Master's Thesis

in Theoretical Computer Science

Approximate Real Function Maximization and Query Complexity

Student:	Carsten Rösnick
Email:	roesnick@mail.upb.de
Matriculation Number:	63 82 981
Field of Study:	Theoretical Computer Science
Advisor (TUD):	Prof. Dr. Martin Ziegler
Advisor (UPB):	Prof. Dr. Friedhelm Meyer auf der Heide

Paderborn. Darmstadt. March 3, 2011.

Abstract

Consider the problem of approximate function maximization: Given a continuous real function with domain $[0,1]$, what is its approximate maximum of absolute error 2^{-n}? More precisely, we are interested in the complexity of approximating the real functional $\textsc{Max} : C[0,1] \ni f \mapsto \max_{0\le x\le 1} \in \mathbb{R}$ up to a prescribed absolute error 2^{-n}. Regarding the complexity, we examine quantitative and parameterized bounds; that is, the complexity is measured in parameters of the input function f, e. g., in its Lipschitz constant. For that, we model the approximation of real functionals by using methods from recursive analysis where computations of functions are black-boxed via *function oracles*. As an extension, the retrievable information about a function is defined by *protocols*. At first, we analyze the mutual simulation of these protocols, their computability and their computational complexity. Second, we determine the query complexity of approximating the MAX-functional relative to selected protocols. As a result, the simulation complexity often depends on the function's Lipschitz constant and the notion of a *modulus of strong unicity*. Furthermore, it is necessary to give the Lipschitz constant to any simulation algorithm as long as non-adaptive information is concerned. However, no such limitation holds for adaptive information and access to a range approximating protocol.

Zusammenfassung

Betrachtet sei das Problem der approximativen Maximumswertberechnung: Gegeben eine reellwertige stetige Funktion über dem Einheitsintervall $[0,1]$, bestimme approximativ ihr Maximum mit absolutem Fehler 2^{-n}. Genauer interessieren wir uns für die Komplexität der approximativen Berechnung von $\textsc{Max} : C[0,1] \ni f \mapsto \max_{0\le x\le 1} \in \mathbb{R}$ mit gegebenem (absolutem) Fehler 2^{-n}. Wir bestimmen dazu quantitative parametrisierte Schranken, d. h. Schranken in Abhängigkeit von Parametern der Eingabefunktion wie beispielsweise der Lipschitz-Konstante. Zur approximativen Berechnung von MAX verwenden wir Modelle und Methoden aus der berechenbaren Analysis: Die Berechnung einer Funktion wird ausgelagert in eine Black Box und einem Algorithmus als *Funktionsorakel* mitgegeben. Unsere Erweiterung dieses Modells beschreibt welche Information solch eine Black Box zurückgeben kann als *Protokoll*. Wir untersuchen zunächst die gegenseitige Simulierbarkeit dieser Protokolle, als auch ihre Berechenbarkeit und Berechnungskomplexität. Anschließend widmen wir uns der Frage der Anfragekomplexität bezüglich verschiedern Protokolle zur hinreichend genauen Approximation von MAX. Wie sich zeigt hängt die Simulationskomplex-

ität der Protokolle zumeist von der Lipschitz-Konstante also auch von einem *Modul der starken Eindeutigkeit* ab. Bezüglich nicht-adaptiver Information zeigt sich, dass die Lipschitz-Konstante notwendig ist zur Approximation von MAX. Dies ist hingegen nicht der Fall bei adaptiver Information bei Zugriff auf ein Protokoll approximierend den Wertebereich einer Funktion.

Contents

List of Figures

List of Tables

1 Introduction

A significant amount of (mathematical) problems we face on an everyday basis is not discrete, but continuous and as such often require approaches from numerical analysis. Although most numerical methods are sufficient in theory (let aside stability issues), when it comes to a practical implementation, then every algorithm has to deal with the finite nature of floating point calculations; even for problems as (seemingly) easy as computing integrals $\int_0^x f(t)\,dt$ or maximum values $\max_{0\leq t\leq x} f(t)$.

At least since 1985 when the standard IEEE floating point[1] description has been conceived, examples of disastrous round-off errors has risen (as, for example, Ménissier-Morain mentions and references in [MM05, Section 1.1]). Such examples are not only direct consequences of the (in some cases) insufficient representation of floating point values via mantissa, basis and exponent (which causes a *non-uniform* distribution of "IEEE-floats" along the real line), but they could also arise as a lack of a concise mathematical derivation bounding the intermediate errors during computations. Those are at least one motivation to compute approximations based on (arithmetical) operations of controlled precision in the field of *exact real arithmetic*. Models for exact real arithmetic are intended to bring both worlds together, that is, to accept the discrete nature of our basic underlying computation model (the Turing machine) while performing every arithmetical operation with the desired resulting error in mind. This implies carrying out each intermediate calculation with an at worst significantly higher precision than required for the output itself. Moreover, even the input error now depends on the output error, requiring any system for exact real arithmetic to provide any input with arbitrarily high accuracy.

Models of computation for exact real arithmetic meet this requirement of providing (countably of even uncountably) infinite information about the inputs. This is achieved through various approaches. As one of them, the model introduced by Ko and Friedman grants any approximation algorithm for, say a real function f, access to an oracle ϕ for the input $x \in \mathbb{R}$

[1] IEEE 754: Standard for Binary Floating-Point Arithmetic; http://754r.ucbtest.org/standards/754.pdf. For an introduction in and also a discussion about the difficulties in dealing with floating point arithmetic, the paper by Goldberg [Gol91] appears to be a valuable starting point.

so that ϕ returns approximations of x within an arbitrary user-specified accuracy. Their approach also extends quite naturally to functions of higher type, e. g., to functionals mapping real functions to real functions. Having fixed the model with oracle access to the inputs enables us to talk about classical complexity questions. That is, how many arithmetical operations as well as queries to the fixed oracle are necessary (at least or at worst, corresponding to whether we would like to gather lower or upper bounds, respectively) to compute the end result for the given problem.

Abstract goal, its relevance and related work

Now, to circle back to our "easy problems" like function maximization $\textsc{Max} : [0,X] \to \mathbb{R} \ni f \mapsto \max_{0\le t\le X} f(t)$, the motivating question for this thesis sums up as follows: How hard (in a complexity theoretic sense) is function maximization relative to different oracles, each encoding a different bit of information about the input function? Due to their relativistic nature, those bounds then reflect to what degree approximate function maximization depends on properties (of classical analysis) of the input function, thus exploiting parts of the problem's inner structure. Besides for parameterized bounds of approximative function maximization being of theoretic interest, they are also of practical relevance like for concrete implementations on frameworks like the iRRAM [Mü01] or RealLib [Lam07].

This approach has much in common with the notion of *information-based complexity* (IBC), introduced by Traub et al. (see [TWW83, TWW88]). The main difference between IBC and our approach is that within this thesis, we are not only interested in the minimal amount of information needed about the input to approximate $\max f$ (as typical in IBC), but also (and in particular) in what *concrete analytic properties* have influence on the (information) complexity of function maximization. From the side of computational complexity, Kreinovich et al. [KLRK98] treat problems like approximating a function's range which in fact is quite useful for function maximization. However, like with IBC, their work also does not cover our specific problem. To the best of our knowledge, neither the former approaches, nor the work by Ko and Friedman [Ko79, KF82, Ko82, Fri84, Ko91, Ko98] (which worked extensively in the field of recursive analysis, including function maximization), or others address the question raised above, the main motivator for this thesis.

