Optimal monomial quadratization for ODE systems*

Andrey Bychkov[1] and Gleb Pogudin[2]

[1] Higher School of Economics, Myasnitskaya str., 101978 Moscow, Russia
abychkov@edu.hse.ru
[2] LIX, CNRS, École Polytechnique, Institute Polytechnique de Paris, Palaiseau, France gleb.pogudin@polytechnique.edu

Abstract. Quadratization is a transform of a system of ODEs with polynomial right-hand side into a system of ODEs with at most quadratic right-hand side via the introduction of new variables. Quadratization problem is, given a system of ODEs with polynomial right-hand side, transform the system to a system with quadratic right-hand side by introducing new variables. Such transformations have been used, for example, as a preprocessing step by model order reduction methods and for transforming chemical reaction networks.

We present an algorithm that, given a system of polynomial ODEs, finds a transformation into a quadratic ODE system by introducing new variables which are monomials in the original variables. The algorithm is guaranteed to produce an optimal transformation of this form (that is, the number of new variables is as small as possible), and it is the first algorithm with such a guarantee we are aware of. Its performance compares favorably with the existing software, and it is capable to tackle problems that were out of reach before.

Keywords: differential equations · branch-and-bound · quadratization.

1 Introduction

The *quadratization* problem considered in this paper is, given a system of ordinary differential equations (ODEs) with polynomial right-hand side, transform it into a system with quadratic right-hand side (see Definition 1). We illustrate the problem on a simple example of a scalar ODE:

$$x' = x^5. \tag{1}$$

* The article was prepared within the framework of the HSE University Basic Research Program. GP was partially supported by NSF grants DMS-1853482, DMS-1760448, DMS-1853650, CCF-1564132, and CCF-1563942 and by the Paris Ile-de-France region. The authors are grateful to Mathieu Hemery, François Fages, and Sylvain Soliman for helpful discussions. The work has started when G. Pogudin worked at the Higher School of Economics, Moscow. The authors would like to thank the referees for their comments, which helped us improve the manuscript.

The right-hand side has degree larger than two but if we introduce a new variable $y := x^4$, then we can write:

$$x' = xy, \quad \text{and} \quad y' = 4x^3x' = 4x^4y = 4y^2. \tag{2}$$

The right-hand sides of (2) are of degree at most two, and every solution of (1) is the x-component of some solution of (2).

A problem of finding such a transformation (*quadratization*) for an ODE system has appeared recently in several contexts:

- One of the recent approaches to *model order reduction* [11] uses quadratization as follows. For the ODE systems with quadratic right-hand side, there are dedicated model order reduction methods which can produce a better reduction than the general ones. Therefore, it can be beneficial to perform a quadratization first and then use the dedicated methods. For further details and examples of applications, we refer to [11,15,16,20].
- Quadratization has been used as a preprocessing step for *solving differential equations numerically* [6,12,14].
- Applied to *chemical reaction networks*, quadratization allows one to transform a given chemical reaction network into a bimolecular one [13].

It is known (e.g. [11, Theorem 3]) that it is always possible to perform quadratization with new variables being monomials in the original variables (like x^4 in the example above). We will call such quadratization *monomial* (see Definition 2). An algorithm for finding some monomial quadratization has been described in [11, Section G.]. In [13], the authors have shown that the problem of finding an optimal (i.e. of the smallest possible dimension) monomial quadratization is NP-hard. They also designed and implemented an algorithm for finding a monomial quadratization which is practical and yields an optimal monomial quadratization in many cases (but not always, see Section 3).

In this paper, we present an algorithm that computes an optimal monomial quadratization for a given system of ODEs. To the best of our knowledge, this is the first practical algorithm with the optimality guarantee. In terms of efficiency, our implementation compares favorably to the existing software [13] (see Table 3). The implementation is publicly available at https://github.com/AndreyBychkov/QBee/. Our algorithm follows the classical Branch-and-Bound approach [17] together with problem-specific search and branching strategies and pruning rules (with one using the extremal graph theory, see Section 5.2).

Note that, according to [2], one may be able to find a quadratization of lower dimension by allowing the new variables to be arbitrary polynomials, not just monomials. We restrict ourselves to the monomial case because it is already challenging (e.g., includes an APX-hard [2]-sumset cover problem, see Remark 6) and monomial transformations are relevant for some application areas [13].

The rest of the paper is organized as follows. In Section 2, we state the problem precisely. In Section 3, we review the prior approaches, most notably [13]. Sections 4 and 5 describe our algorithm. Its performance is demonstrated and compared to [13] in Section 6. Sections 7 and 8 contain remarks on the complexity and conclusions/open problems, respectively.

2 Problem Statement

Definition 1. *Consider a system of ODEs*

$$x_1' = f_1(\bar{x}), \quad \ldots, \quad x_n' = f_n(\bar{x}), \tag{3}$$

where $\bar{x} = (x_1, \ldots, x_n)$ *and* $f_1, \ldots, f_n \in \mathbb{C}[\mathbf{x}]$. *Then a list of new variables*

$$y_1 = g_1(\bar{x}), \ldots, y_m = g_m(\bar{x}), \tag{4}$$

is said to be a quadratization *of* (3) *if there exist polynomials* $h_1, \ldots, h_{m+n} \in \mathbb{C}[\bar{x}, \bar{y}]$ *of degree at most two such that*

- $x_i' = h_i(\bar{x}, \bar{y})$ *for every* $1 \leqslant i \leqslant n$;
- $y_j' = h_{j+n}(\bar{x}, \bar{y})$ *for every* $1 \leqslant j \leqslant m$.

