Convexity and Concavity Detection in Computational Graphs: Tree Walks for Convexity Assessment

Article *in* Informs Journal on Computing · February 2010

DOI: 10.1287/ijoc.1090.0321 · Source: DBLP

Technical Report

R. Fourer · C. Maheshwari · A. Neumaier · D. Orban* · H. Schichl

Convexity and Concavity Detection in Computational Graphs

Tree Walks for Convexity Assessment

December 18, 2008

Abstract. We examine symbolic tools associated with two modeling systems for mathematical programming, which can be used to automatically detect the presence or absence of convexity and concavity in the objective and constraint functions, as well as convexity of the feasible set in some cases. The COCONUT solver system Schichl (2004b) focuses on nonlinear global continuous optimization and possesses its own modeling language and data structures. The DR.AMPL meta-solver (Fourer and Orban, 2007) aims to analyze nonlinear differentiable optimization models and hooks into the AMPL Solver Library (Gay, 2002). Our symbolic convexity analysis may be supplemented, when it returns inconclusive results, with a numerical phase that may detect non-convexity. We report numerical results using these tools on sets of test problems for both global and local optimization.

1. Introduction

We consider continuous deterministic optimization problems of the form

$$\begin{array}{ll} \underset{x\in\mathbb{R}^n}{\text{minimize}} & f(x) \\ \text{subject to} & c_{\mathcal{E}}(x) = 0 \\ & c_{\mathcal{I}}(x) \geq 0, \end{array} \tag{1.1}$$

and are interested in two cases. In the first, the structure of the problem is such that finding a global minimizer is tractable. In the second, the objective f and constraint functions $c_{\mathcal{E}}$ and $c_{\mathcal{I}}$ are smooth and the problem is either so large or so nonlinear that we are content with local solutions. We are also interested in determining what information can be available on the problem based on its representation in a modeling language such as AMPL (Fourer et al., 2002).

During the solution process, it is useful to have as much information as possible on the structure of the objective and constraint functions. Convexity and concavity, if present, have important consequences for the structure of the problem and the nature of solutions. The Karush-John first order optimality conditions simplify (Neumaier and Schichl, 2003) if some constraint functions are known to be convex or concave. If the objective function is convex, all equality constraints are linear and all inequalities are concave, every local minimum is also global. Convexity of the objective or constraint functions may influence the core linear algebra techniques used to solve subproblems at each iteration.

R. Fourer: Northwestern University, Department of Industrial Engineering and the Management Sciences, Evanston IL, USA. E-mail: 4er@iems.northwestern.edu

C. Maheshwari: India Institute of Technology, Department of Mechanical Engineering, Guwahati, India. E-mail: chandrakant721@yahoo.com

A. Neumaier: University of Vienna, Department of Mathematics, Vienna, Austria. E-mail: Arnold.Neumaier@univie.ac.at

D. Orban: GERAD and École Polytechnique de Montréal, Département de Mathématiques et Génie Industriel, Montréal (Québec), Canada. E-mail: Dominique.Orban@gerad.ca

H. Schichl: University of Vienna, Department of Mathematics, Vienna, Austria. E-mail: hermann.schichl@esi.ac.at

* Research partially supported by NSERC Discovery Grant 299010-04

As a convention, we use the terminology *constraint function* to denote one of the functions c_i appearing in the constraints of (1.1) for $i \in \mathcal{E} \cup \mathcal{I}$. In contrast, a *constraint* is a condition such as $c_i(x) = 0$ or $c_i(x) \geq 0$ defining a subset of $\mathbb{R}^n$. In the present context, we will use two interpretations of the term *convexity*. In the first, we are concerned with the convexity of the constraint function c_i. In the second, we examine the convexity of the set $\{x \in \mathbb{R}^n \mid c_i(x) = 0\}$ or $\{x \in \mathbb{R}^n \mid c_i(x) \geq 0\}$. The latter influences the convexity of the whole feasible set, which is an intersection of such subsets. The former may influence the linear algebra used in the course of a numerical method to solve (1.1). For instance, active-set methods will only consider a subset of the constraints at a time and will typically require the solution of a system of linear equations with the Hessian of a Lagrangian as coefficient matrix. If all constraint functions taking part in this Lagrangian are convex functions—irrespective of whether or not the conditions imposed by means of those constraint functions define convex subsets of $\mathbb{R}^n$—, tailored solution methods may be employed.

This report describes the convexity and concavity detection methods implemented in the inference engines `simple_convexity` of the COCONUT solver system (Schichl, 2004b) and in the DR.AMPL meta-solver (Fourer and Orban, 2007).

The COCONUT environment (Schichl, 2004b,a) is an open source modular environment for global optimization aimed at the integration of the existing approaches to continuous global optimization and constraint satisfaction. The application programmer's interface (API) is designed to make the development of the various module types independent of each other and independent of the internal model representation. It is a collection of `C++` classes protected by the LGPL license model, so that it could be used as part of commercial software. It also possesses an interface to the AMPL and GAMS modeling languages. The `simple_convexity` module is one of its inference modules, implementing some of the methods described in this article. One of the small supplementary programs, `analyze_convexity`, can be used for convexity analysis of optimization and constraint satisfaction problems.

DR.AMPL (Fourer and Orban, 2007) is an optimization problem analysis layer that builds upon the tools offered by the AMPL Solver Library (Gay, 2002) to provide tools for convexity assessment but also for problem classification and simplification. In particular, it offers some level of nonlinear preprocessing that goes beyond the default AMPL presolve. By categorizing problems, it is able to offer solver recommendation to the user. DR.AMPL is licensed under the LGPL. Its convexity assessment is twofold. First, a symbolic proving analysis is performed. If the latter is inconclusive, a numerical disproving analysis is performed. We present both in great detail in the next sections.

Detection methods naturally have their limitations, which we will discuss, but the implementations described below are flexible enough to be expanded as experience grows.

The application of symbolic mathematics to the detection of properties of optimization problems has been described by Stoutmeyer (1978) and one of those properties is convexity. Other studies are under way concerning automatic convexity and concavity detection. Nenov et al. (2004) assess convexity based on the notion of the *Hessian sign*. The approach of Grant et al. (2006, 2008); Grant and Boyd (2008); Mattingley and Boyd (2008) is more constructive and describes a framework to *build* convex functions and *disciplined* convex problems. Chinneck (2001) approaches the problem with a heuristic point of view and attempts to disprove convexity.

This paper is organized as follows. Section 2 recalls the definition and usage of a directed acyclic graph when processing nonlinear expressions. Section 3 summarizes general rules allowing the convexity or concavity of the composition of functions to be inferred. Section 4 applies these rules to the many operators found in modeling languages for nonlinear programming while §5 examines numerical methods to disprove convexity. Preliminary numerical experience is reported in §7 and some concluding remarks are given in §8.

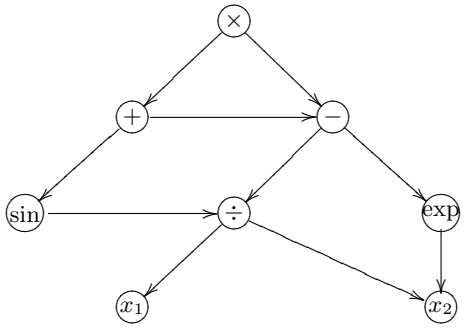

Fig. 2.1. A DAG representing the expression (2.1). A directed edge from a node to another indicates that the first node is an operator and the second is one of the operands. This graph has 3 common expressions: x_2, x_1/x_2 and $x_1/x_2 - e^{x_2}$. They are given by the nodes with more than one incident arc.

2. The Directed Acyclic Graph

The directed acyclic graph, or DAG, is a recursive data structure fit to hold nonlinear expressions in such a way that evaluation and computation of partial derivatives by means of automatic differentiation is efficient. Essentially, the DAG for an expression is the Kantorovich graph for that expression (Kantorovich, 1957), in which the repeated occurence of a sub-expression is fully exploited. In this context, it is also referred to as a *computational graph* (Bauer, 1974). As a simple example, borrowed from Griewank (2000), consider the nonlinear expression of two variables

$$f(x_1, x_2) = (\sin(x_1/x_2) + x_1/x_2 - e^{x_2})\,(x_1/x_2 - e^{x_2})\,. \tag{2.1}$$

The DAG for (2.1) can be represented as in Fig. 2.1 where reuse of common sub-expressions is apparent.

As we see from Fig. 2.1, variables and constants are leaves of the DAG. All other nodes are operators, such as the addition, division, multiplication or exponentiation. The terminology *walking* the DAG refers to starting from the root and visiting each node in turn, in a prescribed order. As we show in the next sections, convexity of a nonlinear expression such as (2.1) may be assessed by walking the DAG for the expression. We deliberately use the terminology *convexity assessment* to denote processes that attempt to prove and/or disprove convexity or concavity properties of a function. As already mentioned, convexity assessment tools have their limitations. In the sequel, we present two different types of convexity assessment tools that are applied together to ensure as conclusive results as possible.

By storing common expression only once, the DAG is advantageously used by AMPL to store and evaluate but also differentiate nonlinear expressions by means of automatic differentiation. The latter makes extensive use of DAG walks and offers the possibility of evaluating gradients and Hessian-vector products at a small multiple of the cost of evaluating the function value Griewank (2000). The open source AMPL Library (Gay, 2002) gives access to the various DAGS associated to a problem and is available from `www.netlib.org/ampl/solvers` or `netlib.sandia.gov/ampl/solvers`. It is an essential toolbox for the convexity assessment techniques described below. For instance, it also offers the possibility to compute full Hessians, sparse or dense, in addition to Hessian-vector products, albeit with more effort. To this end, it takes advantage of the group partially-separable structure of the problem functions.

3. Convexity Detection

In this section, we review basic rules constituting sufficient conditions for the convexity or concavity of the composition of two functions and how this is related to the internal representation of (1.1). By

their recursive nature, these rules readily extend to the convexity or concavity of the composition of an arbitrary number of functions.

Both in the COCONUT system and in the AMPL modeling language, every nonlinear function playing a role in (1.1) is represented as a directed acyclic graph (DAG) (Fourer and Orban, 2007; Gay, 2002; Schichl and Neumaier, 2003). A fundamental tool towards convexity detection is bound propagation: a technique by which it is possible to infer an overestimate of the range of a nonlinear function given bounds on the variables on which it depends (Griewank, 2000; Fourer and Orban, 2007). Bound propagation and the recursive rules described below for convexity detection are naturally related to interval arithmetic and constraint programming.