Approach and concrete results

The approach taken to the main question about parameterized bounds for approximative function maximization splits up into two steps. First, we devise, compare and explore different ways (i. e., oracles, denoted as *protocols*) of accessing an unknown continuous real function via black-box queries that model the actual computation of f as supported by certified numerics like interval arithmetic [KLRK98] or the iRRAM [Mü01]. Each of these protocols is devoted to cover a certain analytic property of its encoded continuous function f. In Chapter 3 we introduce eight of them, namely protocol 0, 1, 2, Y, L, X, Coeff and SLP. Second, we determine the query complexity of the MAX-functional by varying the attached protocol. Since each protocol addresses another analytic property, this leads to explicit quantitative and parameterized both upper and lower bounds on the complexity of MAX. In the following, we give a brief overview over all protocols and present the results obtained when simulating the behaviour of one protocol by another. Subsequently, we summarize the results deduced for MAX' query complexity.

Our underlying notion of computability is encoded by protocol 0: For every error $\varepsilon > 0$ and every input $x \in \mathrm{dom}(f)$ it gives an ε–approximation $p \in \mathbb{Q}$ to $f(x)$, i. e., $|f(x) - p| < \varepsilon$. In contrast, protocol 1 does not return an approximation of a single function value, but a close approximation (measured in the function's Lipschitz constant L) to *all* function values over a prescribed x-interval $[(a-1)/2^n, (a+1)/2^n]$ as depicted in Figure 1.1. Albeit their different nature, protocols 0 and 1 turn out to be asymptotically equivalent when the function's Lipschitz constant is kept fixed (Theorem 3.2.6).

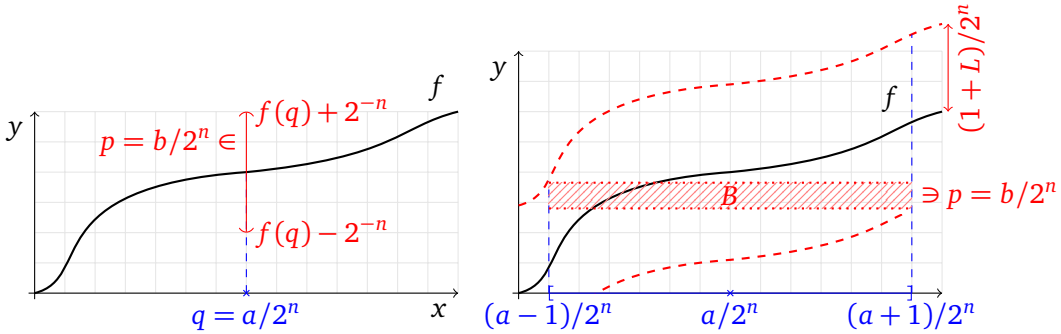


Figure 1.1: Schematic of protocols 0 (left) and 1 (right). The output $p = b/2^n$ of protocol 0 is contained in an open interval of radius $1/2^n$ around $a/2^n$, where for protocol 1 it is contained in "striped bar" B of elements whose distance to every function value $f(x)$ on $[(a-1)/2^n, (a+1)/2^n]$ is less than $(1+L)/2^n$.

Protocol Y provides an over-estimation (of relative error) of f's image over a prescribed x-interval. Conversely, given a point $q \in \mathrm{dom}(f)$, protocol X provides an (under-)estimated x-interval (of relative error) centered at q so that f's image over that x-interval does not exceed a prescribed length. These relations are depicted in Figure 1.2. Comparing both

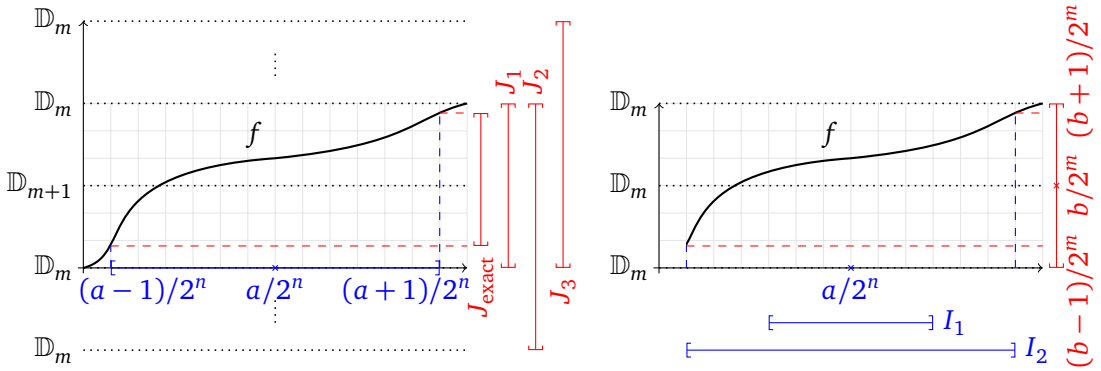


Figure 1.2: Schematic of protocols Y (left) and X (right). Protocol Y returns one of the intervals J_i (containing J_{exact}, the exact function's range over $[(a-1)/2^n, (a+1)/2^n]$). In contrast, protocol X returns one of the intervals I_j and a point $b/2^m$ so that $f(I_j)$ is contained in $[(b-1)/2^m, (b+1)/2^m]$.

protocols with the formerly introduced protocols 0 and 1 lead to the following results:

(a) The simulation time for protocols 0 and 1 by protocol X is constant (Theorem 3.6.3(a)), while for protocol Y it is linearly bounded in the function's Lipschitz constant (Theorem 3.4.7(a)).

(b) The simulation time for protocol $\mathrm{Y}_{\mathrm{mod}}$ (a slightly modified version of protocol Y) by protocol 1 depends on the notion of a *strong modulus of unicity* $\underline{m}(\cdot)$ which, roughly said, states that the given function cannot be "too flat". In addition to $\underline{m}(\cdot)$, the simulation time for protocol X by 1 also depends on both the desired y-interval length and f's Lipschitz constant, assumed that $\mathrm{dom}(f) = [0,1]$.

Protocol L, like protocol Y, approximates the function's range, though the relative error now depends on the function's Lipschitz constant as opposed to the length of the exact range as for protocol Y. As it turns out, protocol L is much easier to simulate by protocol 1 than protocol Y is. When the function's Lipschitz constant grows, the simulation time becomes almost constant (Theorem 3.5.3(b)).

Protocol 2 extends protocol 0 such that not only an ε–approximation to $f(x)$ is returned, but also one for $f'(x)$. Both protocols Coeff and SLP encode an approximation polynomial for

f. Their main difference is its representation: Where protocol Coeff returns its coefficients with prescribed error, protocol SLP on the other hand encodes it as a (finite) chain of basic arithmetical operations (addition and multiplication), called *straight-line program*. Given the fact that the interpolation polynomial which protocol Coeff basically returns is easily differentiable, we will conjecture that even if f has an L'-Lipschitz continuous first derivative protocol 2 still remains impossible to simulate.

The precise results are summarized in Table 1.2, modulo the protocol comparisons that either only left to conjectures or are still open.

Regarding the query complexity to approximate $\max f$ relative to different protocols, we obtain the following results:

(a) *Negative results.* Having given access to either protocol 0 or 2, $\max f$ is *not* approximable within arbitrary an error $\varepsilon > 0$ without any additional knowledge about f (Theorems 4.1.1 and 4.1.6). The same result (relative to protocol 2) turns out to be true even for infinitely differentiable functions (Corollary 4.1.7), but may not extend further to analytic functions (Conjecture 4).

(b) *Positive result.* In contrast to the former results, there exists an algorithm relative to protocol Y that without knowledge of L computes an ε–approximation of $\max f$ (with $f : [0,X] \to \mathbb{R}$ being L-Lipschitz continuous) using up to $\mathcal{O}((1+L\cdot X)\cdot\varepsilon)$ queries and arithmetical operation.