The number m *will be called* the order of quadratization. *A quadratization of the smallest possible order will be called* an optimal quadratization.

Definition 2. *If all the polynomials* $g_1, \ldots, g_m$ *are monomials, the quadratization is called* a monomial quadratization. *If a monomial quadratization of a system has the smallest possible order among all the monomial quadratizations of the system, it is called* an optimal monomial quadratization.

Now we are ready to precisely state the main problem we tackle:

Input A system of ODEs of the form (3).
Output An optimal monomial quadratization of the system.

Example 1. Consider a single scalar ODE $x' = x^5$ from (1), that is $f_1(x) = x^5$. As has been show in (2), $y = x^4$ is a quadratization of the ODE with $g(x) = x^4$, $h_1(x, y) = xy$, and $h_2(x, y) = 4y^2$. Moreover, this is a monomial quadratization.

Since the original ODE is not quadratic, the quadratization is optimal, so it is also an optimal monomial quadratization.

Example 2. The Rabinovich-Fabrikant system [19, Eq. (2)] is defined as follows:

$$x' = y(z - 1 + x^2) + ax, \quad y' = x(3z + 1 - x^2) + ay, \quad z' = -2z(b + xy).$$

Our algorithm finds an optimal monomial quadratization of order three: $z_1 = x^2, z_2 = xy, z_3 = y^2$. The resulting quadratic system is:

$$\begin{aligned} x' &= y(z_1 + z - 1) + ax, & z_1' &= 2z_1(a + z_2) + 2z_2(z - 1), \\ y' &= x(3z + 1 - z_1) + ay, & z_2' &= 2az_2 + z_1(3z + 1 - z_1 + z_3) + z_3(z - 1) \\ z' &= -2z(b + z_2), & z_3' &= 2az_3 + 2z_2(3z + 1 - z_1). \end{aligned}$$

3 Discussion of prior approaches

To the best of our knowledge, the existing algorithms for quadratization are [11, Algotirhm 2] and [13, Algorithm 2]. The former has not been implemented and is not aimed at producing an optimal quadratization: it simply adds new variables until the system is quadratized, and its termination is based on [11, Theorem 2].

It has been shown [13, Theorem 2] that finding an optimal quadratization is NP-hard. The authors designed and implemented an algorithm for finding a small (but not necessarily optimal) monomial quadratization which proceeds as follows. For an n-dimensional system $\bar{x}' = \bar{f}(\bar{x})$, define, for every $1 \leqslant i \leqslant n$,

$$D_i := \max_{1 \leqslant j \leqslant n} \deg_{x_i} f_j.$$

Then consider the set

$$M := \{x_1^{d_1} \dots x_n^{d_n} \mid 0 \leqslant d_1 \leqslant D_1, \dots, 0 \leqslant d_n \leqslant D_n\}. \tag{5}$$

[4, Proof of Theorem 1] implies that there always exists a monomial quadratization with the new variables from M. The idea behind [13, Algorithm 2] is to search for an optimal quadratization *inside* M. This is done by an elegant encoding into a MAX-SAT problem.

However, it turns out that the set M does not necessarily contain an optimal monomial quadratization. As our algorithm shows, this happens, for example, for some of the benchmark problems from [13] (Hard and Monom series, see Table 3). Below we show a simpler example illustrating this phenomenon.

Example 3. Consider a system

$$x_1' = x_2^4, \quad x_2' = x_1^2. \tag{6}$$

Our algorithm shows that it has a unique optimal monomial quadratization

$$z_1 = x_1 x_2^2, \;\; z_2 = x_2^3, \;\; z_3 = x_1^3 \tag{7}$$

yielding the following quadratic ODE system:

$$\begin{aligned} &x_1' = x_2 z_2, && z_1' = x_2^6 + 2x_1^3 x_2 = z_2^2 + 2x_2 z_3, && z_3' = 3x_1^2 x_2^4 = 3z_1^2, \\ &x_2' = x_1^2, && z_2' = 3x_1^2 x_2^2 = 3x_1 z_1. \end{aligned}$$

The degree of (7) with respect to x_1 is larger than the x_1-degree of the original system (6), so such a quadratization will not be found by the algorithm [13].

It would be interesting to find an analogue of the set M from (5) always containing an optimal monomial quadratization as this would allow using powerful SAT-solvers. For all the examples we have considered, the following set worked

$$\widetilde{M} := \{x_1^{d_1} \dots x_n^{d_n} \mid 0 \leqslant d_1, \dots, d_n \leqslant D\}, \quad \text{where } D := \max_{1 \leqslant i \leqslant n} D_i.$$

4 Outline of the algorithm

Our algorithm follows the general *Branch-and-Bound (B&B)* paradigm [17]. We will describe our algorithm using the standard B&B terminology (see, e.g., [17, Section 2.1]).

Definition 3 (B&B formulation for the quadratization problem).

