4-1-2023

Wormholes, Superfast Computations, and Selivanov's Theorem

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-23-14

Wormholes, Superfast Computations, and Selivanov's Theorem

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

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

Abstract. While modern computers are fast, there are still many practical problems that require even faster computers. It turns out that on the fundamental level, one of the main factors limiting computation speed is the fact that, according to modern physics, the speed of all processes is limited by the speed of light. Good news is that while the corresponding limitation is very severe in Euclidean geometry, it can be more relaxed in (at least some) non-Euclidean spaces, and, according to modern physics, the physical space is not Euclidean. The differences from Euclidean character are especially large on micro-level, where quantum effects need to be taken into account. To analyze how we can speed up computations, it is desirable to reconstruct the actual distance values – corresponding to all possible paths – from the values that we actually measure – which correspond only to macro-paths and thus, provide only the upper bound for the distance. In our previous papers – including our joint paper with Victor Selivanov – we provided an explicit formula for such a reconstruction. But for this formula to be useful, we need to analyze how algorithmic is this reconstructions. In this paper, we show that while in general, no reconstruction algorithm is possible, an algorithm *is* possible if we impose a lower limit on the distances between steps in a path. So, hopefully, this can help to eventually come up with faster computations.

Keywords: superfast computations, wormholes, constructive mathematics, metric, interval uncertainty

1. Introduction

We need faster computers. While modern computers are several orders of magnitude faster than several decades ago, there are still computational problems for which the current computer speed is not sufficient. For example, it is potentially possible to predict, with reasonable accuracy, in what direction a tornado will turn in the next 15 minutes – the computations are similar to how we predict tomorrow's weather – but even on the fastest high-performance computers, these predictions will take several hours, making these computations useless.

There are physical limits to the computer speed. Why cannot we design faster computers? There are technological limits – that will be hopefully overcome, but there are also fundamental physical limits. One of such limits is related to the fact that, according to relativity theory (see, e.g., [8, 22]), communications cannot be faster than the speed of light c. (Why this limits computation speed is explained, in some detail, in Section 2.)

So how can we make computer faster? This limitation on computation speed is based on the assumption that locally, space has the usual geometry (close to Euclidean) and the usual topology (of a simply connected region in Euclidean space). Good news is that, according to modern physics, space is not Euclidean, and, due to quantum effects, this non-Euclidean character becomes prevalent on very small spatial scales. In principle, using such non-Euclidean character can drastically speed up computations.

Let us describe all this in precise terms. When we measure the spatial distance $d(a, b)$ between the two points, what we actually measure is how close these points are if we use techniques corresponding to the level of such distances. If we take into account microstructure of space, we may be able to get their faster, and this can lead to faster computations.

There is a simple commonsense analogy of this phenomena: if we measure a distance between two locations in two rooms separated by a wall, a natural way to measure it is to walk from one room to another: this is fastest we macro-objects can go. However, molecules can easily go through the walls – so for them the distance will be smaller.

In mathematical terms, what we measure is not the actual distance $d(a,b)$, but the upper bound $\overline{d}(a,b)$ on this distance. Based on these upper bounds, we would like to reconstruct the actual distance – i.e., the length of the shortest path that we can find between a and b if we use connections of all types.

What is known. In our previous work [2, 10] – including our joint work with Victor Selivanov [10] – we provided an explicit formula for such reconstruction.

Remaining question. But can this formula help? For this formula to be helpful, we need to have an algorithm reconstructing the actual distance from the measured upper bounds.

What we do in this paper. In this paper, we show that:

- while no general reconstruction algorithm is possible,
- a reconstruction algorithm is possible if we impose a lower limit on the distances between steps in a path.

Hopefully, this can help to eventually come up with faster computations.

The structure of this paper. In Section 2, we provide a detailed explanation of physics-based limitations on computer speed. In Section 3, we use this explanation to analyze how physics can help speed up computations. In Section 4, we formulate the corresponding problem in precise terms, mention the previously known result, and describe our new results about the computational aspects of metric reconstruction.

