Quantifier Elimination for Approximate Factorization of Linear Partial Differential Operators

Elena Kartashova[1] and Scott McCallum[2]

[1] RISC, J.Kepler University, Linz, Austria
lena@risc.uni-linz.ac.at
[2] Macquarie University, Sydney, Australia
scott@ics.mq.edu.au

Abstract. This paper looks at the feasibility of applying the quantifier elimination program QEPCAD-B to obtain quantifier-free conditions for the approximate factorization of a simple hyperbolic linear partial differential operator (LPDO) of order 2 over some given bounded rectangular domain in the plane. A condition for approximate factorization of such an operator to within some given tolerance over some given bounded rectangular domain is first stated as a quantified formula of elementary real algebra. Then QEPCAD-B is applied to try to eliminate the quantifiers from the formula. While QEPCAD-B required too much space and time to finish its task, it was able to find a partial solution to the problem. That is, it was able to find a nontrivial quantifier-free sufficient condition for the original quantified formula.

1 Introduction

Let $\mathbb{Q}$ denote the field of all rational numbers, and let R denote the polynomial ring $\mathbb{Q}[x, y]$ in the variables x and y over $\mathbb{Q}$. A *linear partial differential operator (LPDO)* in x and y over $\mathbb{Q}$ is an element of the noncommutative ring $R[\partial_x, \partial_y]$, where ∂_x and ∂_y denote the usual derivation operators on R. An LPDO is of *order* n if the highest order derivations occurring in it are of the n-th order.

While factorization of linear ordinary differential operators is well studied and has a well developed algorithmic theory, the theory of factorization of LPDOs is much more difficult. One constructive factorization method for LPDOs – Beals-Kartashova (BK) factorization – was introduced in [1]. The method could roughly be described as a straightforward search for first order factors of a given LPDO from the *left*, so to speak. One simply expresses a given LPDO of order n as a symbolic product of a first order factor on the left and an $(n-1)$st order factor on the right. One writes down a system of equations for the symbolic coefficients of the factors. Then one tries to solve these equations. Usually one or more *factorization conditions* are thereby derived, necessary and sufficient for the existence of such factors. In case the factorization conditions are fulfilled, the factors can be obtained, and the method applied recursively to the factor on

M. Kauers et al. (Eds.): MKM/Calculemus 2007, LNCS 4573, pp. 106–115, 2007.

the right. A simple LPDO of order 2, together with its factorization condition, and its factorization when the condition is satisfied, is presented in Section 2.

The idea to use BK-factorization for the *approximate* factorization of an LPDO over some bounded domain is discussed in [2]. It is motivated by the important application area of numerical simulations. The processing time for such numerical simulations could be substantially reduced if instead of computation with one LPDO of order n we could proceed with n LPDOs all of order 1. In numerical simulations the coefficients of the given operator are given within some tolerance. It is thus not necessary to fulfil the factorization conditions exactly, but instead within some given tolerance, and over some bounded domain. This leads to the idea of *approximate factorization conditions* for an LPDO over some bounded domain. Approximate factorization conditions would be expected to be formulated using quantifiers over the real numbers.

The idea of the present paper is to look into the feasibility of obtaining quantifier-free approximate factorization conditions using *quantifier elimination by cylindrical algebraic decomposition (QE by CAD)* [3]. This idea was suggested to us by Bruno Buchberger [4]. In Section 2 of this paper we formulate an approximate factorization condition for a so-called hyperbolic LPDO of order 2 with simple polynomial coefficients. We use the language of Tarski algebra to do this. In Section 3 we provide a brief synopsis of QE by CAD. In Section 4 we report the results of applying a computer program for carrying out QE by CAD to the approximate factorization condition obtained in Section 2 in an attempt to find a quantifier-free version of the condition. We find that, while our program could not solve the problem given using a reasonable amount of time and space, it was able to find a partial solution to the problem. More specifically, it was able to find a nontrivial quantifier-free sufficient condition for the given quantified formula.