- The search space *is a set of all monomial quadratizations of the input system* $\bar{x}' = \bar{f}(\bar{x})$.
- The objective function *to be minimized is the number of new variables introduced by a quadratization.*
- *Each* subproblem *is defined by a set of new monomial variables* $z_1(\bar{x}), \dots, z_\ell(\bar{x})$ *and the corresponding subset of the search space is the set of all quadratizations including the variables* $z_1(\bar{x}), \dots, z_\ell(\bar{x})$.

Definition 4 (Properties of a subproblem). *To each subproblem (see Definition 3) defined by new variables* $z_1(\bar{x}), \dots, z_\ell(\bar{x})$*, we assign:*

1. *the set of* generalized variables*, denoted by* V*, consisting of the polynomials* $1, x_1, \dots, x_n, z_1(\bar{x}), \dots, z_\ell(\bar{x})$*;*
2. *the set of* nonsquares*, denoted by* NS*, consisting of all the monomials in the derivatives of the generalized variables which do not belong to* $V^2 := \{v_1 v_2 \mid v_1, v_2 \in V\}$*. In particular, a subproblem is a quadratization iff* $\mathrm{NS} = \varnothing$.

Example 4. We will illustrate the notation introduced in Definition 4 on a system $x' = x^4 + x^3$ and a new variable $z_1(x) = x^3$. We have $z_1' = 3x^2x' = 3x^6 + 3x^5$. Therefore, for this subproblem, we have:

$$V = \{1, x, x^3\}, \quad V^2 = \{1, x, x^2, x^3, x^4, x^6\}, \quad \mathrm{NS} = \{x^5\}.$$

In order to organize a B&B search in the search space defined above, we define several subroutines/strategies answering the following questions:

- *How to set the original bound?* [4, Theorem 1] implies that the set M from (5) gives a quadratization of the original system, so it can be used as the starting incumbent solution.
- *How to explore the search space?* There are two subquestions:
  - *What are the child subproblems of a given subproblem (branching strategy)?* This is described in Section 4.1.
  - *In what order we traverse the tree of the subproblems?* We use DFS (to make new incumbents appear earlier) guided by a heuristic as described in Algorithm 1.
- *How to prune the search tree (pruning strategy)?* We use two algorithms for computing a lower bound for the objective function in a given subtree, they are described and justified in Section 5.

4.1 Branching strategy

Let $\bar{x}' = \bar{f}(\bar{x})$ be the input system. Consider a subproblem defined by new monomial variables $z_1(\bar{x}), \dots, z_\ell(\bar{x})$. The child subproblems will be constructed as follows:

1. among the nonsquares (NS, see Definition 4), choose any monomial $m = x_1^{d_1} \dots x_n^{d_n}$ with the value $\prod_{i=1}^n (d_i + 1)$ the smallest possible;
2. for every decomposition $m = m_1 m_2$ as a product of two monomials, define a new subproblem by adding the elements of $\{m_1, m_2\} \setminus V$ (see Definition 4) as new variables. Since $m \in$ NS, at least one new variable will be added.

The score function $\prod_{i=1}^n (d_i + 1)$ is twice the number of representations $m = m_1 m_2$, so this way we reduce the branching factor of the algorithm.

Lemma 1. *Any optimal subproblem $z_1(\bar{x}), \dots, z_\ell(\bar{x})$ is a solution of at least one of the children subproblems generated by the procedure above.*

Proof. Let $z_1(\bar{x}), \dots, z_n(\bar{x})$ be any solution of the subproblem. Since m must be either of the form $z_i z_j$ or z_j, it will be a solution of the child subproblem corresponding to the decomposition $m = z_i z_j$ or $m = 1 \cdot z_j$, respectively.

Example 5. Figure 1 below show the graph representation of system $x' = x^4 + x^3$ from Example 4. The starting vertex is $\varnothing$. The underlined vertices correspond to optimal quadratizations, so the algorithm will return one of them. On the first step, the algorithm chooses the monomial x^3 which has two decompositions $x^3 = x \cdot x^2$ and $x^3 = 1 \cdot x^3$ yielding the left and the right children of the root, respectively. The subproblem $\{x^3\}$ was described in more details in Example 4.

The score function $\prod_{i=1}^n (d_i + 1)$ for the decompositions $x^3 = x \cdot x^2$ and $x^3 = 1 \cdot x^3$ takes values 6 and 4, respectively. Hence the algorithm will first explore the branch on the right.

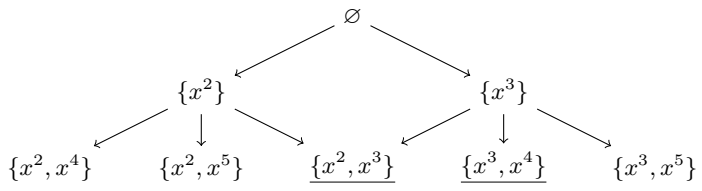

Fig. 1. Graph illustration for equation $x' = x^4 + x^3$

4.2 Recursive step of the algorithm

The recursive step of our algorithm can be described as follows.

Algorithm 1: Branch and Bound recursive step

Input
- polynomial ODE system $\bar{x}' = \bar{f}(\bar{x})$;
- set of new variables $z_1(\bar{x}), \dots, z_\ell(\bar{x})$;
- an optimal quadratization found so far (incumbent) with N new variables.