(c) *Positive results.* With access to protocol 0, approximating $\max f$ within an error $\varepsilon > 0$ takes up to $\mathcal{O}((1+L\cdot X)\cdot\varepsilon)$ queries for L-Lipschitz continuous functions $f : [0,X] \to \mathbb{R}$ (Theorem 4.1.2) where L is *revealed* to any simulation algorithm. Exchanging protocol 0 by 2 and posing the restriction of f having an L'-Lipschitz continuous first derivative (where now L' is given to any approximation algorithm) results in an upper bound on the number of queries of $\mathcal{O}((1+L\cdot X)\cdot\sqrt{\varepsilon})$ (Theorem 4.1.9).

	0	1	Y_{mod}	L	X
0	—	Theorem 3.2.6 $\mathcal{O}(1)$	$0 \to 1 \to Y_{\text{mod}}$ $\mathcal{O}(1+2^{\underline{m}(n)-n})$	$0 \to 1 \to L$ $\mathcal{O}(1+1/L)$	$0 \to 1 \to X$ $\mathcal{O}((1+L)\cdot X \cdot 2^{\underline{m}(m_J+\log(1+L))})$
1	Theorem 3.2.6 $\mathcal{O}(1+L)$	—	Theorem 3.4.7(b) $\mathcal{O}(1+2^{\underline{m}(n)-n})$	Theorem 3.5.3(b) $\mathcal{O}(1+1/L)$	Theorem 3.6.3(b) $\mathcal{O}((1+L)\cdot X \cdot 2^{\underline{m}(m_J+\log(1+L))})$
2	$2 \to 0$ $\mathcal{O}(1)$	$2 \to 0 \to 1$ $\mathcal{O}(1)$	$2 \to^* 1 \to Y_{\text{mod}}$ $\mathcal{O}(1+2^{\underline{m}(n)-n})$	$2 \to^* 1 \to L$ $\mathcal{O}(1+1/L)$	$2 \to^* 1 \to X$ $\mathcal{O}((1+L)\cdot X \cdot 2^{\underline{m}(m_J+\log(1+L))})$
Y	$Y \to 1 \to 0$ $\mathcal{O}(1+L)$	Theorem 3.4.7(a) $\mathcal{O}(1+L)$	—	Theorem 3.5.3(a) $\mathcal{O}(1)$	$Y \to 1 \to X$ $\mathcal{O}((1+L)(1+X \cdot 2^{\underline{m}(m_J+\log(1+L))}))$
L	$L \to 1 \to 0$ $\mathcal{O}(1+L)$	Theorem 3.5.3(b) $\mathcal{O}(1+L)$	Theorem 3.5.3(c) $\mathcal{O}(1+2^{\underline{m}(n)-n})$	—	$L \to 1 \to X$ $\mathcal{O}((1+L)(1+X \cdot 2^{\underline{m}(m_J+\log(1+L))}))$
X	Theorem 3.6.3(a) $\mathcal{O}(1)$	$X \to 0 \to 1$ $\mathcal{O}(1)$	$X \to^* 1 \to Y_{\text{mod}}$ $\mathcal{O}(1+2^{\underline{m}(n)-n})$	$X \to^* Y_{\text{mod}} \to L$ $\mathcal{O}(1+2^{\underline{m}(n)-n})$	—
Coeff	Lemma 3.7.2(a) $\mathcal{O}(1)$	Coeff $\to 0 \to 1$ $\mathcal{O}(1)$	Coeff $\to^* 1 \to Y_{\text{mod}}$ $\mathcal{O}(1+2^{\underline{m}(n)-n})$	Coeff $\to^* 1 \to L$ $\mathcal{O}(1+1/L)$	Coeff $\to^* 1 \to X$ $\mathcal{O}((1+L)\cdot X \cdot 2^{\underline{m}(m_J+\log(1+L))})$

Table 1.2: Comparison matrix of protocol's simulation complexity. The obtained result for simulating protocol *column* by protocol *row*, also written as *row* $\to$ *column*, is split into two parts. The upper half of each cell states either a chain of simulations or the respective theorem. In the lower half, the concrete bound is presented.

Structure

This thesis is now structured as follows. In Chapter 2 we discuss the tools required to answer our main question stated above. They include a comparison of various models of computation, followed by an introduction to (polynomial-time) computability on our model of choice, the model by Ko and Friedman, and related work, but also overviews on mathematical formulations of information and (in a sense align with that) existing frameworksfor exact real arithmetic using those methods. In Chapter 3 we define the actual protocols (i. e., our way to represent access to real functions) and study both their computational complexity and simulation complexity (i. e., the complexity of simulating one call to some protocol by another one). Those encodings as black-boxes are used after all in Chapter 4 to finally approximate real function maximization, followed by concluding remarks and open issues in Chapter 5.

2 Fundamentals

Let us start with a quote by Ker-I. Ko from [Ko98]:

> It is important to point out that certain critical analytic properties seem to affect greatly the inherent complexity of a numerical problem.

This works well as a motivation for our approach to pinpoint the complexity of real functionals for reasons we explain in a moment.

As Alan Turing introduced his famous universal model of computation, the *Turing machine*, it was intended to deal with discrete problems; that is, with problems which on finite inputs (composed over a finite set, the so-called *alphabet*) produce finite outputs (over the same alphabet).[1] While this model works well for the discrete world, it gives rise to a whole lot of questions when looked at from the angle of continuous problems. Just to name some of them:

(a) How to represent reals on discrete machines?

(b) How to re-define or extend the notion of computability to reals and real functions?

(c) Are there relations in computability and complexity between the discrete and the continuous world?

(d) While interesting from a theoretical angle, what about real-world implementations of existing concepts in real computation?

Both question (a) and (b) will be partially answered tied to specific models of computation in Section 2.2, whereas we emphasize the mathematically formulation of computations over the reals rather than addressing the question and comparison of possible representations of reals. Note that in general, there is no need to restrict a computation model to perform computations over a discrete structure (as opposed to, for example, treating the reals as atomic) if one only is interested in lower bound results for certain problems. Although concrete

[1] We assume familiarity with the standard Turing-computability. If it is new to you, we refer to [AB09, Chapter 1] for a neat introduction as well as further reading in the subsequent chapters.

implementations of algorithms are not of our main interest, the conceived lower bounds should nevertheless be as close as possible to those on existing, real implementations.

Subsequently, in Section 2.3 a certain model, the Ko-model, is discussed in further detail. As we will see then, it not only has nice properties and is easy enough to describe, but it is also closely related to a computational model, the Weihrauch-model, for which there are actual implementations of *exact real arithmetic*. They are described in short in Section 2.5.

As explained in the introduction, it is the goal of this thesis to conceive parameterized both lower and upper bounds for a certain maximum operator, namely for $\textsc{Max}(f) = \max_{x \in \mathrm{dom}(f)} f(x)$. (Note that we restrict our analysis of $\textsc{Max} : ([0,X] \to \mathbb{R}) \to \mathbb{R}$ for fixed $X > 0$ to the set of computable functions f with $\mathrm{dom}(f) = [0,X]$; the definition of computability and connected questions are discussed tied down to the Ko model in Section 2.3.) *Parameterization* here means the search for properties Z of functions $f : [0,X] \to \mathbb{R}$ (e. g., a fixed Lipschitz constant) wherein the computational complexity of $\textsc{Max}$ can be measured (in addition to the input, of course). If the complexity breaks down when Z is fixed instead of being treated as variable, then the problem (i. e., the respective set of functions) obviously relies heavily on this particular property. Now we closed the gap to Ko's observation stated in the beginning.

