Symbolic Computation Group

Annual Report 1996

Department of Computer Science
University of Waterloo

1996 Annual Report

Symbolic Computation Group
Department of Computer Science
University of Waterloo
Waterloo ON N2L 3G1
Canada
E-mail: scg@daisy.uwaterloo.ca
URL: http://daisy.uwaterloo.ca/

May 1, 1996

SCG People

Directors:	Keith Geddes, George Labahn
Admin. Assistant:	Jennifer Keir
Co-Researchers:	Rob Corless (Univ. of Western Ontario), Jeff Shallit
Research Associates:	Dave Hare, Kelly Roach
Postdoctoral Fellows:	François Boulier, Martin Mohrenschildt, Ken Parker
Research Assistants:	David Clark, Arne Storjohann
Graduate Students:	Ho Fai James Chan (M.Math 1996), Wolfgang Heidrich (M.Math 1995), Lee Qiao (M.Math 1995), Enrico Au-Yeung, Fred Chapman, Geck Hong, Masoud Kavian
Undergraduate Research Assistants (Full-time and Part-time):	Fiona Beardwood (Spring 95), Michael Buckley (Winter and Spring 95), Blaine Campbell (Spring 95), Kevin Hare (Fall 95), Theodore Kolokolnikov (Winter 96), Marni Mishna (Winter 96), Todd Parsons (95-96), Shaloub Razak (Spring 95).
Visiting Researchers:	Frédéric Chyzak (November 1995), Mark van Hoeij (January 1996), Marc Rybowicz (July - August 1995), Ernst Weniger (January - December 1996)

Summary

The Symbolic Computation Group has as its primary goal the research and development of algorithms for computer algebra, including both symbolic computation and hybrid symbolic-

numeric computation. The algorithms developed are incorporated into the Maple computer algebra system.

Particular areas of research and development which have been targeted during the past year include symbolic integration, computing solutions of ordinary and partial differential equations (both symbolically and numerically), extended-precision numerical evaluation of special functions, facilities for handling piecewise functions, pattern matching, matrix computations, and tensor computations.

Detailed descriptions of progress in the past year are presented in the following sections.

Project Reports

1 Hypergeometric Function Representations

The hypergeometric function F can be defined by

$$\mathrm{F}\left(\vec{a};\vec{b};z\right) = \sum_{j=0}^{\infty} \frac{(\vec{a})_j}{\left(\vec{b}\right)_j} \frac{z^j}{j!} = \sum_{j=0}^{\infty} \Gamma\left(\begin{matrix}\vec{a}+j, \vec{b} \\ \vec{a}, \vec{b}+j, 1+j\end{matrix}\right) z^j$$

Various simple expressions and special functions of applied mathematics can be expressed in terms of F. Hypergeometric functions are applicable to symbolic integration, differential equations, closed-form summation, and difference equations [7], [16], [20]. Various methods

create answers in terms of F [17], [19]. The development of an algorithm to compute formula representations of instances of F described in Roach[22] is an important advance that converts many such answers in terms of F into answers in terms of elementary or special functions.

The basic idea of the algorithm is to enlarge a known table of summations infinitely by the use of differential and contiguity relations. The known table is implemented by a **Lookup** routine. The differential and contiguity relations are implemented by shift operators A_i and B_i and inverse shift operators A_i^{-1} and B_i^{-1}. The **shift operators** A_i and B_i are defined by

$$A_i = \frac{z}{a_i} D + 1$$

$$B_i = \frac{z}{b_i - 1} D + 1$$

Supposing

$$\begin{aligned} L &= z \prod_{j=1}^{p} (z\,D + a_j) - (z\,D) \prod_{j=1}^{q} (z\,D + b_j - 1) \\ &= \alpha_{\mathrm{d}}\left(\vec{a}, \vec{b}, z\right) A_i^d + \ldots + \alpha_0\left(\vec{a}, \vec{b}, z\right) \\ &= \beta_{\mathrm{d}}\left(\vec{a}, \vec{b}, z\right) B_i^d + \ldots + \beta_0\left(\vec{a}, \vec{b}, z\right) \end{aligned}$$

is the differential operator for the differential equation satisfied by $\mathrm{F}(\vec{a}; \vec{b}; z)$, the **inverse shift operators** A_i^{-1} and B_i^{-1} are defined by

$$A_i^{-1} = -\sum_{n=0}^{d-1} \frac{\alpha_{\mathrm{n}+1}\left(\vec{a} - \vec{e}_i, \vec{b}, z\right)}{\alpha_0\left(\vec{a} - \vec{e}_i, \vec{b}, z\right)} A_i^n$$

$$B_i^{-1} = -\sum_{n=0}^{d-1} \frac{\beta_{\mathrm{n}+1}\left(\vec{a}, \vec{b} + \vec{e}_i, z\right)}{\beta_0\left(\vec{a}, \vec{b} + \vec{e}_i, z\right)} B_i^n$$

With some restrictions, the operators A_i, B_i, A_i^{-1}, and B_i^{-1} are used to increment and decrement indices a_i and b_i appearing inside $\mathrm{F}(\vec{a}; \vec{b}; z)$.

If we need to compute $\mathrm{F}(\vec{a}; \vec{b}; z)$ where $\vec{m} = \vec{a} - \vec{a}_0 \in \mathbb{Z}$, $\vec{n} = \vec{b}_0 - \vec{b} \in \mathbb{Z}$, and $\mathrm{F}(\vec{a}_0; \vec{b}_0; z)$ is available to us through **Lookup**, the simplest idea that might occur to us would be to compute $\mathrm{F}(\vec{a}; \vec{b}; z)$ by use of the equation

$$\mathrm{F}\left(\vec{a}; \vec{b}; z\right) = \prod_{i=1}^{p} A_i^{m_i} \prod_{i=1}^{q} B_i^{n_i}\, \mathrm{F}\left(\vec{a}_0; \vec{b}_0; z\right)$$

However, this equation can be wrong and the simple approach will not always work because of restrictions on where A_i, B_i, A_i^{-1}, and B_i^{-1} are defined. Roach[22] presents a better strategy that resembles this simple idea but which applies the operators in a proper sequence starting at a suitable origin.

A sequence $\vec{S}$ of shift and inverse shift operators A_i, B_i, A_i^{-1}, and B_i^{-1} is a **proper sequence** if the composition $S_{|S|} \ldots S_1$ is defined. A pair $(\vec{a}_0; \vec{b}_0)$ is a **suitable origin** if $\vec{a}_0$ and $\vec{b}_0$ are free of nonpositive integers, $\vec{a}_0$ and $\vec{b}_0$ are disjoint, and integer elements of $\vec{b}_0$ are $\geq d = \max(p, q+1)$. A pair $(\vec{a}; \vec{b})$ is **accessible** from a pair $(\vec{a}_0; \vec{b}_0)$ if there exists a proper sequence $\vec{S}$ of shift and inverse shift operators A_i, B_i, A_i^{-1}, and B_i^{-1} such that

$$\mathrm{F}\left(\vec{a}; \vec{b}; z\right) = S_{|S|} \ldots S_1 \, \mathrm{F}\left(\vec{a}_0; \vec{b}_0; z\right)$$

Given any vector $\vec{v}$, define $[\vec{v}]_r$ to be the subvector of elements of $\vec{v}$ which are congruent to r mod 1. Given any permutation π of $\{1, \ldots, |\vec{v}|\}$ define $\pi(\vec{v}) = (v_{\pi(1)}, \ldots, v_{\pi(|\vec{v}|)})$. Then Roach[22] obtains the following very general theorem.

Theorem. *Let*
(1) $\vec{a}$ *and* $\vec{b}$ *be free of nonpositive integers.*
(2) $\vec{a}$ *and* $\vec{b}$ *be disjoint.*
(3) π *sort* $\vec{x} = (a_1, \ldots, a_p, b_1, \ldots, b_q)$ *into nondescending order*
(4) $(\vec{a}_0; \vec{b}_0)$ *be a suitable origin*
(5) $\vec{a} - \vec{a}_0, \vec{b} - \vec{b}_0 \in \mathbb{Z}$
(6) $\vec{x}_0 = (a_{01}, \ldots, a_{0p}, b_{01}, \ldots, b_{0q})$
(7) $[\pi(\vec{x}_0)]_r$ *be nondescending for every* $r \in [0, 1)$
Then $(\vec{a}; \vec{b})$ *is accessible from* $(\vec{a}_0; \vec{b}_0)$.