Output the algorithm replaces the incumbent with a more optimal quadratization containing $z_1(\bar{x}), \dots, z_\ell(\bar{x})$ if such quadratization exists.

(Step 1) if $z_1(\bar{x}), \dots, z_\ell(\bar{x})$ is a quadratization
(a) if $\ell < N$, replace the incumbent with $z_1(\bar{x}), \dots, z_\ell(\bar{x})$;
(b) **return**;

(Step 2) if any of the pruning rules (Algorithm 2 or 3) applied to $z_1(\bar{x}), \dots, z_\ell(\bar{x})$ and N return `True`, **return**;

(Step 3) generate set C of child subproblems as described in Section 4.1

(Step 4) sort C in increasing order w.r.t. $S + n|V|$, where S is the sum of the degrees of the elements in V (V is different for different subproblems as defined in Definition 4);

(Step 5) for each element of C, call Algorithm 1 on it.

5 Pruning rules

In this section, we present two pruning rules yielding a substantial speedup of the algorithm: based on a quadratic upper bound and based on squarefree graphs.

Property 1. Each pruning rule has the following input-output specification:

Input:
- the original ODE system $\bar{x}' = \bar{f}(\bar{x})$;
- already added new variables $z_1(\bar{x}), \dots, z_\ell(\bar{x})$ which are monomials in $\bar{x}$;
- positive integer N.

Output: `True` if it is guaranteed that the set of new variables $z_1(\bar{x}), \dots, z_s(\bar{x})$ cannot be extended to a monomial quadratization of $\bar{x}' = \bar{f}(\bar{x})$ of order less then N. `False` otherwise.

Note that, if `False` is returned, it does not imply that the set of new variables can be extended.

Remark 1. Both pruning rules presented here actually check a stronger condition: whether the set of new variables can be extended by at most $N-s$ variables so that all the monomials NS in the *current* subproblem can be written as a product of two generalized variables. It would be very interesting to strengthen these rules by taking into account the derivatives of the extra new variables.

5.1 Rule based on quadratic upper bound

Remark 2 (Intuition behind the rule). Consider a subproblem with the generalized variables V and set of nonsquares NS (see Definition 4). Assume that it can be quadratized by adding a set W of variables. This would imply that $\mathrm{NS} \subseteq (V \cup W)^2$. This yields a bound

$$|\,\mathrm{NS}\,| \leqslant \frac{(|V|+|W|)(|V|+|W|+1)}{2}. \tag{8}$$

The general ideal of the rule is: since $|V|$ and $|\,\mathrm{NS}\,|$ are known, (8) can be used to find a lower bound for $|W|$. However, a straightforward application of (8) does not lead to noticeable performance improvements. We found that one can do much better by first estimating the number of elements of $\mathrm{NS} \cap (V \cdot W)$ and then applying an argument as in (8) to $\mathrm{NS} \setminus (V \cdot W)$ and W.

Algorithm 2: Pruning rule: based on a quadratic upper bound

(Step 1) Compute the following multiset of monomials in $\bar{x}$

$$D := \{m/v \mid m \in \mathrm{NS}, v \in V, v \mid m\}.$$

(Step 2) Let `mult` be the list of multiplicities of the elements of D sorted in the descending order.

(Step 3) Find the smallest integer k such that

$$|\,\mathrm{NS}\,| \leqslant \sum_{i=1}^{k} \mathrm{mult}[i] + \frac{k(k+1)}{2}. \tag{9}$$

(We use 1-based indexing and set $\mathrm{mult}[i] = 0$ for $i > |\,\mathrm{mult}\,|$)

(Step 4) If $k + \ell \geqslant N$, return `True`. Otherwise, return `False`.

Lemma 2. *Algorithm 2 satisfied the specification described in Property 1.*

Proof. Assume that Algorithm 2 has returned `True`. Consider any quadratization $z_1, \ldots, z_{\ell+r}$ of $\bar{x}' = \bar{f}(\bar{x})$ extending $z_1, \ldots, z_\ell$. We define $\widetilde{V}$, a superset of V, as $\{1, x_1, \ldots, x_n, z_1, \ldots, z_{\ell+r}\}$. By the definition of quadratization, $\mathrm{NS} \subseteq \widetilde{V}^2$. We split NS into two subsets $\mathrm{NS}_0 := \mathrm{NS} \cap (V \cdot \widetilde{V})$ and $\mathrm{NS}_1 := \mathrm{NS} \setminus \mathrm{NS}_0$. For every $1 \leqslant i \leqslant r$, the cardinality of $\mathrm{NS} \cap (z_{\ell+i} \cdot V)$ does not exceed the multiplicity of $z_{\ell+i}$ in the multiset D constructed at **(Step 1)**. Therefore, $|\,\mathrm{NS}_0\,| \leqslant \sum_{i=1}^{r} \mathrm{mult}[i]$. The number of products of the form $z_{\ell+i} z_{\ell+j}$ with $1 \leqslant i \leqslant j \leqslant r$ does not exceed $\frac{r(r+1)}{2}$. Therefore, we have

$$|\,\mathrm{NS}\,| = |\,\mathrm{NS}_0\,| + |\,\mathrm{NS}_1\,| \leqslant \sum_{i=1}^{r} \mathrm{mult}[i] + \frac{r(r+1)}{2},$$

so r satisfies (9). The minimality of k implies $r \geqslant k$. Thus, $r + \ell \geqslant N$, so $z_1, \ldots, z_\ell$ cannot be extended to a quadratization of order less than N.

