Checking Monotonicity Is NP-Hard Even for Cubic Polynomials*

Andrzej Pownuk[1], Luc Longpré[2], and Vladik Kreinovich[2]
Departments of [1]Mathematical Sciences and
[2]Computer Science, University of Texas at El Paso,
500 W. University, El Paso, TX 79968, USA
ampownuk@utep.edu, longpre@utep.edu, vladik@utep.edu

February 25, 2013

Abstract

One of the main problems of interval computations is to compute the range of a given function over given intervals. In general, this problem is computationally intractable (NP-hard) – that is why we usually compute an enclosure and not the exact range. However, there are cases when it is possible to feasibly compute the exact range; one of these cases is when the function is monotonic with respect to each of its variables. The monotonicity assumption holds when the derivatives at a midpoint are different from 0 and the intervals are sufficiently narrow; because of this, monotonicity-based estimates are often used as a heuristic method. In situations when it is important to have an enclosure, it is desirable to check whether this estimate is justified, i.e., whether the function is indeed monotonic. It is known that monotonicity can be feasibly checked for quadratic functions. In this paper, we show that for cubic functions, checking monotonicity is NP-hard.

Keywords: interval computations, monotonicity, NP-hard
AMS subject classifications: 65G20, 65G40, 03D15, 68Q17

It is desirable to check monotonicity. One of the main problems of interval computations is computing the range $\boldsymbol{y}$ of an (algorithmically) given function $f(x_1, \ldots, x_n)$ over n given intervals $\boldsymbol{x}_i = [\underline{x}_i, \overline{x}_i]$, $i = 1, \ldots, n$:

$$\boldsymbol{y} = [\underline{y}, \overline{y}] = \{f(x_1, \ldots, x_n) : x_1 \in \boldsymbol{x}_1, \ldots, x_n \in \boldsymbol{x}_n\}$$

of the function $f(x_1, \ldots, x_n)$ under given intervals. It is known (see, e.g., [1]) that even for quadratic polynomials this problem is, in general, NP-hard.

There are cases when it is possible to feasibly compute the exact range; see, e.g., [2]. One such case is when a function is monotonic (i.e., increasing or decreasing) in each of its variables. In this case, the range of this function can be easily computed. For

*Submitted: February 25, 2013; Revised: ??? Accepted: ???.

example, if a function is increasing with respect to each of its variables, i.e., if for all i and for all possible values $x_1 \in [\underline{x}_1, \overline{x}_1]$, $\ldots$, $x_{i-1} \in [\underline{x}_{i-1}, \overline{x}_{i-1}]$, $x_i \in [\underline{x}_i, \overline{x}_i]$, $x_i' \in [\underline{x}_i, \overline{x}_i]$, $x_{i+1} \in [\underline{x}_{i+1}, \overline{x}_{i+1}]$, $\ldots$, $x_n \in [\underline{x}_n, \overline{x}_n]$, the inequality $x_i \le x_i'$ implies that

$$f(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) \le f(x_1, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots, x_n),$$

then its range can be easily computed as $[f(\underline{x}_1, \ldots, \underline{x}_n), f(\overline{x}_1, \ldots, \overline{x}_n)]$.

One way to check whether a function is monotonic is to find the ranges of each partial derivatives $\dfrac{\partial f}{\partial x_i}$; if none of these ranges contains 0, this means that the function is monotonic [2]. In practice, we can only feasibly compute *enclosures* for these ranges. If none of the enclosures contains 0, this mean that the actual ranges also do not contain 0, so the function is monotonic. However, if one of the enclosures does contain 0, the function may still be monotonic – and 0 may be caused by the excess width of the enclosure.