2. Physics-Based Limitation on Computer Speed: A Brief Reminder

Need for parallelization. When we have a task – e.g., cleaning the building at the end of the working day – that would take too long for one person, a natural idea is to divide this task between several people working in parallel. Similarly, if a computational task takes too much time on a single processor, a natural way to speed up computations is to have several processors working in parallel. This is how high performance computers work: they consist of thousands of processors working in parallel.

Comment. It should be mentioned that at present, this is the main way the computer speed increases: the computation speed of a single processor has remained about the same for some time already.

Parallelization: an optimistic viewpoint. The possibilities of such parallelization are not yet exhausted: new high-performance computers are still being designed that include more processors and that will further increase the resulting computation speed. So, at first glance, it may seem that this way, we can achieve a potentially unlimited speed-up.

Unfortunately, there is a limit on how parallelization can speed up computations. Let us show that a reasonably simple analysis reveals that the speed of light imposes a restriction on the resulting computer speed; see, e.g., [18].

Indeed, suppose that to perform a computational task on a single processor, we need time T_s. How fast can we compute this task if we have a potentially unlimited number of processors working in parallel? Let us assume that we have a parallel computer that finishes this task in time T_p. This means that we start at some location, and get the computation result at the same location after time T_p. During this time, only processors located at the distance $\leqslant c \cdot T_p$ could influence the result: indeed, processors located further away would require time larger than T_p to make sure that their signals affect the observed computation result. Thus, only processor within the ball of radius $R = c \cdot T_p$ can participate in this computation.

How many processors N can we fit into this ball? The volume V of this ball is equal to

$$V = \frac{4}{3} \cdot \pi \cdot R^3 = \frac{4}{3} \cdot \pi \cdot c^3 \cdot T_p^3.$$

Let us denote the smallest possible volume of a processor (that we can design based on the current technology) by ΔV. The volume occupied by all these processors is greater than or equal to $N \cdot \Delta V$. Clearly, this volume cannot exceed V, so we must have $N \cdot \Delta V \leqslant V$ and thus

$$N \leqslant \frac{V}{\Delta V} = \frac{4 \cdot \pi \cdot c^3}{3 \cdot \Delta V} \cdot T_p^3. \tag{1}$$

During the time T_p, each of these N processors performs at most $T_p/\Delta t$ computational steps, where we denoted the shortest time of a single computational operation (within given technology) by Δt. Each parallel computation that requires S steps on each of N processors can be naturally simulated on a single processor:

- first, we simulate the first step of the 1st processor, then the 1st step of the second processor, etc.;
- then, we simulate the second step of the 1st processor, then the 2nd step of the second processor, etc.

Each step of parallel computations can be thus simulated by N sequential steps of a single processor. Thus, the computation time on a single processor is increased by no more than a factor of N: a task that required time T_p on a parallel computer takes time $t_s \leqslant N \cdot T_p$ on a sequential computer. From (2), we therefore conclude that we can perform the original task on a sequential computer in time t_s bounded by

$$t_s \leqslant N \cdot T_p \leqslant \frac{4 \cdot \pi \cdot c^3}{3 \cdot \Delta V} \cdot T_p^3 \cdot T_p = C \cdot T_p^4,$$

where we denoted

$$C \stackrel{\text{def}}{=} \frac{4 \cdot \pi \cdot c^3}{3 \cdot \Delta V}.$$

We know that computing the given task on a sequential computer requires at least time T_s, thus we have $T_s \leqslant t_s$, i.e., $T_s \leqslant C \cdot T_p^4$. Therefore, the parallel computation time T_p is indeed limited from below:

$$T_p \geqslant C^{-1/4} \cdot T_s^{1/4}.$$

3. So How Can We Speed Up Computations: Physical Analysis

Physical assumptions that lead to the above restriction on computation speed. The above restriction is based on two physical assumptions:

- that all communication are limited by the speed of light, and
- the space in Euclidean, so the volume of the ball of radius R is equal to

$$\frac{4}{3} \cdot \pi \cdot R^3.$$

So, to find out how we can speed up computations, it is reasonable to check if these two assumptions can be violated. Let us analyze these two restrictions one by one.

Can communications be faster than speed of light? Maybe. The reason why there is a restriction on the speed of light is that in relativity theory, allowing communications which are faster that the speed of light will lead to going to the past [8, 22]. While physicists seriously consider the possibility of travel to the past – see, e.g., a book [21] by a Nobelist Kip Thorne – at this moment, these are mostly theoretical speculations, not yet supported by any experimental evidence.