5.2 Rule based on squarefree graphs

Remark 3 (Intuition behind the rule). We will illustrate the idea behind the rule in a simple example. Assume that we have five monomials $m_1, \dots, m_5$ such that none of them is a square. Assume also that there is a set V of monomial new variables such that $|V| = 4$ and $m_i \in V^2$ for every i. Since none of m_i's is a square, it can be written as $m_i = z_{i,1}z_{i,2}$ for distinct $z_{i,1}, z_{i,2} \in V$. We can therefore think about a graph with vertices being elements of V and edges given by $m_1, \dots, m_5$. One can check that every graph with four vertices and five edges must contain a four-cycle. Let the cycle consist of edges m_1, m_2, m_3, m_4 in this order. Then, for some numbering of elements in V, we have:

$$m_1 = z_1z_2,\ m_2 = z_2z_3,\ m_3 = z_3z_4,\ m_4 = z_4z_1 \implies m_1m_3 = m_2m_4.$$

Thus, by checking that all pairwise product of $m_1, \dots, m_5$ are distinct, we can verify that $m_1, \dots, m_5 \in V^2$ implies that $|V| > 4$.

In order to take into account the monomials which are squares, we consider not just graphs but pseudographs. We also employ the separation strategy $\mathrm{NS} = (\mathrm{NS} \cap (V \cdot W)) \cup (\mathrm{NS} \setminus (V \cdot W))$ as described in Remark 2.

Definition 5. *A pseudograph G (i.e., a graph with loops and multiple edges allowed) is called $C4^*$-*free *if there is no cycle of length four in G with every two adjacent edges being distinct (repetition of edges and/or vertices is allowed).*

Example 6. A $C4^*$-free pseudograph cannot contain:
- *A vertex with two loops.* If the loops are ℓ_1 and ℓ_2 then the cycle $\ell_1, \ell_2, \ell_1, \ell_2$ will violate $C4^*$-freeness.
- *Multiple edges.* If e_1 and e_2 are edges with the same endpoints, then e_1, e_2, e_1, e_2 will violate $C4^*$-freeness.
- *Two vertices with loops connected by an edge.* If the loops are ℓ_1 and ℓ_2 and the edge is e, then ℓ_1, e, ℓ_2, e will violate $C4^*$-freeness.

Definition 6. *By $C(n, m)$ we denote the largest possible number of edges in a $C4^*$-free pseudograph G with n vertices and at most m loops.*

Remark 4. Note that the example above implies that $C(n, n + k) = C(n, n)$ for every positive integer k because a $C4^*$-free pseudograph cannot contain more than n loops.

The number $C(n, 0)$ is the maximal number of edges in a $C4$-free graph and has been extensively studied (e.g. [1,5,7,9]). Values for $n \leqslant 31$ are available as a sequence A006855 in OEIS [18].

In Algorithm 3, we use the exact values for $C(n, m)$ found by an exhaustive search and collected in Table 1 for $n \leqslant 7$. The script for the search is available at https://github.com/AndreyBychkov/QBee/blob/0.5.0/qbee/no_C4_count.py. For $n > 7$, we use the following bound

$$C(n, m) \leqslant C(n, 0) + m \leqslant \frac{n}{2}(1 + \sqrt{4n - 3}) + m,$$

	m							
n	0	1	2	3	4	5	6	7
1	0	1						
2	1	2	2					
3	3	3	4	4				
4	4	5	5	6	6			
5	6	6	7	7	8	8		
6	7	8	9	9	9	10	10	
7	9	10	11	12	12	12	12	12

Table 1. Exact values for $C(n, m)$ (see Definition 6).

where the bound for $C(n, 0)$ is due to [10, Chapter 23, Theorem 1.3.3].

Algorithm 3: Pruning rule: based on squarefree graphs

(Step 1) Compute a subset $E = \{m_1, \dots, m_e\} \subseteq \mathrm{NS}$ such that all the products $m_i m_j$ for $1 \leqslant i \leqslant j \leqslant e$ are distinct.
(done by traversing NS in a descending order w.r.t. the total degree and appending each monomial if it does not violate the property)

(Step 2) Compute the following multiset of monomials in $\mathbf{x}$

$$D := \{m/v \mid m \in E, v \in V, v \mid m\}.$$

(Step 3) Let `mult` be the list of multiplicities of the elements of D sorted in descending order.

(Step 4) Let c be the number of elements in E with all the degrees being even.

(Step 5) Find the smallest integer k such that

$$|E| \leqslant \sum_{i=1}^{k} \mathrm{mult}[i] + C(k, c). \tag{10}$$

(We use 1-based indexing and set $\mathrm{mult}[i] = 0$ for $i > |\,\mathrm{mult}\,|$)

(Step 6) If $k + \ell \geqslant N$, return `True`. Otherwise, return `False`.