For illustration purposes, we consider here a simple example and refer the interested reader to (Fourer and Orban, 2007) for further information. In an AMPL model, bounds on the variables $-\infty \leq x_i^{\mathrm{L}} \leq x_i \leq x_i^{\mathrm{U}} \leq +\infty$, $i = 1, \ldots, n$, are specified and stored separately from other constraints. It is not difficult to walk the DAG of an expression and propagate those bounds to obtain an estimate of the range of the whole expression. To simplify, assume there is a single variable x and the explicit bounds $-6 \leq x \leq 8$ are specified. Consider now the expression $f(x) = \exp(\sin(1/x))$. The bounds on x are first propagated into the expression $1/x$ to produce $-\infty \leq 1/x \leq \infty$. Next, these last bounds are introduced into $\sin(1/x)$ to yield $-1 \leq \sin(1/x) \leq 1$ and finally into the exponential, to give $0.367879 \leq f(x) \leq 2.718282$. Since the exponential is nondecreasing, the lower bound is e^{-1} while the upper bound is e. This very simple example illustrates the need for an additional tool in bound propagation: a monotonicity assessment tool. In turn, assessing monotonicity may require bound propagation. In essence, constants can be considered as both nondecreasing and nonincreasing. Variables are nondecreasing expressions. We can now build upong these two base cases and reason recursively. For instance, the sum of two nondecreasing functions is nondecreasing. The sine of an expression f is nondecreasing if either

(a) f is nondecreasing and $\cos(f) \geq 0$, or
(b) f is nonincreasing and $\cos(f) \leq 0$,

and so forth. All operators supported by a given modeling language may be covered in this way. Both the bound propagation and monotonicity assessment are features of DR.AMPL and their design is covered in (Fourer and Orban, 2007).

In what follows, for any given (possibly nonlinear) function $f : \mathbb{R}^n \to \mathbb{R}$, we assume that we have a means to infer a range $[\underline{f}, \overline{f}]$ such that for all $x \in \mathrm{dom} f$, $\underline{f} \leq f(x) \leq \overline{f}$, where $-\infty \leq \underline{f} \leq \overline{f} \leq +\infty$. A generic pseudo-code sample to parse a DAG could be represented as in Fig. 3.1, which serves as a template for the implementation of a convexity-proving engine. In this generic pseudo-code, we assumed that each node has two children for simplicity of exposition. In practice, some operators are unary while others have more than two children. For example, division has two children, the absolute value has a single child and the minimum of a number of expressions has a variable number of children.

The convexity detection procedure is to descend through all nodes of the DAG in forward mode after bound propagation. The ranges of the intermediate nodes are used to aid convexity detection. We distinguish the properties *linear*, *convex*, *concave*, and *unknown convexity status*. Note that a linear function is both convex and concave.

Every node is considered as a function taking input from its child nodes, and we distinguish whether the elementary function represented by the node on its range has some convexity property, and whether its arguments have certain convexity properties. The recursive argument relies on the fact that leaves of the DAG can only be constants or variables, which are linear expressions and thus both convex and concave. Based on this knowledge, the following rules help decide whether the composite function $f \circ g$ possesses convexity properties (Boyd and Vandenberghe, 2004).

Rule I. If the function f defined by the current node depends on one argument only and is convex on the range of its argument then $f \circ g$ is convex if either

```
int OperatorProcessor( node ) {

   switch( node->operator ) {
       case leaf:                    /* Variable or constant */
           return appropriate_value;

       case other;
           OperatorProcessor( node->left  );  /* left  child */
           OperatorProcessor( node->right );  /* right child */
           process( node->operator );
   }
}
```

Fig. 3.1. Generic pseudo-code for parsing a DAG, assuming two children per node.

(a) the child function g is linear,
(b) the child function g is convex and f is nondecreasing over $[\underline{g}, \overline{g}]$,
(c) the child function g is concave and f is nonincreasing over $[\underline{g}, \overline{g}]$,

Rule II. If the function f defined by the current node depends on one argument only and is concave on the range of its argument then $f \circ g$ is concave if either
(a) the child function g is linear,
(b) the child function g is convex and f is nonincreasing over $[\underline{g}, \overline{g}]$,
(c) the child function g is concave and f is nondecreasing over $[\underline{g}, \overline{g}]$,

Similar rules can be stated for multivariate functions $f(g_1(x), \ldots, g_m(x))$. In this case, we denote $[\underline{g}, \overline{g}]$ the domain $[\underline{g_1}, \overline{g_1}] \times \ldots \times [\underline{g_m}, \overline{g_m}] \subseteq \mathbb{R}^m$.

Rule III. If the function f defined by the current node depends on several arguments $g_1, \ldots, g_m$ and is convex over $[\underline{g}, \overline{g}]$ then $f \circ g$ is convex if for all $i = 1, \ldots, m$ either
(a) the child function g_i is linear,
(b) f is nondecreasing in its i-th argument over $[\underline{g_i}, \overline{g_i}]$ and the child function g_i is convex,
(c) f is nonincreasing in its i-th argument over $[\underline{g_i}, \overline{g_i}]$ and the child function g_i is concave.

Rule IV. If the function f defined by the current node depends on several arguments $g_1, \ldots, g_m$ and is concave over $[\underline{g}, \overline{g}]$ then $f \circ g$ is concave if for all $i = 1, \ldots, m$ either
(a) the child function g_i is linear,
(b) f is nondecreasing in its i-th argument over $[\underline{g_i}, \overline{g_i}]$ and the child function g_i is concave,
(c) f is nonincreasing in its i-th argument over $[\underline{g_i}, \overline{g_i}]$ and the child function g_i is convex.

Using the prototype in Fig. 3.1, it becomes possible to infer the required monotonicity properties (Fourer and Orban, 2007).

4. Convexity Information for Elementary Functions

In the following we give a list of the elementary functions that the algorithm can currently exploit. Most of the rules below are inferred from Rule I–Rule IV or (Boyd and Vandenberghe, 2004). Unless stated in the following rules, convexity and concavity are inconclusive. We use the notation $f > 0$, $f < 0$ and $f \neq 0$ where f is a function as a shorthand for $f(x) > 0$, $f(x) < 0$ and $f(x) \neq 0$ for all $x \in \mathrm{dom} f$ respectively. We first cover basic operators.

Unary minus. $-f$ is convex if and only if f is concave. It is concave if and only if f is convex.

Sum. A sum $f(x) = \sum_{i=1}^{n} f_i(x)$ of convex functions is convex. If all the children f_i are convex then f is convex. Because of the previous property, if all the children f_i are concave, then f is concave.

Product. If g is a constant function, fg is convex if and only if [f is convex and $g \geq 0$] or [f is concave and $g \leq 0$]. It is concave if and only if [f is concave and $g \geq 0$] or [f is convex and $g \leq 0$]. If g is not constant, the product need not be convex or concave.

Quotient. If g is a constant function, f/g is convex if and only if [f is convex and $g > 0$] or [f is concave and $g < 0$]. It is concave if and only if [f is convex and $g < 0$] or [f is concave and $g > 0$]. If f is constant and $g \neq 0$ is non-constant, then f/g is convex if [$f \geq 0$ and $g > 0$ is concave] or [$f \leq 0$ and $g < 0$ is convex]. It is concave if [$f \geq 0$ and $g < 0$ is convex] or [$f \leq 0$ and $g > 0$ is concave].

We now examine all functions which can be represented by nodes of the DAG in our implementations.

Minimum. The minimum of a finite number of concave functions is concave.

$$f(x) = \min\{c, f_1(x), f_2(x), \ldots, f_n(x)\},$$

where c is a constant. The function f is concave if either all f_i, $i = 1, \ldots, n$ are concave, or if $\underline{f_i} \geq \overline{f}$ for each nonconcave f_i. Otherwise, nothing can be concluded.

Maximum. The maximum of a finite number of convex functions is convex.

$$f(x) = \max\{c, f_1(x), f_2(x), \ldots, f_n(x)\},$$

where c is a constant. The function f is convex if either all f_i, $i = 1, \ldots, n$ are convex, or if $\overline{f}_i \leq \underline{f}$ for each nonconvex f_i. Otherwise, nothing can be concluded.

Absolute value, Square, Hyperbolic cosine. $f(x) = |x|$, $f(x) = x^2$ and $f(x) = \cosh(x)$ are a convex functions, decreasing on $\mathbb{R}^-$ and increasing on $\mathbb{R}^+$. Hence, $f \circ g$ is
(a) convex if either g is linear, g is convex and $\underline{g} \geq 0$, or g is concave and $\overline{g} \leq 0$.
(b) concave if either g is convex and $\overline{g} \leq 0$, or g is concave and $\underline{g} \geq 0$.

Square root. $f(x) = \sqrt{x}$, $x \geq 0$ is concave. The function $f \circ g$ is concave if g is concave. Note that $\underline{g} \geq 0$ must hold. Moreover, $f \circ g$ is convex if g is convex and quadratic.

Exponential. $f(x) = e^x$ is convex increasing. The function $f \circ g$ is convex if g is convex.

Gauss. $f(x) = e^{-(x-m)^2/s^2}$ with $s > 0$ is concave if $x \in [m - s/\sqrt{2}, m + s/\sqrt{2}]$ and convex otherwise. Moreover, f is increasing for $x \leq m$ and decreasing for $x \geq m$. Therefore $f \circ g$ is convex if either
(a) g is concave and $\underline{g} \geq m + s/\sqrt{2}$,
(b) g is convex and $\overline{g} \leq m - s/\sqrt{2}$.
Similarly, $f \circ g$ is concave if $[\underline{g}, \overline{g}] \subseteq [m - s/\sqrt{2}, m + s/\sqrt{2}]$ and either
(a) g is linear,
(b) g is convex and $\underline{g} \geq m$,
(c) g is concave and $\overline{g} \leq m$.

Logarithm. $f(x) = \log(x)$, $x > 0$ is concave increasing. Therefore, $f \circ g$ is concave if g is concave. Note that $\underline{g} > 0$ must hold.

Sine. $f(x) = \sin(x)$. The function $f \circ g$ is neither convex nor concave if either $\overline{g} - \underline{g} > \pi$ or $\sin(\underline{g}) \sin(\overline{g}) < 0$. If $\overline{g} - \underline{g} \leq \pi$, $\sin(\underline{g}) \geq 0$ and $\sin(\overline{g}) \geq 0$, then $f \circ g$ is concave if either
(a) g is linear,
(b) g is convex and $\cos(g) \leq 0$,
(c) g is concave and $\cos(g) \geq 0$.
If $\overline{g} - \underline{g} \leq \pi$, $\sin(\underline{g}) \leq 0$ and $\sin(\overline{g}) \leq 0$, $f \circ g$ is convex if either
(a) g is linear,
(b) g is concave and $\cos(g) \leq 0$,
(c) g is convex and $\cos(g) \geq 0$.