The **Lookup** routine part of the algorithm supplies the initial "table" that is extended infinitely by the use of shift operators and inverse shift operators as described above. Currently, the **Lookup** routine consists of an actual table of 71 different $[\vec{a}_0, \vec{b}_0, B, C, M, \rho]$ entries called **certificates** plus procedures which implement 19 different formulas, each of which, in effect, add infinitely many more certificates to the lookup table. The numbers of different formulas implemented so far are summarized by the following table:

${}_0F_0$	1	${}_0F_1$	1
${}_1F_0$	1	${}_1F_1$	3
${}_1F_2$	3	${}_2F_1$	4
${}_0F_3$	2	${}_0F_q$	1
Deriv	1	Lerch Φ	1
PFD Dupl	1		

Roach[22] describes the Derivative Formula, the Lerch Φ Formula, and the PFD Duplication Formula shown here.

The main accomplishment of the algorithm described in Roach[22] is the essential reproduction of 1504 formulas in 9 tables of representations of $\mathrm{F}(\vec{a}; \vec{b}; z)$ listed in *Integrals and Series, Volume 3: More Special Functions* [20]. The following examples are some formulas produced by this algorithm.

$$\mathrm{F}\left(-\frac{3}{2}, -\frac{1}{2}; 2; z\right) = -\frac{4 + 24\,z - 28\,z^2}{15\,z\,\pi}\,\mathrm{K}\left(\sqrt{z}\right) + \frac{4 + 56\,z + 4\,z^2}{15\,z\,\pi}\,\mathrm{E}\left(\sqrt{z}\right)$$

$$\mathrm{F}\left(-\frac{1}{2}, 1; \frac{1}{4}, \frac{1}{2}, \frac{3}{4}; z\right)$$
$$= 1 + z^{1/4}\sqrt{2}\sqrt{\pi}\, e^{2\sqrt{z}}\,\mathrm{erf}\left(\sqrt{2}\, z^{1/4}\right) - z^{1/4}\sqrt{2}\sqrt{\pi}\, e^{-2\sqrt{z}}\,\mathrm{erfi}\left(\sqrt{2}\, z^{1/4}\right)$$
$$- 2\sqrt{z}\,\pi\,\mathrm{erf}\left(\sqrt{2}\, z^{1/4}\right)\,\mathrm{erfi}\left(\sqrt{2}\, z^{1/4}\right)$$

$$\mathrm{F}\left(\frac{3}{2}; \frac{5}{2}, 5; z\right)$$
$$= -\frac{432 - 24\, z + 96\, z^2}{5\, z^3}\, I_0\left(2\sqrt{z}\right) + \frac{432 + 192\, z + 48\, z^2}{5\, z^{7/2}}\, I_1\left(2\sqrt{z}\right)$$
$$- \frac{48}{5\, z}\,\pi\,\left(I_0\left(2\sqrt{z}\right)\,\mathbf{L}_1\left(2\sqrt{z}\right) - I_1\left(2\sqrt{z}\right)\,\mathbf{L}_0\left(2\sqrt{z}\right)\right)$$

$$\mathrm{F}\left(-\frac{1}{2}, \frac{1}{2}, 1; \frac{3}{2}, \frac{5}{2}; z\right)$$
$$= -\frac{3 - 3\, z}{16\, z} - \frac{3 - 3\, z^2}{32\, z^{3/2}}\,\left(\log\left(1 - \sqrt{z}\right) - \log\left(1 + \sqrt{z}\right)\right) + \frac{3}{8\sqrt{z}}\,\mathrm{Li}_2\left(\sqrt{z}\right)$$
$$- \frac{3}{8\sqrt{z}}\,\mathrm{Li}_2\left(-\sqrt{z}\right)$$

This algorithm has already been extended to compute representations for $\mathrm{F}(\vec{a}; \vec{b}; -z)$, therefore making the algorithm encompass even more elementary and special functions. In the future, we expect to generalize this algorithm to Meijer G function representations, representations of multiple hypergeometric functions (e.g. Appell and Lauricella functions [8]) and to investigate $\mathrm{F}(\vec{a}; \vec{b}; 1)$ representations for $p = q + 1$, a special case the current algorithm does not fully address.

2 Symbolic Integration

The Meijer G function is defined by the contour integral

$$\mathrm{G}\left(\vec{a}; \vec{b}; \vec{c}; \vec{d}; z\right) = \frac{1}{2\,\pi\, i}\oint_L \Gamma\left(\begin{matrix}1 - \vec{a} + y, \vec{c} - y \\ \vec{b} - y, 1 - \vec{d} + y\end{matrix}\right)\, e^{y\, z}\, dy$$

where L is one of three types of integration paths $L_{\gamma + i\infty}$, L_∞, and $L_{-\infty}$. The Meijer G function is important to the theory of definite integration because, first, it is a generalization of the hypergeometric function F and can represent a wide range of simple expressions and special functions of applied mathematics and, second, because there are some general theorems about definite integrals of integrands involving the Meijer G function.

The following theorem shows how to integrate a single G function.

Theorem. (One G Function.)

$$\int_0^\infty z^t \, \mathrm{G}\left(\vec{a};\vec{b};\vec{c};\vec{d};u\log(z)+v\right)\,dz = \frac{1}{u}\,e^{-\left(\frac{t+1}{u}\right)v}\,\Gamma\left(\begin{matrix}-\vec{a}+1-\frac{t+1}{u},\vec{c}+\frac{t+1}{u}\\ \vec{b}+\frac{t+1}{u},-\vec{d}+1-\frac{t+1}{u}\end{matrix}\right)$$

The next theorem shows how to integrate a product of two G functions.

Theorem. (Two G Functions.)

$$\begin{aligned}&\int_0^\infty z^t \, \mathrm{G}\left(\vec{a}_1;\vec{b}_1;\vec{c}_1;\vec{d}_1;u\log(z)+v_1\right)\,\mathrm{G}\left(\vec{a}_2;\vec{b}_2;\vec{c}_2;\vec{d}_2;u\log(z)+v_2\right)\,dz\\ &\quad=\frac{1}{u}\,e^{-\left(\frac{t+1}{u}\right)v_2}\\ &\qquad\times\mathrm{G}\Bigg(\vec{a}_1,-\vec{c_2}-\frac{t+1}{u}+1;\vec{b}_1,-\vec{d_2}-\frac{t+1}{u}+1;\\ &\qquad\qquad\vec{c_1},-\vec{a}_2-\frac{t+1}{u}+1;\vec{d_1},-\vec{b}_2-\frac{t+1}{u}+1;v_1-v_2\Bigg)\end{aligned}$$

Products of more than two G functions can be integrated in terms of so-called multiple hypergeometric functions which may or may not be able to simplify into more recognizable functions.

Since Heaviside functions can be represented by Meijer G functions, either or both of the limits of integration 0 and ∞ can be changed to positive reals by making the integrand more complicated.

The integrand need not be a polynomial in Meijer G functions, but could be expressible as a converging infinite sum of Meijer G function products. In this case, an infinite sum can be integrated and frequently simplified into Zeta functions or hypergeometric functions.

Differentiation under the integral sign combined with the Meijer G approach can be applicable to integrands containing logarithms. Easy to recognize substitutions that convert various types of integrals into other integrals that are more suited to the Meijer G approach is also quite common.