Lemma 3. *Algorithm 3 satisfied the specification described in Property 1.*

Proof. Assume that Algorithm 2 has returned `True`. Consider any quadratization $z_1, \dots, z_{\ell+r}$ of $\bar{x}' = \bar{f}(\bar{x})$ extending $z_1, \dots, z_\ell$. We define $\widetilde{V}$, a superset of V, as $\{1, x_1, \dots, x_n, z_1, \dots, z_{\ell+r}\}$. By the definition of quadratization, $E \subseteq \mathrm{NS} \subseteq \widetilde{V}^2$. Similarly to the proof of Lemma 2, we split E into two subsets

$$E_0 := E \cap (V \cdot \widetilde{V}) \quad \text{and} \quad E_1 := E \setminus E_0.$$

For every $1 \leqslant i \leqslant r$, the cardinality of $E \cap (z_{\ell+i} \cdot V)$ does not exceed the multiplicity of $z_{\ell+i}$ in the multiset D from **(Step 2)**. Therefore, $|E_0| \leqslant \sum_{i=1}^{r} \mathrm{mult}[i]$.

Consider a pseudograph G with r vertices numbered from 1 to r corresponding to $z_{\ell+1}, \ldots, z_{\ell+r}$, respectively. For every element $m \in E_1$, we fix a representation $m = z_{\ell+i} z_{\ell+j}$, and add an edge connecting vertices i and j in G (this will be a loop of $i = j$). We claim that pseudograph G will be $C4^*$-free. Indeed, if there is a cycle formed by edges $m_1, m_2, m_3, m_4 \in E_0$, then we will have $m_1 \cdot m_3 = m_2 \cdot m_4$. Moreover, $\{m_1, m_3\} \cap \{m_2, m_4\} = \varnothing$, so such a relation contradicts the condition on E imposed by **(Step 1)**. Finally, a monomial $m \in E$ can correspond to a loop in G only if it is a square, that is, all the degrees in m are even. Hence E_1, the total number of edges in G, does not exceed $C(r, c)$

In total, we have

$$|E| = |E_0| + |E_1| \leqslant \sum_{i=1}^{r} \mathrm{mult}[i] + C(r, c),$$

so r satisfies (10). The minimality of k implies that $r \geqslant k$. Thus, $r + \ell \geqslant N$, so $z_1, \ldots, z_\ell$ cannot be extended to a quadratization of order less than N.

Remark 5 (Cycles of even length). One can modify this rule to use graphs not containing cycles of even length. In this case, the set E from **(Step 1)** of Algorithm 3 would satisfy the condition that there are no multi-subsets of equal cardinality and with equal product. However, this approach did not work that well in practice, in particular, due to the overhead for finding such E.

5.3 Performance of the pruning rules

Table 2 below shows the performance of our algorithm with a different combination of the pruning rules employed. It shows that the rules substantially speed up the computation and that Algorithm 3 is particularly successful in higher dimensions.

ODE system	Dimension	No pruning	Alg. 2	Alg. 3	Alg. 2 & 3
Circular(8)	2	4293 ± 445	497 ± 5	526 ± 8	453 ± 7
Hill(20)	3	3.4 ± 0.1	3.0 ± 0.1	2.4 ± 0.1	2.4 ± 0.1
Hard(2)	3	106.3 ± 1.0	19.6 ± 1.1	20.1 ± 0.6	16.7 ± 0.6
Hard(4)	3	360.1 ± 5.6	107.5 ± 2.4	108.8 ± 2.1	96.6 ± 1.5
Monom(3)	3	552.9 ± 10.9	85.7 ± 4.2	124.7 ± 5.5	84.2 ± 3.3
Cubic Cycle(6)	6	187.3 ± 0.8	43.6 ± 0.6	20.0 ± 0.5	20.1 ± 0.3
Cubic Cycle(7)	7	2002 ± 6.4	360.7 ± 1.1	150.2 ± 1.3	160.9 ± 5.9
Cubic Bicycle(7)	7	1742 ± 89	73.2 ± 0.6	29.8 ± 0.3	30.5 ± 0.2
Cubic Bicycle(8)	8	4440+	175.4 ± 4.0	64.8 ± 0.5	68.9 ± 0.7

Table 2. Comparison of the pruning rules used by our algorithm. Values in the cells represent an average time with the standard deviation in seconds.

6 Performance and Results

We have implemented our algorithm in Python, and the implementation is available at https://github.com/AndreyBychkov/QBee/tree/0.5.0. We compare our algorithm with the one proposed in [13]. For the comparison, we use the set of benchmarks from [13] and add a couple of new ones (described in the Appendix).

The results of the comparison are collected in Table 3. All computation times are given either in milliseconds or in seconds and were obtained on a laptop with the following parameters: Intel(R) Core(TM) i7-8750H CPU @ 2.20GHz, WSL Windows 10 Ubuntu 20.04, CPython 3.8.5. From the table, we see that the only cases when the algorithm from [13] runs faster are when it does not produce an optimal quadratization (while we do). Also, cases, when the algorithm from [13] is not able to terminate, are marked as "—" symbol.