To come up with reasonable parameterizations, we take a two-step approach: First we define protocols; second we use them as oracles to determine the number of queries to them it takes to approximate $\textsc{Max}(f)$ within a prescribed error. Splitting the problem of approximate function maximization into two parts not only is a good idea because it is align with Ko's formulation of computation of functionals, but it also allows to analyze the complexity of gathering the information provided by the attached protocols independent from the problem's query complexity, examined by varying the attached protocols. Note that the second step leads to insights about the critical piece(s) of information (of the input) a problem depends on, thus exploiting also the problem's internal structure a little bit further. This intuition is align with the approach taken by Traub et al. with *information-based complexity*. In Section 2.1 we examine this field in more detail. For an example of a problems with (partially) unknown internal structure, consider *Bloch's constant*.

Protocols (treated in depth in Chapter 3) are our way of encoding real functions. Clearly, their continuous nature voids any attempt to uniquely map each real function to a finite representation. Therefore, our protocols are merely giving *approximative information* about their encoded functions. A mathematical formulation of information in connection with real computation is given in Section 2.1.

How this chapter is organized

To summarize, in this chapter we discuss the tools required to answer our main question, formulated in the introduction. It includes a comparison of various models of computation (Section 2.2) followed by an introduction to (polynomial-time) computability (Section 2.3) on the model we like to investigate further, the model by Ko and Friedman (Section 2.2.2). Joining that discussion, we present a brief selection of results (Section 2.4) for and work related to (Section 2.6) real function maximization based on Ko's notions. The overview is completed by Traub's formulation of (partial) information (roughly speaking, partial information about a function is that piece of it used on a discrete machine of finite [or even countably infinite] running time; cf. Section 2.1) and a glimpse into two of the most important implementations (frameworks) for real exact arithmetic (Section 2.5).

2.1 IBC: Information Complexity of Continuous Problems

Information-based complexity (see [TWW83, TWW88]) is a framework for measuring the complexity of continuous problems when given only partial (or even contaminated) information about the problem itself. Similar to computational complexity, the IBC approach imposes a cost measure on the problem space plus the restriction on any algorithm (using certain information and running for a finite amount of time before printing the final result) to read (and therefore to use) only a finite portion of the information about the given problem. Thus, it is equivalent to say that the algorithm only had access to the "cut-down" information in the first place. The question captured by the notion of *information complexity* now is what the, in a sense, smallest possible information is that an algorithm needs to successfully compute all instances of a given problem.

For instance, consider our maximization problem $\textsc{Max}(f) = \max_{0\le t\le x} f(t)$ formulated earlier. If the information is comprised of function values $f(x_i)$ in a finite amount of points x_i, and the goal is to compute a close approximation to $\textsc{Max}(f)$ with some prescribed error $\varepsilon > 0$, then the question becomes what the minimum number of function evaluations is needed to compute such a so-called *ε–approximation* of $\textsc{Max}(f)$ over, say, all continuous functions $f : [0,1] \to \mathbb{R}$.

The subsequent paragraphs are devoted to explain how to adapt IBC so that it fits in the framework of exact real arithmetic.

2.1.1 "Classical" IBC

First, we discuss the basic notions in information-based complexity, followed by an exemplary translation of our function maximization problem into this framework. Let f be some problem (a function for instance), and F the *problem space* (e. g., the set of functions we are interested in). Then partial (and finite) information $N(f)$ about $f \in F$ is modeled as a mapping $N : F \to \text{pow}(H)$ from a problem instance to a finite tuple of information, where H denotes a set of allowed information with $\text{pow}(H)$ being the power set of it. Assume $S(f)$ would be the correct solution of f (with $S : F \to G$ being the *solution operator*) and $\mathscr{A} : \text{pow}(H) \to G$ an algorithm operating on a given information $N(f)$, mapping it to a set of possible outcomes G. Then $\mathscr{A}$ provides an ε–approximation of f if and only if $\| \mathscr{A}(N(f)) - S(f) \|_G \leq \varepsilon$ (for some fixed norm $\| \cdot \|_G$ on G).

For the problem of real function maximization, the solution operator clearly is MAX, where the problem space F is set (for instance) to all continuous real functions $f : [0,1] \to \mathbb{R}$. With that fixed, both the "information set" H and the "solution set" G are set equal to the whole real line, i. e., $H = G = \mathbb{R}$.

Information

In detail, information N is defined by a tuple of finitely many mappings $L_i \in \Lambda$ with Λ being the set of *permissible information operations*. Hence, $N(f) = (L_1(f), \ldots, L_{n(f)}(f))$ for some $n(f) \in \mathbb{N}$. This notation does not impose a connection between the different information operations (i. e., such a tuple is comprised of independent pieces of information). In such a case, we say the information N is *non-adaptive*, and denote it by N^{non} if necessary or not clear by context. (As a side remark, N being non-adaptive implies that $n(f)$ is actually independent of f.)

In the presence of *adaptive* information, the ith information operation depends on the current history of information. I. e., $N(f)$ is given as

$$N(f) = (L_1(f), L_2(f; y_1), \ldots, L_{n(f)}(f; y_1, \ldots, y_{n(f)-1}))$$

where $y_i = L_i(f; y_1, \ldots, y_{i-1})$ for $1 < i \leq n(f)$ is the *exact* information of the ith information operation L_i. The number $n(f)$ of information operations is said to be the *cardinality*

of information N. For non-adaptive information, we also denote $n(f)$ by $\mathrm{card}(N)$ to emphasize that the number of information operations is actually independent from the problem instance.

Cost Measure and Radius of Information

As mentioned, the access to information is priced. For the sake of simplicity we assume in the following a unit cost model where each access to a piece of information accounts for one time step. With this set, the worst-case cost of computing an ε-approximation for S on F on some given information N is defined as

$$\mathrm{comp}(\varepsilon) = \inf_{\mathscr{A}} \mathrm{cost}(\mathscr{A}, N) = \inf_{\mathscr{A}} \sup_{f \in F} \left(\mathrm{cost}(\mathscr{A}, f) + \mathrm{cost}(\mathscr{A}, N(f)) \right).$$

The combinatorial costs (i. e., the actual computations performed to approximate $S(f)$ based on information $N(f)$ about f) of $\mathscr{A}$ are described by $\mathrm{cost}(\mathscr{A}, N(f))$, while the information cost (time consumed to gather the information) is expressed by $\mathrm{cost}(\mathscr{A}, f)$. While the worst-case computational costs $\mathrm{comp}(\varepsilon)$ are of interest when analyzing optimal approximation algorithms, they do not emphasize the information cost. However, as we will see in Chapter 4, the combinatorial costs are (at least on our case) bounded from above by the information cost when dealing with approximative function maximization.

As mentioned before, and also used in above's formulation of information and computation costs, we are interested in the computation of an ε–approximation of $S(f)$. Although, so far we have no condition to impose on the information N so that such an approximation actually exists. The proper notion for the existence of an ε–approximation is called the *radius of information*, denoted by $r(N)$. Informally, it measures the radius (in a fixed norm) of the smallest ball containing all problem instances $f \in F$ which lead to the *same* information under a fixed information N. Thus, they also result in the same solution, and (consistent with one's intuition) an increase in the amount of information might decrease the radius of such ball. This led to the nice result that for given information N, an ε–approximation exists if and only if the radius of information satisfies $r(N) \leq \varepsilon$.