The Meijer G function has various basic properties [7], [16], [20]. Among the more interesting of these are the Duplication Formula:

Theorem. (Duplication Formula.)

$$\begin{aligned}&\mathrm{G}\left(\vec{a};\vec{b};\vec{c};\vec{d};\frac{z}{n}\right)\\ &\quad=(2\,\pi)^{-(n-1)\,\beta/2}\;n^{1+\delta/2-\sigma}\;\mathrm{G}\left(\Delta\left(\vec{a},n\right);\Delta\left(\vec{b},n\right);\Delta\left(\vec{c},n\right);\Delta\left(\vec{d},n\right);z+n\,\delta\log(n)\right)\end{aligned}$$

and Slater's Theorem:

Theorem. (Slater's Theorem.) *If $\delta < 0$ or ($\delta = 0$ and $\mathrm{Re}(z) < 0$) and the elements of $\vec{c}$ are distinct mod 1, then*

$$\begin{aligned}&\mathrm{G}\left(\vec{a};\vec{b};\vec{c};\vec{d};z\right)\\&\quad=\sum_{h=1}^{p}\left(\Gamma\left(\begin{matrix}1-\vec{a}+c_h, c^*-c_h\\ \vec{b}-c_h, 1-\vec{d}-c_h\end{matrix}\right)\ e^{c_h\,z}\right.\\&\qquad\left.\times\,\mathrm{F}\left(1-\vec{a}+c_h, 1-\vec{b}+c_h; 1-c^*+c_h, 1-\vec{d}+c_h; (-1)^{n-p}\ e^{z}\right)\right)\end{aligned}$$

where $c^* = \vec{c}$ with c_h omitted.

Example 1. We show how to compute

$$\int_0^\infty \frac{(a\,x+b)^{\beta-1}}{(c\,x+d)^{\beta+1}}\,dx = -\frac{b^\beta\,d^{-\beta}-a^\beta\,c^{-\beta}}{(a\,d-b\,c)\,\beta}$$

The two binomials in the integrand can both be expressed as Meijer G functions.

$$(a\,x+b)^{\beta-1} = \frac{b^{\beta-1}\,\mathrm{G}\left(1;;-\beta+1;;-\log(x)+\log\left(\frac{b}{a}\right)\right)}{\Gamma\left(-\beta+1\right)}$$

$$\frac{1}{(c\,x+d)^{\beta+1}} = \frac{d^{-\beta-1}\,\mathrm{G}\left(1;;\beta+1;;-\log(x)+\log\left(\frac{d}{c}\right)\right)}{\Gamma\left(\beta+1\right)}$$

So, applying the Two G Functions Theorem we get

$$\begin{aligned}&\int_0^\infty \frac{(a\,x+b)^{\beta-1}}{(c\,x+d)^{\beta+1}}\,dx\\&\quad=\int_0^\infty \frac{b^{\beta-1}\,d^{-\beta-1}}{\Gamma\left(-\beta+1\right)\,\Gamma\left(\beta+1\right)}\,\mathrm{G}\left(1;;-\beta+1;;-\log(x)+\log\left(\frac{b}{a}\right)\right)\\&\qquad\times\,\mathrm{G}\left(1;;\beta+1;;-\log(x)+\log\left(\frac{d}{c}\right)\right)\,dx\\&\quad=-\frac{b^{\beta-1}\,d^{-\beta}\sin\left(\pi\,\beta\right)}{c\,\pi\,\beta}\,\mathrm{G}\left(1,-\beta+1;;1,-\beta+1;;\log\left(\frac{b}{a}\right)-\log\left(\frac{d}{c}\right)\right)\\&\quad=\frac{a^\beta\,c^{-\beta}-b^\beta\,d^{-\beta}}{d\,\beta\,a}\,\mathrm{F}\left(1;;\frac{b\,c}{a\,d}\right)\end{aligned}$$

where the final Meijer G function is reduced to a single hypergeometric F function by Slater's Theorem. This hypergeometric F function

$$\mathrm{F}\left(1;;\frac{b\,c}{a\,d}\right) = \frac{a\,d}{a\,d-b\,c}$$

is easily computed by the algorithm in Roach[22]. So, finally, we get the answer

$$\int_0^\infty \frac{(a\,x+b)^{\beta-1}}{(c\,x+d)^{\beta+1}}\,dx = -\frac{b^\beta\,d^{-\beta}-a^\beta\,c^{-\beta}}{(a\,d-b\,c)\,\beta}$$

Example 2. We show how to compute

$$\int_0^\infty \sqrt{x}\,\mathrm{J}_{2\,\nu+1}\left(b\sqrt{x}\right)\,\mathrm{J}_\nu\left(c\,x\right)\,dx = -\frac{b}{2\,c^2}\,\mathrm{J}_{\nu+1}\left(\frac{b^2}{4\,c}\right)$$

The two Bessel functions in the integrand can both be expressed as Meijer G functions.

$$\mathrm{J}_{2\,\nu+1}\left(b\sqrt{x}\right) = \mathrm{G}\left(;;\nu+\frac{1}{2};-\nu-\frac{1}{2};\log(x)+2\log\left(\frac{b}{2}\right)\right)$$

$$\mathrm{J}_\nu\left(c\,x\right) = \mathrm{G}\left(;;\frac{\nu}{2};-\frac{\nu}{2};2\log(x)+2\log\left(\frac{c}{2}\right)\right)$$

Applying the Duplication Formula for Meijer G functions to the first Meijer G function produces

$$\mathrm{G}\left(;;\nu+1;-\nu;\log(x)+2\log\left(\frac{b}{2}\right)\right)$$
$$= 2\,\mathrm{G}\left(;;\frac{\nu}{2}+1,\frac{\nu}{2}+\frac{1}{2};\frac{1}{2}-\frac{\nu}{2},-\frac{\nu}{2};2\log(x)+4\log\left(\frac{b}{4}\right)\right)$$

So, applying the Two G Functions Theorem we get

$$\int_0^\infty \sqrt{x}\,\mathrm{J}_{2\,\nu+1}\left(b\sqrt{x}\right)\,\mathrm{J}_\nu\left(c\,x\right)\,dx$$
$$= 2\int_0^\infty \sqrt{x}\,\mathrm{G}\left(;;\frac{\nu}{2}+1,\frac{\nu}{2}+\frac{1}{2};\frac{1}{2}-\frac{\nu}{2},-\frac{\nu}{2};2\log(x)+4\log\left(\frac{b}{4}\right)\right)$$
$$\times\,\mathrm{G}\left(;;\frac{\nu}{2};-\frac{\nu}{2};2\log(x)+2\log\left(\frac{c}{2}\right)\right)\,dx$$
$$= \frac{\sqrt{2}}{c^{3/2}}\,\mathrm{G}\left(;;\frac{2\,\nu+3}{4};-\frac{2\,\nu+1}{4};-4\log(2)+4\log\left(\frac{b}{2}\right)-2\log\left(\frac{c}{2}\right)\right)$$
$$= -\frac{2^{-4-3\,\nu}\,c^{-3-\nu}\,b^{3+2\,\nu}}{\Gamma\left(\nu+2\right)}\,\mathrm{F}\left(;\nu+2;-\frac{b^4}{64\,c^2}\right)$$

where the final Meijer G function is reduced to a single hypergeometric F function by Slater's Theorem. This hypergeometric F function

$$\mathrm{F}\left(;\nu+2;-\frac{b^4}{64\,c^2}\right) = \frac{8\,c\,\left(\nu+1\right)\,\Gamma\left(\nu+1\right)\,\left(\frac{b^2}{8\,c}\right)^{-\nu}}{b^2}\,\mathrm{J}_{\nu+1}\left(\frac{b^2}{4\,c}\right)$$

is easily computed by the algorithm in Roach[22]. So, finally, we get the answer

$$\int_0^\infty \sqrt{x}\,\mathrm{J}_{2\,\nu+1}\left(b\sqrt{x}\right)\,\mathrm{J}_\nu\left(c\,x\right)\,dx = -\frac{b}{2\,c^2}\,\mathrm{J}_{\nu+1}\left(\frac{b^2}{4\,c}\right)$$

3 Elliptic Integration

Let $R(x,y)$ be a rational function in the variables x and y with y^2 a polynomial in x of degree 3 or 4. For the past two versions, Maple has had the ability to reduce elliptic integrals of the form

$$\int_a^b R(x,y)dx$$

to its Legendre normal form in terms of Legendre's elliptic F, E, Π and K functions. In the past year, work has gone into improving the computation of such a normal form. In particular, the ability to work with symbolic quantities has been greatly improved. For example, we now obtain the following result.