Can space be non-Euclidean? Absolutely, physical space actually is non-Euclidean. In contrast, the assumption that the space is Euclidean is known to be only an approximation. According to general relativity theory, the actual space is curved, i.e., its geometry is different from Euclidean [8, 16, 22] – this is actually one of the main ideas behind General Relativity.

This non-Euclidean character of space can potentially speed up computations. Since the above restriction on computation speed is based in the assumption that the space is Euclidean, it is reasonable to conjecture that in non-Euclidean spaces – at least in some of them – we can speed up computations. And indeed, many non-Euclidean spaces can potentially lead to an exponential speed-up. For example, in the Lobachevsky space (historically the first non-Euclidean space), the volume of a ball of radius R grows exponentially with R, so we can fit exponentially many processors into the ball and thus, reach an exponential speedup; see, e.g., [12, 18].

The real space is not exactly Lobachevsky one, but there are other examples of physically meaningful non-Euclidean spaces in which a drastic computational speed up is possible; see, e.g., [18].

Quantum effects add to the non-Euclidean character of physical space. The non-Euclidean character of space is also occurs for very short distance, when quantum effects happen. At sufficiently small distances, space-time becomes what is called a *foam* [16], where many points are connected by so-called *wormholes* [21], short "tunnels" traveling through which requires much shorter distance than what we measure on the macro-level (i.e., without taking these wormholes into account).

If we could use these microscopic wormholes to send signals, we would thus be able to effectively reduce the distances between processors, reduce communication time, and thus, speed up computations.

4. Resulting Mathematical Problem: Formulation, Known Results, New Results

Mathematical formulation of the problem. Let us assume that we have a set X – its elements be called *spatial points*, or, alternatively, *spatial locations*.

- *What we know:* for each pair of points $a, b \in X$, we know the value $\overline{d}(a, b) > 0$.
- *What we want:* we want to compute, for each pair of point $a, b \in X$, the value

$$d(a, b) \stackrel{\text{def}}{=} \inf \left\{ \sum_{i=1}^{m-1} \overline{d}(a_0, a_{i+1}) \right\}, \tag{2}$$

where the infimum is taken over all chains $a_0, a_1, \ldots, a_m$ for which $a_0 = a$ and $a_m = b$.

Comment. In [2], we have shown that the function $d(a, b)$ described by the expression (2) is:

- symmetric, i.e., $d(a, b) = d(b, a)$ and
- satisfies the triangle inequality $d(a, c) \leqslant d(a, b) + d(b, c)$.

Thus, $d(a, b)$ is what is called a *pseudometric*. When we also have $d(a, b) > 0$ for all $a \neq b$, this becomes a metric.

Towards the formulation of the computational problem. The formula (2) implicitly assumes that for all a and b, we know the exact value of the macro-distance $\overline{d}(a, b)$. However, our information about each value $\overline{d}(a, b)$ come from measurements, and measurements are never 100% accurate, there is always a difference between the measurement result $\widetilde{d}(a, b)$ and the actual value $\overline{d}(a, b)$ of the corresponding quantity; see, e.g., [20]. In many practical situations, the only information that we have about this difference is the upper bound Δ on its absolute value. In this case, once we get the measurement result $\widetilde{d}(a, b)$, the only information that we have about the actual value $\overline{d}(a, b)$ is that it is contained in the interval $[\widetilde{d}(a, b) - \Delta, \widetilde{d}(a, b) + \Delta]$. Such *interval uncertainty* is ubiquitous in practice; see, e.g., [9, 13, 15, 17].

At each moment of time, we only know the values $\overline{d}(a, b)$ with some accuracy Δ. We can perform the measurements with higher and higher accuracy, but a question is: will there be a stage at which, based on this information, we will be able to reconstruct at least some value $d(a, b)$ with a desired accuracy ε?

