1-1-2024

Every Feasibly Computable Reals-to-Reals Function Is Feasibly Uniformly Continuous

Olga Kosheleva
The University of Texas at El Paso, olgak@utep.edu

Vladik Kreinovich
The University of Texas at El Paso, vladik@utep.edu

Comments:
Technical Report: UTEP-CS-24-01

Every Feasibly Computable Reals-to-Reals Function Is Feasibly Uniformly Continuous

Olga Kosheleva and Vladik Kreinovich

Abstract It is known that every computable function is continuous; moreover, it is computably continuous in the sense that for every $\varepsilon > 0$, we can compute $\delta > 0$ such that δ-close inputs lead to ε-close outputs. It is also known that not all functions which are, in principle, computable, can actually be computed: indeed, the computation sometimes requires more time than the lifetime of the Universe. A natural question is thus: can the above known result about computable continuity of computable functions be extended to the case when we limit ourselves to feasible computations? In this paper, we prove that this extension is indeed possible.

1 What We Do in This Paper

What is known. It is known that every computable reals-to-reals function is continuous; moreover, it is computably uniformly continuous on each bounded region; this was first proven in [3, 4] in the late 1950s; see, e.g., [5].

Why do we need to go beyond this result. Since the 1960s, we know that not everything that is, in principle, computable can actually be computed: if computations require more time than the lifetime of the Universe, we clearly cannot actually perform these computations. A special notion of *feasible* computations was introduced to denote computations that are, in principle, practically possible; see, e.g., [1, 2].

At present, the usual formalization of this notion is by equating feasible with polynomial time: an algorithm $Y = f(X)$ is feasible if for each input X, its computation time is bounded by a polynomial $P(\text{len}(X))$ of the length of the inputs.

Olga Kosheleva
Department of Teacher Education, University of Texas at El Paso
500 W. University, El Paso, Texas 79968, USA, e-mail: olgak@utep.edu

Vladik Kreinovich
Department of Computer Science, University of Texas at El Paso
500 W. University, El Paso, Texas 79968, USA, e-mail: vladik@utep.edu

It is desirable to see if the above result can be extended to the case when we limit ourselves to feasible computations.

What we do in this paper. In the paper, we prove that the above continuity result can indeed be extended to the feasible case.

The structure of this paper is as follows: in Section 2, we provide a brief reminder of what are computable functions and how we can formulate and prove the corresponding result. The desired extension to the feasible case is provided in Section 3.

2 Computable Reals-to-Reals Functions and Related Result: A Brief Reminder

What do we mean by a computable reals-to-reals function: towards a natural definition. By the very meaning of the word "computable", a computable real-valued function $f(x_1, \ldots, x_k)$ of k real-valued inputs is a function whose values can be computed based on the inputs. Such functions are used to process *data* $x_1, \ldots, x_k$.

The goal of this data processing is to estimate the value of some other quantity y which is related to the quantities $x_1, \ldots, x_n$ by the formula $y = f(x_1, \ldots, x_k)$. For example, we want to estimate tomorrow's temperature y at some spatial location based on the current values $x_1, \ldots, x_k$ of different meteorological quantities at this and nearby locations.

However, in the ideal world, data are the actual values of the corresponding physical quantities. The way we learn the values x_i is by measurement: either by direct measurements, or by processing the results of appropriate auxiliary measurements. It is therefore important to take into account that measurements are never absolutely accurate, they always have some accuracy – often described by the number of digits m in the corresponding binary representation, so that the accuracy is 2^{-m}. In other words, instead of knowing the actual values $a_1, \ldots, a_k$ of the corresponding quantities, we only know the measurement results $x_1, \ldots, x_k$ which are 2^{-m}-close to these values, i.e., for which $|x_i - a_i| \le 2^{-m}$ for all i from 1 to k. Since the known values x_i are only approximations to the actual values a_i, the result $f(x_1, \ldots, x_k)$ of data processing is only an approximation to the desired ideal value $f(a_1, \ldots, a_k)$.

We want to make sure that the result $y = f(x_1, \ldots, x_k)$ of data processing is close to the desired (ideal) value $b = f(a_1, \ldots, a_k)$, and we need to know what is the accuracy of the estimate y, i.e., how close is y to the desired value b: if we do not know this accuracy, i.e., if the difference $y - b$ can be arbitrarily large, then the estimate y is useless – since it does not impose any restrictions on a at all.

In practice, we want to estimate b with some given accuracy. For example, for temperature, with the accuracy of a few degrees. It may be that the existing accuracy with which we know x_k is not enough to achieve the desired accuracy of y – this happens when the sensors are not very accurate. In this case, to get the value b with the desired accuracy, we need to perform more accurate measurements – and we

need to make sure for some accuracy of measuring the inputs, we will get the desired accuracy in y.

Also, it is important to take into account that in practice, all physical quantities are bounded: speeds are bounded by the speed of light, distances on Earth are bounded by the size of the Earth, etc. So, it makes sense to consider functions defined on a box

$$[\underline{a}_1, \overline{a}_1] \times \ldots \times [\underline{a}_k, \overline{a}_k].$$

Thus, we arrive at the following definition.

What are computable reals-to-reals function: natural definition and the known result. Let us start with the definition.