This also motivates the definition of an *ε-cardinality number*

$$m(\varepsilon) = \min_{N} \{ \mathrm{card}(N) \mid N \text{ such that } r(N) \leq \varepsilon \}.$$

It describes the query complexity we like to measure in the framework of IBC. More precisely, it describes the "smallest" possible information (measured in its cardinality) sufficient to compute an ε–approximation of $S(f)$ for all problem instances f. With the combinatorial costs for computing an ε-approximation of MAX being bounded by the information cost, we get $\mathrm{comp}(\varepsilon) = \mathcal{O}(m(\varepsilon))$.

2.1.2 Handling Noisy Information

So far, information $N(f)$ itself is partial, but still exact. For an example, consider $N(f) = (L_1(f), \ldots, L_{n(f)}(f))$ with information operations given as evaluations $L_i(f) := f(x_i)$ of some continuous function $f : [0,1] \to \mathbb{R}$ in pairwise distinct points $x_i \in [0,1]$. In the beginning of this section we posed the aimed connection to exact real arithmetic. To establish such a connection, the information itself has to be represented with a prescribed error. To express this intention in terms of our example: Every information N gets attached some error vector $\vec{\delta} = (\delta_1, \ldots, \delta_{\mathrm{card}(N)}) \in \mathbb{R}^{\mathrm{card}(N)}$ such that $L_i(f) = f(x_i) + \delta_i \in \mathbb{D}$ (i. e., $L_i(f) = a/2^n$ for some $a \in \mathbb{Z}$, $n \in \mathbb{N}$ for reasons we explain in the subsequent sections). This concept is called *noisy information*.

In the presence of noisy information, the cardinality of N not only depends on f, but also on the (in general unknown) noise $\vec{\delta}$; see [TWW88, Section 12.2.1, p. 435]. For non-adaptive information we get the equality $N(f, \vec{\delta}) = N(f) + \vec{\delta}$ (where the vector addition is defined as usual, i. e., componentwise).

Approximation Error and Radius of Information

Let $A^{-1}(\vec{z})$ be the set of exact information underlying some noisy information $\vec{z}$, i. e., $\vec{z} = \vec{y} + \vec{\delta}$ for some exact information $\vec{y}$ and noise $\vec{\delta}$. As an (abstract) example consider

$$A^{-1}(\vec{z}) = \{N(f) \mid \exists f \in F, \vec{\delta}' \in E(f,N) : N(f,\vec{\delta}') = N_{\vec{z}}(f) + \vec{\delta}' = \vec{z}\} \tag{2.1}$$
$$\text{with } E(f,N) = \{\vec{\delta}' \in \mathbb{R}^{\mathrm{card}(N)} \mid \|\vec{\delta}'\|_\infty \leq \eta\},$$

where (2.1) is the general form of $A^{-1}(\vec{z})$, and $E(f,N)$ denotes the set of possible noise $\vec{\delta}$ based on N and f. Also note the dependence of information $N(f,\vec{\delta})$ on the noise $\vec{\delta}$. For non-adaptive information we have the relation $N(f,\vec{\delta}) = N(f) + \vec{\delta}$, where for adaptive information we have $N(f,\vec{\delta}) = N_{\vec{z}}(f) + \vec{\delta}$. That is, because for adaptive information

the information operations (and thus the number of them) depend on the history of noisy information, i. e., $L_{i,\vec{z}}(f) = L_i(f; z_1, \ldots, z_{i-1})$. So, $N_{\vec{z}}(f) = (L_{1,\vec{z}}(f), \ldots, L_{n(f,\vec{\delta}),\vec{z}}(f))$.

Given the set of exact information $A^{-1}(\vec{z})$ corresponding to some noisy information $\vec{z}$, the set of problem instances $f \in F$ that have the same information from N's point of view is given as $N^{-1}A^{-1}(\vec{z})$.[2] Consequently, the solution operator S operates on this set instead of the whole problem space F. The resulting set of solution elements, namely $SN^{-1}A^{-1}(\vec{z})$, is equivalent[3] to the ε–approximation $\mathscr{A}(\vec{z})$ gathered based on the noisy information $\vec{z}$. Figure 2.1 depicts a schematic of the overall construction. As a side remark, the combination of N and E in

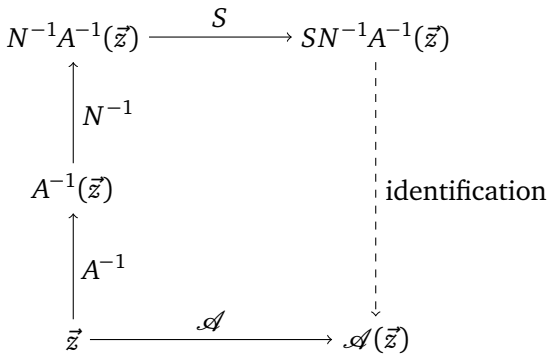


Figure 2.1: Indistinguishable solution elements, resulting from different problem instances that share the *same* noisy information $\vec{z}$.

fact models a protocol Z (that is, an oracle encoding a function f, which we will discuss in depth in Chapter 3), because Z also gives deterministic information about functions $f \in F$, but only with a prescribed error – which is modeled by E.

As mentioned in the introduction, an ε-approximation (on *exact* and fixed information N) only exists if $\| \mathscr{A}(N(f)) - S(f) \|_G \leq \varepsilon$. Given instead some noisy information $\vec{z}$, this can be rephrased to

$$\| \mathscr{A}(\vec{z}) - S(f) \|_G \leq \varepsilon \quad \forall f \in SN^{-1}A^{-1}(\vec{z}). \tag{2.2}$$

Hence, informally speaking, the wider the set $SN^{-1}A^{-1}(\vec{z})$ (measured in $\| \cdot \|_G$), the bigger ε has to be so that an ε–approximation exists (where everything is fixed but the algorithm $\mathscr{A}$).

[2] As a common abbreviation, by $N^{-1}A^{-1}$ we actually mean the composition $N^{-1} \circ A^{-1}$ where mapping A^{-1} is applied first, followed by N^{-1}.

[3] Here, equivalency has to be interpreted as follows: Given some noisy information $\vec{z}$, then for every $f \in SN^{-1}A^{-1}(\vec{z})$ we have $\mathscr{A}(\vec{z}) = \mathscr{A}(E(f,N))$, i. e., those problem instances result in the same approximation.

Equation (2.2) motivates the notation of a *local radius of (noisy) information* of N at $\vec{z}$, denoted by $r(N,E,\vec{z})$, and given as

$$r(N,E,\vec{z}) := \operatorname{rad}(SN^{-1}A^{-1}(\vec{z})) := \inf_{g\in G} \sup_{s\in SN^{-1}A^{-1}(\vec{z})} \| g-s \|_G.$$

Accordingly, the *(global) radius of (noisy) information* is defined as

$$r(N,E) = \sup_{(\exists f\in F)\ \vec{z}-N_{\vec{z}}(f)\in E(f,N)} r(N,E,\vec{z}).$$

Therefore, an ε-approximation (in presence of noisy information) exists if and only if N satisfies $r(N,E) \leq \varepsilon$ for fixed E. It is our job to define N, E, and the cardinality $n(f)$ of N, such that $r(E,N) \leq \varepsilon$, where ε is fixed, but arbitrary.

2.1.3 Existing Results

For linear problems (see [TWW88, Section 4.5.1, p. 56]) there are nice results like *adaption of information does not significantly help*, and results on optimal information measured in the radius or diameter of information. Unfortunately, one condition on a problem set to be called *linear* is it that S is a linear operator[4]; which MAX clearly is *not*.

2.2 Models of Computation

It is one motivator for us both to understand the underlying model of computation and to choose it wisely because the protocols we examine in Chapter 3 are modeled with an eye on implementing it on an existing framework like iRRAM or RealLib (see Section 2.5 for a description).