First (negative) result. The following negative result shows that, in general, reconstruction is not algorithmically possible: namely, no matter with what accuracy Δ we know the value $\overline{d}(a,b)$, all we can conclude about the reconstructed value $d(a,b)$ is that this value is somewhere between 0 and $\overline{d}(a,b)$. We will show that this is true even for metrics which are close to the Euclidean metric $d_E(a,b)$ on an n-dimensional space $\mathbb{R}^n$.

Proposition 1. *Let n be an integer, let a_0, b_0 be two points in $\mathbb{R}^n$, let d_0 be a positive number between 0 and $d_E(a_0,b_0)$, and let $\Delta > 0$. Then, there exists a function $\overline{d}(a,b)$ for which:*

- *for all a and b, we have $\overline{d}(a,b) \in [d_E(a,b) - \Delta, d_E(a,b) + \Delta]$, and*
- *for the function $d(a,b)$ described by the formula (2), we have*

$$d(a_0,b_0) = d_0.$$

Proof. Let us denote $\alpha \overset{\text{def}}{=} d_0/d_E(a_0,b_0)$, and let us define the following function $\overline{d}(a,b)$:

- for the points a and b for which $d(a,b) > \Delta$, we take $\overline{d}(a,b) = d_E(a,b)$, and
- for the points a and b for which $d(a,b) \leqslant \Delta$, we take $\overline{d}(a,b) = \alpha \cdot d_E(a,b)$.

One can see that this function is indeed Δ-close to $d_E(a,b)$.

The shortest distance $d(a,b)$ corresponding to this function $\overline{d}(a,b)$ is obtained when all the pairs in the chain a_i are Δ-close, so we have $d(a,b) = \alpha \cdot d_E(a,b)$ for all a and b. In particular, for $a = a_0$ and $b = b_0$, we have $d(a_0,b_0) = \alpha \cdot d_E(a_0,b_0)$. By definition of α, this means that $d(a_0,b_0) = d_0$. The proposition is proven.

Second (positive) result. Let us now prove that the problem of reconstructing the metric $d(a,b)$ becomes algorithmic if, in the formula (2), we limit ourselves to chains in which there is a lower bound δ on each distance $d(a_i,a_{i+1})$. Of course, this modification of the definition (2) only makes sense when $\overline{d}(a,b) > \delta$: for points a and b for which $\overline{d}(a,b) \leqslant \delta$, already the chain $a_0 = a$ and $a_1 = b$ provides a better estimate than any chain with longer links.

To formulate (and prove) this result, we use constructive (computable) mathematics; see, e.g., [1, 3–7, 11, 14, 19, 23], in which it is defined what it means for a mathematical objects is computable.

Let us recall the definitions of computability that we will use. A real number x is called *computable* if there exists an algorithm that, given a natural number n, returns a rational number r_n for which $|x - r_n| \leqslant 2^{-n}$. A point in an n-dimensional space $\mathbb{R}^n$ is called *computable* if all its coordinates are computable.

We say that a compact set $S \subseteq \mathbb{R}^n$ is *computably compact* if there is an algorithm that, given a natural number n, returns a finite list of computable elements of S such that every elements from S is 2^{-n}-close to one of the elements from this finite list.

A function $f(x)$ from a computably compact set $S \subseteq \mathbb{R}^n$ is called *computable* if the following two algorithms exist:

- an algorithm that, given a point $x \in S$ with rational coordinates and a natural number n, returns a rational number y_n for which $|f(x) - y_n| \leqslant 2^{-n}$, and
- an algorithm that, given a natural number n, returns a natural number m such that for all $a, b \in S$ for $d_E(a,b) \leqslant 2^{-m}$, we have $|f(a) - f(b)| \leqslant 2^{-n}$.

There are known algorithms for computing minimum and maximum of a computable function.

Comment. As we have mentioned, in this paper, we only cite definitions that are needed for our purpose. In line with this plan, what we have just described is the definition of a computable function of a computably compact set. This is all we need in this paper; for clarity, it should be mentioned that in computable mathematics, there exist more general definitions for functions defined on more general sets.