Definition 1. *We say that a reals-to-reals function $f(a_1, \ldots, a_k)$ defined on a box*

$$[\underline{a}_1, \overline{a}_1] \times \ldots \times [\underline{a}_k, \overline{a}_k]$$

is computable *if there exists an algorithm that, for each tuple $a = (a_1, \ldots, a_k)$ of real numbers, given a natural number n, computes a 2^{-n}-approximation to the value $b = f(a_1, \ldots, a_k)$. In addition to the usual computational steps, this algorithm can ask, for each i and for each natural number j, for a 2^{-j}-approximation to the value a_i.*

Definition 2. *We say that a function $f(a_1, \ldots, a_k)$ is* computably uniformly continuous *if there is an algorithm that, given a natural number n, computes a natural number m for which, if $|a_i - a_i'| \le 2^{-m}$ of all i, then*

$$|f(a_1, \ldots, a_k) - f(a_1', \ldots, a_k')| \le 2^{-n}.$$

Proposition 1. *Every computable function is computably uniformly continuous.*

Proof. Let $f(a_1, \ldots, a_k)$ be a computable function, and let n be given. Let us show how we can now compute the desired value m.

For this purpose, for each $m = 0, 1, 2, \ldots$, we consider all possible tuples $a = (a_1, \ldots, a_k)$ in which each all the values a_i are proportional to 2^{-m}. To each such tuple, we apply the algorithm $f(a_1, \ldots, a_n)$. If for one of these tuples, the algorithm for computing the function $f(a_1, \ldots, a_n)$ asks for a 2^{-j}-approximation to one of the values a_i with $j > m$, we stop computations related to this m, and repeat the same procedure for the next value m, etc.

Let us prove that this procedure stops, i.e., that we will get m for which computations for each such tuple will finish without asking for more accurate approximations. For such m, the algorithm asks for only 2^{-j}-approximations with $j \le m$ – and for any tuple $(a_1, \ldots, a_k)$, these approximations can be made proportional to 2^{-m}.

Let us prove the existence of such m by contradiction. Let us assume that no such m exists. This means that for each m, there exists a tuple $a(m)$ for which providing the 2^{-m}-approximations is not enough. All these tuples $a(m)$ belong to a bounded box which is a compact set. It is known that from every sequence in a compact set we

can extract a converging subsequence $a(s_1), a(s_2), \ldots$ with $s_j \to \infty$. For the limit ℓ of this subsequence, the tuples $a(s_j)$ form the corresponding approximations. Thus, for this limit tuple ℓ, the algorithm $f(a_1, \ldots, a_m)$ will never stop:

- by definition of $a(s_1)$, this algorithm cannot stop by using only 2^{-s_1}-approximations;
- by definition of $a(s_2)$, it cannot stop by using only 2^{-s_2}-approximations, etc.

This contradicts to our definition of a feasibly computable function, according to which the function should be applicable to any tuple. This contradiction shows that the desired value m does exit. The proposition is proven.

3 New Result

Definition 3. *We say that a reals-to-reals function $f(a_1, \ldots, a_k)$ defined on a box*

$$[\underline{a}_1, \overline{a}_1] \times \ldots \times [\underline{a}_k, \overline{a}_k]$$

is feasibly computable *if there exists an algorithm that, for each tuple $(a_1, \ldots, a_k)$ of real numbers, given a natural number n, computes a 2^{-n}-approximation to the value $a = f(a_1, \ldots, a_k)$ in time bounded by some polynomial of n. In addition to the usual computational steps, this algorithm can ask, for each i and for each natural number j, for a 2^{-j}-approximation to the value a_i.*

Definition 4. *We say that a function $f(a_1, \ldots, a_k)$ is* feasibly uniformly continuous *if there is a feasible algorithm that, given a natural number n, computes a natural number m for which, if $|a_i - a_i'| \le 2^{-m}$ of all i, then*

$$|f(a_1, \ldots, a_k) - f(a_1', \ldots, a_k')| \le 2^{-n}.$$

Proposition 2. *Every feasibly computable function is feasibly uniformly continuous.*

Proof. By definition of a feasibly computable function, all the values j – for which the algorithm for computing this function requests for the 2^{-j}-approximation to the inputs – must be computed in time bounded by some polynomial of n. Generating each bit in the requested integer j requires at least one computational step. This means that the number of bits in j cannot exceed $P(b(n))$, where $b(n) \approx \log_2(n)$ is the length of the number n – i.e., the number of bits in the input m. Thus, if for $m = 2^{P(b(n))}$, we know the 2^{-m}-approximation to a_i, we can get answers to all such questions and compute the value $f(a_1, \ldots, a_k)$ for all possible inputs. So, the function $f(a_1, \ldots, a_k)$ is indeed feasibly uniformly continuous, for the value $m = 2^{P(b(n))}$ that can be feasibly computed from n. The proposition is proven.

Acknowledgments

This work was supported in part by the National Science Foundation grants 1623190 (A Model of Change for Preparing a New Generation for Professional Practice in Computer Science), HRD-1834620 and HRD-2034030 (CAHSI Includes), EAR-2225395 (Center for Collective Impact in Earthquake Science C-CIES), and by the AT&T Fellowship in Information Technology.

It was also supported by the program of the development of the Scientific-Educational Mathematical Center of Volga Federal District No. 075-02-2020-1478, and by a grant from the Hungarian National Research, Development and Innovation Office (NRDI).

References

1. V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Computational Complexity and Feasibility of Data Processing and Interval Computations*, Kluwer, Dordrecht, 1998.
2. C. Papadimitriou, *Computational Complexity*, Addison-Wesley, Reading, Massachusetts, 1994.
3. G. Tseytin, “Algorithmic operators in constructive complete separable metric spaces", *Doklady Akademii Nauk SSSR*, 1959, Vol. 128, pp. 49–52 (In Russian).
4. G. Tseytin, “Algorithmic operators in constructive metric spaces", *Trudy Matematicheskogo Instituta im. Steklova*, 1962, Vol. 67, pp. 295–361 (In Russian).
5. K. Weihrauch, *Computable Analysis*, Springer Verlag, Berlin, 2000.