From the practical viewpoint, the use of monotonicity is a reasonable idea: when all the partial derivatives $\dfrac{\partial f}{\partial x_i}$ computed at the midpoint with coordinates $\widetilde{x}_i = \dfrac{\underline{x}_i + \overline{x}_i}{2}$ are non-zero, then, when the derivatives are continuous, for sufficiently small radii Δ_i, the derivatives are non-zero for all points x from the box

$$[\widetilde{x}_1 - \Delta_1, \widetilde{x}_1 + \Delta_1] \times \ldots \times [\widetilde{x}_n - \Delta_n, \widetilde{x}_n + \Delta_n].$$

So, if measurement accuracy is high enough, i.e., if the upper bounds Δ_i on the corresponding uncertainty are small enough, practitioners assume that the function is monotonic and use the above simple estimate for the range.

In many practical situations, it is important to check whether this estimate is indeed an enclosure. For example, we are designing an engineering system, and we want to make sure that the value of some critical quantity $y = f(x_1, \ldots, x_n)$ (temperature, pollution level, etc.) cannot exceed a given threshold y_0 no matter what combination of parameters x_i from the given ranges $\boldsymbol{x}_i$ we take. If we make this conclusion based on an estimate which misses some values of $f(x_1, \ldots, x_n)$, we may design a defective system.

To justify that the monotonicity-based estimate is an enclosure, it is desirable to check whether the function is indeed monotonic on a given box.

Checking monotonicity: what is known. For a *quadratic* function $f(x_1, \ldots, x_n)$, all partial derivatives are linear. For a linear function, we can feasibly compute its range, so we can feasibly check whether a given quadratic function is monotonic.

New result. In this paper, we show that already for cubic polynomials, checking monotonicity is NP-hard.

Comment. It is widely believed that P≠NP. In this case, NP-hardness means that it is not possible to have a feasible (= polynomial time) algorithm that always computes the desired range; see, e.g., [1, 3].

Definition 1.

- *We say that a function* $f(x_1, \ldots, x_n)$ *is* non-strictly increasing with respect to a variable x_i *if for every set of values* $x_1, \ldots, x_{i-1}, x_i, x_i', x_{i+1}, \ldots, x_n$ *for which* $x_i \le x_i'$, *we have*

$$f(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) \le f(x_1, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots, x_n).$$

- *We say that a function* $f(x_1, \ldots, x_n)$ *is* non-strictly decreasing with respect to a variable x_i *if for every set of values* $x_1, \ldots, x_{i-1}, x_i, x_i', x_{i+1}, \ldots, x_n$ *for which* $x_i \le x_i'$, *we have*

$$f(x_1, \ldots, x_{i-1}, x_i, x_{i+1}, \ldots, x_n) \ge f(x_1, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots, x_n).$$

- *We say that a function a function* $f(x_1, \ldots, x_n)$ *is* monotonic with respect to a variable x_i *if it is either strictly increasing or strictly decreasing with respect to this variable.*
- *We say that a function a function* $f(x_1, \ldots, x_n)$ *is* monotonic *if it is monotonic with respect to all its variables* $x_1, \ldots, x_n$.

Proposition. *The following problem is NP-hard:*

- *given: a cubic polynomial* $P(x_1, \ldots, x_n)$ *with rational coefficients and* n *intervals* $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n$ *with rational endpoints;*
- *check: whether the restriction of the polynomial* $P(x_1, \ldots, x_n)$ *to the box* $\boldsymbol{x}_1 \times \ldots \times \boldsymbol{x}_n$ *is monotonic.*

Proof. By definition, a problem is NP-hard if every problem from the class NP can reduced to this problem (see, e.g., [1, 3]). Thus, to prove that our problem is NP-hard, it is sufficient to prove that one of the known NP-hard problems can be reduced to our problem: indeed, in this case, every problem from the class NP can be reduced to the known NP-hard problem and thus, by transitivity of reduction, to our problem.

As such a known NP-hard problem, we will take a propositional satisfiability problem for 3-CNF propositional formulas, i.e., for Boolean expressions F of the type $F_1 \,\&\, \ldots \,\&\, F_k$, where each F_k has the form $a \vee b$ or $a \vee b \vee c$, and a, b, and c are *literals*, i.e., propositional variables $z_1, \ldots, z_v$ or their negations $\neg z_i$. An example of such a formula is $(z_1 \vee z_2 \vee \neg z_3) \,\&\, (\neg z_1 \vee z_2)$. A formula F is called *satisfiable* if there exist truth values of the corresponding variables $z_1, \ldots, z_v$ which make this Boolean expression true.

