Representations for feasibly approximable functions

Michal Konečný Eike Neumann
Aston University, Birmingham, United Kingdom
{m.konecny, neumaef1}@aston.ac.uk

Given a continuous real function $f\colon [-1,1] \to \mathbb{R}$, two of the most basic computational tasks are the computation of its integral and its range. Both problems are generally perceived to be "easy" by practitioners (given that the domain is one-dimensional). Hence it was surprising when Ko and Friedman [4] proved that these problems are "hard" in a well-defined sense: the maximisation operator

$$\mathrm{Max}\colon C([-1,1]) \to C([-1,1]),\ f \mapsto \lambda x.\, \max_{t \leq x} f(t)$$

maps polytime computable functions to polytime computable functions if and only if P = NP, and the integration operator

$$\mathrm{Int}\colon C([-1,1]) \to C([-1,1]),\ f \mapsto \lambda x.\, \int_{-1}^{x} f(t)\,\mathrm{dt}$$

maps polytime computable functions to polytime computable functions if and only if P = #P.

In contrast, Müller [6] showed that both operators map polytime computable *analytic* functions to polytime computable functions. Uniform generalisations of this were obtained by Labhalla, Lombardi, and Moutai [5], and by Kawamura, Müller, Rösnick, and Ziegler [2]: In [5] it is shown that both the integration functional and the maximisation functional[1] are uniformly polytime computable on every fixed level of the Gevrey-hierarchy, and in [2] it is shown that the functionals are uniformly polytime computable on the whole hierarchy, if the parameters which control the growth of the derivatives are provided as complexity parameters.

Polytime computable Gevrey functions correspond to *feasibly polynomially approximable* functions, in the sense that for any given Gevrey function one can uniformly in polynomial time compute a sequence of polynomials with dyadic rational coefficients which converge fast to the given function, and vice versa (see [2, Theorem 2.3 (b)] and [5, Théorème 5.2.7]). This is a natural starting point for generalising the results to more general classes of functions - i.e. feasibly approximable functions with respect to different choices of approximants. Natural choices beyond polynomials include piecewise polynomials and rational functions. We will work in the context of second-order polytime computability as introduced by Kawamura and Cook [1]. As in [5] we will study feasibly approximable functions as polytime computable points with respect to a Cauchy representation.

Definition 1. We introduce the following representations of the space $C([-1,1])$:

1. Let Fun denote the standard representation of $C([-1,1])$, where a function is represented by its values on dyadic rational numbers and a modulus of continuity.

2. Let Poly denote the Cauchy representation of $C([-1,1])$ which is induced by the discrete subspace $\mathbb{D}[x]$ of polynomials with dyadic rational coefficients.

[1] i.e. the uncurried versions of the operators

3. Let PPoly denote the Cauchy representation of $C([-1,1])$ which is induced by the discrete subspace of continuous piecewise polynomials with dyadic rational coefficients and dyadic rational breakpoints.

4. Let Frac denote the Cauchy representation of $C([-1,1])$ which is induced by the discrete subspace of rational functions $R(x) = P(x)/Q(x)$ with dyadic rational coefficients, where the denominator is minorised by 1, i.e. $Q(x) \geq 1$ for all $x \in [-1,1]$.

5. Let PFrac denote the Cauchy representation of $C([-1,1])$ which is induced by the discrete subspace of continuous piecewise rational functions with dyadic rational coefficients and breakpoints, where all denominators are minorised by 1.

The relationship between the introduced representations with respect to polytime reducibility has first been studied in [5], albeit in a somewhat different framework. They leave the precise relationship between Fun and Frac, and between Frac and PPoly as an open question. We can give a complete overview:

Theorem 2. *We have second-order polytime reductions and equivalences*

$$\mathrm{Poly} < \mathrm{PPoly} \equiv \mathrm{Frac} \equiv \mathrm{PFrac} < \mathrm{Fun}\,.$$

This motivates the study of feasibly piecewise-polynomially approximable functions as the largest class of feasibly approximable functions which are typically considered in practice. We first remark that the operators under consideration do become second-order polytime computable with respect to these representations:

Theorem 3. *The operators* Max *and* Int *are second-order polytime* $(\mathrm{PPoly}, \mathrm{PPoly})$*-computable.*

As noted, the class of feasibly piecewise-polynomially approximable functions contains all polytime computable Gevrey functions. Moreover we have the following closure properties:

Theorem 4. *The class of feasibly piecewise-polynomially approximable functions is uniformly closed under addition, multiplication, bounded division (i.e. division by a function which is minorised by* 1*), composition, pairwise maximisation, taking antiderivatives, taking square roots, and taking absolute values.*

The class of feasibly polynomially approximable functions shares some, but not all of these closure properties. In particular, feasibly polynomially approximable functions are not closed under pairwise maximisation or taking square roots. These results suggest that the class of feasibly piecewise-polynomially approximable functions contains all univariate real functions that are usually considered in practice. Theorem 3 shows that maximisation and integration are uniformly polytime computable on these functions. This is a possible explanation for the observed discrepancy between practical feasibility and the Ko-Friedman results.

These theoretical results motivate to study the relationship between the various representations and the complexity of the functionals and operators thereon from a practical perspective. We have started to implement the representations and operators presented here in AERN[2], a Haskell library for exact real arithmetic, and plan to report on benchmarks which compare the various choices.

Our implementation of Poly and PPoly uses polynomials with dyadic coefficients in the Chebyshev basis, together with a uniform error bound. The representation Fun is implemented as a Haskell function type which maps dyadic rational balls to dyadic rational balls.

Range computation for polynomials is based on the real root isolation algorithm found in [**?**]. Division is implemented differently for Poly and PPoly: in Poly we use Chebyshev interpolation,

[2]Code and preliminary benchmark results available at: `http://tinyurl.com/aern2-fnreps`

while in PPoly we use linear interpolation in conjunction with the Newton-Raphson division method.

We have tested our implementations on various "well-behaved" and "ill-behaved" functions. In general, our implementation of PPoly and Poly is much faster than Fun when it comes to integration. On well-behaved analytic functions with multiple maxima, our implementation of PPoly and Poly is also much faster when it comes to maximisation. For simple non-smooth functions and analytic functions with singularities near the real line, PPoly tends to be more efficient than Poly. However, there are examples where Chebyshev interpolation yields much better results than spline interpolation. For more complicated functions, particularly such which involve composition, the representation Fun tends to be much faster than PPoly or Poly when it comes to maximisation.

References

[1] A. Kawamura and S. A. Cook. Complexity theory for operators in analysis. *ACM Transactions on Computation Theory*, 4(2):5, 2012.

[2] A. Kawamura, N. Müller, C. Rösnick, and M. Ziegler. Computational benefit of smoothness: Parameterized bit-complexity of numerical operators on analytic functions and Gevrey's hierarchy. *Journal of Complexity*, 31(5):689 – 714, 2015.

[3] K.-I. Ko. *Complexity Theory of Real Functions*. Birkhäuser, 1991.

[4] K.-I. Ko and H. Friedman. Computational complexity of real functions. *Theoretical Computer Science*, 20:323–352, 1982.

[5] S. Labhalla, H.Lombardi, and E.Moutai. Espaces métriques rationnellement présentés et complexité, le cas de l'espace des fonctions réelles uniformément continues sur un intervalle compact. *Theoretical Computer Science*, 250:265–332, 2001.

[6] N. T. Müller. Uniform computational complexity of Taylor series. In *Automata, Languages and Programming*, volume 267 of *Lecture Notes in Computer Science*, pages 435–444. Springer, 1987.