Cosine. $f(x) = \cos(x)$. The function $f \circ g$ is neither convex nor concave if either $\overline{g} - \underline{g} > \pi$ or $\cos(\underline{g})\cos(\overline{g}) < 0$. If $\overline{g} - \underline{g} \leq \pi$, $\cos(\underline{g}) \geq 0$ and $\cos(\overline{g}) \geq 0$, $f \circ g$ is concave if either
(a) g is linear,
(b) g is convex and $\sin(g) \leq 0$,
(c) g is concave and $\sin(g) \geq 0$.
If $\overline{g} - \underline{g} \leq \pi$, $\cos(\underline{g}) \leq 0$ and $\cos(\overline{g}) \leq 0$, $f \circ g$ is convex if either
(a) g is linear,
(b) g is concave and $\sin(g) \leq 0$,
(c) g is convex and $\sin(g) \geq 0$.

Tangent. $f(x) = \tan(x)$ is concave increasing on every interval of the form $(-\pi/2 + k\pi, k\pi]$ and convex increasing on every interval of the form $[k\pi, \pi/2 + k\pi)$, $k \in \mathbb{Z}$. Thus $f \circ g$ is nonconvex and nonconcave if either $\overline{g} - \underline{g} > \pi/2$ or $\tan(\underline{g})\tan(\overline{g}) < 0$. If $\overline{g} - \underline{g} \leq \pi/2$, $f \circ g$ is
(a) convex if $\tan(\underline{g}) \geq 0$, $\tan(\overline{g}) \geq 0$ and g is convex,
(b) concave if $\tan(\underline{g}) \leq 0$, $\tan(\overline{g}) \leq 0$ and g is concave.

Arc sine. $f(x) = \operatorname{asin}(x)$ is concave increasing on $[-1, 0]$, concave increasing on $[0, 1]$ and undefined outside $[-1, 1]$. Thus $f \circ g$ is
(a) convex if g is convex and $0 \leq \underline{g} \leq \overline{g} \leq 1$,
(b) concave if g is concave and $-1 \leq \underline{g} \leq \overline{g} \leq 0$.

Arc cosine. $f(x) = \operatorname{acos}(x)$ is convex decreasing on $[-1, 0]$, concave decreasing on $[0, 1]$ and undefined outside $[-1, 1]$. Thus $f \circ g$ is
(a) convex if g is concave and $-1 \leq \underline{g} \leq \overline{g} \leq 0$,
(b) concave if g is convex and $0 \leq \underline{g} \leq \overline{g} \leq 1$.

Arc tangent. $f(x) = \operatorname{atan}(x)$ is convex increasing on $\mathbb{R}^-$ and concave increasing on $\mathbb{R}^+$. Thus $f \circ g$ is
(a) convex if g is convex and $\overline{g} \leq 0$,
(b) concave if g is concave and $\underline{g} \geq 0$.

Arc tangent of a quotient. $f(x, y) = \operatorname{atan}(y/x)$ is declared inconclusive for now.

Hyperbolic sine. The same rules as for the odd positive power apply.

Hyperbolic tangent. $f(x) = \tanh(x)$ is convex increasing on $\mathbb{R}^-$ and concave increasing on $\mathbb{R}^+$. Thus, $f \circ g$ is
(a) convex if g is convex and $\overline{g} \leq 0$,
(b) concave if g is concave and $\underline{g} \geq 0$.

Inverse hyperbolic sine. $f(x) = \operatorname{asinh}(x)$ is convex increasing on $\mathbb{R}^-$ and concave increasing on $\mathbb{R}^+$. Thus, $f \circ g$ is
(a) convex if g is convex and $\overline{g} \leq 0$.
(b) concave g is concave and $\underline{g} \geq 0$.

Inverse hyperbolic cosine. $f(x) = \operatorname{acosh}(x)$ is concave increasing for $x \geq 1$. Thus $f \circ g$ is concave if g is concave. Note that $\underline{g} \geq 1$ must hold.

Inverse hyperbolic tangent. $f(x) = \operatorname{atanh}(x)$ is concave increasing on $(-1, 0]$ and convex increasing on $[0, 1)$. It is undefined outside $(-1, 1)$. Thus $f \circ g$ is
(a) convex if g is convex and $0 \leq \underline{g} \leq \overline{g} < 1$,
(b) concave if g is concave and $-1 < \underline{g} \leq \overline{g} \leq 0$.

Even positive power. $f(x) = x^{2k}$, $k \in \mathbb{N}$, is
(a) constant if $k = 0$,
(b) convex everywhere, decreasing on $\mathbb{R}^-$ and increasing on $\mathbb{R}^+$.

We eliminate the case $k = 0$ for then $f \circ g$ is constant. Hence $f \circ g$ is convex if g is linear, or g is convex and $\underline{g} \geq 0$, or g is concave and $\overline{g} \leq 0$,

Even negative power. $f(x) = x^{-2k}$, $k \in \mathbb{N}_0$, is convex increasing on $\mathbb{R}^-$ and convex decreasing on $\mathbb{R}^+$. Hence $f \circ g$ is
(a) convex if g is convex and $\overline{g} \leq 0$, or g is concave and $\underline{g} \geq 0$.
(b) concave if g is convex and $\underline{g} \geq 0$, or g is concave and $\overline{g} \leq 0$.

Odd positive power. $f(x) = x^{2k+1}$, $k \in \mathbb{N}$, is
(a) linear if $k = 0$,
(b) concave increasing on $\mathbb{R}^-$ and convex increasing on $\mathbb{R}^+$ if $k > 0$.
If $k = 0$, $f \circ g = g$ has the convexity properties of g. If $k > 0$, $f \circ g$ is
(a) convex if g is convex and $\underline{g} \geq 0$,
(b) concave if g is concave and $\overline{g} \leq 0$.

Odd negative power. $f(x) = x^{-2k-1}$, $k \in \mathbb{N}$, is concave decreasing on $\mathbb{R}^-$ and convex decreasing on $\mathbb{R}^+$. Hence, $f \circ g$ is
(a) convex if g is concave and $\underline{g} \geq 0$,
(b) concave if g is convex and $\overline{g} \leq 0$.

Power 1. $f(x) = \alpha^x$, $\alpha > 0$. For $0 < \alpha < 1$, $f \circ g$ is convex if g is concave. For $\alpha \geq 1$, $f \circ g$ is convex if g is convex.

Nonintegral power. $f(x) = x^\alpha$, $\alpha \in \mathbb{R}\backslash\mathbb{Z}$, is undefined if $x < 0$. Whenever $\underline{g} \geq 0$, $f \circ g$ is convex if either
(a) $\alpha > 1$ and g is convex,
(b) $\alpha < 0$ and g is concave,
and $f \circ g$ is concave if either
(a) $0 < \alpha < 1$ and g is concave,
(b) $\alpha < 0$ and g is convex.

Power 3. If either f or g is constant, f^g reduces to one of the previous cases. The other cases are treated as inconclusive for now.

5. Convexity Disproving

If the systematic rules of §4 happen to fail for the function under consideration, convexity is inconclusive. This does not mean that the function is nonconvex, nor does it mean that it is concave. On the other hand, we might try, by other means, to *disprove* convexity. This has been done for AMPL models in the mProbe (Chinneck, 2001) software, by sampling function values on feasible line segments and checking that the definition of convexity is satisfied. In Dr.AMPL it is possible to arrange for a *convexity disprover* to be called on all those problem functions for which the symbolic phase returned an inconclusive result. The convexity disprover is a user-supplied module with a prespecified prototype required to return the value 0 in case of success and a positive value in case of failure in disproving local convexity. For instance, the user might write a function similar to that used by mProbe and plug it into Dr.AMPL. Because the objective and constraint functions are typically more complicated than quadratic functions, a possibility is to disprove convexity for given values of the variables. For instance, let ψ denote a problem function. Since we are assuming that ψ is twice continuously differentiable, we have access to the Hessian matrix $\nabla^2\psi(x)$ for a given value of x, e.g., $x = 0$, the starting point specified by the user, or any other value in the domain of ψ. The function ψ is convex at this particular x if and only if $\nabla^2\psi(x)$ is positive semi-definite. Even though this latter property is intricate to assess numerically, we may attempt to verify whether or not it fails to hold. If it does fail to hold, we have a proof—subject to rounding errors—that ψ is not convex at x, and therefore that ψ is not convex. In order to see whether $\nabla^2\psi(x)$ is positive semi-definite or not, we might attempt to compute its Cholesky

factorization. If the factorization exists, the matrix is positive definite and ψ is convex at x. However, the cost of the numerical method employed to attempt to disprove convexity should not be substantial compared to the cost of solving the problem.

By default, DR.AMPL supplies a convexity disprover that follows a similar yet inherently different numerical approach by attempting to exhibit a direction of negative curvature, i.e., a direction $d \in \mathbb{R}^n$ such that $d^T \nabla^2 \psi(x) d < 0$. To that end, it formulates the quadratic optimization problem with trust-region constraint

$$\begin{array}{ll} \underset{d \in \mathbb{R}^n}{\text{minimize}} & g^T d + \frac{1}{2} d^T \nabla^2 \psi(x) d \\ \text{subject to} & \|d\|_2^2 \leq \Delta, \end{array} \tag{5.1}$$

where $\| \cdot \|_2$ and $\Delta > 0$ are the Euclidean norm and a trust-region radius respectively. The vector g may be chosen as $\nabla \psi(x)$, or another appropriate vector as we discuss below. This quadratic program is in turn approximately solved by means of the truncated conjugate gradient of Steihaug (1983) and Toint (1981), often used in nonlinear optimization implementations. On large problems, solving (5.1) is typically faster than computing a sparse Cholesky factorization but the latter may be favored on small problems since then, there is no reason to truncate the conjugate gradient iterations. Alternatively, one might elect to choose $\Delta = +\infty$.

An improved convexity disprover is available if the user chooses to use the Generalized Trust-Region Lanczos method GLTR (Gould et al., 1999), now part of the GALAHAD optimization library (Gould et al., 2003a). This algorithm may be set to stop once it hits the trust-region boundary, in which case it reduces to the standard disprover, or to further explore the boundary in order to improve on the current iterate. At a potentially slightly higher computational cost, GLTR may return more accurate results.

In both the truncated conjugate gradient and the GLTR, numerical errors are accounted for by declaring that d is a direction of negative curvature as soon as $d^T \nabla^2 \psi(x) d \leq -100\epsilon$ where ϵ is the machine epsilon—typically, $\epsilon \approx 2 \cdot 10^{-16}$ on architectures implementing the IEEE double precision standard. In both algorithms, the role of the trust-region is to bound the numerical effort invested into disproving convexity. Its radius is chosen as

$$\Delta = \max(10, \|\nabla \psi(x)\|_2^2 / 10).$$

The point x at which convexity is thus assessed is chosen to be feasible with respect to the bound constraints of problem (1.1), if any. Note that bound constraints may be present because no presolve was performed and the original problem had bound constraints, because presolve did not eliminate some of them, or because bounds on the variables appeared as a result of presolving other expressions in the problem. Note also that DR.AMPL attempts to further tighten bounds or simplify them (Fourer and Orban, 2007). It is the latter tighter bounds that are referred to in this section.