Proposition 2. *Let n be an integer, let $S \subseteq \mathbb{R}^n$ be a computably compact set, let $\overline{d}(a,b)$ be a computable function for $a,b \in S$, and let $\delta > 0$ be a computable number. Then, for each pair of computable elements $a,b \in S$ for which $\overline{d}(a,b) \geqslant \delta$ and for each desired accuracy ε, we can compute a number $D(a,b)$ which is ε-close to the value*

$$d(a,b) \stackrel{\text{def}}{=} \inf \left\{ \sum_{i=1}^{m-1} \overline{d}(a_0, a_{i+1}) \right\}, \tag{3}$$

where the infimum is taken over all chains $a_0, a_1, \ldots, a_m$ for which $a_0 = a$, $a_m = b$, and $d_E(a_i, a_{i+1}) \geqslant \delta$ for all i.

Proof. For each a and b for which $d(a,b) \geqslant \delta$, by definition, we have $d(a,b) \leqslant \overline{d}(a,b)$. Since each distance $\overline{d}(a_i, a_{i+1})$ in an m-element chain is larger than or equal to δ, the whole length of this chain is larger than or equal to $m \cdot d$. Since we are looking for the infimum, it makes no sense to consider chains for which $m \cdot \delta > \overline{d}(a,b)$. Thus, it is sufficient to only consider chains for which $m \cdot \delta \leqslant \overline{d}(a,b)$, i.e., for which $m \leqslant \overline{d}(a,b)/\delta$.

For each possible m, as one can easily check, the set of all corresponding chain is computably compact, and the chain's length is a computable function of a tuple consisting of the chain's elements. So, we can compute the smallest length over all chains of length m. Then, to find the desired minimu, we take the smallest of the minima corresponding to the chains of all lengths $m \cdot \delta \leqslant \overline{d}(a,b)$. The proposition is proven.

Comment. A similar result holds if instead of subsets of Euclidean space, we consider functions on general computably compact sets. In general, a metric space (X, ρ) with metric $\rho(a,b)$ is called *computable* if there is a dense sequence of elements x_n in this set for which there is an algorithm that, given natural numbers n and m, computes the distance $\rho(x_n, x_m)$. (For real numbers, rational numbers form such a sequence.)

Other definitions are similar to definitions related to real numbers and Euclidean space, but with elements x_n instead of rational numbers or points with rational coefficients. An element x of the metric space is called computable if for every natural number n there exists a natural number m for which $\rho(x, x_m) \leqslant 2^{-n}$. A function $f(x)$ from a computably compact metric space X is called *computable* if the following two algorithms exist:

- an algorithm that, given natural numbers n and m, returns a rational number y_n for which $|f(x_m) - y_n| \leqslant 2^{-n}$, and
- an algorithm that, given a natural number n, returns a natural number m such that for all $a,b \in S$ for $\rho(a,b) \leqslant 2^{-m}$, we have $|f(a) - f(b)| \leqslant 2^{-n}$.

There are known algorithms for computing minimum and maximum of a computable function.

In this more general case, we also have a statement similar to Proposition 2, but slightly different:

Proposition 3. *Let n be an integer, let (X, ρ) be a computably compact metric space, let $\overline{d}(a,b)$ be a computable function for $a,b \in S$,and let $0 < \delta_- < \delta_+$ be computable numbers. Then, for each pair of computable elements $a,b \in S$ for which $\overline{d}(a,b) \geqslant \delta_+$, we can compute a number $\delta \in (\delta_-, \delta_+)$ for which, for each desired accuracy ε, we can compute a number $D(a,b)$ which is ε-close to the value*

$$d(a,b) \stackrel{\text{def}}{=} \inf \left\{ \sum_{i=1}^{m-1} \overline{d}(a_0, a_{i+1}) \right\}, \tag{4}$$

where the infimum is taken over all chains $a_0, a_1, \ldots, a_m$ for which $a_0 = a$, $a_m = b$, and $\rho(a_i, a_{i+1}) \geqslant \delta$ for all i.

Proof. The proof is similar to the proof of Proposition 2, the only difference is that we need to select a number $\delta \in (\delta_-, \delta_+)$ for which the set of chains is computably compact. This selection is made possible by a result from [4, 5] according to which for each computable function $F(x)$ on computably compact metric space and for each pair $\delta_- < \delta_+$ of computable real numbers, we can compute a value $\delta \in (\delta_-, \delta_+)$ for which the set $\{x : F(x) \geqslant \delta\}$ is computably compact.