assume$(0 < k, k < 1, 1 < z, z < 1/k)$:

$$\int_0^z \sqrt{(1-x^2)\,(1-k^2x^2)}dx$$

$$= 1/3\,z\sqrt{k^2z^4+1-z^2-k^2z^2} + \left(1/3 - 1/3\,k^{-2}\right)\,K(k) + \frac{(1/3+1/3\,k^2)\,E(k)}{k^2}$$

$$-1/3\,i\left(2\,F\left(\sqrt{\frac{z^2-1}{1-k^2}}z^{-1},\sqrt{1-k^2}\right) - \left(1+\,k^2\right)\Pi\left(\sqrt{\frac{z^2-1}{1-k^2}}z^{-1},1-k^2,\sqrt{1-k^2}\right)\right)$$

Maple now also converts elliptic integrals represented in trigonometric form

$$\int_a^b R(\sin(x),\cos(x),y)dx$$

where y^2 is a quadratic form in $\sin(x)$ and $\cos(x)$. For example,

assume$(0 < k, k < 1)$:

$$\int_0^{1/2\,\pi} \frac{\sqrt{1+k^2\left(\sin(x)\right)^2}}{2+\sin(x)}dx = k^2\left(1/4\,\frac{\pi}{k} + 1/2\,\arcsin\left(\frac{-1+k^2}{1+k^2}\right)k^{-1}\right) - 1/4\,\frac{\pi\,(1+4\,k^2)}{\sqrt{3+12\,k^2}}$$

$$-1/2\,\arctan\left(\frac{-1+2\,k^2}{\sqrt{3+12\,k^2}}\right)\left(1+4\,k^2\right)\frac{1}{\sqrt{3+12\,k^2}} - 2\,k^2\,K\left(\frac{k}{\sqrt{1+k^2}}\right)\frac{1}{\sqrt{1+k^2}}$$

$$+2/3\,\left(1+4\,k^2\right)\Pi\left(-1/3,\frac{k}{\sqrt{1+k^2}}\right)\frac{1}{\sqrt{1+k^2}}$$

Additional work has been done to ensure that the elliptic functions evaluate over the entire complex plane. Thus we now have

$$\int_0^1 \frac{1}{(1+x^4)\sqrt{(1-x^2)(1-1/4x^2)}}dx = 1/4 \sum_{\alpha=\mathrm{RootOf}(Z^4+1)} \Pi\left(\frac{1}{\alpha^2}, \frac{1}{2}\right)$$

which evaluates numerically to 1.293415691.

Further work on this project is continuing. In particular, the normal form is only unique up to a Landen transform, and so a number of different answers are possible. We plan to investigate a new algorithm that reduces the number of algebraic quantities in the result. For example,

$$\int_0^{\frac{1}{2}} \frac{\sqrt{1+x^4}}{1-x^4}dx$$

produces a result containing a few hundred lines of output. However, it is possible to determine that this integral is, in fact, elementary and takes the form

$$-\frac{\sqrt{2}}{4}\arctan\left(\frac{\sqrt{17}\sqrt{2}}{4}\right) + \frac{\sqrt{2}}{8}\ln\left(\frac{4\sqrt{17}\sqrt{2}}{9} + \frac{25}{9}\right) + \frac{\pi\sqrt{2}}{8}.$$

This project is being carried out by Professor G. Labahn.

4 Integral Transforms

During the past year, we have completed approximately half of the second phase of this project. The first phase consisted of the creation of a new Maple package for computing a variety of new integral transforms (e.g., Hilbert, Fourier Sine, etc.). The second phase involves strengthening the capabilities of the package to cover all the material in classical tables of transforms. This work has been completed for the Fourier and Laplace transforms. The remaining transforms will be completed with the aid of an undergraduate student, R. Shahidi, this coming summer.

The primary investigations on this topic have been done by an undergraduate student, K. Hare, under the direction of Professor G. Labahn.

5 Hybrid Symbolic-Numeric Solutions of ODEs

Research work is continuing into the enhanced problem-solving capabilities which can be achieved by hybrid symbolic-numeric techniques, building on previous successes in hybrid methods for definite integration [9, 10]. For his Masters project under the supervision of Professor K.O. Geddes, H.F.J. Chan [4] has investigated methods for the solution of systems of ordinary differential equations based on a divided-difference calculus proposed by Kahan. This approach requires a computer algebra system to carry out the divided-difference calculus, and then a problem-specific numerical method is generated which can be very efficient even for high precision requirements. While the results are promising, more work is required before this approach can be incorporated into a production system.

A project carried out by F. Beardwood, on a work-term as a senior undergraduate and supervised by Professor K.O. Geddes, implemented improved algorithms for computing series

solutions of ODEs. In particular, techniques were implemented for developing series solutions of a linear ODE around an irregular singular point.

6 Partial Differential Equations

For the past six months we have been designing and implementing a package for the symbolic manipulation of polynomial differential equations, with ordinary or partial derivatives. The package provides functionalities for differential equations close to that offered by the Gröbner basis package for algebraic polynomials.

Given any system of polynomial differential equations S (homogenous or inhomogenous), the package computes a representation of the radical $\mathfrak{b}$ of the differential ideal generated by S. This can be used to solve problems such as:

1. To decide membership in $\mathfrak{b}$ through simple reductions. Thus, given any two differential polynomials p and q one can use the representation to decide if p is equivalent to q modulo S and hence, to compute in differential fields or rings presented by generators and relations. Such problems arise for instance when studying symmetries of differential and partial differential equations.

2. To obtain information about the structure of the set of the solutions of S by computing invariants of this set. Information of this type includes determining if S has any solutions (triviality of the ideal $\mathfrak{b}$), or determining if there are functions in S which can be chosen freely (i.e. the differential dimension of $\mathfrak{b}$). One can also express some quantities in terms of some others. This is important in a number of applications in Physics, the compatibility conditions for partial differential equations and, in the case of Nonlinear Control Theory, observability, parameter identification, input–output inversion, etc.

3. To compute formal power series solutions of S, in the case where the initial conditions do not cancel the denominators of the terms of the series.

The primary research in this work concerns the underlying mathematical theory of differential algebra. This work is being carried out by François Boulier, a postdoctoral fellow with our group.

An optimization of the keystone Rosenfeld's lemma and an analogue of Buchberger's second criterion were proved. These results save unnecessary computations during the computation of the representation of $\mathfrak{b}$. These results have been submitted [3] to IMACS'96, and reported at the closing session of the "Special Year in Differential Algebra and Algebraic Geometry" organized by Professors Sit and Hoobler at the City College of New York in January 1996.

Example 1

```
> read differential_algebra;
```

The package allows to manipulate differential polynomials of differential polynomial rings in many different differential indeterminates, endowed with many derivations over quite general ground fields.

The field $K = \mathbb{Q}(a, b)$ defined below is a purely transcendental field extension of the field of rational numbers. The differential polynomial ring R is the ring of all the differential polynomials in two differential indeterminates u and v, endowed with derivations w.r.t. x and y, over $G = K(x, y)$. The ground field G or R contains K as subfield of constants and the two independent variables x and y associated with the derivations.

```
> K := ground_field (generators = [a, b]);
                                  K := your_field

> R := differential_ring (field_of_constants = K, derivations = [x,y],
                                                  ranking = [tord[u, v]]);
                                  R := your_ring
```

A ranking is a total ordering over the set of all the derivatives of the differential indeterminates of R. Given a ranking, any differential polynomial (which does not belong to the ground field) has a leader. Here, all the derivatives of u and v are ordered according to *tord* which is an orderly ranking. The package provides a function which describes rankings.

```
> which_ranking_between (u, v, R);
tord: _U [theta] > _V [phi] when
    |theta| > |phi| or
    |theta| = |phi| and theta > phi for the lexicog. order: x > y or
    theta = phi and _U > _V according to: u > v

> p1 := a * v[] * u[x,x] - u[x];
> p2 := u[x,y];
> p3 := u[y,y]^2 - b;
> leader (p1, R);
```

$$u_{xx}$$

The main function of the package is called *Rosenfeld–Gröbner*. It is described in [2] but the implementation involves many nontrivial optimizations which were developed in the Symbolic Computation Group.

It represents the radical *ideal* of the differential ideal generated by p_1, p_2 and p_3 as an intersection of two regular differential ideals presented by triangular systems of differential equations and inequations.

```
> ideal := Rosenfeld_Groebner ([p1,p2,p3], R);
                          ideal := [regular, regular]

> map (equations, ideal);
```

$$[[u_{yy}^2 - b,\ u_x],\ [a\,v\,u_{xx} - u_x,\ u_{xy},\ u_{yy}^2 - b,\ v_y]]$$

```
> map (inequations, ideal);
```

$$[[u_{yy}],\ [v,\ u_{yy}]]$$

The representation computed by *Rosenfeld–Gröbner* provides an algorithm to decide membership in *ideal* through simple reductions: a differential polynomial belongs to *ideal* if and only if its reduced form is $[0,\ 0]$.

The computations below show that v_y belongs to the second regular ideal but not to the first and that u_x belongs to the first regular ideal but not to the second. Therefore, the decomposition is minimal. In general, the decomposition may contain redundant regular ideals. Deciding the inclusion between regular differential ideals presented by systems of equations and inequations is an open problem.

We see also that $u_x v_y$ belongs to *ideal*. Thus the ideal is not prime.

```
> reduced_form (v[y], ideal);
```

$$[v_y,\ 0]$$

```
> reduced_form (u[x], ideal);
```

$$[0,\ u_x]$$

```
> reduced_form (u[x]*v[y], ideal);
```

$$[0,\ 0]$$

Formal power series can be computed from regular differential ideals but are only valid for initial conditions which do not cancel the denominators of the terms (the inequations of the system). The general problem: given a system of polynomial PDEs and a set of initial conditions, " does there exist a formal power series solution of the system for these conditions ? " is undecidable.

```
> unevaluated_formal_power_series (u(x,y), [x,y], 4, ideal [1]);
```

$$u(0,0) + u_y(0,0)\,y + 1/2\,RootOf(\_Z^2 - b)\,y^2$$

```
> unevaluated_formal_power_series (u(x,y), [x,y], 4, ideal [2]);
```

$$u(0,0) + u_x(0,0)\,x + u_y(0,0)\,y - 1/2\,\frac{u_x(0,0)\,x^2}{a\,v(0,0)} + 1/2\,RootOf(\_Z^2 - b)\,y^2$$

$$+1/6\,\frac{(u_x(0,0)\,a\,v_x(0,0) - u_x(0,0))\ x^3}{a^2\,(v(0,0))^2}$$

```
> unevaluated_formal_power_series (v(x,y), [x,y], 4, ideal [2]);
```

$$v(0,0) + v_x(0,0)\,x + 1/2\,v_{xx}(0,0)\,x^2 + 1/6\,v_{xxx}(0,0)\,x^3$$

Example 2

Consider the problem of computing the compatibility conditions of the system $u_{zz} - y u_{xx} = 0$, $u_{yy} = 0$. This example is quite famous (it has been treated by Janet, Pommaret and Mansfield) but easy, since it is linear. To compute the compatibility conditions, we just call v the first equation, w the second one and we eliminate u.

```
> read differential_algebra;
> R := differential_ring (derivations = [x,y,z], ranking = [u,[v,w]]);
                                  R := your_ring

> which_ranking_between (u, v, R);
             elimination: u [theta] > v [phi] for all theta, phi

> which_ranking_between (v, w, R);
tord: _U [theta] > _V [phi] when
    |theta| > |phi| or
    |theta| = |phi| and theta > phi for the lexicog. order: x > y > z or
    theta = phi and _U > _V according to: v > w

> p1 := v[] - u[z,z] + y * u[x,x];
> p2 := w[] - u[y,y];

> ideal := Rosenfeld_Groebner ([p1,p2], R);
                                ideal := [regular]

> map (equations, ideal);
```

$$[[-y^4 w_{xxxx} + 2 y^3 w_{xxzz} - y^3 v_{xxyy} + y^2 v_{yyzz} + 2 y^2 v_{xxy} - y^2 w_{zzzz} - 2 y v_{xx} + 2 u_{zzzz} - 2 v_{zz}$$
$$y^3 w_{xx} + y^2 v_{yy} - y^2 w_{zz} - 2 y v_y + 2 y u_{yzz} + 2 v - 2 u_{zz}$$
$$v - u_{zz} + y u_{xx}$$
$$-w + u_{yy}$$
$$w_{xxxxxx} y^3 + y^2 v_{xxxxyy} - 3 y^2 w_{xxxxzz} - 2 y v_{xxxxy} + 3 y w_{xxzzzz} - 2 y v_{xxyyzz}$$
$$+2 v_{xxyzz} - w_{zzzzzz} + 2 v_{xxxx} + v_{yyzzzz}$$
$$w_{xxy} y + 3 w_{xx} + v_{yyy} - w_{yzz}]]$$

Looking at the equations, we see that the two last equations are free of u. These equations give the compatibility conditions of the initial system.

7 Special Functions

Maple improved in many ways in its handling of special functions during the last fiew years with the two most significant of these being the extension to the complex plane of the arbitrary precision evaluation of the Bessel functions (and the introduction into Maple of

many Bessel-related functions), and the improvements to series codes to properly handle branch cut information.

In addition to these, work was done (and continues to be done) on extending or improving the evaluation of other special functions over the complexes; improving the random testing of floating point code; and researching the mathematics of the Lambert W function.

7.1 Bessel functions

Through Maple V Release 3, Bessel functions could only be evaluated for real orders and arguments, and the implementation was poor with respect to both efficiency and accuracy. With Release 4, Maple is able to evaluate these functions for complex orders and arguments. This new implementation is accurate, and generally more efficient for the real cases. There is still room for improvement for certain (hard) cases, so work is continuing on these functions.

In addition to extending the four basic Bessel functions (J, Y, K and I) to the complexes, several families of Bessel related functions were introduced into Maple, complete with symbolic manipulation capabilities and arbitrary precision evaluation over the complexes. These were:

- Hankel functions: HankelH1 and HankelH2; these functions are also known as the Bessel functions of the third kind
- Kelvin functions: KelvinBer, KelvinBei, KelvinHer, KelvinHei, KelvinKer, KelvinKei; these functions are sometimes known as Thompson functions
- Struve functions: StruveH, StruveL
- Anger and Weber functions: AngerJ, WeberE

Furthermore, the Airy functions were overhauled (and renamed to AiryAi and AiryBi), and their first derivatives included (available as AiryAi(1,x) and AiryBi(1,x)). Higher derivatives can be evaluated in terms of the 0'th and 1'st derivatives.

Maple is able to recognize and solve the differential equations satisfied by each of these (families of) functions, and is able to convert between them, compute series expansions, and manipulate them using recurrence relation identities.

7.2 Improvements to series

It can often happen that a limit computation requires knowledge of local behaviour of a function near a branch cut (or branch point) of the function. Such local information is supposed to be obtainable from series(), but until recently, series did not account for branch cuts. For example, we have this evolution:

Maple V Release 1:

```
> series(ln(x),x=-1,3);
                                                2             3
```

```
ln(-1) - (x + 1) - 1/2 (x + 1)  + O((x + 1) )
```

Maple V Release 2 and Release 3:

```
> series(ln(x),x=-1,3);
Error, (in series/ln) series of ln(x) over branch cut does not exist
```

Maple V Release 4:

```
> series(ln(x),x=-1,3);
                                                    2             3
          - I csgn(I x) Pi - (x + 1) - 1/2 (x + 1)  + O((x + 1) )
```

The Release 1 result is wrong, as it suggests that $\ln(x)$ is analytic in a neighbourhood of -1, which is not true for the principal branch function (which is what $\ln(x)$ represents in Maple). The Release 2 & 3 approach was a stopgap measure to avoid returning completely incorrect results. The Release 4 result is the correct one.

In addition to ln, the branch cut information has been encoded into the series expansions for arctan, arctanh, arccot, arccoth, and polylog.

Conspicuously absent from this list are the functions arcsin, arccos, arccsc, arcsec and their hyperbolic relatives. The reason for this is that the correct series expansions on the branch cuts for these functions involve terms of the form $\sqrt{1-x}^n$, where n is a non-negative integer. When n is odd, the square root term remains. However, series() insists on rewriting such terms in the form $i^n\sqrt{x-1}^n$, which is a mathematically invalid transformation. The inability of series() to return results involving powers of $(ax+b)$, where $a \neq 1$, is a very serious weakness in the design of series() and the series data structure. Without this ability, correct series expansions for many functions with branch cuts will not be possible in Maple.

Another serious deficiency in the design of series() is highlighted by attempts to code the series for the Bessel functions. For example, the series for $J_\nu(z)$ can be written in the form

$$(\frac{z}{2})^\nu \sum_{k=0}^{\infty} \frac{(-z^2/4)^k}{k!\,\Gamma(\nu+k+1)}$$

but unless ν is an explicitly given rational number, series() cannot handle the $(z/2)^\nu$ term. This shortcoming severely limits the usability of series with respect to functions which are "analytic up to a factor".

7.3 Other special functions

Peter Borwein provided a new algorithm for the evaluation of the Riemann ζ function (Maple's Zeta(x)) which has much potential, particularly for high precision. While some work was done with respect to including this algorithm in Maple's evalf/Zeta, this study is not yet complete, so the algorithm is not part of the Release 4 version of evalf/Zeta.

Dawson's integral, known in Maple by the name dawson(x), has been extended to complex arguments. This extension was relatively straightforward, using the identity relating Dawson's integral to the error function.

The two-argument arctan function, which is most commonly used to compute the argument (also known as the *phase*) of a complex number, also has a natural extension to complex input arguments, via the formula

$$\arctan(y,x) = -i\ln\left(\frac{x+iy}{\sqrt{x^2+y^2}}\right)$$

This extension has been implemented.

The hypergeometric series is usually expressed using the Pochhammer symbol, which, for complex z and non-negative integer n is defined to be

$$(z)_n = z(z+1)\ldots(z+n-1)\,.$$

This can be more compactly written in the form $\Gamma(z+n)/\Gamma(z)$, and this form immediately provides the extension of the Pochhammer symbol to non-integer (and negative integer) n. Maple now implements this symbol, for complex z and n, with the name pochhammer(z,n).

Study of the mathematical properties of the Lambert W function has continued, resulting in a paper which has appeared in Comptes Rendu, entitled "Sur l'inversion de $y^\alpha e^y$ au moyen de nombres de Stirling associés", by D.J. Jeffrey, R.M. Corless, D.E.G. Hare and D.E. Knuth, and a paper which is to appear in The Mathematical Scientist, entitled "Unwinding the branches of the Lambert W function", by D.J. Jeffrey, R.M. Corless and D.E.G. Hare.

7.4 Accounting for architecture differences

Maple now runs on both 32-bit and 64-bit architectures. The Maple software floating point number systems on these different architectures are themselves substantially different, a fact which Maple itself can recognize and utilize. To accomplish this, a routine called Maple_floats was written, which is essentially a query engine operating somewhat along the lines of the evalhf interface to the hardware floats parameters. For example, evalhf(Digits) returns the number of digits (base 10) available from the underlying hardware double precision number system. Maple_floats(MAX_DIGITS) returns the maximum number of digits (base 10) available from Maple itself.

There is a set of keywords whose values are substituted for in expressions passed to Maple_floats(). These keywords are consistently named (unlike the evalhf keywords, which were inherited from the IEEE standard), using all capital letters and underscores between words.

This routine has option remember, so it is quite efficient to use. Several evalf routines, such as exp, GAMMA, etc, now use Maple_floats to enable them to tailor their behaviour to the particular architecture Maple is running on.

7.5 Testing

In 1993, a system of regularly testing floating point code at randomly generated arguments and precisions was implemented. This system has been extremely successful in locating problem areas and ensuring the robustness of the code.

The system underwent several revisions in 1994 and 1995, and has been extensively enhanced. In the first place, tests of routines taking different numbers and types of arguments have been unified, so that it is even simpler now to add new functions to the test runs.

Secondly, tests of special cases, such as integer arguments, 0 arguments, purely imaginary arguments, etc, were included.

Thirdly, a new test designed to verify that a Maple evalf/ routine has the same continuity properties as the function it is supposed to be evaluating was developed. This test checks to see that small changes in input values which cause significant differences in execution path through the code yield only small changes in output values. Typical situations are code which handles exact integer arguments differently from general arguments and real arguments versus arguments with small imaginary parts. This test also validates continuity onto branch cuts to ensure that changes to the code don't change the closure properties with respect to the branch definitions.

Finally, a test designed to verify that Maple evalf/ routines satisfy the same mathematical identities as the functions they are supposed to evaluate is being developed. This test is still in its infancy, and there are some difficulties which will need to be understood and overcome.

8 Piecewise Functions

The piecewise function facilities of Maple have been extended. Based on Technical Report CS-96-14, "A Normal Form for Function Rings Of Piecewise Functions" by Martin v. Mohrenschildt, the algorithm to compute composition of piecewise functions has been improved.

Example 1

$$f(x) = \begin{cases} x^2 & x \geq -1 \text{ and } x \leq 1 \\ x & \text{otherwise} \end{cases}$$

Simplified:

$$f(x) = \begin{cases} x & x < -1 \\ x^2 & x \leq 1 \\ x & 1 < x \end{cases}$$

Define a second function:

$$g(x) = \begin{cases} x+1 & 0 \le x \\ 3\,x & \text{otherwise} \end{cases}$$

Compositions:

$$\text{simplify}(g(f(x))) = \begin{cases} 3\,x & x < -1 \\ x^2+1 & x \le 1 \\ x+1 & 1 < x \end{cases}$$

$$\text{simplify}(f(g(x)) \begin{cases} 3\,x & x < -1/3 \\ 9\,x^2 & x < 0 \\ x+1 & 0 \le x \end{cases}$$

Example 2

$$f(x) = \frac{cx}{|x|+1}$$

We can convert this to piecewise notation:

$$f(x) = \begin{cases} \frac{cx}{-x+1} & x \le 0 \\ \frac{cx}{x+1} & 0 < x \end{cases}$$

$$g = piecewise(x < -5, 3(x-5), x >= -5 \text{ and } x <= 2, 0, 2 < x, x-2)$$

$$g(x) = \begin{cases} 3\,x - 15 & x < -5 \\ 0 & x \le 2 \\ x-2 & 2 < x \end{cases}$$

For $c = 1$

$$\text{simplify}(g(f(x))) = 0$$

For $c = 3$

$$\text{simplify}(g(f(x))) = \begin{cases} 0 & x \le 2 \\ \frac{x-2}{x+1} & 2 < x \end{cases}$$

For $c = 6$

$$\text{simplify}(g(f(x))) = \begin{cases} -3\,\frac{11\,x-5}{x-1} & x < -5 \\ 0 & x \le 1/2 \\ 2\,\frac{2\,x-1}{x+1} & 1/2 < x \end{cases}$$

Example 3

Some examples using the composition operator @.

```
> m := x->piecewise(x <=0,-x,x>0,x);

              m := x -> piecewise(x <= 0, -x, 0 < x, x)

> convert( abs(2-abs(x-1)), piecewise , x);

                        { -1 - x        x <= -1
                        {
                        { 1 + x         x <= 1
                        {
                        { 3 - x          x < 3
                        {
                        { x - 3         3 <= x

> convert( m(2-m(x-1)), piecewise , x);

                        { -1 - x        x <= -1
                        {
                        { 1 + x         x <= 1
                        {
                        { 3 - x          x < 3
                        {
                        { x - 3         3 <= x

> convert( (m@m)(x), piecewise , x);   # this is abs(abs(x))

                           { -x        x <= 0
                           {
                           { x         0 < x
```

9 Pattern Matching and Functional Programming

Maple did not previously have a general pattern matching facility. Implementing a pattern matcher gave us the possibility to enhance Maple's concepts of functional programming and applying rules. In this report by Martin v. Mohrenschildt, we describe the different functions implemented to support the above ideas in Maple.

The functions which have been implemented are: *patmatch, compiletable, tablelook, define, definemore, applyrule.*

9.1 Pattern Matching

The function **patmatch** implements real pattern matching into Maple. This functionality is entirely created in the Maple library.

Syntax: patmatch(expr, pattern) or patmatch(expr, pattern, 's')

The patmatch function returns true if it is able to match expr to pattern, and false otherwise. If the patmatch is successful, s is assigned a substitution set such that subs(s, pattern) = expr.

A pattern is an expression containing variables typed by ::. For example, a::radnum means that a is matched to an expression of type radnum. Note that in a sum, e.g. a::realcons+x, a can be 0, and in a product, e.g. a::realcons*x, a can be 1 (not 0). This behavior can be overridden using the special keyword nonunit around the type. For example, a::nonunit(realcons)*x does not match x.

Some examples:

```
> patmatch(x,a::realcons*x+b::realcons,'la');la;
                                    true

                                [a = 1, b = 0]

> patmatch(sqrt(3)*x-ln(4)*Pi/5-exp(1),a::realcons*x+b::realcons,'la');la;
                                    true

                        1/2
                [a = 3    , b = - 1/5 ln(4) Pi - exp(1)]

> patmatch(exp(1/2*Pi),exp(n::radnum*Pi),'la');la;
                                    true

                                [n = 1/2]
```

The pattern matcher knows about these special default values:

$exp^{a::type}$	matches also exp if $a = 1$ if 1 is of type $type$
$a :: type + exp$	matches also exp if $a = 0$ if 0 is of type $type$
$a :: type * exp$	matches also exp if $a = 1$ if 1 is of type $type$

We have the special constructor **nonunit(..)** to make sure that the matching of the above examples does not happen.

a::nonunit(algebraic)+b::nonunit(algebraic)	matches a sum of two or more terms
a::nonunit(algebraic)+b::algebraic	matches a single term or the sum of terms

```
> patmatch(a+b+c,A::nonunit(algebraic)+B::nonunit(algebraic),'la');la;
                                    true

                                [A = a, B = b + c]
```

```
> patmatch(a,A::nonunit(algebraic)+B::nonunit(algebraic),'la');
                                    false

> patmatch(a+exp(x)-Pi,A::nonunit(algebraic)+B::nonunit(algebraic),'la');la;
                                    true

                            [A = a, B = exp(x) - Pi]

> patmatch(a+exp(x)-Pi,A::nonunit(algebraic)+B::algebraic,'la');la;
                                    true

                            [A = a, B = exp(x) - Pi]

> patmatch(a,A::nonunit(algebraic)+B::algebraic,'la');la;
                                    true

                                [A = a, B = 0]
```

We have the special keyword **conditional** to express patterns having additional conditions. This is used for programming patterns in tables with additional conditions on the pattern. The syntax is *conditional(pattern,condition)* and *conditional(pattern=right-hand side, condition)* for rules in tables or define. For example, it can be used for patterns of this kind:

```
int(a::algebraic,x::name)=a*x,_type(a,freeof(x)).
```

This is not the same as `int(a::freeof(x),x::name)` since at the point the pattern matcher matches a, x is not yet known. Note that the condition has to be unevaluated or in inert form: use an _ in front of every name. For example, `_type(a,.freeof(x))`.

```
> patmatch(2*x+5,conditional(a::integer*x+b::integer,a^2<b),'la');la;
                                    true

                                [a = 2, b = 5]

> patmatch(2*x+2,conditional(a::integer*x+b::integer,a^2<b),'la');
                                    false

> patmatch(11*x+6,conditional(a::integer*x+b::integer,a>b and _type(a,prime)
> and not a<0),'la');
                                    true
```

9.2 Functional Programming

The idea of functional programming is implemented with the function **define** in Maple. With **define** it is possible to give evaluation and simplification rules for functions and operators. **define** creates a procedure applying the properties to the function. Note that properties are used in the order they are given, hence if a pattern is more general than another one it has to be first, e.g. $f(x) = x$ before $f(a :: algebraic) = g^a$. With **definemore** it is possible to add additional properties to a function or operator which was defined by **define**.

The **define** function understands the following keywords:

linear	a function linear in the first argument
multilinear	a function linear in all arguments
orderless	a function with no order in the arguments
flat	a function not being nested
diff(function,x)=exp	define the derivative to be exp

```
Examples:
> define(f,linear,f(1)=tt);
> f(2*x+4);
                                  2 f(x) + 4 tt

> define(g,g((a::algebraic)^n::nonunit(integer))=n*g(a),g(a::realcons)=a);
> g(x^2);
                                     2 g(x)

> g((x-2)^(-1));
                                   -g(x - 2)

> g(Pi*sqrt(2));
                                       1/2
                                    Pi 2
```

To define commutative and associative operations use the keywords orderless and flat. Orderless means that there is no order in the arguments of a function, and flat means that for example f(a,f(b,c)) is f(a,b,c).

```
> define(h,orderless,flat);
> h(a,b)-h(b,a);
                                        0

> h(a,h(b,c))-h(h(a,b),c);
                                        0
```

Functional programming:

```
> define(fac,fac(0)=1,fac(n::posint)=n*fac(n-1));
> fac(5);
                                    120

> define(fib,fib(0)=1,fib(1)=1,fib(n::posint)=fib(n-1)+fib(n-2));
> fib(7);
                                     21


# An example of programming a one-line GCD:
> define(GCD,GCD(a::integer,0)=a,GCD(a::integer,b::integer)=GCD(b,a mod b));
> GCD(6,3);
                                     3

> GCD(120,12*120);
                                    120

> GCD(13,11);
                                     1
```

An example of an integrator created with define:

```
>  define(INT,linear,
>      conditional(INT(a::algebraic,X::name)=a*X,_type(a,freeof(X))),
>      INT(X::name,X::name)=1/2*X^2);
```

With this definition it works for any variables:

```
> INT(2*x+4,x);
                                      2
                                     x  + 4 x

> INT(z+x,z);
                                        2
                                   1/2 z  + x z

> definemore(INT,
>       conditional(INT( 1/(a::algebraic*X::name+b::algebraic),X::name)
>       =ln(a*X+b)/a,_type(a,freeof(X) &and _type(b,freeof(X)))));
> INT(1/x,x);
                                    ln(x)

> INT(1/(2*x+3),x);
```

```
                                1/2 ln(2 x + 3)

> definemore(INT,INT(\
> (x::name)^n::nonunit(integer),x::name)=1/(n+1)*x^(n+1));
> INT(2*x^2,x);
                                          3
                                       2/3 x
> INT(1/x,x);
                                       ln(x)

> definemore(INT,
> conditional(INT(sin(a::algebraic*x+b::algebraic),x::name)
      =-1/a*cos(a*x+b),_type(a,freeof(x)\
> &and _type(b,freeof(x)))));
```

Still if INT does not know the pattern it returns unevaluated:

```
> INT(sin(x)+cos(x),x);
                           -cos(x) + INT(cos(x), x)
```

And a example of what Maple can compute knowing that a function is linear and it's derivative is a:

```
> define(f,linear,f(1)=1,diff(f(x),x)=a);
> diff(f(x),x);
                                       a

> f(0);
                                       0

> int(f(x),x);
                                            2
                                 x f(x) - 1/2 x  a

> int(exp(f(x)),x);
                                    exp(f(x))
                                    ---------
                                        a

> int(f(sin(x)),x);
                      x f(sin(x)) - (cos(x) + x sin(x)) a
```

```
> limit(f(x),x=0);
                                    0

> series(f(x),x=1);
                                                        6
                          1 + a (x - 1) + O((x - 1) )
```

Derivatives of functions can be defined using the keyword *diff*:

```
> define(P,diff(P(x),x)=1/P(x));
> diff(P(x),x);
                                     1
                                    ----
                                    P(x)

> diff(P(exp(z)),z);
                                   exp(z)
                                  ---------
                                  P(exp(z))

> diff(1/P(x^2),x);
                                      x
                                  -2 ------
                                        2 3
                                     P(x )
```

9.3 Applying Rules

With **applyrule** a rule or a list of rules can be applied to a given expression. **applyrule** computes the fixed point, i.e. it applies the rules until no rule can be applied any more. It is more powerful than the command **algsubs**, but it does not do mathematical transformations as **algsubs** does.