For comparison purposes, it should be kept in mind that in most cases, the conjugate gradient approach, the truncated Lanczos approach and even the Cholesky factorization approach have minimal cost compared to that of solving the problem with a given optimization method. The reason for this is that in the course of the iterations of an optimization method to minimize the fuction $\psi(x)$ unconstrained, a single step would be typically computed by solving (5.1) or by attempting a factorization of $\nabla^2 \psi(x)$ and performing backsolves to obtain the solution to a linear system of equations. Many such steps are typically required to solve the problem.

Two outcomes may occur in the solution of (5.1). If, while following the conjugate-gradient or Lanczos path, the boundary of the trust region is met without having found any direction of negative curvature, disproof of convexity is declared inconclusive. If a direction of negative curvature is encountered in the process, this direction is followed until the boundary of the trust region is met. At that point, the minimization ends and the disprover reports that negative curvature was found. This disproves positive semi-definiteness of the Hessian matrix $\nabla^2 \psi(x)$, convexity of ψ about x, and thus convexity of ψ. Of course, this numerical process is local in the sense that only convexity of ψ about x is assessed.

However, the function, if it is not quadratic, might very well be convex around x and nonconvex in other feasible regions. To circumvent this additional difficulty, a number of different regions are chosen by selecting a number of points $x_1, \ldots, x_p$ satisfying the bound constraints and repeating the above conjugate gradient or Lanczos minimization for each of them. In later releases of DR.AMPL, feasibility of these points with respect to other constraints will be able to be enforced as well.

An appropriate choice of the vector g in (5.1) may be important. Indeed, the natural choice $g = \nabla\psi(x)$ often leads to a minimization ending in a single iteration because $\|g\|$ is too large and the first direction taken by both algorithms is $d_0 = -g$. To circumvent this difficulty and give the disprover more of a chance to identify negative curvature, a random vector g may be chosen.

Proving and disproving convexity are tools of different nature which may work jointly to ensure that as few cases as possible are missed, while keeping the computational load moderate in the disproving phase. In the case where convexity is deemed to hold in the symbolic phase, we can be sure that the objective function is convex. Another possible outcome is where convexity cannot be proved symbolically, but the numerical procedure succeeds in disproving it. In this case, we know for a fact that the function is nonconvex. A final possibility, where convexity could neither be proved symbolically, nor disproved numerically is a totally inconclusive case. The latter occurs on problems which are either convex but not encompassed by the rules of §4, or on nonconvex problems for which a region of nonconvexity has not been identified.

Because of the uncertainty inherent to finite-precision arithmetic, we believe that such numerical procedures should not be used for convexity *proving*. Indeed, because of roundoff errors, the Cholesky factorization of a positive definite matrix may fail if the smallest eigenvalue is sufficiently small. Similarly, GLTR computes the curvature along a direction by evaluating dot products of the form $p^T\nabla^2\psi(x)p$ for some vectors p. Because of cancellation, even if $\nabla^2\psi(x)$ is positive definite, this dot product may evaluate to some positive number that rounds down to zero or even to a negative number. This uncertainty makes it difficult to differentiate between a curvature value which is actually negative and a "false negative."

It goes without saying that the procedures outlined in this section can equally be applied for disproving concavity. This is achieved by changing the sign of the function ψ everywhere.

Before presenting more extensive numerical results in §7, let us take a brief look at a sample output from DR.AMPL on problem `elec` from the COPS collection (Dolan et al., 2004). The problem, which consists in arranging a given number of electrons on a sphere so as to minimize the total Coulomb potential between the charged particles, is stated as

$$\begin{array}{ll} \underset{x,y,z}{\text{minimize}} & \displaystyle\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}\left[(x_i-x_j)^2+(y_i-y_j)^2+(z_i-z_j)^2\right]^{-1/2} \\ \text{subject to} & x_i^2+y_i^2+z_i^2=r^2, \qquad i=1,\ldots,n, \end{array}$$

where $n \in \mathbb{N}_0$ is the number of electrons and $r > 0$ is the radius of the sphere. The objective function of this problem is not convex and each constraint function is clearly convex. Note that by this, we do not mean that they define a convex feasible set. The symbolic convexity-proving phase of DR.AMPL on this problem with $n = 100$ produces the output of Fig. 5.1. In this output, we first see that the objective is being minimized and there are no objectives being maximized. DR.AMPL was not able to prove convexity of this objective function. In the second part of the report, all constraint functions were proved convex. Since they all are equality constraints, we still expect the feasible set to be nonconvex.

The numerical convexity-disproving phase produces the output show in Fig. 5.2. In this output, the term *nonconvex* is understood as both nonconvex and nonconcave. The first part of the report shows that both convexity and concavity of the objective function were successfully disproved. The second part shows that all constraint functions were rightfully claimed as being convex, since convexity disproving failed.

```
Symbolic Convexity Proving of Objectives and Constraints

 Nonlinear objective 'coulomb_potential' has not been proved convex or concave over the bound constraints.
 Detected     0/1      nonlinear convex  objectives being minimized,
              0/0      nonlinear convex  objectives being maximized,
              0/1      nonlinear concave  objectives being minimized,
              0/0      nonlinear concave  objectives being maximized,

 Detected     0/0      nonlinear convex  inequality constraint functions,
            100/100    nonlinear convex    equality constraint functions,
              0/0      nonlinear concave inequality constraint functions,
              0/100    nonlinear concave   equality constraint functions.
```

Fig. 5.1. Symbolic convexity-proving phase on problem `elec`.

```
Numerical Convexity Disproving of Objectives and Constraints

 Attempting to disprove convexity of objectives
  Processing objective 'coulomb_potential'
   Convexity was disproved numerically
   Concavity was disproved numerically
 In hindsight,
 Detected     0/1      nonlinear convex       objectives being minimized,
              0/0      nonlinear convex       objectives being maximized,
              0/1      nonlinear concave      objectives being minimized,
              0/0      nonlinear concave      objectives being maximized,
              1/1      nonlinear nonconvex    objectives being minimized,
              0/0      nonlinear nonconvex    objectives being maximized,
              0/1      nonlinear inconclusive objectives being minimized,
              0/0      nonlinear inconclusive objectives being maximized,
              0/1      nonlinear objectives misclassified (0.0 %).

 Attempting to disprove convexity of constraints
 In hindsight,
 Detected     0/0      nonlinear convex       inequality constraint functions,
            100/100    nonlinear convex         equality constraint functions,
              0/0      nonlinear concave      inequality constraint functions,
              0/100    nonlinear concave        equality constraint functions.
              0/0      nonlinear nonconvex    inequality constraint functions.
              0/100    nonlinear nonconvex      equality constraint functions.
              0/0      nonlinear inconclusive inequality constraint functions.
              0/100    nonlinear inconclusive   equality constraint functions.
              0/100    nonlinear constraints misclassified (0.0 %).
```

Fig. 5.2. Numerical convexity-disproving phase on problem `elec`.

Note that any convexity disprover based on the Hessian matrix $\nabla^2\psi(x)$ implicitly assumes that ψ is twice continuously differentiable. This poses a difficulty for constraint functions of the form $c(x,t) = \|x\| - t$, where $\|\cdot\|$ is the Euclidian norm. Such functions are convex, and correctly identified as such by the rules on the square root of §4. Constraint of the form $\|Ax - b\| - t \leq 0$ are typically referred to as *second-order cone constraints* and occur frequently in practice (Lobo et al., 1998).

6. Convexity of the Feasible Set

New section

Convexity assessment of the feasible set follows as a corollary to the convexity assessment of constraint functions. The feasible set is viewed as an intersection of larger sets, each of which being defined by an individual constraint. The foremost rule is thus that the intersection of finitely many convex sets is a convex set. Therefore, if any one constraint cannot be proved to define a convex set, the convexity of the whole feasible set is deemed *inconclusive*. In particular, if the convexity of a constraint function is *inconclusive*, the convexity of the feasible set is *inconclusive*.

Both Dr.ampl and coconut transform inequality constraints so they take the general form $c_i^{\text{L}} \leq c_i(x) \leq c_i^{\text{U}}$ where $c_i^{\text{L}} \in \mathbb{R} \cup \{-\infty\}$ and $c_i^{\text{U}} \in \mathbb{R} \cup \{+\infty\}$. Denoting as before $\underline{c_i}$ and $\overline{c_i}$ inferred lower

and upper bounds on the values of the constraint function c_i, the set Ω_i defined by this constraint is convex if either (a) c_i is a convex function and $c_i^{\text{L}} \leq \underline{c_i}$, or (b) c_i is a concave function and $\overline{c_i} \leq c_i^{\text{U}}$.

An equality constraint is considered to (provably) define a convex set only if its constraint function is affine. The feasible set will thus be deemed *inconclusive* if there exists a nonlinear equality constraint.

Note that if the problem contains constraints formulated as $g(x) \leq h(x)$, the latter is readily rephrased $g(x) - h(x) \leq 0$ or $h(x) - g(x) \geq 0$. All of the above rules apply to either case. For instance, the constraint function of the first reformulation will be deemed convex if and only if g is convex and h is concave. If such is the case, the set defined by this constraint will be deemed convex. The second case is similar; the set that it defines is convex if and only if $h(x) - g(x)$ is concave, which occurs if and only if g is convex and h is concave. The procedure is identical for equality constraints written as $g(x) = h(x)$.

7. Numerical Results

We present in this section results of the DAG parsing techniques outlined in the previous sections on standard collections of medium to large-scale nonlinear programs. Dr.AMPL results are given in §7.2 while COCONUT results are reported in §7.3. The first series of tests, described in the next section, validates the approach on a set of problems with known convexity properties. Problems in this set have nonlinear objectives, nonlinear constraints, or both, and at least one of the objective or the feasible set is verifiably convex.

All Dr.AMPL tests were carried out on a Pentium IV 3.5GHz processor running Linux. The main Dr.AMPL executable and library were compiled with the Gnu gcc compiler version 4.0.3 and the convexity disprover was compiled with the Gnu g95 Compiler version 4.0.1. For both compilers, level 3 optimization was used. For the purpose of double-checking results in the present development version of the software, the convexity/concavity disproving phase is launched even when the symbolic phase is conclusive. Of course, in practice, there is no need to do so.

