Computable Numbers, Computable Sets, Computable Functions, And How It Is All Related to Interval Computations

Vladik Kreinovich

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

http://www.cs.utep.edu/vladik

1. What Is a Computable Number

- From the physical viewpoint, real numbers x describe values of different quantities.
- We get values of real numbers by measurements.
- Measurements are never 100% accurate, so after a measurement, we get an approximate value r_k of x.
- In principle, we can measure x with higher and higher accuracy.
- So, from the computational viewpoint, a real number is a sequence of rational numbers r_k for which, e.g.,
$$|x - r_k| \leq 2^{-k}.$$
- By an algorithm processing real numbers, we mean an algorithm using r_k as an "oracle" (subroutine).
- This is how computations with real numbers are defined in *computable analysis*.

2. Relation to Interval Analysis

- Once we know:
  - the measurement result $\widetilde{x}$ and
  - the upper bound Δ on the measurement error $\Delta x \stackrel{\text{def}}{=} \widetilde{x} - x$,

  we can conclude that the actual value x belongs to the interval $[\widetilde{x} - \Delta, \widetilde{x} + \Delta]$.
- In interval analysis, this is all we know:
  - we performed measurements (or estimates),
  - we get intervals, and
  - we want to extract as much information as possible from these results.
- In particular, we want to know what can we conclude about $y = f(x_1, \ldots, x_n)$, where f is a known algorithm.

3. Computable vs. Interval Analysis (cont-d)

- In computable (constructive) analysis:
  - we take into account that eventually,
  - we will be able to measure each x_i with higher and higher accuracy.
- In other words, for each quantity,
  - instead of a *single* interval,
  - we have a *sequence* of narrower and narrower intervals,
  - a sequence that eventually converging to the actual value.
- "*Interval analysis is applied constructive analysis*" (Yuri Matiyasevich, of 10th Hilbert problem fame).

4. Constructive vs. Computable Analysis

- There is a subtle difference between *constructive analysis* and *computable analysis*.
- Crudely speaking, constructive analysis only considers objects that can be algorithmically constructed.
- E.g., we only allow computable real numbers.
- In contrast, computable analysis takes into account that inputs can be non-computable.
- E.g., measurement results are often random.
- Computable analysis checks what we can compute based on such – possibly non-computable – inputs.

5. Computable Analysis: Typical Questions

- In general:
  - once we know x_i with more and more accuracy,
  - we can usually find $y = f(x_1, \ldots, x_n)$ with more and more accuracy.
- This means that the corresponding function $f(x_1, \ldots, x_n)$ is computable.
- One of the possible questions is: which questions about $y = f(x_1, \ldots, x_n)$ we will be able to eventually answer?
- Example: if $y > 0$, then we will eventually be able to confirm this.
- On the other hand, no matter how accurately we measure, we will never be able to check whether $y = 0$.

6. What Is a Computable Set

- In a computer, we can only store finitely many objects – i.e., a finite set, with computable distances.
- It is therefore reasonable to define a computable set as a set S that:
  - can be algorithmically approximated, with any given accuracy,
  - by finite sets.
- Approximated means that every $x \in S$ is 2^{-n}-close to one of the elements from the approximating finite set.
- Elements of these finite sets approximate our set with higher and higher accuracy.

7. What Is a Computable Set (cont-d)

- A computer has a linear memory, so it is convenient to place these elements into an infinite sequence

$$x_1, \quad x_2, \ldots$$

- Elements from this sequence approximate any element from the given set.
- Thus, this sequence must be *everywhere* dense in this set.
- In practice, we do not know the exact values of the elements.
- We only have approximations to elements of the set.
- Based on these approximations, we can never know whether the resulting set is closed or not.
- For example, whether a set of real numbers is the interval $[-1, 1]$ or the same interval minus 0 point.

8. What Is a Computable Set (final)

- To ignore such un-detectable differences, it is reasonable to assume:
  - that our set is *complete*,
  - i.e., that it includes the limit of each converging sequence.
- Thus, we arrive at the following definition.

9. What is a Computable Set: Definition

- By a *computable set*, we mean a complete metric space with an everywhere dense sequence $\{x_i\}$ for which:
    - $\exists$ an algorithm that, given i and j, computes the distance $d(x_i, x_j)$ (with any given accuracy), and
    - There exists an algorithm that:
        - given a natural number n,
        - returns a natural number $N(n)$ for which every point $x_1, x_2, \ldots$ is 2^{-n}-close to one of the points

$$x_1, \ldots, x_{N(n)}.$$

- By a *computable element* x of a computable set, we mean an algorithm that:
    - given a natural number n,
    - returns an integer $i(n)$ for which $d(x, x_{i(n)}) \leq 2^{-n}$.

10. What is a Computable Function: Intuitive Idea

- A computable function f should be able:
  - given a computable real number (or, more generally, a computable element of a computable set),
  - to compute the value $f(x)$ with any given accuracy.
- Computable elements x are given by their approximations.
- Thus, to compute $f(x)$ with a given accuracy 2^{-n}, we need to:
  - determine how accurately we need to compute x to achieve the desired accuracy 2^{-n} in $f(x)$,
  - then use the corresponding approximation to x to compute the desired approximation to $f(x)$.
- So, we arrive at the following definition.

11. What Is a Computable Function: Definition

- We say that a function $f(x)$ from a computable set to real numbers is computable if:
  - first, we have an algorithm that, given n, returns m for which $d(x,x') \le 2^{-m}$ implies that
    $$|f(x) - f(x')| \le 2^{-n}, \text{ and}$$
  - second, we have an algorithm that, given i, computes $f(x_i)$.
- The existence of m for every n is nothing else but uniform continuity.
- So, in effect, we want $f(x)$ to be effectively uniformly continuous.

12. Examples of Positive Results

- We can algorithmically compute the maximum M of a computable function $f(x)$ on a computable set X:
  - for given $\varepsilon > 0$, we know what accuracy $\delta > 0$ we need for x to get $f(x)$ with accuracy ε;
  - so, we find a δ-net $\{x_1, \ldots, x_n\} \subseteq X$,
  - then $\max(f(x_1), \ldots, f(x_n))$ is the desired ε-approximation to M.
- A more complex example: for every computable function $f(x)$ on a computable set:
  - for every four rational numbers $\underline{r}_1 < \overline{r}_1 < \underline{r}_2 < \overline{r}_2$,
  - we can algorithmically find values $b_1 \in (\underline{r}_1, \overline{r}_1)$ and $b_2 \in (\underline{b}_2, \overline{b}_2)$ for which

$$\{x : b_1 \le f(x) \le b_2\} \text{ is a computable set.}$$

13. Examples of Negative Results

- No algorithm is possible that, given two numbers x and y, would check whether $x = y$.
- This follows from the halting problem: it is not possible to check whether a given algorithm halts on given data.
- More complex examples:
  - No algorithm is possible that, given f, returns x such that $f(x) = 0$.
  - No algorithm is possible that, given f, returns x s.t. $f(x) = \max\limits_{y \in K} f(y)$ (but $\max\limits_{y \in K} f(y)$ *is* computable.)
  - No algorithm is possible that, given f, returns x such that $f(x) = x$.