A computational model is comprised of (a) a given (finite or infinite) alphabet Γ, (b) a (finite) set of of basic operations available to perform an actual computation, and (c) the notion of costs attached to each of those operations. In *computable analysis* (also: *recursive analysis*) one is mostly interested in computability questions of numerical problems; that is (loosely speaking), if there is an algorithm that produces a reasonable approximation (or even an exact result) to its given problem instance (e. g., a continuous function).

[4] An operator $S : F \to G$ is called *linear* if $S(f+f') = S(f)+S(f')$ for all $f, f' \in F$.

First, we provide a brief glimpse on how computable analysis developed over time; thereafter, we discuss these models in details. The origin of computable analysis dates back to a paper by Alan Turing [Tur37] where he defined computable real numbers and functions, thus describing approximative computation over the reals on integer machines (i. e., on his famous model of Turing machines). Later, Grzegorczyk (1957) used Turing's notion of computability to introduce a hierarchy on primitive recursive functions. Ko and Friedman (1982), and Pour-El and Richards (1989) took that approach and extended it further by an extensive treatment of complexity questions: How is computable analysis related to discrete complexity theory? How to define, e. g., polynomial-time computability over the reals? What is the computational complexity of standard problems in numerical analysis, like maximization, root finding, differentiation and integration? Later on, Weihrauch (1997) not only came up with sort of a generalization to some of the formerly introduced models, but also related them, even though it turned out that they are not all equivalent (as opposed to the situation in discrete complexity theory, where we have the famous Church-Turing thesis; cf. [AB09, Chapter 1]).

An overview on the history of approaches (including a lot more references) in real computation (even though it is formulated compared to the Blum-Shub-Smale (BSS) model; see Section 2.2) can also be found in [BCSS98, Section 1.8]. For a more elaborate (and more technical) discussion on (the much wider field of) real computation, we refer to [Zie07, Chapter 2].

2.2.1 TTE / Weihrauch Model

The *type-2 model of effectivity* (TTE), introduced by Weihrauch in [Wei87], rather is a "host" for other models (a meta-model for say) than a completely new approach to computable analysis. It explicitly introduces the notion of *naming systems*. To describe and understand the use of naming systems, we first take a look at the most usual form of computations, represented as partial functions $f :\subseteq \Sigma^* \to \Sigma^*$ mapping one word from Σ^* to another one from Σ^*, while the actual computation is realized over an alphabet Γ.[5] A naming system now basically is an interpretation (a mapping) from Σ^* onto the machine alphabet Γ.

For example, consider the field of real numbers $\mathbb{R}$. Elements $x \in \mathbb{R}$ can be encoded using decimal fractions. Then, an infinite decimal fraction acts as a name for a real x; e. g.,

[5] As usual in computer science, a set Σ^* is comprised of any character string (word) over Σ of finite length including the empty word, i. e., $\Sigma^* = \bigcup_{n\geq 0} \Sigma^n$.

$3.14159265\ldots$ is a name for π. In general, in TTE, infinite sequences are used as names for reals $x \in \mathbb{R}$.

A comparison of TTE with several other approaches in computable analysis can be found, for example, in [Wei00, Section 9].

2.2.2 Ko(-Friedman) Model

In Ko's model (we may also call it as the *Ko-Friedman* model due to Friedman's contributions) reals are represented by fast converging sequences of dyadic rationals, e. g., by Cauchy sequences. A real number $x \in \mathbb{R}$ has such a representation if there is a sequence $\{d_n\}$ of dyadic rationals $d_n \in \mathbb{D}_n$ where $\mathbb{D}_n = \{q \in \mathbb{Q} \mid (\exists a \in \mathbb{Z})\ q = a/2^n\}$, so that $|\, d_n - x \,| \leq 2^{-n}$ for all $n \in \mathbb{N}$, and it is encoded by a computable function $\phi : \mathbb{N} \to \mathbb{D}$ where $\phi(n) := d_n$.

Bottom line, Ko-machines are now basically Turing machines with oracle access to such a function ϕ as opposed to getting the real x as a direct input. directly. As a technical detail, ϕ does not map n to a dyadic rational $d_n = a/2^n$, but to a dyadic rational interval of length 2^{-n+1} centered at d_n, encoded by (d_n, n).

Thus, the definition and understanding of the Ko-model leads to the conclusion stated in [Wei00, Theorem 9.4.3]; namely, TTE and Ko's model are essentially the same. I. e., Ko-machines can be effectively simulated by TTE-machines and vice versa. Also see [Wei00, p. 254], for a reasonable deduction.

2.2.3 Interval Arithmetic

Interval Arithmetic (IA), even if it came from a somewhat different angle than the formerly presented models, is of interest (especially the book by Kreinovitch et al.; [KLRK98]) because it brings both worlds together, namely the floating point community with the classical complexity theoreticians. The field of interval arithmetic is devoted to provide methods for and unravel the complexity of computing controlled estimates. As an example: given a function $f(x_1, \ldots, x_n)$ of n real variables, it is considered to be a basic problem of IA to compute the range of possible values $f(I_1, \ldots, I_n) = \{f(x_1, \ldots, x_n) \mid x_i \in I_i\}$ where the inputs are not exact reals, but intervals describing the set of possible (and erroneous) inputs.

In [KLRK98], they dealed with results about the complexity of polynomial equations with finitely many and infinitely many roots, for real, rational and even integer valued polynomials. As one result (which in particular is of interest for us) Gaganov proved that even for

rational functions (i. e., polynomials given by a fraction $f(X)/g(X)$ of two polynomials with rational coefficients) the problem of computing the function's range on some interval with rational endpoints is, in general, NP-hard.

Embedding IA in this thesis, it does not significantly differ from Weihrauch's approach. In both models, we only care about machines holding after a finite amount of time. Therefore, a TTE-machine could only read and write a finite amount of intervals on its input and output tape, respectively, so why not only giving the machine the same interval as an input as for the IA-machine. Then it would act like a machine in interval arithmetic – at least when considering the *relative* error approximation.

Kreinovitch et al. have shown (see [KLRK98, Appendix D]) that computing roots is easier when a function has multiple roots rather than having a unique root. Is an analogous statement also true for maxima, i. e., a function's maximum is easier to compute when there are multiple maximum points? In Section 4.2.3 we show that this is *not* the case; at least when it comes to compute approximations of *maximum values* instead of *maximum points*.

2.2.4 Blum-Shub-Smale (BSS) Model

Compared to, say, the formerly introduced models of Weihrauch and Ko, the BSS-model merely is a theoretical approach to handle continuous problems over an arbitrary ring R, e. g., the reals. As a major deviation to those other models we have seen so far, the elements of such a ring R are assumed to be *atomic*, thus eliminating the need for a reasonable naming system.

Nevertheless, there is a connection between BSS machines and the theory of information-based complexity as already discussed in Section 2.1. As a short recap, in IBC we are interested in the information required about a continuous problem in order to compute a reasonable approximation of it. There, the information is obtained by use of a given subroutine. As usual, the complexity of the respective algorithm solving the given problem is measured both in the number of arithmetical operations and by how often the subroutine (to obtain the information) is called. The same measure applies to BSS-machines.

2.3 Complexity Theory on the Ko-model

In this section we summarize both the basic definitions and results concerning the model by Ko and Friedman one need to know for the rest of this thesis.