All COCONUT tests were carried out on a Pentium IV 3.4GHz processor running Linux. The COCONUT system was compiled with the GNU C++ compiler version 4.0.3 without optimization. All testing was performed on DAGs generated by `ampl2dag` and simplified by `dag_simplify`.

7.1. Preliminary Validation

In order to validate the convexity engines implemented in Dr.AMPL and COCONUT, we start by reporting convexity assessment results on a test set made of problems available in analytic form in the literature, originating from the AMPL translation of CUTEr models (Gould et al., 2003b), including some Hock and Schittkowski (1981) problems and from Robert Vanderbei's collection of nonlinear models Vanderbei (2008). The problems could be divided into two categories: problems with no constraints or with linear constraints only, and problems with at least one nonlinear constraint. Results on this test set with COCONUT are reported in Table 7.1. We do not report Dr.AMPL results separately since they are nearly identical save for some variations in the CPU time. The few differences are detailed below. The columns of Table 7.1 are as follows: n is the number of variables, m is the total number of constraints, m_n is the number of nonlinear constraints, n_b is the number of bounds, "Obj" is the convexity assessment of the objective function and #cvx, #gen and #inc are the number of convex, general (both nonconvex and nonconcave) and inconclusive nonlinear constraints. The column Ω gives the convexity assessment of the feasible set and t is the running time in seconds. All problems went through the AMPL presolve phase with default options, i.e., 10 passes.

The convexity assessment of the objective function is reported as "Cvx" if it is found to be convex, "Lin" if it is linear, and "Con" if it is constant—the problem is then a feasibility problem. All are minimization problems. The convexity assessment of Ω is reported as "Unc" if the problem is unconstrained, "Lin" if all constraint functions are linear, "Box" if the problem only has bound constraints, and "Cvx" if there is at least one nonlinear constraint and if the feasible set is found to be convex. In both categories, "Inc" means that the analysis in inconclusive.

A few comments are in order to clarify the origin of some of the problems and the variations between the two sets of results.

Problems with names of the form xyz-2 are a variation on problem xyz in which the objective is moved into the constraints. The result is a feasibility problem. When the objective is a convex function, it is used to build a constraint which defines a convex set. For instance, Problem hs064 is the 64-th Hock and Schittkowski problem, which has a convex objective and a convex feasible set, while hs064-2 is a variation in which the problem is converted into a feasibility problem by imposing that the objective be less than some prescribed value. The latter thus also has a convex feasible set.

Problem arwhead-l is a larger version of arwhead.

Problem argauss is the 9-th problem of Moré, Garbow, and Hillstrom (1981). Its feasible set is defined by a level set of some convex Gaussian function. At the time of this writing, Gaussian functions are not supported in Dr.ampl. On this problem, Dr.ampl cannot disprove the constraint function's convexity but can disprove its concavity; as a result, the convexity assessment of the feasible set is inconclusive.

In total, inconclusive results were drawn on five problems out of 171: allinitc, hs012, hs029, nb_L1_eps and optmass. We now comment on those cases in more detail.

The objective function of allinitc is a sum of polynomials and trigonometric functions. However, its convexity could not be disproved due to the values of the variables—namely, zero. Upon setting the initial variables to 1, Dr.ampl correctly disproves convexity.

The objective function of hs012 is the strictly convex quadratic $x^2 + y^2 - xy$. Symbolic convexity proving is inconclusive because of the cross term, as is numerical disproving. However, dr.ampl is able to prove that the function is not concave. ◀

The objective function of hs029 is nonconvex. It has the form $-xyz$. However, because of the initial values of the variables—all zero—convexity cannot be disproved. Upon setting the initial variables to $(1, 1, 1)$, Dr.ampl correctly disproves convexity.

Problem nb_L1_eps is a narrow band 3-dimensional beam pattern optimization problem. Its objective function is a sum of Euclidian norms of linear terms. Dr.ampl is able to correctly prove that the objective function is convex.

The objective function of optmass is diff-convex and thus, is neither convex nor concave. However, convexity could not be disproved due to the values of the variables—namely, zero. Upon setting the initial variables to 1, Dr.ampl correctly disproves convexity.

Table 7.1: Convexity assessment results on test set with known convexity properties.

Problem	n	m	m_n	n_b	Obj	#cvx	#gen	#inc	Ω	t (s)
airport	84	42	42	84	Cvx	42	0	0	Cvx	0.08
airport3	84	43	43	84	Con	43	0	0	Cvx	4.39
allinitc	3	1	1	2	Inc	1	0	0	Cvx	0.01
ampl	2	2	1	2	Con	1	0	0	Cvx	0.00
antenna	363	481	157	1	Con	157	0	0	Cvx	2.94
argauss	3	1	1	0	Con	1	0	0	Cvx	0.00
arglina	100	0	0	0	Cvx	0	0	0	Unc	0.04

Table 7.1: Convexity assessment results on test set with known convexity properties (continued.)

Problem	n	m	m_n	n_b	Obj	#cvx	#gen	#inc	Ω	t (s)
arglina-2	100	1	1	0	Con	1	0	0	Cvx	1.98
arglinb	10	0	0	0	Cvx	0	0	0	Unc	0.00
arglinc	8	0	0	0	Cvx	0	0	0	Unc	0.00
arglinc-2	8	1	1	0	Con	1	0	0	Cvx	0.01
arwhead	100	0	0	0	Cvx	0	0	0	Unc	0.01
arwhead-l	5000	0	0	0	Cvx	0	0	0	Unc	0.23
aug2d	20192	9996	0	0	Cvx	0	0	0	Lin	1.64
aug2dc	20200	9996	0	198	Cvx	0	0	0	Lin	1.07
aug2dcqp	20200	9996	0	20200	Cvx	0	0	0	Lin	1.03
aug2dqp	20192	9996	0	20192	Cvx	0	0	0	Lin	1.01
batch	46	69	1	46	Cvx	1	0	0	Cvx	0.01
bdqrtic	1000	0	0	0	Cvx	0	0	0	Unc	0.07
biggsb1	1000	0	0	999	Cvx	0	0	0	Box	0.03
booth	2	1	1	0	Con	1	0	0	Cvx	0.00
bqp1var	1	0	0	1	Cvx	0	0	0	Box	0.00
braess	4	5	1	4	Con	1	0	0	Cvx	0.00
braess_new	5	5	1	5	Con	1	0	0	Cvx	0.00
brownden	4	0	0	0	Cvx	0	0	0	Unc	0.00
bt3	5	3	0	0	Cvx	0	0	0	Lin	0.00
catenary	198	101	100	0	Con	100	0	0	Cvx	0.04
cb2	3	3	3	0	Lin	3	0	0	Cvx	0.00
cb3	3	3	3	0	Lin	3	0	0	Cvx	0.00
chaconn1	3	3	3	0	Lin	3	0	0	Cvx	0.00
chaconn2	3	3	3	0	Lin	3	0	0	Cvx	0.00
chenhark	1000	0	0	1000	Cvx	0	0	0	Box	0.04
cliff	2	0	0	0	Cvx	0	0	0	Unc	0.00
clplatea	4970	0	0	0	Cvx	0	0	0	Unc	0.57
clplateb	4970	0	0	0	Cvx	0	0	0	Unc	0.57
clplatec	4970	0	0	0	Cvx	0	0	0	Unc	0.58
cvxbqp1	10000	0	0	10000	Cvx	0	0	0	Box	0.32
cvxqp1	1000	500	0	1000	Cvx	0	0	0	Lin	0.13
cvxqp2	10000	2500	0	10000	Cvx	0	0	0	Lin	0.69
cvxqp3	10000	7500	0	10000	Cvx	0	0	0	Lin	1.15
demymalo	3	3	1	0	Lin	1	0	0	Cvx	0.00
dixon3dq	10	0	0	0	Cvx	0	0	0	Unc	0.00
dqdrtic	5000	0	0	0	Cvx	0	0	0	Unc	0.09
dqrtic	5000	0	0	0	Cvx	0	0	0	Unc	0.13
dtoc1l	14985	9990	0	0	Cvx	0	0	0	Lin	2.36
dtoc3	14996	9997	0	0	Cvx	0	0	0	Lin	0.88
engval1	5000	0	0	0	Cvx	0	0	0	Unc	0.23
fccu	19	8	0	0	Cvx	0	0	0	Lin	0.00
fir_convex	11	243	91	1	Con	91	0	0	Cvx	0.07
genhs28	10	8	0	0	Cvx	0	0	0	Lin	0.00
gigomez1	3	3	1	0	Lin	1	0	0	Cvx	0.00
gouldqp2	699	349	0	699	Cvx	0	0	0	Lin	0.03
gouldqp3	699	349	0	699	Cvx	0	0	0	Lin	0.03
gpp	250	498	249	0	Cvx	249	0	0	Cvx	0.86
gridneta	8964	6724	0	325	Cvx	0	0	0	Lin	0.74
gridnetb	13284	6724	0	0	Cvx	0	0	0	Lin	0.71
gridnetc	7564	3844	0	2521	Cvx	0	0	0	Lin	0.41
hager1	10000	5000	0	0	Cvx	0	0	0	Lin	0.36
hong	4	1	0	4	Cvx	0	0	0	Lin	0.00
hs003	2	0	0	1	Cvx	0	0	0	Box	0.00
hs004	2	0	0	2	Cvx	0	0	0	Box	0.00

Table 7.1: Convexity assessment results on test set with known convexity properties (continued.)

Problem	n	m	m_n	n_b	Obj	#cvx	#gen	#inc	Ω	t (s)
hs011	2	1	1	0	Cvx	1	0	0	Cvx	0.00
hs012	2	1	1	0	Inc	1	0	0	Cvx	0.00
hs014	2	2	1	0	Cvx	1	0	0	Cvx	0.00
hs021	2	1	0	2	Cvx	0	0	0	Lin	0.00
hs022	2	2	1	0	Cvx	1	0	0	Cvx	0.00
hs028	3	1	0	0	Cvx	0	0	0	Lin	0.00
hs029	3	1	1	0	Inc	1	0	0	Cvx	0.00
hs030	3	1	1	3	Cvx	1	0	0	Cvx	0.00
hs034	3	2	2	3	Lin	2	0	0	Cvx	0.00
hs043	4	3	3	0	Cvx	3	0	0	Cvx	0.00
hs048	5	2	0	0	Cvx	0	0	0	Lin	0.00
hs049	5	2	0	0	Cvx	0	0	0	Lin	0.01
hs050	5	3	0	0	Cvx	0	0	0	Lin	0.00
hs051	5	3	0	0	Cvx	0	0	0	Lin	0.00
hs052	5	3	0	0	Cvx	0	0	0	Lin	0.00
hs053	5	3	0	5	Cvx	0	0	0	Lin	0.00
hs064	3	1	1	3	Cvx	1	0	0	Cvx	0.00
hs064-2	3	2	2	3	Con	2	0	0	Cvx	0.00
hs065	3	1	1	3	Cvx	1	0	0	Cvx	0.00
hs066	3	2	2	3	Lin	2	0	0	Cvx	0.00
hs072	4	2	2	4	Lin	2	0	0	Cvx	0.00
hs072-2	4	3	2	4	Con	2	0	0	Cvx	0.00
hs118	15	17	0	15	Cvx	0	0	0	Lin	0.00
hs21mod	7	1	0	6	Cvx	0	0	0	Lin	0.00
hs3mod	2	0	0	1	Cvx	0	0	0	Box	0.00
hues-mod	10000	2	0	10000	Cvx	0	0	0	Lin	336.84
huestis	10000	2	0	10000	Cvx	0	0	0	Lin	332.05
immun	19	6	0	19	Cvx	0	0	0	Lin	0.01
jbearing100	5000	0	0	5000	Cvx	0	0	0	Box	0.56
jbearing25	2500	0	0	2500	Cvx	0	0	0	Box	0.29
jbearing50	2500	0	0	2500	Cvx	0	0	0	Box	0.28
jbearing75	3750	0	0	3750	Cvx	0	0	0	Box	0.41
ksip	20	1000	0	1	Cvx	0	0	0	Lin	1.06
linear	24	20	0	20	Cvx	0	0	0	Lin	0.00
liswet1	10002	10000	0	0	Cvx	0	0	0	Lin	0.74
liswet10	10002	10000	0	0	Cvx	0	0	0	Lin	0.74
liswet11	10002	10000	0	0	Cvx	0	0	0	Lin	0.78
liswet12	10002	10000	0	0	Cvx	0	0	0	Lin	1.22
liswet2	10002	10000	0	0	Cvx	0	0	0	Lin	1.73
liswet3	10002	10000	0	0	Cvx	0	0	0	Lin	0.74
liswet4	10002	10000	0	0	Cvx	0	0	0	Lin	0.74
liswet5	10002	10000	0	0	Cvx	0	0	0	Lin	0.74
liswet6	10002	10000	0	0	Cvx	0	0	0	Lin	0.74
liswet7	10002	10000	0	0	Cvx	0	0	0	Lin	0.77
liswet8	10002	10000	0	0	Cvx	0	0	0	Lin	0.76
liswet9	10002	10000	0	0	Cvx	0	0	0	Lin	0.76
lotschd	12	7	0	12	Cvx	0	0	0	Lin	0.02
lsqfit	2	1	0	1	Cvx	0	0	0	Lin	0.00
madsschj	81	158	158	0	Lin	158	0	0	Cvx	2.05
makela1	3	2	1	0	Lin	1	0	0	Cvx	0.00
makela2	3	3	3	0	Lin	3	0	0	Cvx	0.00
makela3	21	20	20	0	Lin	20	0	0	Cvx	0.00
markowitz	8	2	1	8	Con	1	0	0	Cvx	0.01
meyer3	3	0	0	0	Cvx	0	0	0	Unc	0.00

Table 7.1: Convexity assessment results on test set with known convexity properties (continued.)

Problem	n	m	m_n	n_b	Obj	#cvx	#gen	#inc	Ω	t (s)
mifflin1	3	2	1	0	Lin	1	0	0	Cvx	0.00
mifflin2	3	2	2	0	Lin	2	0	0	Cvx	0.00
minmaxbd	5	20	20	0	Lin	20	0	0	Cvx	0.00
nasty	2	0	0	0	Cvx	0	0	0	Unc	0.00
nb_L1_eps	1708	2381	793	0	Inc	793	0	0	Cvx	90.59
nondquar	10000	0	0	0	Cvx	0	0	0	Unc	0.37
optmass	66	55	11	0	Inc	11	0	0	Cvx	0.01
optprloc	30	29	25	30	Cvx	25	0	0	Cvx	0.01
oslbqp	8	0	0	8	Cvx	0	0	0	Box	0.00
palmer1c	8	0	0	0	Cvx	0	0	0	Unc	0.00
palmer1d	7	0	0	0	Cvx	0	0	0	Unc	0.00
palmer2c	8	0	0	0	Cvx	0	0	0	Unc	0.00
palmer3c	8	0	0	0	Cvx	0	0	0	Unc	0.00
palmer4c	8	0	0	0	Cvx	0	0	0	Unc	0.00
palmer5c	6	0	0	0	Cvx	0	0	0	Unc	0.00
palmer5d	4	0	0	0	Cvx	0	0	0	Unc	0.00
palmer6c	8	0	0	0	Cvx	0	0	0	Unc	0.00
palmer7c	8	0	0	0	Cvx	0	0	0	Unc	0.00
palmer8c	8	0	0	0	Cvx	0	0	0	Unc	0.00
polak1	3	2	2	0	Lin	2	0	0	Cvx	0.00
polak2	11	2	2	0	Lin	2	0	0	Cvx	0.00
polak4	3	3	3	0	Lin	3	0	0	Cvx	0.00
portfl1	12	1	0	12	Cvx	0	0	0	Lin	0.01
portfl2	12	1	0	12	Cvx	0	0	0	Lin	0.01
portfl3	12	1	0	12	Cvx	0	0	0	Lin	0.01
portfl4	12	1	0	12	Cvx	0	0	0	Lin	0.01
portfl6	12	1	0	12	Cvx	0	0	0	Lin	0.01
powell20	1000	1000	0	0	Cvx	0	0	0	Lin	0.06
power	1000	0	0	0	Cvx	0	0	0	Unc	0.02
probpenl	500	0	0	500	Cvx	0	0	0	Box	0.03
qp4	79	31	0	50	Cvx	0	0	0	Lin	0.12
qpcboei1	372	285	0	372	Cvx	0	0	0	Lin	0.32
qpcboei2	143	122	0	143	Cvx	0	0	0	Lin	0.05
qpcstair	385	356	0	379	Cvx	0	0	0	Lin	7.17
quartc	10000	0	0	0	Cvx	0	0	0	Unc	0.27
rosenmmx	5	4	4	0	Lin	4	0	0	Cvx	0.00
s383	14	2	1	14	Con	1	0	0	Cvx	0.00
sambal	17	10	0	0	Cvx	0	0	0	Lin	0.00
sample	4	2	2	4	Lin	2	0	0	Cvx	0.00
sim2bqp	2	0	0	1	Cvx	0	0	0	Box	0.00
simbqp	2	0	0	1	Cvx	0	0	0	Box	0.00
steenbra	432	108	0	432	Cvx	0	0	0	Lin	0.03
svanberg	5000	5000	5000	5000	Cvx	5000	0	0	Cvx	1.22
synthes1	6	6	2	6	Cvx	2	0	0	Cvx	0.00
tame	2	1	0	2	Cvx	0	0	0	Lin	0.00
tame-2	2	2	1	2	Con	1	0	0	Cvx	0.00
tame1	2	2	1	2	Con	1	0	0	Cvx	0.00
tridia	10000	0	0	0	Cvx	0	0	0	Unc	0.29
tridia-2	30	1	1	0	Con	1	0	0	Cvx	0.01
turkey	512	276	0	411	Cvx	0	0	0	Lin	0.63
ubh1	17997	12000	0	6003	Cvx	0	0	0	Lin	1.75
vardim	100	0	0	0	Cvx	0	0	0	Unc	0.00
yao	2000	1999	0	2	Cvx	0	0	0	Lin	0.14
zecevic2	2	2	0	2	Cvx	0	0	0	Lin	0.00

Table 7.1: Convexity assessment results on test set with known convexity properties (continued.)

Problem	n	m	m_n	n_b	Obj	#cvx	#gen	#inc	Ω	t (s)
zecevic2-2	2	3	1	2	Con	1	0	0	Cvx	0.00

7.2. DR.AMPL Results on the COPS Collection

This section presents numerical results on the COPS (Dolan et al., 2004; Bondarenko et al., 2004) test set, version 2.0. Table 7.2 presents summary results reporting problem names, dimension, convexity of objective function and timing statistics. For each problem, AMPL was asked to return tight bounds using the option `option var_bounds 2` and two presolve options were used. The first, `option presolve 0`, disables AMPL's presolve altogether while the second, `option presolve 10`, asks for up to 10 presolve passes. For convexity disproving, a single point x satisfying the bounds was chosen to solve problem (5.1). The point x is either the starting point x_0 specified in the AMPL model if it satisfies the bound constraints, or the projection of x_0 into the bounds.

In Table 7.2, the number of constraints m does not include simple bounds. It may, and does, happen that only bounds remain after AMPL's presolve phase, causing $m = 0$ to be reported. The number of bound constraints is reported in the column titled n_b. The nature of the objective is reported as *linear* if the objective is a linear function, as *constant* if the problem is a pure feasibility problem, as *convex* if convexity was deemed to hold in the symbolic phase and could not be numerically disproved, as *concave* if concavity was proved symbolically and could not be disproved numerically, and as *nonconvex* if neither convexity or concavity could be proved in the symbolic phase but were disproved numerically. Finally, it is reported as *inconclusive* if convexity and concavity could neither be proved nor disproved. The timings reported for a problem do not include AMPL's presolve phase but only take into account the convexity/concavity proving/disproving tests. We emphasize that for the purpose of double-checking results, the convexity disprover is run not only on problem functions whose convexity was deemed inconclusive in the symbolic phase, but also on all those that were deemed convex. This does not include linear problem functions.

7.3. COCONUT Results on the COPS Collection

In this section we present the test results for the COCONUT system (version 3.0 build 202), on the COPS test set version 3.0. Table 7.3 presents summary statistics on the COPS 3.0 collection. Note that problems were preprocessed and that some problem sizes may differ from those in the COPS 2.0 collection. For convexity analysis the COCONUT tool `analyze_convexity` was used.

The `analyze_convexity` tool calls the constraint propagator and uses the `simple_convexity` inference engine to test all functions involved for convexity. It returns convexity information for the objective function and all constraints, as well as a summary information for the feasible set and the problem classification. The information returned for the *objective function* is `constant` if it can be proved to be a constant function or a pure feasibility problem is given, as `linear` if it is an affine function. The objective is reported as `convex` if convexity could be proved and the problem is a minimization problem or if concavity could be proved and the problem is a maximization problem, it is reported as `concave` in the reverse cases. Finally, the return value is `Inconclusive` if neither convexity nor concavity could be proved.

Although convexity of individual constraint functions in important elsewhere in the COCONUT framework, the `analyze_convexity` tool reports on convexity of the sets defined by the individual constraints. With this in mind, a single constraint is reported to be `linear` if it is an affine function

Problem	Pre	n	m	m_n	n_b	Obj	#cvx	#gen	#inc	Ω	t (s)
Bearing	0	2704	208	0	2704	Cvx	0	0	0	Cvx	0.43
	10	2500	0	0	2500	Cvx	0	0	0	Cvx	0.41
Camshape	0	800	1603	801	800	Lin	1	800	0	Inc	0.83
	10	800	1600	801	800	Lin	1	800	0	Inc	0.90
Catmix	0	2403	1602	1600	801	Lin	0	800	800	Inc	3.69
	10	2401	1600	1600	801	Lin	0	801	799	Inc	3.61
Chain	0	402	203	1	0	Ncvx	0	1	0	Inc	0.02
	10	400	201	1	0	Ncvx	0	1	0	Inc	0.02
Channel	0	1600	1600	800	0	Cons	0	800	0	Inc	2.29
	10	1598	1598	800	0	Cons	0	800	0	Inc	2.25
Elec	0	300	100	100	0	Ncvx	100	0	0	Inc	0.67
	10	300	100	100	0	Ncvx	100	0	0	Inc	0.66
Gasoil	0	2003	2003	1600	0	Cvx	0	1596	4	Inc	4.12
	10	2001	1998	1600	3	Cvx	0	1596	4	Inc	4.12
Glider	0	1006	1411	800	0	Lin	0	800	0	Inc	6.32
	10	999	800	800	601	Lin	0	800	0	Inc	6.03
Marine	0	4815	4807	3200	0	Cvx	0	3200	0	Inc	21.01
	10	4815	4792	3200	15	Cvx	0	3200	0	Inc	21.65
Methanol	0	1205	1205	900	0	Cvx	0	900	0	Inc	2.68
	10	1202	1197	900	5	Cvx	0	900	0	Inc	2.45
Minsurf	0	2704	3696	0	0	Inc	0	0	0	Cvx	0.38
	10	2500	0	0	2500	Inc	0	0	0	Cvx	0.36
Pinene	0	1005	1005	750	0	Cvx	0	438	312	Inc	0.76
	10	1000	995	750	5	Cvx	0	438	312	Inc	0.75
Polygon	0	100	1376	10	0	Ncvx	0	10	0	Inc	0.00
	10	98	1273	10	98	Ncvx	4	6	0	Inc	0.00
Robot	0	1811	1213	1200	1207	Lin	0	200	1000	Inc	2.62
	10	1799	1201	1200	1202	Lin	0	200	1000	Inc	2.77
Rocket	0	1605	2408	1200	401	Lin	0	1200	0	Inc	3.82
	10	1601	1200	1200	1601	Lin	0	1200	0	Inc	3.81
Steering	0	507	510	400	0	Lin	0	100	300	Inc	0.09
	10	500	401	400	103	Lin	0	100	300	Inc	0.09
Torsion	0	2704	2704	0	0	Cvx	0	0	0	Cvx	0.31
	10	2500	0	0	2500	Cvx	0	0	0	Cvx	0.19

Table 7.2. DR.AMPL results on the COPS 2.0 test set. In this table, "Pre" is the value of `presolve`. The other column headers are as in Table 7.1. The number of concave constraints is not reported since it is zero for all problems. Here, t is the running time in seconds for the convexity proving and disproving phases.

combined with either one-sided or two-sided constraints, as `convex` if it defines a convex subset of $\mathbb{R}^n$, and as `concave` if flipping the sign of the constraint function defines a convex subset. More precisely, we will say that the set $\{x \in \mathbb{R}^n \mid c(x) \geq 0\}$ is *concave* if the function c is convex. If a problem has only linear or concave constraints, we will say that it is *concavely constrained*. The fact that a problem is concavely constrained can be important for global optimization insofar as in this case, without additional constraint qualification, the Kuhn-Tucker conditions are valid on all local optima.

The system reports a constraint to be `nonconvex` if it can be proved to define a nonconvex or empty subset of $\mathbb{R}^n$, most notably this is the result for nonlinear equality constraints, but also for provably convex or concave functions with two-sided inequality constraints. Finally, the return value for a constraint is `Inconclusive` if none of the above cases is present.

For the feasible set the system reports `linear` if all constraints are affine, and `convex` if all constraints were proven to be convex. The very rare result `nonconvex` is reported if the feasible set could be proved to be nonconvex. In all other cases the system reports `Inconclusive`.

For the problem classification, the possible return values are

`linear` if the objective function is linear and all constraints are linear,
`convex, linearly constrained` if the objective function is convex and all constraints are linear (i.e. the problem is linearly constrained),
`concave, linearly constrained` if the objective function is concave and all constraints are linear,
`convex` if the objective function is convex and the feasible set is convex,
`concave, convexly constrained` if the objective function is concave and the feasible set is convex,
`linear CSP` if the objective function is constant and all constraints are linear (i.e. a linear constraint satisfaction problem),
`convex CSP` if the objective function is constant and the feasible set is convex,
`concave CSP` if the objective function is constant and the feasible set is concavely constrained,
`linearly constrained` if the objective function is inconclusive and all constraints are linear,
`convexly constrained` if the objective function is inconclusive and the feasible set is convex,
`concavely constrained` if the objective function is inconclusive and the feasible set is concavely constrained.
`nonconvex` if the objective function is inconclusive or any constraint is nonconvex or inconclusive.
`linear, unconstrained` if the objective function is linear and there are no constraints.
`convex, unconstrained` if the objective function is convex and there are no constraints.
`concave, unconstrained` if the objective function is concave and there are no constraints.
`linear, box constrained` if the objective function is linear and there are only simple bound constraints.
`convex, box constrained` if the objective function is linear and there are only simple bound constraints.
`concave, box constrained` if the objective function is linear and there are only simple bound constraints.
`trivial` if the objective function is constant and there are only simple bound constraints or no constraints at all.

The COCONUT environment only uses symbolic convexity proving techniques. All timing results include constraint propagation, and the time required for problem I/O, excluding the automatic AMPL to DAG transformation, which can take minutes for the largest test problems.

Problem	n	m	m_n	n_b	Obj	#cvx	#gen	#inc	Ω	t (s)
Bearing	2500	0	0	2500	Cvx	0	0	0	Box	0.30
Camshape	800	1600	801	800	Lin	1	0	800	Inc	1.23
Catmix	2548	2008	900	450	Lin	0	900	0	Inc	0.24
Chain	299	225	77	0	Lin	0	0	77	Inc	0.06
Channel	4798	4798	800	0	Cons	0	800	0	Inc	0.78
Elec	300	100	100	0	Inc	0	100	0	Inc	1.09
Gasoil	3601	3598	1600	3	Cvx	0	1	1599	Inc	0.45
Glider	1303	1204	903	301	Lin	0	601	302	Inc	0.09
Marine	9615	9592	3200	15	Cvx	0	3200	0	Inc	0.59
Methanol	2702	2697	900	5	Cvx	0	0	900	Inc	0.51
Minsurf	2500	0	0	2500	Inc	0	0	0	Box	0.64
Pinene	1450	1445	750	5	Cvx	0	302	448	Inc	0.24
Polygon	98	1273	1223	98	Inc	47	0	1176	Inc	0.13
Robot	2197	1599	1198	1200	Lin	0	800	398	Inc	0.26
Rocket	2401	2000	1600	1601	Lin	0	800	800	Inc	0.28
Steering	1000	801	400	202	Lin	0	400	0	Inc	0.12
Torsion	2500	0	0	2500	Cvx	0	0	0	Cvx	0.15

Table 7.3. COCONUT results on the COPS 3.0 test set. The column headers are as in Table 7.1.

Note that while there seems to be a discrepancy between the form of the objective of problem `chain` in Tables 7.2 and 7.3, both results are correct. In version 2.0 of the COPS test set, the objective of `chain` is nonlinear, indeed nonconvex, while in version 3.0 the same problems is reformulated so as to have a linear objective and to move the initial nonlinear objective into the constraints.

7.4. Discussion of the Numerical Results

Given that many problems from the COPS test set are discretized control problems, they often feature nonlinear equality constraints. Thus, the feasible set of all such problems was cast as *nonconvex*. The three problems with a convex feasible set—`Bearing`, `Minsurf` and `Torsion`—have linear constraints only. For each problem, each constraint was examined symbolically and numerically. The AMPL presolve phase affects convexity assessment in some cases, such as `Polygon`, where from the 10 nonlinear constraints, all are deemed *inconclusive* when `presolve=0` but four of them are determined to be convex when `presolve=10`. DR.AMPL classifies all equality and inequality constraints as either *convex*, *concave*, *nonconvex* or *inconclusive*, where *nonconvex* is understood as nonconvex and nonconcave. This provides a finer analysis of the feasible set. Full details on the numerical results will be available and updated on the main DR.AMPL web site at `www.gerad.ca/~orban/drampl`.

The main source of nonconvexity in the problems of Table 7.2 is the product between two or more variables, that is, terms of the form $x_i y_j$ where both x and y are problem variables or defined variables. Such terms are also a source of frequent inconclusive cases. For example, consider the function of two variables $f(x, y) = (x-y)^2$. DR.AMPL easily proves that f is convex because it consists in the squaring of a linear term. However, were this function equivalently rewritten $f(x, y) = x^2+y^2-2xy$, its convexity could neither be proved nor disproved by our graph-based techniques. It could not be proved because of the product of two variables and it could not be disproved because f is indeed convex.

This is the reason why the objective function of problem `hs012` failed to be detected as convex in §7.1. Had the objective x^2+y^2-xy been written instead $(x-\frac{1}{2}y)^2+\frac{3}{4}y^2$ or $\frac{1}{4}(x+y)^2+\frac{3}{4}(x-y)^2$, it would have been detected as a sum of squares of linear terms and identified as convex. Many convex polynomials can be stated as such a a sum of squares (Choi et al., 1995; Reznick, 2000). However, there may be infinitely many such formulations of a given convex polynomial and finding one is not straightforward in general, especially as the number of variables increases. Expressing a convex quadratic as a sum of squares can be done, for example, by computing the eigenvalue and eigenvector decomposition of its Hessian matrix. If we wish to introduce a numerical proof of convexity for the quadratic case, however, then a Cholesky factorization of the Hessian may enable us to obtain the same results more efficiently. In either case there is the risk of an incorrect conclusion due to numerical inaccuracies in the computations; a more reliable assessment could be obtained by use of interval arithmetic, but possibly at substantially greater cost. Nevertheless, we intend to consider the introduction of these approaches in future versions of our software. ◀

As another example, the constraint function $xy - 1$ used in the set of constraints $xy - 1 \leq 0$ where x and y are guaranteed to be positive is not a convex function. However, upon introducing the change of variable $x = \exp(w)$ and $y = \exp(z)$, the constraint may be equivalently rewritten $\exp(w + z) \leq 1$, which is convex and will correctly be identified as such.[1] Inconclusive cases need not only be caused by products of variables. Consider the simpler function $g(x) = (x - 1)^4$. Again, if the function is written in this form then DR.AMPL easily proves convexity since g consists in a linear term to an even positive power. However, if expanded as $g(x) = x^4 - 4x^3 + 6x^2 - 4x + 1$, this expression becomes a sum of nonlinear terms and, according to the rules of §4, the sum will only be deemed convex if all terms are convex. Here, the term $-4x^3$ prevents detection of the convexity of the sum and hence, convexity of $g(x)$.