ODE system	Biocham time	Biocham order	Our time	Our order
Circular(3), **ms**	83.2 ± 0.1	3	5.1 ± 0.1	3
Circular(4), **ms**	106.7 ± 2.3	4	164.8 ± 32.3	4
Circular(5), **ms**	596.2 ± 10.9	4	20.0 ± 0.1	4
Circular(6), **s**	37.6 ± 0.4	5	4.2 ± 0.1	5
Circular(8), **s**	—	—	453.3 ± 6.9	6
Hard(3), **s**	1.09 ± 0.01	11	8.6 ± 0.2	9
Hard(4), **s**	20.2 ± 0.3	13	96.9 ± 1.5	10
Hill(5), **ms**	87.8 ± 0.9	2	4.6 ± 0.0	2
Hill(10), **ms**	409.8 ± 5.6	4	49.7 ± 1.3	4
Hill(15), **s**	64.1 ± 0.4	5	0.34 ± 0.1	5
Hill(20),**s**	—	—	2.4 ± 0.1	6
Monom(2), **ms**	96.4 ± 1.6	4	15 ± 0.1	3
Monom(3), **s**	0.44 ± 0	13	84.2 ± 3.3	10
Cubic Cycle(6), **s**	—	—	20.1 ± 0.3	12
Cubic Cycle(7), **s**	—	—	160.9 ± 5.9	14
Cubic Bicycle(7), **s**	—	—	30.5 ± 0.2	14
Cubic Bicycle(8), **s**	—	—	68.9 ± 0.7	16

Table 3. Comparison of our implementation with the algorithm [13] on a set benchmarks

.

7 Remarks on the complexity

It has been conjectured in [13, Conjecture 1] that the size of an optimal monomial quadratization may be exponential in the number of monomials of the input system in the worst case. Interestingly, this is not the case if one allows monomials with negative powers (i.e., Laurent monomials): Proposition 1 shows that

there exists a quadratization with the number of new variables being linear in the number of monomials in the system.

Proposition 1. *Let $\bar{x}' = \bar{f}(\bar{x})$, where $\bar{x} = (x_1, \ldots, x_n)$, be a system of ODEs with polynomial right hand sides. For every $1 \leqslant i \leqslant n$, we denote the monomials in the right-hand side of the i-th equation by $m_{i,1}, \ldots, m_{i,k_i}$. Then the following set of new variables (given by Laurent monomials) is a quadratization of the original system:*

$$z_{i,j} := \frac{m_{i,j}}{x_i} \text{ for every } 1 \leqslant i \leqslant n,\ 1 \leqslant j \leqslant k_i.$$

Proof. Since $m_{i,j} = z_{i,j}x_i$, the original equations can be written as quadratic in the new variables. Let the coefficient in the original system in front of $m_{i,j}$ be denoted by $c_{i,j}$. We consider any $1 \leqslant i \leqslant n$, $1 \leqslant j \leqslant k_j$:

$$z'_{i,j} = \sum_{s=1}^{n} f_s(\mathbf{x}) \frac{\partial z_{i,j}}{\partial x_s} = \sum_{s=1}^{n} \sum_{r=1}^{k_s} c_{s,r} m_{s,r} \frac{\partial z_{i,j}}{\partial x_s}.$$

Since $\frac{\partial z_{i,j}}{\partial x_s}$ is proportional to $\frac{z_{i,j}}{x_s}$, the monomial $m_{s,r}\frac{\partial z_{i,j}}{\partial x_s}$ is proportional to a quadratic monomial $z_{s,r}z_{i,j}$, so we are done.

Remark 6 (Relation to the [2]-sumset cover problem). The [2]-sumset cover problem [3] is, given a finite set $S \subset \mathbb{Z}_{>0}$ of positive integers, find a smallest set $X \subset \mathbb{Z}_{>0}$ such that $S \subset X \cup \{x_i + x_j \mid x_i, x_j \in X\}$. It has been shown in [8, Proposition 1] that the [2]-sumset cover problem is APX-hard, moreover, the set S used in the proof contains 1. We will show how to encode this problem into the optimal monomial quadratization problem thus showing that the latter is also APX-hard (in the number of monomials, but not necessarily in the size of the input). For $S = \{s_1, \ldots, s_n\} \subset \mathbb{Z}_{>0}$ with $s_1 = 1$, we define a system

$$x'_1 = 0, \quad x'_2 = \sum_{i=1}^{n} x_1^{s_i}.$$

Then a set $X = \{1, a_1, \ldots, a_\ell\}$ is a minimal [2]-sumset cover of S iff $x_1^{a_1}, \ldots, x_1^{a_\ell}$ is an optimal monomial quadratization of the system.

8 Conclusions and Open problems

In this paper, we have presented the first practical algorithm for finding an optimal monomial quadratization. Our implementation compares favorably with the existing software and allows us to find better quadratizations for already used benchmark problems. We were able to compute quadratization for ODE systems which could not be tackled before.

We would like to mention several interesting open problems:

1. Is it possible to describe a finite set of monomials which must contain an optimal quadratization? This would allow using SAT-solving techniques of [13] as described in Section 3.
2. As has been shown in [2], general polynomial quadratization may be of a smaller dimension than an optimal monomial quadratization. This poses a challenge: design an algorithm for finding optimal polynomial quadratization (or at least a smaller one than an optimal monomial).
3. How to search for optimal monomial quadratizations if negative powers are allowed (see Section 7)?
4. How to design a faster algorithm for approximate quadratization (that is, finding a quadratization which is close to the optimal) with guarantees on the quality of the approximation?