2.3.1 Computable Reals and Real Functions

Reals $x \in \mathbb{R}$ have to be presented by in a finite way; e. g., by some rational $q \in \mathbb{Q}$ such that $\| x - q \| \leq \varepsilon$ for arbitrary $\varepsilon > 0$. There are several representations out there: Encoding by a sequence of nested intervals (e. g., Weihrauch), by a fast converging sequence of dyadic rationals (e. g., Ko and Friedman), signed integer representation (e. g., Weihrauch; cf. Section 2.2). Besides the fact that Ko's representation arises quite naturally, he argues that his representation of reals (defined hereafter) has also been shown to be mostly equivalent to other formulations of computable reals, but with the addition of further good properties (cf. [KF82, p. 326]).

Definition 2.3.1 (computable real)**.** Let $x \in \mathbb{R}$ be an arbitrary real number.

(a) We say, a function $\phi : \mathbb{N} \to \mathbb{D}$ *binary converges to* x if $| x - \phi(n) | \leq 2^{-n}$ and $\phi(n) \in \mathbb{D}_n$ (i. e., $\phi(n) = a/2^n$ for some $a \in \mathbb{Z}$).

(b) A real number x is said to be *computable* if there is a (Turing) computable function $\phi : \mathbb{N} \to \mathbb{D}$ that binary converges to x.

Based on this definition, the set of rational number clearly is computable, but so are prominent reals (transcendentals!) like π or Euler's constant, too (cf. [Sko08]). As a counterexample, a *Chaitin constant* is a non-computable real number (see, e. g., [CDS02]). In fact, the set of non-computable real numbers forms a proper subset of all transcendental reals.

Next we define the notion of computable real functions based on the formerly presented concept of binary converging computable functions.

Definition 2.3.2 (computable real function)**.** A function $f : \mathbb{R} \to \mathbb{R}$ is *computable* (or, recursive) if there is a function-oracle TM (OTM) M such that for each $x \in \mathbb{R}$ and each $\phi : \mathbb{N} \to \mathbb{D}$ that binary converges to x, the function ψ computed by M with oracle ϕ binary converges to $f(x)$.

Note: We will use the terms “computable” and “recursive” interchangeably. For the definition of computable reals (and thus, the concept of functions binary converging to them), see Definition 2.3.1. Without having explicitly mentioned it in the definition of computable functions, we say, a function $f : S \to \mathbb{R}$ with $S \subseteq \mathbb{R}$ is computable if it is computable in each $x \in S$.

When defining computability this way, some simple functions become non-computable like, e. g., step functions. Although there is literature considering weaker notions of computability, so far none of them has been proven to be *the* notion to use. We accept this restriction to, amongst others, circumvent the need for multi-valued functions. Besides that, this model (compared to others) preserves the basic properties known from classical analysis (cf. [KF82, Section 2] for further references). Then the following theorem presents the, if you will, law underlying this computability concept that *computability implies continuity*.

Theorem 2.3.3 (computability versus continuity)**.** *If $f : [0,X] \to \mathbb{R}$ is computable on $[0,X]$, then f also is continuous on $[0,X]$. Conversely, if f is continuous on $[0,X]$, then there is a set-oracle $E \subseteq \mathbb{N}$ such that f is recursive in E.*

For the direction of computability implying continuity, the main idea is presented in [Wei00, Theorem 1.3.4]. For the converse direction, see [KF82, Theorem 2.2].

Intuitively, a function becomes a lot easier to approximate if we have upper and lower bounds on its slope. Hereinafter, we define both terms more precisely and discuss both their existence and computability.

Definition 2.3.4 (modulus functions)**.** Let $f : [0,X] \to \mathbb{R}$ be a continuous function.

(a) A function $\overline{m}(\cdot)$ is called *modulus of continuity (of f on $[0,X]$)* if

$$(\forall x, y \in [0,X])(\forall n \in \mathbb{N})\, |x - y| \leq 2^{-\overline{m}(n)} \Rightarrow |f(x) - f(y)| \leq 2^{-n}.$$

(b) Similarly, a function $\underline{m}(\cdot)$ is called *modulus of strong unicity* (of f on $[0,X]$) if

$$(\forall x, y \in [0,X])(\forall n \in \mathbb{N})\, |x - y| > 2^{-n} \Rightarrow |f(x) - f(y)| > 2^{-\underline{m}(n)}.$$

In short, computability causes f to have a modulus of continuity $\overline{m}(\cdot)$. Moreover, $\overline{m}(\cdot)$ also is computable; not on arbitrary domains, but on compact ones. That is because on compact sets, the concepts of continuity and *uniform* continuity match. The following theorem uses this observation.

Theorem 2.3.5 (computability vs. computable modulus of continuity). *If $f : [0,X] \to \mathbb{R}$ is recursive on $[0,X]$, then f also has a* recursive *modulus of continuity $\overline{m}(\cdot)$ on $[0,X]$.*

For a proof, see [KF82, Corollary 2.2.1, p. 331].

Notice that the computability analysis of a modulus of strong unicity is beyond the scope of this thesis. However, we want to point out that $\underline{m}(\cdot)$ is monotonically non-decreasing as implied by its definition:

$$\begin{aligned} & (\forall n, i \in \mathbb{N})(\forall x, y \in \mathbb{R})\, |x - y| > 2^{-n} > 2^{-(n+i)} \Rightarrow |f(x) - f(y)| > 2^{-\underline{m}(n+i)} \\ \Rightarrow \quad & (\forall n \in \mathbb{N})\ \underline{m}(n) \leq \underline{m}(n+1). \end{aligned}$$

We use this fact later on, especially in Chapter 3.

2.3.2 Polynomial-Time Computability

We define the time complexity of both recursive reals and recursive real functions. According to classical Turing machines (i. e., those operating on $\mathbb{N}$), the time complexity is defined by the *maximum* number of moves[6] $T_M(n)$ it takes M to produce the output based on arbitrary inputs (in unary representation) of length n. For function-oracle Turing machines (as in Definition 2.3.2), the time complexity also incorporates both the time it takes M to write an n-bit string on the query tape (n steps), and the query to the oracle itself (1 step). The following definition summarizes notations from [KF82, Section 3].

Definition 2.3.6 (time complexity). (a) A recursive real number $x \in \mathbb{R}$ has time complexity $\leq T$ if there is a TM M computing a function ϕ such that ϕ binary converges to x, and the time complexity function T_M is bounded above by T.

(b) A recursive real function $f : \mathbb{R} \to \mathbb{R}$ has time complexity $\leq T$ if there is an oracle Turing machine (OTM) M computing a function ψ such that ψ binary converges to $f(x)$ (i. e., $|M^x(n) - f(x)| \leq 2^{-n}$ for all $n \in \mathbb{N}$) and the time complexity function T_M is bounded above by T.

[6] Here and in the classical Turing machine model, an altering of cell contents incorporates a head movement, thus allowing us to only count the number of moves. Note that this implies a unified cost model where each operation counts for one time unit.
This is in contrast to, e. g., the BSS model; see [AB09], Section 16.3, p. 331.

(c) For $X \in \mathbb{R}_{>0}$, let $\mathrm{PF}_{C[0,X]} = \mathrm{TIME}_{C[0,X]}(\mathrm{poly})$ be the set of *polynomial-time computable real functions* $f : [0,X] \to \mathbb{R}$ where

$$\mathrm{TIME}_{C[0,X]}(\mathscr{C}) = \left\{ f \in C[0,X] \;\middle|\; \begin{array}{l} f \text{ is recursive and has time complexity} \\ \leq T \text{ for some } T \in \mathscr{C} \end{array} \right\}.$$

Often, when it is either clear from the context or does not matter, we omit the domain and instead write $\mathrm{TIME}(\mathscr{C})$ and PF, respectively.

The polynomial-time computability of reals is postponed until Section 2.4.

As we have already seen a connection between computability and a computable modulus of continuity (cf. Theorem 2.3.5), a similar result holds when the set of functions is restricted to those being polynomial-time computable.