[1] This example was kindly suggested to us by an anonymous referee.

These simple examples illustrate that the user of our software must still be concerned to some degree with expressing objectives and constraints in factored form or, more generally, in "convex form." ◀ Although excellent software has long been available for symbolic manipulation of nonlinear expressions, and the use of such software for convexity detection in some cases has been investigated (as by Stoutmeyer (1978)), tools such as those described in the present paper have not yet reached a level of maturity that would permit the introduction of symbolic factorizations and other transformations that reveal convexity. For now, the knowledgeable modeler must be prepared to bring symbolic software to bear on the formulation before it is submitted to such tools as COCONUT or DR.AMPL, in order to develop models that can be solved most efficiently and reliably.

Finally, a comment is in order regarding the timings reported in Table 7.2. The convexity disproving phase typically accounts for around 99% of the total time required by the convexity analysis. In later versions of the code, constraints which will have been proven convex will no longer need to undergo convexity disproving, which will bring the convexity analysis time to a few hundredth of a second in most cases. For example, on problem *elec*, the convexity proving phase takes less than an hundredth of a second, causing a time of 0 seconds to be reported. The remaining 0.67 seconds are all used by the disproving phase.

8. Conclusion

Automatic rules for detection of convexity and concavity have been presented in the framework of modeling languages for smooth global and local optimization. Nonlinear functions are represented by directed acyclic graphs, the recursive nature of which allows us to infer convexity and monotonicity properties and to propagate bounds. Numerical results illustrating the rules as implemented in the COCONUT system and in the DR.AMPL meta-solver have been presented. In the DR.AMPL package, work is under way to allow user-selected convexity disprovers, which will ease comparison with other software, e.g., (Chinneck, 2001).

A symbolic analysis by itself is insufficient for convexity assessment, as is a numerical analysis. The two pieces of software illustrated in this paper show how the two may be combined so that as few cases as possible are misclassified. Assessing the convexity properties of large-scale problems requires that we work in finite precision. Yet the numerical procedure used to disprove convexity properties may fail for one of several reasons: the limitations of finite precision do not lead to a conclusive diagnosis, the numerical data was analyzed with unfortunate input—such as the values of the variables at which the Hessian matrix is analyzed—, or perhaps other, problem-dependent, reasons. On the other hand, the symbolic phase is best suited to produce a certificate of convexity and not to attempt to disprove it. It may however also fail because it is composed of a finite set of rules.

The two pieces of software illustrated in this paper analyze each problem function in turn. Determining convexity of the constraint functions and convexity of the feasible set are different procedures and their conclusions have different impacts on the use or design of optimization software and on the properties of the problem.

Computational graph walks may be used to determine whether a function can be expressed as the difference between two convex functions—the so-called *diff-convexity*. This may be a desirable feature of the packages described in this paper as there exist algorithms which can take advantage of such structure, e.g., (Voorhis and Al-Khayyal, 2003). This is a rather straightforward extension since a sufficient condition for a function to be diff-convex is that it be a sum of terms, each of which is either convex or concave.

The authors are aware that there are more known rules for inferring convexity properties, e.g., (Avriel et al., 1988), but these do not follow simple patterns and are not yet implemented. Broader nuances of convexity such as pseudoconvexity (Boyd and Vandenberghe, 2004) are not examined in this research as it remains unclear how software can take advantage of it.

Several classes of problems are naturally convex, such as semidefinite programs or second-order cone programs (Boyd and Vandenberghe, 2004). Some second-order cone constraints can currently be detected as convex by DR.AMPL. An extension of DR.AMPL to properly categorize and detect second-order cone programs is the subject of current research. Categorizing semidefinite programs and assessing their convexity properties promises to be challenging. On the one hand, modeling languages must be extended to allow the description of semidefinite programs. On the other hand, verifying convexity of, say, linear matrix equalities or inequalities is both delicate and computationally intensive.

In practical situations, the large amount of numerical software available for smooth optimization makes the choice difficult to the novice or unfamiliar user. Tools such as DR.AMPL have additional features that detect problem structure and are able to recommend certain solvers in place of others (Fourer and Orban, 2007). In future research, it would be worthwhile examining the efficiency of an approach where a given problem is first analyzed structurally—this includes convexity assessment—and then passed to an appropriate, recommended, solver.

Acknowledgements. The authors acknowledge the constructive comments of two anonymous referees and the associate editor which improved the presentation of this paper.

References

Avriel, M., W.E. Diewert, S. Schaible, I. Zang. 1988. *Generalized Concavity*. Plenum, New York.

Bauer, F. L. 1974. Computational graphs and rounding error. *SIAM Journal on Numerical Analysis* **11** 87–96.

Bondarenko, A., D. Bortz, E. D Dolan, M. Merritt, J. J. Moré, T. S. Munson. 2004. www.mcs.anl.edu/~more/cops.

Boyd, S., L. Vandenberghe. 2004. *Convex Optimization*. Cambridge University Press.

Chinneck, J. 2001. Analyzing Mathematical Programs using MProbe. *Annals of Operations Research* **104** 33–48.

Choi, M. D., T. Y. Lam, B. Reznick. 1995. Sums of squares of real polynomials. *Proceedings of Symposia in Pure Mathematics*, vol. 58. 103–126.

Dolan, E. D., J. J. Moré, T. S. Munson. 2004. Benchmarking optimization software with COPS 3.0. Technical Report ANL/MCS-TM-273, Argonne National Laboratory.

Fourer, R., D. M. Gay, B. W. Kernighan. 2002. *AMPL: A Modeling Language for Mathematical Programming*. 2nd ed. Duxbury Press, Brooks/Cole Publishing Company.

Fourer, R., D. Orban. 2007. The DrAmpl meta solver for optimization. Technical Report G-2007-10, GERAD, Montreal, Canada. www.gerad.ca/~orban/drampl.

Gay, D. M. 2002. Hooking your solver to AMPL. www.ampl.com/REFS/HOOKING.

Gould, N. I. M., S. Lucidi, M. Roma, Ph. L. Toint. 1999. Solving the trust-region subproblem using the Lanczos method. *SIAM Journal on Optimization* **9** 504–525.

Gould, N. I. M., D. Orban, Ph. L. Toint. 2003a. GALAHAD—a library of thread-safe Fortran 90 packages for large-scale nonlinear optimization. *Transactions of the ACM on Mathematical Software* **29** 353–372.

Gould, N. I. M., D. Orban, Ph. L. Toint. 2003b. CUTEr and SifDec, a Constrained and Unconstrained Testing Environment, revisited. *Transactions of the ACM on Mathematical Software* **29** 373–394.

Grant, M. C., S. Boyd. 2008. Graph implementations for nonsmooth convex programs. V. Blondel, S. Boyd, H. Kimura, eds., *Recent Advances in Learning and Control (a tribute to M. Vidyasagar)*. Springer. To appear.

Grant, M. C., S. Boyd, Y. Ye. 2006. Disciplined convex programming. L. Liberti, N. Maculan, eds., *Global Optimization: From Theory to Implementation*. Nonconvex Optimization and its Applications series, Springer, Netherlands.

Grant, M. C., S. Boyd, Y. Ye. 2008. CVX: Matlab software for disciplined convex programming. www.stanford.edu/~boyd/cvx. Web page and software.

Griewank, A. 2000. *Evaluating Derivatives: Principles and Techniques of Algorithmic Differentiation*. No. FR19 in Frontiers in Applied Mathematics, SIAM.

Hock, W., K. Schittkowski. 1981. *Test Examples for Nonlinear Programming Codes*, vol. 187, chap. Lectures Notes in Economics and Mathematical Systems. Springer Verlag, Berlin.

Kantorovich, L. V. 1957. On a mathematical symbolism convenient for performing machine calculations. *Dokl. Akad. Nauk SSSR* **113** 738–741. (in Russian).

Lobo, M. S., L. Vandenberghe, S. Boyd, H. Lebret. 1998. Applications of second-order cone programming. *Linear Algebra and its Applications* 193–228.

Mattingley, J., S. Boyd. 2008. CVXMOD: Convex optmization software in Python. www.cvxmod.net. Web page and software.

Moré, J. J., B. S. Garbow, K. W. Hillstrom. 1981. Testing unconstrained optimization software. *Transactions of the ACM on Mathematical Software* **7** 17–41.

Nenov, I. P., D. H. Fylstra, L. V. Kolev. 2004. Convexity Determination in the Microsoft Excel Solver Using Automatic Differentiation Techniques. Technical Report, Frontline Systems Inc., Incline Village NV, USA.

Neumaier, A., H. Schichl. 2003. Sharpening the karush-john optimality conditions. Preprint, University of Vienna, Vienna, Austria.

Reznick, B. 2000. Some concrete aspects of hilbert's 17th problem. *Contemporary Mathematics* **253** 251–272.

Schichl, H. 2004a. The COCONUT environment. www.mat.univie.ac.at/coconut-environment.

Schichl, H. 2004b. The COCONUT project home page. www.mat.univie.ac.at/coconut.

Schichl, H., A. Neumaier. 2003. Interval analysis on directed acyclic graphs for global optimization. Preprint, University of Vienna, Vienna, Austria.

Steihaug, T. 1983. The conjugate gradient method and trust regions in large scale optimization. *SIAM Journal on Numerical Analysis* **20** 626–637.

Stoutmeyer, D. R. 1978. Automatic categorization of optimization problems: An application of computer symbolic mathematics. *Operations Research* **26** 773–788.

Toint, Ph. L. 1981. Towards an efficient sparsity exploiting newton method for minimization. I. S. Duff, ed., *Sparse Matrices and Their Uses*. Academic Press, London, UK, 57–88.

Vanderbei, R. J. 2008. Nonlinear optimization models. www.orfe.princeton.edu/~rvdb/ampl/nlmodels. Web page and software, accessed on 15 December 2008.

Voorhis, T. Van, F. A. Al-Khayyal. 2003. Difference of convex solutions of quadratically constrained optimization problems. *European Journal of Operations Research* **148** 349–362.