References

1. Abreu, M., Balbuena, C., Labbate, D.: Adjacency matrices of polarity graphs and of other C4-free graphs of large size. Designs, Codes and Cryptography **55**(2-3), 221–233 (2010), https://doi.org/10.1007/s10623-010-9364-1
2. Alauddin, F.: Quadratization of ODEs: Monomial vs. non-monomial. SIAM Undergraduate Research Online **14** (2021), https://doi.org/10.1137/20s1360578
3. Bulteau, L., Fertin, G., Rizzi, R., Vialette, S.: Some algorithmic results for [2]-sumset covers. Information Processing Letters **115**(1), 1–5 (2015), https://doi.org/10.1016/j.ipl.2014.07.008
4. Carothers, D.C., Parker, G.E., Sochacki, J.S., Warne, P.G.: Some properties of solutions to polynomial systems of differential equations. Electron. J. Diff. Eqns. **2005**(40), 1–17 (2005), http://emis.impa.br/EMIS/journals/EJDE/Volumes/2005/40/carothers.pdf
5. Clapham, C.R.J., Flockhart, A., Sheehan, J.: Graphs without four-cycles. Journal of Graph Theory **13**(1), 29–47 (1989), https://doi.org/10.1002/jgt.3190130107
6. Cochelin, B., Vergez, C.: A high order purely frequency-based harmonic balance formulation for continuation of periodic solutions. Journal of Sound and Vibration **324**(1-2), 243–262 (2009), https://doi.org/10.1016/j.jsv.2009.01.054
7. Erdös, P., Rényi, A., Sós, V.: On a problem of graph theory. Studia Sci. Math. Hungar. **1**, 215–235 (1966)
8. Fagnot, I., Fertin, G., Vialette, S.: On finding small 2-generating sets. In: Lecture Notes in Computer Science, pp. 378–387. Springer Berlin Heidelberg (2009), https://doi.org/10.1007/978-3-642-02882-3_38
9. Füredi, Z.: On the number of edges of quadrilateral-free graphs. Journal of Combinatorial Theory, Series B **68**(1), 1–6 (1996), https://doi.org/10.1006/jctb.1996.0052
10. Graham, R., Grotschel, M., Lovász, L.: Handbook of Combinatorics, vol. 2. North Holland (1995)
11. Gu, C.: QLMOR: A projection-based nonlinear model order reduction approach using quadratic-linear representation of nonlinear systems. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems **30**(9), 1307–1320 (2011), https://doi.org/10.1109/TCAD.2011.2142184
12. Guillot, L., Cochelin, B., Vergez, C.: A Taylor series-based continuation method for solutions of dynamical systems. Nonlinear Dynamics **98**(4), 2827–2845 (2019), https://doi.org/10.1007/s11071-019-04989-5

13. Hemery, M., Fages, F., Soliman, S.: On the complexity of quadratization for polynomial differential equations. In: Computational Methods in Systems Biology, pp. 120–140. Springer International Publishing (2020), https://doi.org/10.1007/978-3-030-60327-4_7
14. Karkar, S., Cochelin, B., Vergez, C.: A high-order, purely frequency based harmonic balance formulation for continuation of periodic solutions: The case of non-polynomial nonlinearities. Journal of Sound and Vibration **332**(4), 968–977 (2013), https://doi.org/10.1016/j.jsv.2012.09.033
15. Kramer, B., Willcox, K.E.: Balanced truncation model reduction for lifted nonlinear systems (2019), https://arxiv.org/abs/1907.12084
16. Kramer, B., Willcox, K.E.: Nonlinear model order reduction via lifting transformations and proper orthogonal decomposition. AIAA Journal **57**(6), 2297–2307 (2019), https://doi.org/10.2514/1.J057791
17. Morrison, D.R., Jacobson, S.H., Sauppe, J.J., Sewell, E.C.: Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. Discrete Optimization **19**, 79–102 (2016), https://doi.org/10.1016/j.disopt.2016.01.005
18. OEIS Foundation Inc.: The on-line encyclopedia of integer sequences, http://oeis.org
19. Rabinovich, M.I., Fabrikant, A.L.: Stochastic self-modulation of waves in nonequilibrium media. J. Exp. Theor. Phys **77**, 617–629 (1979)
20. Ritschel, T.K., Weiß, F., Baumann, M., Grundel, S.: Nonlinear model reduction of dynamical power grid models using quadratization and balanced truncation. at-Automatisierungstechnik **68**(12), 1022–1034 (2020), https://doi.org/10.1515/auto-2020-0070

Appendix: Benchmark systems

Most of the benchmark systems used in this paper (in Tables 3 and 3) are described in [13]. Here we show additional benchmarks we have introduced:

1. *Cubic Cycle(n).* For every integer $n > 1$, we define a system in variables $x_1, \ldots, x_n$ by
$$x_1' = x_2^3,\ x_2' = x_3^3,\ \ldots,\ x_n' = x_1^3.$$
2. *Cubic Bicycle(n).* For every integer $n > 1$, we define a system in variables $x_1, \ldots, x_n$ by
$$x_1' = x_n^3 + x_2^3,\ x_2' = x_1^3 + x_3^3,\ \ldots,\ x_n' = x_{n-1}^3 + x_1^3.$$