Theorem 2.3.7. *If $f : [0,X] \to \mathbb{R}$ is a polynomial-time computable function, then it also has a polynomially bounded modulus of continuity $\overline{m}(\cdot)$.*

For a proof, we refer to [KF82, Theorem 3.1].

What we do not know is whether polynomial-time computability of f implies f' also to be polynomial-time computable, or not (assumed that f is differentiable). Nevertheless, below we state a known relationship between the complexity of f' and the modulus function.

Theorem 2.3.8. *If $f : [0,1] \to \mathbb{R}$ is a both continuously differentiable and polynomial-time computable function, then f' also is polynomial-time computable if and only if f' has a bounded modulus of continuity on $[0,1]$.*

For a proof, see [KF82, Theorem 5.2].

2.4 Existing Results on Function Maximization

We start by defining computable real functionals as an extension to the formerly introduced concept of computable real functions.

Definition 2.4.1 (computable functional)**.** Let F be a real functional which maps real functions to real numbers. We say, F is *computable* if there is a two-oracle machine M_F so that $|M_F^f(n) - F(f)| \leq 2^{-n}$. The oracles are:

(a) A modulus function $\overline{m}(\cdot)$ of f on $\mathrm{dom}(f)$; on input $n \in \mathbb{N}$ it returns how many significant bits it needs of $x \in \mathbb{R}$ to get out an approximation of $f(x)$ within error 2^{-n}.

(b) A function-oracle f; on inputs $n \in \mathbb{N}$ and $q \in \mathbb{D}_{\overline{m}(n)}$ it returns a dyadic rational approximation $e \in \mathbb{D}_n$ so that $|e - f(x)| \leq 2^{-n}$ for all $x \in \mathrm{int}(q, \overline{m}(n)) := [q - 2^{-\overline{m}(n)}, q + 2^{-\overline{m}(n)}]$.

We say, a computable functional F is polynomial-time computable if for any function-oracle f whose modulus of continuity $\overline{m}(\cdot)$ is polynomially bounded[7], the computation time of M_F^f is bounded by a polynomial. Given this situation, is MAX polynomial-time computable? Further, let $f : [0,X] \to \mathbb{R}$ be a polynomial-time computable real function. Is, in this setting, $\max f$ (polynomial-time) computable? The following theorem summarizes our knowledge.

Theorem 2.4.2. *(a) The maximization functional* Max *is computable over the set of computable functions, but it is not computable in polynomial-time.*

(b) The maximum value of a recursive real function again is recursive.

(c) If f is a polynomial-time computable real function, then $\max f$ *is a polynomial-time computable real number* relative *to an* NP*-oracle.*

For actual proofs (or at least plausibility arguments) of above's theorem we refer to [KF82, Section 7]. The functional MAX is recursive over the set of recursive functions due to Theorem 2.4.2(b); just equip TM M_{MAX} with an oracle to f (say, the machine M_f computing f).

As a marginal note, Ko showed interesting connections between NP-real numbers of tally sets[8] and maxima that could be attained by an arbitrary polynomial-time computable real function $f : \mathbb{R} \to \mathbb{R}$: both sets are equal. This implies a connection (one of several) to classical complexity theory, namely, for a proven hardness of computing an arbitrary functions maxima (i. e., in exponential-time) it is necessary to have $\mathrm{P} \neq \mathrm{NP}$ (Theorem 2.4.2(c); also cf. [Ko98, Theorems 3.10 and 3.11]). Another connection to the famous $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$ problem has been shown by Friedman in [Fri84]: Some maximum operators (e. g., $\mathrm{MAX}_h(f) = h(f;\cdot)$ with $h(f,x) = \max\{f(y) \mid 0 \leq y \leq x\}$) are mapping polynomial-time computable functions to polynomial-time computable functions if and only if $\mathrm{P} = \mathrm{NP}$.

[7] Due to Theorem 2.3.7: If f does not have a polynomially bounded modulus of continuity, then f is certainly not polynomially-time computable, thus justifying this restriction.

[8] A *tally set* T is a set over a single alphabet, e. g., $T \subseteq \{0\}^*$.

Given a computable real function $f \in C^1[0,X]$, a naive approach to approximate $\max f$ goes as follows: Compute the roots r_i of f', evaluate f in each r_i (i. e., $m_i = M_f^{r_i}(\cdot)$ for some TM $M_f^{r_i}$) and finally return $\max_i m_i$.[9] The following theorem voids this attempt.

Theorem 2.4.3. *(cf. [Ko98, Theorem 3.12]) There is a polynomial-time computable real function $f : [0,X] \to \mathbb{R}$ that has an uncountable number of roots but* none *of them is computable.*

As a consequence, even in case of $f : [0,X] \to \mathbb{R}$ having a polynomial-time computable first derivative (see Theorem 2.3.8) there is no algorithm that based on the roots of f' (i. e., the candidates for approximations of $\max f$) approximates $\max f$ within a prescribed error 2^{-n} for all $n \in \mathbb{N}$.

In contrast, all roots of an analytic, polynomial-time computable real function f are polynomial-time computable (cf. [KF82, Theorem 4.3.2]). But it is conjectured that there is a function $g \in C^\infty[0,1]$ such that g is polynomial-time computable, but it has a root that is *not* a polynomial-time computable real number. So, as long as the conjecture stands, there seems to be a complexity gap between real analytic and C^∞ functions, not only for differentiation, but presumably also for approximate function maximization. This assumed gap will be again of interest in Section 4.1.2.

2.5 Frameworks for Exact Real Arithmetic

While the models discussed in Section 2.2 are (by definition) of theoretical nature, there are actual implementations present. Below we briefly introduce two of them, namely the iRRAM and the RealLib; it appears that those two are the most accepted and widespread approaches as frameworks for exact real arithmetic.

iRRAM

The iRRAM (interactive Real-RAM) (see [Mü01] for an extensive documentation), maintained by N. Müller and first introduced in [Mü96], partly implements the type-2 model of effectivity already discussed in Section 2.2.1. As a framework written in C++, programs for the iRRAM are also written in plain C++ syntax with the extension of pre-defined data

[9] Note that there is a difference between *approximating* the root and computing an *approximative root*; in the first case we have $|x - \phi(n)| \leq 2^{-n}$ (which is computable due to Theorem 2.4.2(b)), where in the second case it is $|f(\phi(n))| \leq 2^{-n}$ (which is not necessarily close to the real root; see [Ko98], Section 3.4.1).

types (actually objects) like REAL and DYADIC. For a glimpse of how real exact arithmetic is achieved, consider the representation of reals. A real number $x \in \mathbb{R}$ gets assigned a name (an enclosure, i. e., an arbitrary (infinite) sequence of [not necessarily nested] rational intervals) $\mathscr{J} = ((l_i, u_i))_{i=0,1,\ldots}$ out of $\varrho^{-1}(x)$, where ϱ is a partial surjection $\varrho : \mathbb{J}^\infty \to \mathbb{R}$ with $\mathbb{J} := \{(l,u) \in \mathbb{Q} \times \mathbb{Q} \mid l \leq u\}$. Thus, a computation actually becomes a multi-valued function mapping one interval sequence to another one (maybe even in a non-deterministic fashion).

RealLib

As B. Lambov mentioned in [Lam07], designing a framework for exact real arithmetic requires a balancing between providing a user-friendly interface on the one hand side (it should behave as and [even more important] "feel" like standard floating point arithmetic) and elevating the user's understanding of controlled precision arithmetic (by providing a comprehensive set of tools) on the other hand side; in short, a light but sufficiently complex framework. