Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

WINFRIED NEUN
THOMAS STURM[1]
STEFAN VIGERSKE[2,3]

# Supporting Global Numerical Optimization of Rational Functions by Generic Symbolic Convexity Tests

[1] Departamento de Matemáticas, Estadística y Computación, Universidad de Cantabria, 39071 Santander, Spain
[2] Humboldt Universität zu Berlin, 10099 Berlin, Germany
[3] Supported by the DFG Research Center MATHEON *Mathematics for key technologies* in Berlin.

ZIB-Report 10-01 (February 2010)

# Supporting Global Numerical Optimization of Rational Functions by Generic Symbolic Convexity Tests

Winfried Neun[§], Thomas Sturm[¶], Stefan Vigerske[‖]

**Abstract**

Convexity is an important property in nonlinear optimization since it allows to apply efficient local methods for finding global solutions. We propose to apply symbolic methods to prove or disprove convexity of rational functions over a polyhedral domain. Our algorithms reduce convexity questions to real quantifier elimination problems. Our methods are implemented and publicly available in the open source computer algebra system Reduce. Our long term goal is to integrate Reduce as a "workhorse" for symbolic computations into a numerical solver.

## 1 Introduction

Convexity is an important property in nonlinear optimization since it allows to apply efficient local methods for finding global solutions. However, proving convexity of a general nonlinear function is a non-trivial task, for which no general methods are known. In this paper we propose to apply methods originated in computer logic to prove or disprove convexity in the special case of *rational functions*.

A *nonlinear optimization problem* (NLP) is a problem of the following form:

$$\begin{aligned} \text{minimize} \quad & g_0(x), \\ \text{such that} \quad & g_i(x) \leq 0, \quad i = 1, \ldots, m, \\ & x \in X, \end{aligned} \tag{P}$$

where $X \subseteq \mathbb{R}^n$ is polyhedral and $g_i : \mathbb{R}^n \to \mathbb{R}$, $i = 0, \ldots, m$, are functions that are differentiable for all $x \in X$. The function $g_0$ is the *objective function*, and the $g_i(x) \leq 0$ for $i = 1, \ldots, m$ are *constraints*. The *feasible set* of (P) is the set of *feasible points* in $X$ that satisfy all constraints. In the following we assume that the feasible set is non-empty.

Nonlinear optimization problems arise in various applications, e.g., engineering design, logistics, manufacturing, and the chemical and biological sciences [2, 12, 13, 14, 21].

We recall that a set $A$ is called *convex*, if for all $x$, $y \in A$ and for all $\lambda \in [0, 1]$ we have $\lambda x + (1 - \lambda)y \in A$. Further, a function $f : \mathbb{R}^n \to \mathbb{R}$ is called *convex* on a convex set $A \subseteq \mathbb{R}^n$ if for all $x$, $y \in X$ and for all $\lambda \in [0, 1]$ we have

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y).$$

[§]Zuse Institute Berlin, Takustraße 7, 14195 Berlin, Germany, `neun@zib.de`

[¶]Departamento de Matemáticas, Estadística y Computación, Universidad de Cantabria, 39071 Santander, Spain, `sturmt@unican.es`

[‖]Humboldt University Berlin, Unter den Linden 6, 10099 Berlin, Germany, `stefan@math.hu-berlin.de`, supported by the DFG Research Center Matheon *Mathematics for key technologies* in Berlin

A feasible point $\bar{x}$ of (P) is said to be a *global optimum* of (P), if there is no other feasible point that has a lower objective value than $\bar{x}$. Further, a feasible point $\bar{x}$ is said to be a *local optimum* of (P), if there exists no other feasible point in a neighbourhood of $\bar{x}$ which has a lower objective value than $\bar{x}$. Clearly, every global optimum of (P) is also a local optimum. However, in general, not every local optimum is also a global optimum.

Under some regularity conditions, there exist efficient numerical algorithms to find a local optimum of (P) [17]. On the other hand, finding a global optimum solution of (P) is very difficult in general. However, if the set of feasible points is *convex* and the objective function $g_0(x)$ is *convex* on the feasible set, then it can be easily seen that also every local optimum of (P) is a global optimum. Thus, in case that an NLP is known to be convex, methods for finding local optima can be applied for the global optimization of an NLP. A sufficient condition for the feasible set to be convex is that every function $g_i(x)$, $i = 1, \ldots, m$, is convex on $X$.

Even algorithms for solving nonconvex NLPs [1, 21] can profit from information about convexity of a subset of the constraint functions. Such methods usually construct a convex relaxation of (P), the optimal value of which yields a lower bound on the global optimum of (P). The lower bound allows to evaluate the quality of a feasible solution of (P) and to direct the search for a better solution. The convex relaxation is thereby obtained by replacing each function $g_i(x)$, $i = 0, \ldots, m$, by a *convex underestimator* $\breve{g}_i(x)$ that is convex and pointwise less than or equal to $g_i(x)$ on $X$. The tighter these underestimators are, the better are the lower bounds that can be expected. Unfortunately, there exists no method to construct a tightest convex underestimator for a given function in general. Clearly, if the algorithm knows that a function $g_i(x)$ is already convex, then $\breve{g}_i(x)$ can be chosen to equal $g_i(x)$.

Existing deterministic methods for proving or disproving the convexity of a function (given as composition of elementary expressions) with respect to bounds on its variables are based on walking an expression tree and applying convexity rules for function compositions [11], or estimating the spectra of the Hessian matrix or its sign in case of a univariate function [15, 16], or deciding positive semidefiniteness of the interval Hessian [16].

All these approaches may give inconclusive results. For example, the method from [11]—even though very fast—may fail to detect convexity of the function $f(x) = -x/(1 + x)$ on the set $X = [0, 1]$, since it includes no rules for concluding convexity of a quotient of two non-constant functions. Formulating the function as $f(x) = 1/(1+x)-1$, however, convexity is proven, since the enumerator of $1/(1+x)$ is a positive constant, and the denominator $1 + x$ is concave. The second and third method, in contrast, have no problem in proving convexity for $f(x)$, since they only need to prove positivity of the second derivative $f''(x) = 2/(1 + x)^3$.

Nevertheless, for the function $f(x) = 2x^7 - 7x^4 + 84x^2 + 42$ a method like [16] may fail to prove convexity of $f(x)$ on the interval $[-1, 2]$. In this example the second derivative is $f''(x) = 84(x^5 - x^2 + 2)$. Replacing each occurrence of $x$ by $[-1, 2]$ and applying rules for interval arithmetic yields $f''(x) \in 84 \cdot ([-1, 32] - [0, 4] + [2, 2]) = 84 \cdot [-3, 34]$, which allows no conclusive result.

Finally, when proving or disproving convexity of a function over a set all these methods can consider only *simple bound constraints* on the variables $x_i$. These are constraints directly bounding $x_i$ by a number. For instance, the function $f(x) = (x - y)^3$ is obviously convex on the set

$$X = \{\, (x, y) \in \mathbb{R}^2 \mid x, y \in [0, 1],\ x \geq y \,\}.$$

However, [11] would fail to prove convexity of $f(x, y)$ since it only considers the simple bounds

$x, y \in [0, 1]$ and thus does not "see" that $x - y \geq 0$ on $X$.

In this paper, we present a novel symbolic method to prove or disprove convexity of *rational functions* over polyhedral sets. The key idea is to reduce convexity problems to first-order sentences over the reals and to decide these sentences by quantifier elimination methods. Our original contributions are the following:

- We devise a new complete symbolic method for deciding the convexity of rational functions over polyhedral domains.
- Unlike existing methods, our approach is not restricted to simple bound constraints but can process arbitrary multi-linear constraints.
- We apply *positive quantifier elimination*, which has been successfully used for existential problems in the past [18, 19], to universal problems.
- All our methods discussed throughout the paper are efficiently implemented and publicly available in the open-source computer algebra system Reduce.
- We provide and discuss a comprehensive set of benchmark computations to demonstrate the feasibility of our method for established benchmark suites from the NLP community.

The plan of the paper is as follows: In Section 2 we make precise the special case of the convexity problem addressed in this paper. In Section 3 we motivate and develop our various reductions of the problem to suitable first-order sentences. In Section 4 we introduce the concept of positive quantifier elimination and provide an algorithmic reduction of certain convexity problems to make these accessible to this more efficient variant of quantifier elimination. In Section 5 we give asymptotic upper bounds on the time complexity of our method. Section 6 illuminates our work from a software systems point of view and discusses some future plans for our project. In Section 7 we discuss and analyze comprehensive benchmarks carried out for our method. In Section 8 we finally summarize and evaluate our results.

## 2 Problem Definition

We are now going to precisely state the particular problem addressed in the remainder of our paper. Recall that the exact role of this problem for nonlinear global optimization has been made explicit in the Introduction.

We consider rational functions $f \in \mathbb{Q}(x_1, \ldots, x_n)$ as formal objects that establish defining terms for real functions $f : \mathbb{R}^n \to \mathbb{R}$. The *domain* of $f = p/q \in \mathbb{Q}(x_1, \ldots, x_n)$, denoted by $\mathrm{dom} f$, is the set of all points $x \in \mathbb{R}^n$ for which the denominator $q$ does not vanish. Note that every real function defined in term of sums, products, and divisions of variables and rational constants can be described by a rational function.

A set $X \subseteq \mathbb{R}^n$ is called *polyhedral*, if it can be written as the intersection of finitely many halfspaces

$$X = \{\, x \in \mathbb{R}^n \mid Ax \leq b \,\}, \quad A \in \mathbb{Q}^{n \times m}, \quad b \in \mathbb{Q}^m.$$

A rational function $f$ is called *convex* on a polyhedral set $X$ if for all $x$, $y \in X$ and for all $\lambda \in [0, 1]$ we have

$$f(\lambda x + (1 - \lambda) y) \leq \lambda f(x) + (1 - \lambda) f(y). \tag{1}$$

The question we aim to answer is the following: *Given a rational function $f \in \mathbb{Q}(x_1, \ldots, n_n)$ and a polyhedral set $X \subseteq \mathrm{dom} f \subseteq \mathbb{R}^n$, decide whether or not $f$ is convex on $X$.*

## 3 Method

We are going to encode various criteria for convexity into first-order sentences over the language $(0, 1, +, -, \cdot)$ of ordered rings, which is also known as the *Tarski algebra* [20]. These sentences are then checked automatically using real quantifier elimination procedures.

Recall our problem statement in the previous section: We are given $f \in \mathbb{Q}(x_1, \dots, x_n)$, which has got integer coefficients. In addition, we are given $X \subseteq \mathrm{dom} f$ polyhedral, which is naturally described by a conjunction $\gamma$ of linear constraints, i.e. a quantifier-free formula in our language. To start with, our definition (1) above can be directly translated into a first-order formula:

**Lemma 1** (Naive Convexity Condition)**.** *Consider a function $f = \frac{p}{q} \in \mathbb{Q}(x_1, \dots, x_n)$ and a formula $\gamma$ in $x_1$, ..., $x_n$ describing a polyhedral set $X = \{ x \in \mathbb{R}^n \mid \gamma(x) \} \subseteq \mathrm{dom} f$. Let $y_1$, ..., $y_n$, $\lambda$ be new variables, and denote:*

$$\begin{aligned} \delta &= \gamma[y_1/x_1, \dots, y_n/x_n], \\ f_0 &= f(\lambda x_1 + (1-\lambda) y_1, \dots, \lambda x_n + (1-\lambda) y_n), \\ f_1 &= \lambda f + (1-\lambda) f(y_1, \dots, y_n), \end{aligned}$$

*where $f_0$, $f_1$ are obtained via evaluation homomorphisms into $\mathbb{Q}(x_1, \dots, x_n, y_1, \dots, y_n)$ and computation in that field. Denote by $N$ and $D$ the numerator and the denominator of $f_0 - f_1$, respectively. Then $f$ is convex on $X$ if and only if the following first-order sentence holds:*

$$\Phi_1(f, \gamma) = \forall x_1 \dots \forall x_n \forall y_1 \dots \forall y_n \forall \lambda (\gamma \wedge \delta \longrightarrow ND \leq 0). \quad \square$$

This formulation has got $2n + 1$ universal quantifiers. As we will make precise in theory in Section 5 and demonstrate by means of comprehensive example computations in Section 7, the number of quantifiers is the dominant measure of complexity for real quantifier elimination in our case.

As a first optimization we are now going to reduce the number of quantifiers from $2n+1$ to $n + 1$. Under some natural conditions, convexity of $f$ is directly related to certain properties of its Hessian $\nabla^2 f$. Recall that a matrix $A \in \mathbb{R}^{n \times n}$ is *positive semidefinite* if $zAz \geq 0$ and *positive definite* if $zAz > 0$ for all $z \in \mathbb{R}^n$. Assume now that $f$ is twice continuously differentiable on $X$. Then the following are equivalent:

(i) $f$ is convex on $X$,

(ii) for all $x \in X$ the matrix $(\nabla^2 f)(x) \in \mathbb{R}^{n \times n}$ is positive semidefinite,

(iii) for all $x \in X$ the matrix $(\nabla^2 f)(x) \in \mathbb{R}^{n \times n}$ has got exclusively non-negative eigenvalues.

This gives rise to the following algorithm, which produces an alternative first-order sentence describing convexity:

**Subalgorithm 1** (Non-negative Eigenvalues)**.**

**Input** *a formula $\gamma$ in variables $x_1$, ..., $x_n$ and a function $f = \frac{p}{q} \in \mathbb{Q}(x_1, \dots, x_n)$ that is twice continuously differentiable on $X = \{ x \in \mathbb{R}^n \mid \gamma(x) \} \subseteq \mathrm{dom} f$, which must be polyhedral.*

**Output** *a first-order sentence that is equivalent to* true *if and only if $f$ is convex over $X$.*

1. *Compute $\nabla^2 f \in \mathbb{Q}(x_1, \dots, x_n)^{n\times n}$. Notice that due to the differentiation rules, the denominators of the entries of $\nabla^2 f$ are powers of $q$.*

2. *Compute the characteristic polynomial*

$$\chi = \det(\nabla^2 f - \lambda I) \in \mathbb{Q}(x_1, \dots, x_n)[\lambda]$$

*where $I$ denotes the $n \times n$ unit matrix.*

3. *Since $\chi$ is a linear combination of products of the entries of the matrix $\nabla^2 f - \lambda I \in \mathbb{Q}(x_1, \dots, x_n)[\lambda]^{n\times n}$, it follows that all denominators occurring in the coefficients of $\chi$ are once more powers of the denominator $q$ of $f$ and thus do not vanish over $X$. Let $\Delta$ denote the* gcd *of all these coefficient denominators and rewrite $\chi = \frac{\tilde{\chi}}{\Delta}$, where $\tilde{\chi} \in \mathbb{Q}[x_1, \dots, x_n, \lambda]$, $\Delta \in \mathbb{Q}[x_1, \dots, x_n]$. Then for any choice $x_1, \dots, x_n \in X$ we have $\chi(\lambda) = 0$ if and only if $\tilde{\chi}(\lambda) = 0$.*

*Altogether we finally obtain:*

$$\Phi_{\mathrm{iii}}(f, \gamma) := \forall x_1 \dots \forall x_n \forall \lambda (\gamma \longrightarrow \lambda < 0 \longrightarrow \tilde{\chi} \neq 0). \quad \square$$

Recall from the problem statement in the previous section our definition (1) of convexity of a function $f : \mathbb{R}^n \to \mathbb{R}$ on a convex set $X \subseteq \mathrm{dom} f$. Similarly to this, $f$ is called *strictly convex* on $X$ if for all $x, y \in X$ with $x \neq y$ and all $\lambda \in [0, 1]$ we have

$$f(\lambda x + (1-\lambda)y) < \lambda f(x) + (1-\lambda) f(y). \tag{2}$$

It is easy to see that strict convexity implies convexity. We are now going to exploit this fact for a heuristic test that requires only $n$ quantifiers.

Quite naturally, a similar equivalence as above holds for strict positiveness, but here yet another equivalent enters the stage, which is most interesting from an algorithmic point of view:

(i′) $f$ is strictly convex on $X$,

(ii′) for all $x \in X$ the matrix $(\nabla^2 f)(x) \in \mathbb{R}^{n\times n}$ is positive definite,

(iii′) for all $x \in X$ the matrix $(\nabla^2 f)(x) \in \mathbb{R}^{n\times n}$ has got exclusively positive eigenvalues,

(iv′) for all $x \in X$ the determinants of all principal subminors of $(\nabla^2 f)(x) \in \mathbb{R}^{n\times n}$ are positive.

This new condition (iv′) can then be effectively expressed as a first-order formula in the Tarski algebra, which gives rise to the following algorithm:

**Subalgorithm 2** (Positive Principal Subminors)**.**

**Input** *a formula $\gamma$ in variables $x_1, \dots, x_n$ and a function $f = \frac{p}{q} \in \mathbb{Q}(x_1, \dots, x_n)$ such that $f$ is twice continuously differentiable on $X = \{ x \in \mathbb{R}^n \mid \gamma(x) \} \subseteq \mathrm{dom} f$, which must be polyhedral.*

**Output** *a first-order sentence that is equivalent to* true *if and only if $f$ is strictly convex over $X$.*

1. *Compute $\nabla^2 f \in \mathbb{Q}(x_1, \dots, x_n)^{n\times n}$. Notice that due to the differentiation rules, the denominators of the entries of $\nabla^2 f$ are powers of $q$.*

2. *Compute the determinants $\frac{u_1}{v_1}$, ..., $\frac{u_n}{v_n} \in \mathbb{Q}(x_1, \dots, x_n)$ of the principal subminors of $\nabla^2 f$.*

3. *From the Leibniz formula for the determinant it is clear that the denominators $v_1$, ..., $v_n$ are again powers of $q$. Since $f$ is differentiable on $X$ it is in particular continuous. Consequently $q$ does not vanish on $X$ and neither do the $v_1$, ..., $v_n$. Hence for $x_1$, ..., $x_n \in X$ a condition $\frac{u_i}{v_i} > 0$ can be equivalently rewritten as $u_i v_i > 0$.*

*Altogether we finally obtain:*

$$\Phi_{\mathrm{iv}'}(f,\gamma) := \forall x_1 \dots \forall x_n \left(\gamma \longrightarrow \bigwedge_{i=1}^{n} u_i v_i > 0\right). \quad \square$$

On the basis that this algorithm yields only a sufficient condition for convexity, the improvement from $n+1$ to $n$ quantifiers might not appear too striking. There is, however, another measure of complexity that turns Subalgorithm 2 considerably superior to Subalgorithm 1: The degrees of the terms in the formula. The degrees in $\Phi_{\mathrm{iv}'}$ only depend on the degrees in the input $f$ and $\gamma$. In $\Phi_{\mathrm{iii}}$, in contrast, there is additionally the numerator $\tilde{\chi}$ of the characteristic polynomial, the degree of which is bounded from below by the number $n$ of variables.

The degrees are well-known to be a relevant measure for the complexity of real quantifier-elimination [3]. Even more important, low degrees are crucial for the success of the efficient virtual substitution methods [22, 23] primarily used by our implementation. If these methods fail, our implementation has to fall back to partial cylindrical decomposition methods [7], which are not only single but double exponential in the number of universal quantifiers.

Our main algorithm now combines the two subalgorithms in the obvious way:

**Algorithm 1** (Convexity)**.**

**Input** *a formula $\gamma$ in variables $x_1$, ..., $x_n$ and a function $f = \frac{p}{q} \in \mathbb{Q}(x_1, \dots, x_n)$ such that $f$ is twice continuously differentiable on $X = \{\, x \in \mathbb{R}^n \mid \gamma(x) \,\} \subseteq \mathrm{dom} f$, which must be polyhedral.*

**Output** true *if $f$ is convex over $X$,* false *else.*

1. $\Phi := \Phi_{\mathrm{iv}'}(f,\gamma)$ *by Subalgorithm 2.*

2. $\Phi' :=$ realQuantifierElimination$(\Phi)$

3. *if* $\Phi' =$ true *then return* true.

4. $\Phi := \Phi_{\mathrm{iii}}(f,\gamma)$ *by Subalgorithm 1.*

5. $\Phi' :=$ realQuantifierElimination$(\Phi)$

6. *return* $\Phi'$ $\square$

# 4 Positive Quantifier Elimination for Universal Formulas

As indicated in the previous section, we employ quantifier elimination procedures contained in the Redlog [10] package of the open source computer algebra system Reduce to finally decide our sentences. The default real quantifier elimination procedure there applies virtual substitution methods [22, 23] as long as the degrees of the quantified variables admit this and then falls back to partial cylindrical algebraic decomposition [7]. We are going to refer to this procedure as *QE* in the following.

Besides QE, we take an alternative novel approach: We use a *dual version* of *positive quantifier elimination.* Positive quantifier elimination had been originally developed by the second author for existential sentences. This original version has been successfully applied for discovering oscillations in gene regulatory networks in the area of algebraic biology, where it clearly outperformed QE by all means [18, 19].

The essential idea of positive quantifier elimination is to consider problems, where *all* contained variables are known to be positive. Within virtual substitution methods this knowledge can be heuristically exploited in numerous ways; see [18, 19] for details. For the partial cylindrical algebraic decomposition [7] we exploit positivity to some extent as well though much less systematically so far. In the following we are going to refer to positive quantifier elimination as *PQE*.

In the situation of this paper, we can systematically arrive at a positive situation in many cases:

**Lemma 2** (Shift to the 1. Hyper-Quadrant)**.** *Consider a function $f \in \mathbb{Q}(x_1, \dots, x_n)$ and a formula $\gamma$ in $x_1$, …, $x_n$ describing a polyhedral set $X = \{\, x \in \mathbb{R}^n \mid \gamma(x) \,\} \subseteq \mathrm{dom} f$. Assume that $\gamma$ bounds $x_i$ from below by some $a \in \mathbb{Q}$. We set $\hat{\gamma} = \gamma[x_i - a/x_i] \wedge x_i > 0$ silently dropping a (positive) denominator after substitution and obtain $\hat{f} \in \mathbb{Q}(x_1, \dots, x_n)$ by plugging $x_i - a$ for $x_i$ into $f$. Then $x_i$ is positive on the polyhedral set $\hat{X}$ described by $\hat{\gamma}$, and $\hat{f}$ is convex on $\hat{X}$ if and only if $f$ is convex on $X$.*

*Proof.* If $x_i$ is strictly bounded from below by $a$, then we have just moved our problem along the $x_i$-axis via a simple linear transformation. If, in contrast, $\gamma$ guarantees only $a \leq x_i$ then we have turned this in addition into a strict order relation. Since, however, $f$ is smooth on $\mathrm{dom} f$, convexity—in contrast to strict convexity—remains invariant. □

Of course the argument about the smoothness of $f$ in the proof could be avoided by shifting by $a + 1$ instead of $a$. We have observed, however, that this leads to slightly more complicated terms, which can be disadvantageous for the quantifier elimination procedures.

By iterative application of the lemma, we finally arrive at a completely positive situation provided that all variables are explicitly bounded from below. As our examples in Section 7 will demonstrate this is frequently the case for NLP.

In the positive case, we in addition have to take care of the variables $\lambda$ in $\Phi_1$ from Lemma 1 and in $\Phi_{\text{iii}}$ from Subalgorithm 1. To $\Phi_1$ we conjunctively add within the scope of the quantifiers the case $\lambda = 0$ via substitution. In $\Phi_{\text{iii}}$ we substitute within the scope of the quantifiers $-\lambda$ for $\lambda$.

## 5 Complexity

The complexity of our method is dominated by the quantifier elimination step. This is asymptotically bounded by an exponential function in the number of quantifiers, i.e. essentially the dimension $n$ of the domain $\mathbb{R}^n$ of $f$. As there are no quantifier alternations in our case, that bound is only single exponential [8, 22]. Notice that when we have to use Subalgorithm 1 in contrast to Subalgorithm 2, the additional quantifier $\forall\lambda$ contributes exponentially to the complexity. Either do the quantifiers $\forall y_1, \ldots, \forall y_n$ with the naive approach according to Lemma 1.

## 6 System Architecture

The computations of the entities needed here, such as derivatives, matrices, and Hessians are done using the computer algebra system Reduce. The system is the well-known host of the Redlog [1] software system which is essential for the algorithms presented above. Reduce is free software since January 2009 which allows us to manage the communication between several independent tasks, e.g., by modifications and technical add-ons to the base system. The system is hosted at SourceForge[2]. Information on the Reduce system in general can be found at its website[3]. It is considered to make the PSL-based Reduce system available also as a linkable library in the near future, which could be easily used by other (e.g. numerical) software systems.

For the experiments that are discussed in the next section, we have let a numerical optimization software write the function $f$ and the linear inequalities that state the set $X$ into a file, which is then read in by Reduce. The interpretation of the generated outputs is done automatically by using standard Linux tools. However, to allow a seamless integration of Reduce as a service for symbolic computations in an optimization software, a more efficient mechanism for communication is currently developed. We have chosen as basis to send binary objects via shared memory, which avoids the overhead of coding and decoding character strings. In our case the data send to Reduce is relatively large ($f$ and $X$) compared to the results (convex, strictly convex, not convex, or unknown).

It is an interesting option to run this software on a parallel system, since the evaluations of convexity issues for multiple formulas are independent from each other. In fact, we have run our examples on a 16 processors (2.8 GHz each) x86_64 machine with 256 GB of memory under Linux.

## 7 Examples

In order to evaluate our different convexity test methods, we assembled a test set of NLPs and MINLPs from various sources; an MINLP is an NLP where some variables are additionally restricted to take only integer variables. Firstly, the COPS testset [9] is a collection of difficult NLP models which have their origin in various applications. It is frequently used to benchmark NLP solvers. From COPS, we selected instantiations of the models `bearing`, `catmix`, `gasoil`, `glider`, `robot`, and `rocket`. Secondly, we picked some models from the

[1] `http://www.redlog.eu`
[2] `http://reduce-algebra.sourceforge.net`
[3] `http://www.reduce-algebra.com`

"CMU-IBM Cyber-Infrastructure for MINLP" webpage [6], which collects MINLP models from real-world applications. We selected the models "Periodic Scheduling of Continuous Multiproduct Plants" (#34), "Stabilizing controller design and the Belgian chocolate problem" (#57), "The Delay Constrained Routing Problem" (#63), and "Simultaneous Cyclic Scheduling and Control of a Multiproduct CSTR" (#71). Thirdly, we took two instances from a recent paper on solvers for convex MINLPs [4] and one instance from the MINLPLib [5]. These are a constrained layout problem (`clay`), a stochastic service system design problem (`sssd`), and the instance `du-opt5`. Further, we selected a formula for so-called "second-order isotherms" as they appear in the modeling of chromatographic separation processes [2]. Finally, we added the three convex functions that were mentioned in the Introduction as counterexamples to the completeness of existing approaches.

For each NLP, we selected those constraints that are nonlinear and rational functions. For equational constraints $g_i(x) = 0$ we considered also $-g_i(x)$ in order to check for concavity of $g_i(x)$ too. Sets of functions that differ only in the naming of the variables were replaced by one representative.

For each example obtained this way, we have proceeded as follows:

1. (a) Compute $\Phi_1$ according to Lemma 1, and apply QE.
   (b) Compute $\Phi_{\mathrm{iv}'}$ according to Subalgorithm 2, and apply QE.
   (c) If $\Phi_{\mathrm{iv}'}$ did not yield true, i.e. strict convexity, then compute $\Phi_{\mathrm{iii}}$ according to Subalgorithm 1, and apply QE.
2. If the variables in the example are bounded from below, then move the problem to the first hyper-quadrant, and proceed as in 1. (a)–1. (c) but with PQE instead of QE.

For every single computation 1. (a), …, 2. (c) we imposed a timelimit of 10 minutes after which non-finished computations were automatically interrupted. The steps (b) and (c) reflect our proposed Algorithm 1, while the steps (a) are supposed to demonstrate that our algorithmic ideas formulated in Subalgorithm 1 and Subalgorithm 2 outperform the naive approach from Lemma 1. Finally, by considering 1. vs. 2., we are able to judge the efficiency of PQE compared to regular QE.

In Table 1 there are results given for all examples, where PQE is used whenever possible and regular QE else. The columns *example* and *function* show the names of the NLP and the function $f = p/q$ in the NLP that are considered in this line, respectively. The columns $n$, $\deg p$, and $\deg q$ show the number of variables in $f$, the total degree of the polynomial $p$, and the total degree of the polynomial $q$, respectively. The column *curvature* shows whether $f$ was proven to be strictly convex, convex, not convex. It states "unknown", if no method was able to give a result within the time limit. Column PQE indicates whether there was positive quantifier elimination applied. The columns $\Phi_1$, $\Phi_{\mathrm{iv}'}$, $\Phi_{\mathrm{iii}}$ present the corresponding running times in milliseconds, while "$\perp$" is printed if the method hit the time limit of 10 minutes. Recall that $\Phi_{\mathrm{iii}}$ is not considered if $\Phi_{\mathrm{iv}'}$ already yields that the function is strictly convex. In that case we have "–" instead of a running time.

Note that our method was able to prove convexity for all examples mentioned in the introduction including the example $f = (x - y)^3$, where convexity is only given on the region defined by the linear condition $x - y \geq 0$. Furthermore, we were able to decide convexity for the formulas in the instances `catmix100` and `robot50`, whereas [11] reported inconclusive results.

For those examples, where PQE could be used and Table 1 thus gives PQE timings instead of QE timings, Table 2 explicitly compares the computation times of PQE and regular QE.

## 8 Conclusions

Our benchmarking has confirmed that our proposed Algorithm 1 is the most suitable combination of the methods introduced throughout this paper. Furthermore, PQE should probably be used rather than regular QE, although the difference in performance is not at all as striking as with the examples previously reported in the literature [18, 19]. Of course, we do not consider our symbolic method a stand-alone solution which should replace more efficient though incomplete approaches. We think that it would be a reasonable scheme to first try the very fast convexity rules from [11], then to try to disprove convexity numerically, and finally—in the case of rational functions—apply our Algorithm 1. A conclusive result on nonconvexity can be obtained numerically by computing the Hessian $H$ in some points of $X$, and using a robust numerical algorithm to find a vector $z$ such that $z^T Hz < 0$ [11, Sec. 5]. Altogether we consider our work described here an encouraging milestone in our research on integrating Reduce as a symbolic library with state-of-the-art numerical NLP solvers.

## References


[1] C. S. Adjiman and C. A. Floudas. Rigorous convex underestimators for general twice-differentiable problems. *Journal of Global Optimization*, 9:23–40, 1997.

[2] M. Ballerstein, D. Michaels, A. Seidel-Morgenstern, and R. Weismantel. A theoretical study of continuous counter-current chromatography for adsorption isotherms with inflection points. Accepted for publication in Computers & Chemical Engineering, 2009.

[3] S. Basu, R. Pollack, and M.-F. Roy. On the combinatorial and algebraic complexity of quantifier elimination. *Journal of the ACM*, 43(6):1002–1045, 1996.

[4] P. Bonami, M. Kilinç, and J. Linderoth. Algorithms and software for convex mixed integer nonlinear programs. available at Optimization Online, `http://www.optimization-online.org/DB_HTML/2009/10/2429.html`, 2009.

[5] M. R. Bussieck, A. S. Drud, and A. Meeraus. MINLPLib - A Collection of Test Models for Mixed-Integer Nonlinear Programming. *INFORMS Journal on Computing*, 15(1):114–119, 2003. `http://www.gamsworld.org/minlp/minlplib.htm`.

[6] CMU-IBM cyber-infrastructure for MINLP. `http://www.minlp.org`, 2009.

[7] G. E. Collins and H. Hong. Partial cylindrical algebraic decomposition for quantifier elimination. *Journal of Symbolic Computation*, 12(3):299–328, Sept. 1991.

[8] J. H. Davenport and J. Heintz. Real quantifier elimination is doubly exponential. *Journal of Symbolic Computation*, 5(1–2):29–35, Feb.–Apr. 1988.

[9] E. D. Dolan, J. J. Moré, and T. S. Munson. Benchmarking optimization software with COPS 3.0. Technical Report ANL/MCS-273, Mathematics and Computer Science Division, Argonne National Laboratory, 2004. `http://www.mcs.anl.gov/~more/cops`.

[10] A. Dolzmann and T. Sturm. Redlog: Computer algebra meets computer logic. *ACM SIGSAM Bulletin*, 31(2):2–9, June 1997.

[11] R. Fourer, C. Maheshwari, A. Neumaier, D. Orban, and H. Schichl. Convexity and concavity detection in computational graphs: Tree walks for convexity assessment. *INFORMS Journal on Computing*, Articles in Advance, 2009.

[12] I. E. Grossmann, editor. *Global Optimization in Engineering Design*. Kluwer Academic Publishers, 1996.

[13] I. E. Grossmann and Z. Kravanja. Mixed-integer nonlinear programming: A survey of algorithms and applications. In A. Conn, L. Biegler, T. Coleman, and F. Santosa, editors, *Large-Scale Optimization with Applicationa, Part II:Optimal Design and Control*. Springer, 1997.

[14] M. Jüdes, G. Tsatsaronis, and S. Vigerske. Optimization of the design and partial-load operation of power plants using mixed-integer nonlinear programming. In J. Kallrath, P. Pardalos, S. Rebennack, and M. Scheidt, editors, *Optimization in the Energy Industry*, chapter 9. Springer, 2009.

[15] M. Mönnigmann. Efficient calculation of bounds on spectra of hessian matrices. *SIAM Journal on Scientific Computing*, 30(5):2340–2357, 2008.

[16] I. P. Nenov, D. H. Fylstra, and L. V. Kolev. Convexity determination in the Microsoft Excel solver using automatic differentiation techniques. Technical report, Frontline Systems Inc., 2004.

[17] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 2000.

[18] T. Sturm and A. Weber. Investigating generic methods to solve Hopf bifurcation problems in algebraic biology. In K. Horimoto, G. Regensburger, M. Rosenkranz, and H. Yoshida, editors, *Algebraic Biology – Third International Conference (AB 2008), Castle of Hagenberg, Austria*, volume 5147 of *Lecture Notes in Computer Science*, pages 200–215. Springer-Verlag, Berlin, Heidelberg, Germany, 2008.

[19] T. Sturm, A. Weber, E. O. Abdel-Rahman, and M. El Kahoui. Investigating algebraic and logical algorithms to solve Hopf bifurcation problems in algebraic biology. *Mathematics in Computer Science*, 2(3):493–515, March 2009.

[20] A. Tarski. A decision method for elementary algebra and geometry. Prepared for publication by J. C. C. McKinsey. RAND Report R109, August 1, 1948, Revised May 1951, Second Edition, RAND, Santa Monica, CA, 1957.

[21] M. Tawarmalani and N. V. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*. Kluwer Academic Publishers, 2002.

[22] V. Weispfenning. The complexity of linear problems in fields. *Journal of Symbolic Computation*, 5(1&2):3–27, Feb.–Apr. 1988.

[23] V. Weispfenning. Quantifier elimination for real algebra—the quadratic case and beyond. *Applicable Algebra in Engineering Communication and Computing*, 8(2):85–101, 1997.

Table 1: Results and timings of tests for convexity. All times are given in milliseconds, $\perp$ indicates computations that did not finish within 10 minutes of CPU time.

| example | function | $n$ | deg $p$ | deg $q$ | curvature | PQE | $\Phi_1$ | $\Phi_{\mathrm{iv}'}$ | $\Phi_{\mathrm{iii}}$ |
|---|---|---|---|---|---|---|---|---|---|
| **Introduction** | | | | | | | | | |
| intro | $-x/(1+x)$ | 1 | 1 | 1 | strictly convex | ✓ | 20 | 10 | – |
| | $2x^7-7x^4+84x^2+42$ | 1 | 7 | 0 | convex | ✓ | $\perp$ | < 10 | 10 |
| | $(x_1-x_2)^3$ | 2 | 3 | 0 | convex | ✓ | 100 | 10 | < 10 |
| **COPS test set [9]** | | | | | | | | | |
| bearing | e10_0 | 3 | 5 | 0 | not convex | ✓ | 3610 | < 10 | $\perp$ |
| | e10_1 | 1 | 2 | 0 | strictly convex | ✓ | 10 | < 10 | – |
| | e13 | 2 | 2 | 0 | not convex | ✓ | 10 | < 10 | < 10 |
| | e2 | 2 | 2 | 0 | not convex | ✓ | 20 | 10 | 10 |
| | e3_1 | 2 | 2 | 0 | not convex | ✓ | 10 | < 10 | < 10 |
| | e4 | 2 | 2 | 0 | not convex | ✓ | < 10 | < 10 | 10 |
| | e5_0 | 3 | 3 | 0 | not convex | ✓ | 210 | 10 | 590 |
| | e6 | 4 | 2 | 0 | not convex | ✓ | 290 | < 10 | < 10 |
| | -e13 | 2 | 2 | 0 | not convex | ✓ | 20 | < 10 | < 10 |
| | -e2 | 2 | 2 | 0 | not convex | ✓ | 20 | < 10 | 10 |
| | -e3_1 | 2 | 2 | 0 | not convex | ✓ | 10 | < 10 | 10 |
| | -e4 | 2 | 2 | 0 | not convex | ✓ | < 10 | < 10 | < 10 |
| | -e5_0 | 3 | 3 | 0 | not convex | ✓ | 2060 | < 10 | 180 |
| | -e6 | 4 | 2 | 0 | not convex | ✓ | 120 | 10 | 10 |
| catmix100 | e103 | 6 | 2 | 0 | not convex | – | 10 | < 10 | < 10 |
| | e3 | 6 | 2 | 0 | not convex | – | 20 | 10 | < 10 |
| | -e103 | 6 | 2 | 0 | not convex | – | 10 | 10 | 10 |
| | -e3 | 6 | 2 | 0 | not convex | – | 20 | < 10 | 10 |
| gasoil50 | e1100_0 | 2 | 3 | 0 | not convex | – | 20 | < 10 | < 10 |
| | e1100_1 | 2 | 2 | 0 | not convex | – | 10 | < 10 | 10 |
| | e899 | 194 | 2 | 0 | unknown | – | $\perp$ | $\perp$ | $\perp$ |
| | e900 | 3 | 3 | 0 | not convex | – | 40 | < 10 | < 10 |
| | -e1100_0 | 2 | 3 | 0 | not convex | – | 20 | < 10 | < 10 |
| | -e1100_1 | 2 | 2 | 0 | not convex | – | 10 | 10 | < 10 |
| | -e900 | 3 | 3 | 0 | not convex | – | 30 | < 10 | 10 |
| glider50 | e207 | 2 | 4 | 0 | not convex | ✓ | 3650 | < 10 | 30 |
| | e307 | 2 | 3 | 0 | not convex | ✓ | 30 | < 10 | < 10 |
| | e309 | 5 | 2 | 1 | not convex | – | 60 | 10 | 10 |
| | e3 | 1 | 2 | 0 | not convex | ✓ | 10 | 10 | < 10 |
| | e561 | 4 | 2 | 0 | not convex | – | 10 | 10 | 10 |
| | e610 | 3 | 2 | 0 | not convex | – | < 10 | 10 | 10 |
| | -e207 | 2 | 4 | 0 | not convex | ✓ | 149570 | < 10 | 30 |
| | -e307 | 2 | 3 | 0 | not convex | ✓ | 40 | < 10 | < 10 |
| | -e309 | 5 | 2 | 1 | not convex | – | 40 | 10 | 10 |
| | -e3 | 1 | 2 | 0 | strictly convex | ✓ | < 10 | < 10 | – |
| | -e561 | 4 | 2 | 0 | not convex | – | 20 | < 10 | < 10 |
| | -e610 | 3 | 2 | 0 | not convex | – | 10 | 10 | < 10 |
| robot50 | e203 | 5 | 3 | 2 | not convex | – | 4790 | < 10 | $\perp$ |
| | e3 | 3 | 2 | 0 | not convex | – | 10 | < 10 | 10 |
| | e400 | 1 | 2 | 0 | not convex | ✓ | 10 | < 10 | < 10 |
| | -e203 | 5 | 3 | 2 | not convex | – | 60840 | < 10 | $\perp$ |
| | -e3 | 3 | 2 | 0 | not convex | – | < 10 | < 10 | < 10 |
| | -e400 | 1 | 2 | 0 | strictly convex | ✓ | < 10 | < 10 | – |
| rocket50 | e105 | 3 | 2 | 0 | not convex | ✓ | 10 | < 10 | < 10 |
| | e154 | 6 | 3 | 1 | not convex | ✓ | 760 | 10 | 480 |
| | e253 | 3 | 2 | 0 | not convex | ✓ | 20 | < 10 | < 10 |
| | e53 | 1 | 0 | 2 | not convex | ✓ | 30 | < 10 | < 10 |
| | -e105 | 3 | 2 | 0 | not convex | ✓ | 10 | < 10 | < 10 |
| | -e154 | 6 | 3 | 1 | not convex | ✓ | 760 | < 10 | 480 |
| | -e253 | 3 | 2 | 0 | not convex | ✓ | 10 | < 10 | < 10 |
| | -e53 | 1 | 0 | 2 | strictly convex | ✓ | 30 | < 10 | – |
| **minlp.org [6]** | | | | | | | | | |
| minlp_org_34a | balrecs_i1_k2_t1_ | 3 | 2 | 1 | not convex | ✓ | 20 | < 10 | 50 |
| | balrecs_i1_k2_t4_ | 4 | 2 | 1 | not convex | ✓ | 50 | < 10 | 900 |
| | -balrecs_i1_k2_t1_ | 3 | 2 | 1 | not convex | ✓ | 60 | < 10 | 80 |
| | -balrecs_i1_k2_t4_ | 1 | 1 | 0 | unknown | ✓ | $\perp$ | $\perp$ | $\perp$ |
| | -balrecs_i1_k2_t5_ | 4 | 2 | 1 | not convex | ✓ | 30 | 10 | 640 |
| | -object | 45 | 2 | 0 | unknown | – | $\perp$ | 3010 | $\perp$ |
| | object | 45 | 2 | 0 | unknown | – | $\perp$ | 2810 | $\perp$ |
| minlp_org_34b | defrate_i1_k2_ | 3 | 2 | 0 | not convex | ✓ | 10 | < 10 | < 10 |
| | -defrate_i1_k2_ | 3 | 2 | 0 | not convex | ✓ | 10 | < 10 | 10 |
| | -object | 11 | 3 | 0 | unknown | – | $\perp$ | 50 | $\perp$ |
| | object | 11 | 3 | 0 | unknown | – | $\perp$ | 50 | $\perp$ |
| minlp_org_57 | defrx_3_1_ | 3 | 2 | 1 | not convex | ✓ | 50 | < 10 | 720 |
| | -defrx_3_1_ | 3 | 2 | 1 | not convex | ✓ | 30 | < 10 | 360 |
| | -polyy2 | 6 | 3 | 0 | not convex | ✓ | 810 | < 10 | $\perp$ |
| | -polyy3 | 7 | 4 | 0 | unknown | ✓ | $\perp$ | 20 | $\perp$ |
| minlp_org_71 | fecolc_1_1_1_ | 4 | 2 | 0 | not convex | – | 20 | 40 | 100 |
| | -fecolc_1_1_1_ | 4 | 2 | 0 | not convex | – | 20 | 40 | 110 |
| | -obj | 611 | 3 | 1 | unknown | ✓ | $\perp$ | $\perp$ | $\perp$ |
| | -odec_20_3_5_ | 2 | 3 | 0 | not convex | ✓ | 20 | < 10 | 10 |
| | odec_20_3_5_ | 2 | 3 | 0 | not convex | ✓ | 140 | < 10 | 10 |
| **miscellaneous [2, 4, 5]** | | | | | | | | | |
| clay0203h | e107 | 3 | 3 | 1 | strictly convex | ✓ | $\perp$ | 10 | – |
| | e110 | 3 | 3 | 1 | strictly convex | ✓ | $\perp$ | 170 | – |
| sssd-8-4-3 | e30 | 1 | 1 | 1 | strictly convex | ✓ | 10 | 10 | – |
| du-opt5 | e1 | 18 | 2 | 0 | strictly convex | ✓ | $\perp$ | 180 | – |
| isotherms | isotherm | 2 | 2 | 2 | not convex | ✓ | $\perp$ | 670 | 2320 |
| | -isotherm | 2 | 2 | 2 | not convex | ✓ | $\perp$ | 660 | 2290 |

Table 2: Times of regular vs. positive quantifier elimination. All times are given in milliseconds, $\perp$ indicates computations that did not finish within 10 minutes of CPU time.

| example | function | $n$ | deg $p$ | deg $q$ | curvature | $\Phi_1$ | | $\Phi_{\mathrm{iv}'}$ | | $\Phi_{\mathrm{iii}}$ | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | | | QE | PQE | QE | PQE | QE | PQE |
| **Introduction** | | | | | | | | | | | |
| intro | $-x/(1+x)$ | 1 | 1 | 1 | strictly convex | 10 | 20 | 10 | 10 | – | – |
| | $2x^7 - 7x^4 + 84x^2 + 42$ | 1 | 7 | 0 | convex | $\perp$ | $\perp$ | 10 | $< 10$ | 10 | 10 |
| | $(x_1 - x_2)^3$ | 2 | 3 | 0 | convex | 100 | 100 | $< 10$ | 10 | $< 10$ | $< 10$ |
| **COPS test set [9]** | | | | | | | | | | | |
| bearing | e10_0 | 3 | 5 | 0 | not convex | 3640 | 3610 | $< 10$ | $< 10$ | $\perp$ | $\perp$ |
| | e10_1 | 1 | 2 | 0 | strictly convex | $< 10$ | 10 | $< 10$ | $< 10$ | – | – |
| | e13 | 2 | 2 | 0 | not convex | 10 | 10 | 10 | $< 10$ | $< 10$ | $< 10$ |
| | e2 | 2 | 2 | 0 | not convex | 10 | 20 | 10 | 10 | 10 | 10 |
| | e3_1 | 2 | 2 | 0 | not convex | $< 10$ | 10 | $< 10$ | $< 10$ | $< 10$ | $< 10$ |
| | e4 | 2 | 2 | 0 | not convex | 10 | $< 10$ | $< 10$ | $< 10$ | $< 10$ | 10 |
| | e5_0 | 3 | 3 | 0 | not convex | 230 | 210 | $< 10$ | 10 | 600 | 590 |
| | e6 | 4 | 2 | 0 | not convex | 390 | 290 | 10 | $< 10$ | 10 | $< 10$ |
| | -e13 | 2 | 2 | 0 | not convex | 10 | 20 | 10 | $< 10$ | $< 10$ | $< 10$ |
| | -e2 | 2 | 2 | 0 | not convex | 20 | 20 | $< 10$ | $< 10$ | 10 | 10 |
| | -e3_1 | 2 | 2 | 0 | not convex | $< 10$ | 10 | $< 10$ | $< 10$ | $< 10$ | 10 |
| | -e4 | 2 | 2 | 0 | not convex | $< 10$ | $< 10$ | 10 | $< 10$ | $< 10$ | $< 10$ |
| | -e5_0 | 3 | 3 | 0 | not convex | 1970 | 2060 | $< 10$ | $< 10$ | 200 | 180 |
| | -e6 | 4 | 2 | 0 | not convex | 150 | 120 | 10 | 10 | 10 | 10 |
| glider50 | e207 | 2 | 4 | 0 | not convex | 3760 | 3650 | $< 10$ | $< 10$ | 30 | 30 |
| | e307 | 2 | 3 | 0 | not convex | 30 | 30 | $< 10$ | $< 10$ | 10 | $< 10$ |
| | e3 | 1 | 2 | 0 | not convex | $< 10$ | 10 | $< 10$ | 10 | $< 10$ | $< 10$ |
| | -e207 | 2 | 4 | 0 | not convex | 150280 | 149570 | $< 10$ | $< 10$ | 20 | 30 |
| | -e307 | 2 | 3 | 0 | not convex | 50 | 40 | $< 10$ | $< 10$ | 10 | $< 10$ |
| | -e3 | 1 | 2 | 0 | strictly convex | $< 10$ | $< 10$ | $< 10$ | $< 10$ | – | – |
| robot50 | e400 | 1 | 2 | 0 | not convex | $< 10$ | 10 | $< 10$ | $< 10$ | $< 10$ | $< 10$ |
| | -e400 | 1 | 2 | 0 | strictly convex | $< 10$ | $< 10$ | 10 | $< 10$ | – | – |
| rocket50 | e105 | 3 | 2 | 0 | not convex | 10 | 10 | $< 10$ | $< 10$ | $< 10$ | $< 10$ |
| | e154 | 6 | 3 | 1 | not convex | 750 | 760 | 10 | 10 | 500 | 480 |
| | e253 | 3 | 2 | 0 | not convex | 10 | 20 | $< 10$ | $< 10$ | 10 | $< 10$ |
| | e53 | 1 | 0 | 2 | not convex | 30 | 30 | $< 10$ | $< 10$ | $< 10$ | $< 10$ |
| | -e105 | 3 | 2 | 0 | not convex | 10 | 10 | $< 10$ | $< 10$ | 10 | $< 10$ |
| | -e154 | 6 | 3 | 1 | not convex | 720 | 760 | 10 | $< 10$ | 460 | 480 |
| | -e253 | 3 | 2 | 0 | not convex | 20 | 10 | $< 10$ | $< 10$ | $< 10$ | $< 10$ |
| | -e53 | 1 | 0 | 2 | strictly convex | 40 | 30 | $< 10$ | $< 10$ | – | – |
| **minlp.org [6]** | | | | | | | | | | | |
| minlp_org_34a | balrecs_i1_k2_t1_ | 3 | 2 | 1 | not convex | 20 | 20 | 10 | $< 10$ | 50 | 50 |
| | balrecs_i1_k2_t4_ | 4 | 2 | 1 | not convex | 50 | 50 | $< 10$ | $< 10$ | 1160 | 900 |
| | -balrecs_i1_k2_t1_ | 3 | 2 | 1 | not convex | 50 | 60 | $< 10$ | $< 10$ | 80 | 80 |
| | -balrecs_i1_k2_t4_ | 1 | 1 | 0 | unknown | $\perp$ | $\perp$ | $\perp$ | $\perp$ | $\perp$ | $\perp$ |
| | -balrecs_i1_k2_t5_ | 4 | 2 | 1 | not convex | 40 | 30 | $< 10$ | 10 | 790 | 640 |
| minlp_org_34b | defrate_i1_k2_ | 3 | 2 | 0 | not convex | $< 10$ | 10 | 10 | $< 10$ | $< 10$ | $< 10$ |
| | -defrate_i1_k2_ | 3 | 2 | 0 | not convex | 10 | 10 | 10 | $< 10$ | 10 | 10 |
| minlp_org_57 | defrx_3_1_ | 3 | 2 | 1 | not convex | 40 | 50 | 10 | $< 10$ | 700 | 720 |
| | -defrx_3_1_ | 3 | 2 | 1 | not convex | 40 | 30 | $< 10$ | $< 10$ | 360 | 360 |
| | -polyy2 | 6 | 3 | 0 | not convex | 860 | 810 | 10 | $< 10$ | $\perp$ | $\perp$ |
| | -polyy3 | 7 | 4 | 0 | unknown | $\perp$ | $\perp$ | 30 | 20 | $\perp$ | $\perp$ |
| minlp_org_71 | -obj | 611 | 3 | 1 | unknown | $\perp$ | $\perp$ | $\perp$ | $\perp$ | $\perp$ | $\perp$ |
| | -odec_20_3_5_ | 2 | 3 | 0 | not convex | 20 | 20 | $< 10$ | $< 10$ | 10 | 10 |
| | odec_20_3_5_ | 2 | 3 | 0 | not convex | 150 | 140 | $< 10$ | $< 10$ | 10 | 10 |
| **miscellaneous [2, 4, 5]** | | | | | | | | | | | |
| clay0203h | e107 | 3 | 3 | 1 | strictly convex | $\perp$ | $\perp$ | $< 10$ | 10 | – | – |
| | e110 | 3 | 3 | 1 | strictly convex | $\perp$ | $\perp$ | 70 | 170 | – | – |
| sssd-8-4-3 | e30 | 1 | 1 | 1 | strictly convex | 10 | 10 | $< 10$ | 10 | – | – |
| du-opt5 | e1 | 18 | 2 | 0 | strictly convex | $\perp$ | $\perp$ | 1480 | 180 | – | – |
| isotherms | isotherm | 2 | 2 | 2 | not convex | $\perp$ | $\perp$ | 620 | 670 | 2180 | 2320 |
| | -isotherm | 2 | 2 | 2 | not convex | $\perp$ | $\perp$ | 610 | 660 | 2210 | 2290 |