter Oevel, Peter Paule, Arne Storjohann, and Eugene Zima.

Waterloo, 23rd July 2004

Jürgen Gerhard

Table of Contents

1. Introduction	1
2. Overview	7
2.1 Outline	12
2.2 Statement of Main Results	13
2.3 References and Related Works	21
2.4 Open Problems	24
3. Technical Prerequisites	27
3.1 Subresultants and the Euclidean Algorithm	28
3.2 The Cost of Arithmetic	33
4. Change of Basis	41
4.1 Computing Taylor Shifts	42
4.2 Conversion to Falling Factorials	49
4.3 Fast Multiplication in the Falling Factorial Basis	57
5. Modular Squarefree and Greatest Factorial Factorization	61
5.1 Squarefree Factorization	61
5.2 Greatest Factorial Factorization	68
6. Modular Hermite Integration	79
6.1 Small Primes Modular Algorithm	80
6.2 Prime Power Modular Algorithm	85
6.3 Implementation	87
7. Computing All Integral Roots of the Resultant	97
7.1 Application to Hypergeometric Summation	103
7.2 Computing All Integral Roots Via Factoring	109
7.3 Application to Hyperexponential Integration	112
7.4 Modular LRT Algorithm	116
8. Modular Algorithms for the Gosper-Petkovšek Form	121
8.1 Modular GP' -Form Computation	134

9. Polynomial Solutions of Linear First Order Equations	149
9.1 The Method of Undetermined Coefficients	155
9.2 Brent and Kung's Algorithm for Linear Differential Equations	158
9.3 Rothstein's SPDE Algorithm	161
9.4 The ABP Algorithm	165
9.5 A Divide-and-Conquer Algorithm: Generic Case	169
9.6 A Divide-and-Conquer Algorithm: General Case	174
9.7 Barkatou's Algorithm for Linear Difference Equations	179
9.8 Modular Algorithms	180
10. Modular Gosper and Almkvist & Zeilberger Algorithms	195
10.1 High Degree Examples	198
References	207
Index	217

List of Figures and Tables

2.1	Algorithm dependency graph	14
3.1	The d th submatrix of the Sylvester matrix	29
4.1	Running times for shift by 1 with classical arithmetic	44
4.2	Running times for Taylor shift with classical arithmetic	45
4.3	Running times for Taylor shift with classical arithmetic	45
4.4	Running times for Taylor shift with fast arithmetic	48
4.5	Running times for Taylor shift with fast arithmetic	48
4.6	Comparison of running times for Taylor shift	49
4.7	Cost estimates for polynomial basis conversion	57
6.1	Timings for the 1st series with small primes Hermite integration	90
6.2	Timings for the 2nd series with small primes Hermite integration	91
6.3	Timings for the 3rd series with small primes Hermite integration	92
6.4	Timings for the 1st series with prime power Hermite integration	93
6.5	Timings for the 2nd series with prime power Hermite integration	94
6.6	Timings for the 3rd series with prime power Hermite integration	95
9.1	The linear system $Lu = c$ with respect to the monomial basis.	166
9.2	Cost estimates for polynomial solutions of first order equations	192
9.3	Cost estimates for modular algorithms solving first order equations	192

List of Algorithms

3.4	Monic Extended Euclidean Algorithm	29
4.7	Small primes modular Taylor shift	48
4.18	Modular conversion from $\mathcal{M}$ to $\mathcal{M}_{\mathbf{b}}$	55
4.19	Modular conversion from $\mathcal{M}_{\mathbf{b}}$ to $\mathcal{M}$	55
4.23	Evaluation in the falling factorial basis	58
4.25	Interpolation in the falling factorial basis	58
5.2	Yun's squarefree factorization	61
5.6	Small primes modular squarefree factorization	63
5.12	Prime power modular squarefree factorization	67
5.18	Gff computation	70
5.20	Gff computation à la Yun	71
5.24	Small primes modular gff computation	74
6.4	Small primes modular Hermite integration	83
6.9	Prime power modular Hermite integration	85
7.2	Modular bivariate EEA	98
7.12	Prime power modular integral root distances	104
7.20	Prime power modular Man & Wright algorithm	109
7.25	Small primes modular integral residues	113
7.29	Small primes modular LRT algorithm	117
8.2	Gosper-Petkovšek form	121
8.6	Small primes modular shift gcd	123
8.12	Small primes modular Gosper-Petkovšek form	126
8.26	GP' -refinement computation	137
8.28	GP' -form computation	139
8.32	Small primes modular GP' -refinement	140
8.37	Prime power modular GP' -refinement	144
9.12	Method of undetermined coefficients: part I	155
9.15	Method of undetermined coefficients: part II	157
9.20	Brent & Kung's algorithm	159
9.23	Rothstein's SPDE algorithm	162
9.25	Rothstein's SPDE algorithm for difference equations	163
9.27	ABP algorithm for first order differential equations	165
9.31	ABP algorithm for first order difference equations	168
9.35	Difference divide & conquer: generic case	171

9.38	Differential divide & conquer: generic case	173
9.41	Difference divide & conquer: general case	175
9.44	Differential divide & conquer: general case	178
9.46	Barkatou's algorithm	179
9.56	Modular Brent & Kung / ABP algorithm	183
9.60	Modular Barkatou algorithm	187
9.64	Modular difference divide & conquer	189
9.67	Modular differential divide & conquer	191
10.1	Modular Gosper algorithm	195
10.4	Modular Almkvist & Zeilberger algorithm	196