2 Factorization Conditions for a Hyperbolic LPDO of Order 2

Let us consider a hyperbolic LPDO of order 2 in canonical form:

$$H_2 = \partial_x^2 - \partial_y^2 + p\partial_x + q\partial_y + r, \tag{1}$$

where the coefficients p, q, r are arbitrary elements of R (that is, arbitrary polynomials in x and y over $\mathbb{Q}$). We say that H_2 is *factorizable* if H_2 can be expressed in the form

$$H_2 = (\partial_x \pm \partial_y + s)(\partial_x \mp \partial_y + t),$$

for some elements s and t of R. With $\omega = \pm 1$, put

$$\mathcal{R}_\omega = (\partial_x - \omega\partial_y)\left(\frac{p - \omega q}{2}\right) + \frac{p^2 - q^2}{4}.$$

It follows from equation system 2 in [1], specialised for H_2, that H_2 is factorizable if and only if

$$[r = \mathcal{R}_{-1}] \vee [r = \mathcal{R}_1]. \tag{2}$$

If the former disjunct is valid, then

$$H_2 = (\partial_x + \partial_y + \frac{p-q}{2})(\partial_x - \partial_y + \frac{p+q}{2}),$$

and if the latter disjunct is valid then

$$H_2 = (\partial_x - \partial_y + \frac{p+q}{2})(\partial_x + \partial_y + \frac{p-q}{2}).$$

The reader is reminded that multiplication of such LPDOs is in general non-commutative. Hence the former factorization need not imply the latter, and vice versa.

Suppose now that the polynomial coefficients of the operator H_2 are of the first degree: say $p(x,y) = p_3x + p_2y + p_1,\ q(x,y) = q_3x + q_2y + q_1,\ r(x,y) = r_3x + r_2y + r_1$. Then – by expanding the $\mathcal{R}_\omega$ in terms of x and y and equating the coefficients of r and the $\mathcal{R}_\omega$ – the factorization condition (2) can be expressed as a disjunction of two systems of equations in the nine variables p_i, q_j and r_k. The solution of this disjunction of equation systems yields all *exactly* factorizable hyperbolic LPDOs of this type.

For the remainder of this paper we will address the problem of trying to determine and simplify a condition for *approximate* factorization of hyperbolic operators of this type. We use the standard formal language of elementary real algebra, that is, Tarski algebra [3], to formulate a condition for approximate factorization of hyperbolic operators of this type as a quantifier elimination (QE) problem. In addition to the coefficients of the given operator H_2, we assume that we are also given:

(1) a constant ε;

(2) constants M and N, which define a bounded rectangular region in the plane: $-M < x < M$, $-N < y < N$.

With all this given, and with $\omega = \pm 1$, let us consider the quantified formula of elementary real algebra $\phi^* = \phi^*(p_i, q_j, r_k)$ which asserts that "for all x and y in the bounded region $-M < x < M$, $-N < y < N$, we have $-\varepsilon < r(x,y) - \mathcal{R}_\omega(x,y) < \varepsilon$." We wish to eliminate the quantifiers from $\phi^*(p_i, q_j, r_k)$. More precisely, we wish to find a formula of elementary real algebra $\phi' = \phi'(p_i, q_j, r_k)$, free of quantifiers, such that if $\phi'(p_i, q_j, r_k)$ is true then $\phi^*(p_i, q_j, r_k)$ is true. That is, we wish to find conditions on the coefficients of the polynomials $p(x,y)$, $q(x,y)$ and $r(x,y)$ which imply that the function $\mathcal{R}_\omega(x,y)$ differs not too much from one these polynomials, namely $r(x,y)$, throughout the bounded region $-M < x < M$, $-N < y < N$.

3 Synopsis of QE by CAD

Let A be a set of integral polynomials in $x_1, x_2 \ldots, x_r$, where $r \geq 1$. An *A-invariant cylindrical algebraic decomposition (CAD)* of $\mathbf{R}^r$, r-dimensional real space, is a decomposition D of $\mathbf{R}^r$ into nonempty connected subsets called cells such that

1. the cells of D are cylindrically arranged with respect to the variables $x_1, x_2, \ldots, x_r$;

2. every cell of D is a semialgebraic set (that is, a set defined by means of boolean combinations of polynomial equations and inequalities); and

3. every polynomial in A is sign-invariant throughout each cell of D.

The CAD algorithm as originally conceived [3,5] has inputs and outputs as follows. Given such a set A of r-variate polynomials and a nonnegative integer f with $f < r$, the algorithm produces as its output a description of an A-invariant CAD D of $\mathbf{R}^r$, in which explicit semialgebraic defining formulas are provided only for the cells of the CAD D_f of $\mathbf{R}^f$ induced (that is, implicitly determined) by D. The description of D comprises lists of indices and sample points for the cells of D. (Every cell is assigned an *index* which indicates its position within the cylindrical structure of D.)

The working of the original CAD algorithm can be summarized as follows. If $r = 1$, an A-invariant CAD of $\mathbf{R}^1$ is constructed directly, using polynomial real root isolation. If $r > 1$, then the algorithm computes a *projection set* P of $(r-1)$-variate polynomials (in $x_1, \ldots, x_{r-1}$) such that any P-invariant CAD D' of $\mathbf{R}^{r-1}$ can be extended to a CAD D of $\mathbf{R}^r$. If $f = r$ we set $f' \leftarrow f - 1$ and otherwise set $f' \leftarrow f$. Then the algorithm calls itself recursively on P and f' to get such a D'. Finally D' is extended to D. In order to produce semialgebraic defining formulas for the cells of D_f the algorithm must be used in a mode called *augmented projection*.

Thus for $r > 1$, if we trace the algorithm, we see that it computes a first projection set P, eliminating x_r, then computes the projection of P, eliminating x_{r-1}, and so on, until the $(r-1)$-st projection set has been obtained, which is a set of polynomials in the variable x_1 only. This is called the *projection phase* of the algorithm. The construction of a CAD of $\mathbf{R}^1$ invariant with respect to the $(r-1)$-st projection set is called the *base phase*. The successive extensions of the CAD of $\mathbf{R}^1$ to a CAD of $\mathbf{R}^2$, the CAD of $\mathbf{R}^2$ to a CAD of $\mathbf{R}^3$, and so on, until an A-invariant cad of $\mathbf{R}^r$ is obtained, constitute the *extension phase* of the algorithm.

Now we consider the *quantifier elimination (QE) problem* for the elementary theory of the reals: given a quantified formula (known as a *QE problem instance*) of elementary real algebra

$$\phi^* = (Q_{f+1}x_{f+1}) \ldots (Q_r x_r)\phi(x_1, \ldots, x_r)$$

where ϕ is a formula involving the variables $x_1, x_2, \ldots, x_r$ which is free of quantifiers, find a formula $\phi'(x_1, \ldots, x_f)$, free of quantifiers, such that ϕ' is equivalent to ϕ^*. The QE problem can be solved by constructing a certain CAD of $\mathbf{R}^r$. The method is described as follows.

1. Extract from ϕ the list A of distinct non-zero r-variate polynomials occurring in ϕ.

2. Construct lists S and I of sample points and cell indices, respectively, for an A-invariant CAD D of $\mathbf{R}^r$, together with a list F of semialgebraic defining formulas for the cells of the CAD D_f of $\mathbf{R}^f$ induced by D.

3. Using S, evaluate the truth value of ϕ^* in each cell of D_f. (By construction of D, the truth value of ϕ^* is constant throughout each cell c of D_f, hence can be determined by evaluating ϕ^* at the sample point of c.)