In Theorem 3.1 from [1], for each such propositional formula F, we built a quadratic polynomial $f_F(x_1, \ldots, x_n)$ of $n = v + k$ variables $x_i \in [0, 1]$ as follows:

- To each Boolean variable z_i, we put into correspondence a polynomial $f[z_i] = x_i$.
- To each literal $\neg z_i$, we put into correspondence an expression $f[\neg z_i] = 1 - z_i$.
- To each expression F_j of the type $a \vee b$ we put into correspondence an expression $f[F_j] = (f[a] + f[b] + x_{v+j} - 2)^2$.
- To each expression F_j of the type $a \vee b \vee c$ we put into correspondence an expression $f[F_j] = (f[a] + f[b] + f[c] + 2x_{v+j} - 3)^2$.

Finally, we define a quadratic polynomial of $n = v + k$ variables as

$$f_F(x_1, \ldots, x_n) = \sum_{i=1}^{v} x_i \cdot (1 - x_i) + \sum_{j=1}^{k} f[F_j].$$

In Theorem 3.1, we prove that:

- if the formula F is satisfiable, then the lower bound $\underline{f}_F$ of the function $f_F(x_1, \ldots, x_n)$ on the box $[0,1] \times \ldots \times [0,1]$ is equal to 0;
- if the formula F is not satisfiable, then $\underline{f}_F \geq 0.09$.

Each quadratic polynomial f_F is the sum of an expression $x_1 \cdot (1 - x_1)$ and several non-negative terms, So, for $x_1 = 0.5$, the value $f(x_1, \ldots, x_n)$ is greater than or equal to $0.5 \cdot (1 - 0.5) = 0.25$.

In our proof, we reduce this known NP-hard problem to the problem of checking monotonicity. Specifically, for each 3-CNF propositional formula F, we feasibly construct a cubic polynomial $P_F(x_1, \ldots, x_n, x_{n+1})$ which is monotonic on the box $[0,1] \times \ldots \times [0,1] \times [0,1]$ if and only if the formula F is not satisfiable.

This construction is as follows. For a quadratic polynomial $f_F(x_1, \ldots, x_n)$, each partial derivative is a linear function

$$\frac{\partial f_F}{\partial x_i} = a_i + \sum_{j=1}^{n} a_{ij} \cdot x_j.$$

For such a linear function, we can feasibly compute its range $\left[\underline{y}_i(F), \overline{y}_i(F)\right]$ for $x_i \in [0,1]$. Then, we can define the following cubic polynomial:

$$P_F(x_1, \ldots, x_n, x_{n+1}) = (f_F(x_1, \ldots, x_n) - 0.04) \cdot x_{n+1} - \sum_{i=1}^{n} \min\left(0, \underline{y}_i(F)\right) \cdot x_i.$$

Let us prove that the monotonicity of this polynomial is equivalent to $\underline{f}_F \geq 0.09$ and thus, to the fact that the formula F is not satisfiable.

If the formula F is satisfiable, i.e., if $\underline{f}_F = 0$, this means that $f_F(x_1, \ldots, x_n) = 0$ for some values $x_i \in [0,1]$. For these values $x_1, \ldots, x_n$, we have

$$\frac{\partial P_F}{\partial x_{n+1}} = f_F(x_1, \ldots, x_n) - 0.04 = -0.04 < 0,$$

and thus, the cubic function P_F is not increasing with respect to x_{n+1}. On the other hand, when $x_1 = 0.5$ and $f(x_1, \ldots, x_n) \geq 0.25$, we get

$$\frac{\partial P_F}{\partial x_{n+1}} = f_F(x_1, \ldots, x_n) - 0.04 \geq 0.25 - 0.04 = 0.21 > 0,$$

so the function P_F is not decreasing with respect to x_{n+1} either. Thus, the function P_F is not monotonic with respect to x_{n+1} and hence, not monotonic.