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, 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] O. Aberth, *Precise Numerical Analysis Using C++*, Academic Press, New York, 1998.

[2] M. Afravi, V. Kreinovich, and T. Dumrongpokaphan, "Metric spaces under interval uncertainty: towards an adequate definition", In: G. Sidorov and O. Herrera-Alcántara (Eds.), *Advances in Computational Intelligence: Proceedings of the 15th Mexican International Conference on Artificial Intelligence MICAI'2016, Cancun, Mexico, October 25–29, 2016, Part I*, Springer Lecture Notes in Artificial Intelligence, Cham, Switzerland, 2017, Part I, Vol. 10061.

[3] M. J. Beeson, *Foundations of constructive mathematics*, Springer-Verlag, N.Y., 1985.

[4] E. Bishop, *Foundations of Constructive Analysis*, McGraw-Hill, 1967.

[5] E. Bishop and D. S. Bridges, *Constructive Analysis*, Springer, N.Y., 1985.

[6] D.S. Bridges, *Constructive Functional Analysis*, Pitman, London, 1979.

[7] D. S. Bridges and S. L. Vita, *Techniques of Constructive Analysis*, Springer-Verlag, New York, 2006.

[8] R. Feynman, R. Leighton, and M. Sands, *The Feynman Lectures on Physics*, Addison Wesley, Boston, Massachusetts, 2005.

[9] L. Jaulin, M. Kiefer, O. Didrit, and E. Walter, *Applied Interval Analysis, with Examples in Parameter and State Estimation, Robust Control, and Robotics*, Springer, London, 2001.

[10] V. Kreinovich, O. Kosheleva, and V. Selivanov, *Kinematic Metric Spaces Under Interval Uncertainty: Towards an Adequate Definition*, University of Texas at El Paso, Department of Computer Science, Technical Report UTEP-CS-21-100, 2021, https://www.cs.utep.edu/vladik/2021/tr21-100.pdf

[11] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl, *Computational complexity and feasibility of data processing and interval computations*, Kluwer, Dordrecht, 1998.

[12] V. Kreinovich and M. Margenstern, "In some curved spaces, one can solve NP-hard problems in polynomial time", *Notes of Mathematical Seminars of St. Petersburg Department of Steklov Institute of Mathematics*, 2008, Vol. 358, pp. 224–250; reprinted in *Journal of Mathematical Sciences*, 2009, Vol. 158, No. 5, pp. 727–740.

[13] B. J. Kubica, *Interval Methods for Solving Nonlinear Constraint Satisfaction, Optimization, and Similar Problems: from Inequalities Systems to Game Solutions*, Springer, Cham, Switzerland, 2019.

[14] B. A. Kushner, *Lectures on Constructive Mathematical Analysis*, Amer. Math. Soc., Providence, Rhode Island, 1984.

[15] G. Mayer, *Interval Analysis and Automatic Result Verification*, de Gruyter, Berlin, 2017.

[16] C. W. Misner, K. S. Thorne, and J. A. Wheeler, *Gravitation*, W. H. Freeman, New York, 1973.

[17] R. E. Moore, R. B. Kearfott, and M. J. Cloud, *Introduction to Interval Analysis*, SIAM, Philadelphia, 2009.

[18] D. Morgenstein and V. Kreinovich, "Which algorithms are feasible and which are not depends on the geometry of space-time", *Geombinatorics*, 1995, Vol. 4, No. 3, pp. 80–97.

[19] M. Pour-El and J. Richards, *Computability in Analysis and Physics*, Springer-Verlag, New York, 1989.

[20] S. G. Rabinovich, *Measurement Errors and Uncertainty: Theory and Practice*, Springer Verlag, New York, 2005.

[21] K. S. Thorne, *Black Holes and Time Warps: Einstein's Outrageous Legacy*, W. W. Norton, New York, 1995.

[22] K. S. Thorne and R. D. Blandford, *Modern Classical Physics: Optics, Fluids, Plasmas, Elasticity, Relativity, and Statistical Physics*, Princeton University Press, Princeton, New Jersey, 2021.

[23] K. Weihrauch, *Computable Analysis*, Springer Verlag, Berlin, 2000.