4. Construct $\phi'(x_1, \ldots, x_f)$ as the disjunction of the semialgebraic defining formulas of those cells of D_f for which the value of ϕ^* has been determined to be true.

The above algorithm solves any given particular instance of the QE problem in principle. However the computing time of the algorithm grows steeply as the number r of variables occurring in the input formula ϕ increases.

Collins and Hong [8] introduced the method of *partial CAD construction* for QE. This method, named with the acronym QEPCAD, is based upon the simple observation that we can often solve a QE problem by means of a partially built CAD. The QEPCAD algorithm was originally implemented by Hong. A recent implementation, denoted by QEPCAD-B, contains improvements by Brown, Collins, McCallum, and others – see [6]. QEPCAD-B has solved a range of reasonably interesting problems for which the original QE algorithm takes too much time. Nevertheless the worst case computing time of QEPCAD-B remains large (that is, it depends doubly-exponentially on r).

4 Application of QEPCAD to BK-Factorization

We consider only the first simple case of approximate factorization described in Section 2. Using the notation of Section 2, we suppose that ε, M and N have been given specific constant values, say $\varepsilon = M = N = 1$, and we put $\omega = -1$. We consider the formula $\phi^*(p_i, q_j, r_k)$ which asserts that

$$(\forall x)(\forall y)[(|x| < 1 \wedge |y| < 1) \Rightarrow |r(x, y) - \mathcal{R}_{-1}(x, y)| < 1]. \tag{3}$$

We wish to find a formula $\phi'(p_i, q_j, r_k)$, free of quantifiers, such that $\phi'(p_i, q_j, r_k)$ implies $\phi^*(p_i, q_j, r_k)$.

Remark 1. It would be of greatest interest to find the most general such $\phi'(p_i, q_j, r_k)$ – that is, to find quantifier-free $\phi'(p_i, q_j, r_k)$ *equivalent* to $\phi^*(p_i, q_j, r_k)$. But as we'll see it seems that the time and space resources needed to do this are prohibitive. We'll also see that it is not as time consuming, yet hopefully still of interest, to find quantifier-free conditions merely *sufficient* for ϕ^* to be true.

As a first step we rewrote the quantified formula 3 so that the variables p_i, q_j, r_k appear explicitly, and the denominator 4 is cleared from the right hand side of the implication. Expanding in terms of x and y formula 3 thus has the form:

$$(\forall x)(\forall y)[(|x| < 1 \wedge |y| < 1) \Rightarrow |ax^2 + bxy + cy^2 + dx + ey + f| < 4], \tag{4}$$

where a, b, c, d, e, f are integral polynomials in the p_i, q_j, r_k. (For example, $a = q_3^2 - p_3^2$, $b = 2q_2q_3 - 2p_2p_3$, and $c = q_2^2 - p_2^2$.) In fact it is computationally advantageous to use the general form of quantified formula 4, in which a, b, c, d, e, f

occur as distinct *indeterminates*, rather than as polynomial expressions in the p_i, q_j, r_k, because then the total number of variables in the formula is reduced from 11 to 8.

We attempted to find a solution to the above QE problem instance by running the program QEPCAD-B with the quantified formula 4 (in its general form) as its input. The variable ordering used was (a, b, c, d, e, f, x, y). The computer used for this and subsequent experiments was a Sun server having a 292 MHz ultraSPARC risc processor. Forty megabytes of memory were made available for list processing. However the program ran out of memory after a few minutes. The program was executing the projection phase of the algorithm when it stopped. The first three projection steps – that is, successive elimination of y, x and f – were almost complete.

Increasing the amount of memory to eighty megabytes did not help – the program still ran out of memory during the fourth projection step (that is, during elimination of e).