To complete the proof, it is therefore sufficient to show that if the formula F is not satisfiable, i.e., if $\underline{f}_F \geq 0.09$, then the cubic function P_F is indeed monotonic. Specifically, we prove that the function P_F is increasing with respect to all its variables, i.e., that all its derivatives are non-negative. Indeed, in this case, $f_F(x_1, \ldots, x_n) \geq 0.09$ for all x_i and thus,

$$\frac{\partial P_F}{\partial x_{n+1}} = f_F(x_1, \ldots, x_n) - 0.04 \geq 0.09 - 0.04 = 0.05 > 0.$$

For each i from 1 to n, we have

$$\frac{\partial P_F}{\partial x_i} = x_{n+1} \cdot \frac{\partial f_F}{\partial x_i} - \min\left(0, \underline{y}_i(F)\right). \tag{1}$$

To prove that this expression is always non-negative, let us consider two possible cases: $\underline{y}_i(F) \geq 0$ and $\underline{y}_i(F) < 0$.

In the first case, when $\underline{y}_i(F) \geq 0$, we have $\min\left(0, \underline{y}_i(F)\right) = 0$, so we need to prove that $x_{n+1} \cdot \dfrac{\partial f_F}{\partial x_i} \geq 0$. By definition, $\underline{y}_i(F)$ is the minimum of the derivative $\dfrac{\partial f_F}{\partial x_i}$; since this minimum is non-negative, the derivative $\dfrac{\partial f_F}{\partial x_i}$ is non-negative as well. Thus, the product of two non-negative numbers x_{n+1} and $\dfrac{\partial f_F}{\partial x_i}$ is non-negative.

In the second case, when $\underline{y}_i(F) < 0$, we have $\min\left(0, \underline{y}_i(F)\right) = \underline{y}_i(F)$. So, to prove that the expression (1) is non-negative, we need to prove that

$$x_{n+1} \cdot \frac{\partial f_F}{\partial x_i} \geq \underline{y}_i(F).$$

Indeed, by definition of $\underline{y}_i(F)$, we have $\dfrac{\partial f_F}{\partial x_i} \geq \underline{y}_i(F)$. Multiplying both sides of this inequality by a non-negative number x_{n+1}, we conclude that

$$x_{n+1} \cdot \frac{\partial f_F}{\partial x_i} \geq x_{n+1} \cdot \underline{y}_i(F).$$

On the other hand, since $x_{n+1} \in [0, 1]$, we know that $x_{n+1} \leq 1$. Multiplying both sides of this inequality by a negative number $\underline{y}_i(F)$, we conclude that $x_{n+1} \cdot \underline{y}_i(F) \geq \underline{y}_i(F)$. Thus, by transitivity, we conclude that indeed $x_{n+1} \cdot \dfrac{\partial f_F}{\partial x_i} \geq \underline{y}_i(F)$.

The reduction is proven, and so is the proposition.

Acknowledgments. This work was supported in part by the National Science Foundation grants HRD-0734825 and HRD-1242122 (Cyber-ShARE Center of Excellence) and DUE-0926721, by Grants 1 T36 GM078000-01 and 1R43TR000173-01 from the National Institutes of Health, and by a grant on F-transforms from the Office of Naval Research.

References

[1] Kreinovich, V., Lakeyev, A., Rohn, J., Kahl, P.: *Computational Complexity and Feasibility of Data Processing and Interval Computations*, Kluwer, Dordrecht, 1997.

[2] Moore, R. E., Kearfott, R. B., Cloud, M. J.: *Introduction to Interval Analysis*, SIAM Press, Philadelphia, Pennsylvania, 2009.

[3] Papadimitriou, C. H.: *Computational Complexity*, Addison Wesley, San Diego, 1994.