Syntax: **applyrule(rule, expr)** or **applyrule([rule1, rule2, ...], expr)**.

```
> applyrule(a+b=x,f(a+b+c));
                                   f(x + c)

> applyrule(x=y,x^2);
                                     2
                                    y

> applyrule(x^2=y,f(x^2,exp(sin(x)+2*x^2)));
                           f(y, exp(sin(x) + 2 y))
```

```
> applyrule(f(a::integer*x)=a*f(x),f(2*x)+g(x)-p*f(x));
                              2 f(x) + g(x) - p f(x)

> applyrule([a::even=even,a::prime=prime],[1,2,4,3,5,6,4,8,15,21]);
          [1, even, even, prime, prime, even, even, even, 15, 21]

> applyrule(sin(2*x)=2*sin(x)*cos(x),sin(x)+sin(2*x)-cos(x));
                       sin(x) + 2 sin(x) cos(x) - cos(x)
```

10 Matrix Computations

In 1994 we developed two new probabilistic algorithms [23, 24] for the computation of Hermite and Smith forms of polynomial matrices. Since then we have created new deterministic algorithms for computing both Hermite and Smith normal forms of matrices. In the case of Hermite normal forms, the paper [25] provides an asymptotically faster way of computing the form over the integers. In addition, in the important class of rectangular matrices we have also determined a sparse structure for computing the multiplier matrix. This leads to important improvements for computing such applications as one-sided matrix greatest common divisors of integer matrices along with the solution of the accompanying matrix diophantine equations. Also, in the case of Smith normal forms, the paper [26] provides a new algorithm for deterministically computing the form over the integers.

In both cases, the algorithms presented are faster than any previously known results. In particular, in the case of [26] the algorithm computes the invariant factors of an integer matrix in approximately the same time as computing the fraction-free decomposition of the matrix, and hence the determinant of the matrix. Thus the algorithm can be thought of as having optimal complexity, since knowing the invariant factors allows for computation of the determinant.

The primary research on this topic has been done by A. Storjohann. Mr. Storjohann completed his Master's thesis under Professor Labahn in late 1994 and has been employed as a research associate with the group until the end of February 1996. In March 1996 Mr. Storjohann began his Ph.D studies at ETH, Zürich.

11 Tensor Computations

M. Kavian, a Ph.D student co-supervised by K.O. Geddes (with R.G. McLenaghan of Applied Math), is investigating new methods for tensor calculations based on applying sophisticated search techniques to a database of rules. One aspect of the new approach is to apply genetic algorithms which have evolved in applications such as DNA classification. A conference paper [14] reports on progress to date.

References

[1] Beckermann, B. and Labahn G. (1994), "A Uniform Approach for the Fast, Reliable Computation of Matrix-type Padé Approximants", *SIAM J. Matrix Anal. Appl.*, **15**, 804–823.

[2] Boulier, F., Lazard, D., Ollivier, F. and Petitot, M. (1995), "Representation for the Radical of a Finitely Generated Differential Ideal", *Proceedings of ISSAC '95*, 158–166. newblock *SIAM J. Matrix Anal. Appl.*, **15**, 804–823.

[3] Boulier, F. (1996), "Some Improvements of a Lemma of Rosenfeld", Technical Report CS–96–13, University of Waterloo, Ontario.

[4] Chan, H.F.J. (1996), "Hybrid Symbolic-Numeric Solution of Differential Equations", M. Math essay, Department of Computer Science, University of Waterloo, Waterloo.

[5] Cabay, S. and Labahn, G. (1994), "Fast, Numerically Stable Inversion of Mosaic Hankel Matrices", *Proceedings of Mathematical Theory of Networks and Systems (MTNS)*, 625–630.

[6] Corless, R. M., Gonnet, G. H., Hare, D. E. G., Jeffrey, D. J. and Knuth, D. E. (1996), "On the Lambert W Function", to appear in *Adv. Comp. Math.*.

[7] Erdelyi, A. (ed.) (1953), *Higher Transcendental Functions, Volume I*, Robert E. Krieger Publishing Company, Malabar, Florida.

[8] Exton, H. (1976), *Multiple Hypergeometric Functions and Applications*, Ellis Horwood Limited, Chichester, England.

[9] Geddes, K. O. (1995), "Hybrid Symbolic-Numeric Algorithms for Scientific and Engineering Problem Solving", *CD-ROM Proceedings of SEAM'95 (Scientific and Engineering Applications of the Macintosh)*, MacSciTech, Worcester, Massachusetts.

[10] Geddes, K. O. and Labahn, G. (1995), "Symbolic and Numeric Integration in Maple", *Innovative Use of Technology for Teaching and Research in Mathematics (Proceedings of The First Asian Technology Conference in Mathematics, December 1995)*, 377–386, The Association of Mathematics Educators, Singapore.

[11] Heidrich, W. (1996), *Spline Extensions for the MAPLE Plot System*, M. Math thesis, Department of Computer Science, University of Jeffrey, Waterloo, Waterloo.

[12] Jeffrey, D. J., Corless, R. M., Hare, D. E. G. and Knuth, D. E. (1995), "Sur l'Inversion de $y^{\alpha}e^{y}$ au moyen de Nombres de Stirling Associés", *C.R. Acad. Sc. Paris, Série I*, **320**, 1449–1452.

[13] Jeffrey, D. J., Hare, D. E. G. and Corless, R. M. (1996), "Unwinding the Branches of the Lambert W function", to appear in *The Mathematical Scientist*.

[14] Kavian, M., McLenaghan, R. G., and Geddes, K. O. (1996), "MapleTensor: Progress Report on a New System for Performing Indicial and Component Tensor Calculations Using Symbolic Computation", to appear in *Proceedings of ISSAC '96*, ACM, New York.

[15] Labahn G. (1995), "Solving Linear Differential Equations in Maple", *Maple Technical Newsletter*, **2(1)**, 20–28, Birkhäuser, Boston.

[16] Luke, Y. L. (1969), *The Special Functions and Their Approximations, Volume I*, Academic Press, San Diego.

[17] Marichev, O. I. (1983), *Handbook of Integral Transforms of Higher Transcendental Functions: Theory and Algorithmic Tables*, Ellis Horwood Limited, Chichester, England.

[18] Mohrenschildt, M.v. (1996), "A Normal Form for Function Rings of Piecewise Functions", Technical Report CS–96–14, University of Waterloo, Ontario.

[19] Petkovšek, M. and Salvy, B. (1993), "Finding All Hypergeometric Solutions of Linear Differential Equations", *Proceedings of ISSAC '93*, ACM, New York.

[20] Prudnikov, A. P., Brychkov, Yu. A., Marichev O. I. (1990), *Integrals and Series, Volume 3: More Special Functions*, Gordon and Breach Science Publishers.

[21] Qiao, L. (1995), *Performance Visualization Tools for Parallelizing Computer Algebra Algorithms*, M. Math thesis, Department of Computer Science, University of Waterloo, Waterloo.

[22] Roach, K. (1996), "Hypergeometric Function Representations", to appear in *Proceedings of ISSAC '96*, ACM, New York.

[23] Storjohann, A. and Labahn, G. (1995), "Preconditioning of Rectangular Polynomial Matrices for Efficient Hermite Normal Form computation", *Proceedings of ISSAC '95* 119–125.

[24] Storjohann, A. and Labahn, G. (1995), "A Fast Las Vegas Algorithm for Computing the Smith Normal Form of a Polynomial Matrix", accepted by *Linear Algebra and its Applications*.

[25] Storjohann, A. and Labahn, G. (1996), "Asymptotically Fast Computation of the Hermite Normal Form of an Integer Matrix", to appear in *Proceedings of ISSAC '96*, ACM, New York.

[26] Storjohann, A. (1996), "Near Optimal Algorithms for Computing the Smith Normal Form of Integer Matrices", to appear in *Proceedings of ISSAC '96*, ACM, New York.

[27] Van Hoeij, M. (1995), "Factorization of Differential Operators with Rational Function Coefficients", submitted to *Journal of Symbolic Computation*.