Of course a very special, but completely trivial, quantifier-free sufficient condition for our QE problem instance is the formula

$$\phi'(p_i, q_j, r_k) := [\text{all } p_i = 0 \wedge \text{all } q_j = 0 \wedge \text{all } r_k = 0].$$

It could be of some interest to look for partial solutions to (that is, quantifier-free sufficient conditions for) our QE problem instance in which some but not all of the variables p_i, q_j, r_k are equal to zero. For example, suppose that we put $p_2 = q_2 = r_2 = 0$ in (4). We obtain:

$$(\forall x)(\forall y)[(|x| < 1 \wedge |y| < 1) \Rightarrow |a'x^2 + d'x + f'| < 4],$$

where a', d', f' are polynomials in $p_1, p_3, q_1, q_3, r_1, r_3$. (In fact we have $a' = a$ and $d' = d$.) Clearly this formula is equivalent to:

$$(\forall x)[(|x| < 1) \Rightarrow |a'x^2 + d'x + f'| < 4], \tag{5}$$

which we shall denote by $\psi^*(p_i, q_j, r_k)$.

The following theorem shows that a partial solution to the special QE problem instance $\psi^*(p_i, q_j, r_k)$ (that is, a quantifier-free sufficient condition for ψ^*) leads to a partial solution to the QE problem instance ϕ^* (that is, a quantifier-free sufficient condition for ϕ^*).

Theorem 1. *Suppose that $\psi'(p_i, q_j, r_k)$ is a quantifier-free formula, involving only $p_1, p_3, q_1, q_3, r_1, r_3$, which implies $\psi^*(p_i, q_j, r_k)$. Then the quantifier-free formula $\psi'(p_i, q_j, r_k) \wedge p_2 = 0 \wedge q_2 = 0 \wedge r_2 = 0$ implies $\phi^*(p_i, q_j, r_k)$.*

► Let p_i, q_j, r_k be real numbers and let (with slight abuse of notation) a, b, c, d, e, f denote the values of the polynomials a, b, c, d, e, f at the particular p_i, q_j, r_k. Assume $\psi'(p_i, q_j, r_k) \wedge p_2 = 0 \wedge q_2 = 0 \wedge r_2 = 0$. Then $\psi^*(p_i, q_j, r_k) \wedge p_2 = 0 \wedge q_2 = 0 \wedge r_2 = 0$ is true, by hypothesis. Take real numbers x and y, with $|x| < 1$ and $|y| < 1$, and (with slight abuse of notation) let a', d', f' denote the values of the polynomials a', d', f' at the particular p_i, q_j, r_k. Then

$$|a'x^2 + d'x + f'| < 4,$$

by virtue of (5) (since $|x| < 1$). Hence (4) is true (since $a = a'$, $d = d'$, $f = f'$, $b = c = e = 0$). ■

The above discussion suggests that it would be worthwhile to try to find a solution to the simplified, special QE problem instance ψ^* using the program QEPCAD-B. We use the more general form of (5), in which a', d', f' occur as indeterminates, and hence reduce by 3 the number of variables in the formula. For simplicity of the notation hereafter we use the variables a, b, c in place of a', d', f', and thus treat the formula:

$$(\forall x)[(|x| < 1) \Rightarrow |ax^2 + bx + c| < 4]. \tag{6}$$

We ran program QEPCAD-B with (6) as its input. Eighty megabytes of memory were made available for list processing. After 191 seconds the program produced the following quantifier-free formula equivalent to (6):

```
c - b + a + 4 >= 0 /\ c - b + a - 4 <= 0 /\
c + b + a + 4 >= 0 /\ c + b + a - 4 <= 0 /\
[ 4 a c - b^2 + 16 a > 0 \/ 4 a c - b^2 - 16 a > 0 \/
[ b^2 - 16 a = 0 /\ b^2 + 16 a > 0 ] \/
[ b^2 - 16 a < 0 /\ b - 2 a >= 0 ] \/
[ b^2 - 16 a < 0 /\ b + 2 a <= 0 ] \/
[ b^2 - 16 a > 0 /\ b + 2 a >= 0 ] \/
[ b^2 - 16 a > 0 /\ b - 2 a <= 0 ] \/
[ b^2 - 16 a = 0 /\ c - b + a + 4 > 0 /\ c - b + a - 4 < 0 ] ].
```

We could transform this formula into a partial solution $\psi'(p_i, q_j, r_k)$ to ψ^* by setting $a = q_3^2 - p_3^2$, $b = 4r_3 - (p_3 - q_3)(p_1 + q_1) - (p_1 - q_1)(p_3 + q_3)$, and $c = 4r_1 - 2(p_3 + q_3) - (p_1 - q_1)(p_1 + q_1)$.

It is possible to induce the program to produce an arguably even simpler solution formula using less computing time by making three separate runs of QEPCAD-B. The first run uses the command

```
assume [a < 0].
```

After just 1.9 seconds the program produced the following quantifier-free formula equivalent to (6) under the assumption $a < 0$:

```
c - b + a + 4 >= 0 /\ c - b + a - 4 <= 0 /\
c + b + a + 4 >= 0 /\ c + b + a - 4 <= 0 /\
[ b - 2 a <= 0 \/ b + 2 a >= 0 \/ 4 a c - b^2 - 16 a > 0 ].     (7)
```

The above formula is perhaps more elegant and understandable. For it is a slight improvement of (that is, slightly more compact than) a formula seen to be equivalent to it (under assumption $a < 0$) which is quite straightforward to derive by hand from (6) using elementary properties of the parabola $y = ax^2 + bx + c$ on the interval $(-1, +1)$:

```
[ 2 a - b >= 0 /\ a + b + c + 4 >= 0 /\ a - b + c - 4 <= 0] \/
[2 a + b >= 0 /\ a - b + c + 4 >= 0 /\ a + b + c -4 <= 0] \/
[2 a - b < 0 /\ 2 a + b < 0 /\ 4 a c - b^2 - 16 a > 0 /\
 a - b + c + 4 >= 0 /\ a + b + c + 4 >= 0].                    (8)
```

Remark 2. To derive by hand (8) from (6) under the assumption $a < 0$, one has to notice that function $f(x) = ax^2 + bx + c$ has its maximum value for $f'(x) = 2ax+b = 0$, that is, for $x = -b/(2a)$, and consider three cases separately: (1) $-b/(2a) \leq -1$, (2) $-b/(2a) \geq +1$, and (3) $-1 < -b/(2a) < +1$. For each of the above three cases one can then write down necessary and sufficient conditions for (6) to be true. For example, in Case 1, (6) is clearly equivalent to $-4 \leq a + b + c \wedge a - b + c \leq 4$. After treating each of the above cases, we obtain (8) by forming the disjunction of the formulas corresponding to the cases.

To obtain a complete solution to the QE problem instance (6) we need to run QEPCAD two more times, for the cases $a > 0$ and $a = 0$, respectively. For the second run we use the command

```
assume [a > 0].
```

and obtain after 1.9 seconds the following quantifier-free formula equivalent to (6) under the assumption $a > 0$:

```
c - b + a + 4 >= 0 /\ c - b + a - 4 <= 0 /\
c + b + a + 4 >= 0 /\ c + b + a - 4 <= 0 /\
[ b + 2 a <= 0 \/ b - 2 a >= 0 \/ 4 a c - b^2 + 16 a > 0 ].    (9)
```

For the third run we put $a = 0$ in (6) and use the command

```
assume [b /= 0].
```

After 60 milliseconds the program produced the following formula equivalent to (6) with $a = 0$ under assumption $b \neq 0$:

```
c - b + 4 >= 0 /\ c - b - 4 <= 0 /\
c + b + 4 >= 0 /\ c + b - 4 <= 0                               (10)
```

This is immediately seen to be correct! Finally we could obtain a complete solution to (6) by combining (7) for $a < 0$, (9) for $a > 0$, (10) for $b \neq a = 0$ and the formula `c - 4 < 0 /\ c + 4 > 0` (for $a = b = 0$). In fact a simple and elegant way to achieve such a combination is as follows:

```
c - b + a + 4 >= 0 /\ c - b + a - 4 <= 0 /\
c + b + a + 4 >= 0 /\ c + b + a - 4 <= 0 /\
[ b - 2 a <= 0 \/ b + 2 a >= 0 \/
4 a c - b^2 - 16 a > 0 \/ a >= 0] /\
[ b + 2 a <= 0 \/ b - 2 a >= 0 \/
4 a c - b^2 + 16 a > 0 \/ a <= 0].                             (11)
```

5 Discussion

As we remarked in Section 3 the worst case computing time of QEPCAD-B grows steeply as the number of variables in the given QE problem instance increases. Indeed, as is suggested by the results reported in Section 4, a complete solution of the QE problem instance (4) by QEPCAD-B using a reasonable amount of time and space seems to be unlikely for the foreseeable future.

Nevertheless the results of Section 4 also suggest that QEPCAD-B could be of help in searching for certain kinds of sufficient conditions for (3), especially those which involve setting some of the variables to zero.

We briefly mention here another kind of approach which a person could use to derive another kind of sufficient condition for (4) by hand. We simply notice that a sufficient condition for (4) is:

$$|a| < \frac{4}{6} \wedge |b| < \frac{4}{6} \wedge |c| < \frac{4}{6} \wedge |d| < \frac{4}{6} \wedge |e| < \frac{4}{6} \wedge |f| < \frac{4}{6}.$$

The above sufficient condition is unlikely to be obtained in a reasonable amount of time and space using QEPCAD-B applied to (4), even if one issues `assume` commands. The number of variables involved is probably too big. However a version of QEPCAD-B which is planned for the future, which will have the capability to determine adjacency relationships amongst the cells of the partial CAD, could be of some use in analyzing certain topological properties of the truth set in nine-dimensional space of the corresponding quantifier-free formula in p_i, q_j, r_k obtained from the above.

Acknowledgements

This paper has its origin in discussions between the authors at RISC-Linz during the second of S.M. to RISC-Linz in 2005. S.M. would like to thank Professors Franz Winkler and Bruno Buchberger, and all their colleagues and staff at RISC, for their hospitality during his stay. S.M. would also like to acknowledge helpful discussions and communications with Professors Daniel Lazard and Chris Brown. E.K. acknowledges the support of the Austrian Science Foundation (FWF) under projects SFB F013/F1304.

References

1. Beals, R., Kartashova, E.: Constructively factoring linear partial differential operators in two variables. TMPh 145(2), 1510–1523 (2005)
2. Kartashova, E., Rudenko, O.: Invariant Form of BK-factorization and its Applications. In: Calmet, J., Seiler, W.M., Tucker, R.W. (eds.) Proc. GIFT-2006, Universitätsverlag Karlsruhe, pp. 225–241 (2006)
3. Collins, G.E.: Quantifier elimination for real closed fields by cylindrical algebraic decomposition. In: Brakhage, H. (ed.) Automata Theory and Formal Languages. LNCS, vol. 33, pp. 134–183. Springer, Heidelberg (1975)

4. Personal communication with Bruno Buchberger (2005)
5. Arnon, D., Collins, G., McCallum, S.: Cylindrical algebraic decomposition I: the basic algorithm. JSC 13(4), 878–889 (1984)
6. Brown, C.: QEPCAD B: a program for computing with semi-algebraic sets using CADs. ACM SIGSAM Bulletin 37(4), 97–108 (2003)
7. Caviness, B.F., Johnson, J.R. (eds.): Quantifier Elimination and Cylindrical Algebraic Decomposition. Springer, Berlin (1998)
8. Collins, G.E., Hong, H.: Partial cylindrical algebraic decomposition for quantifier elimination. JSC 12(3), 299–328 (1991)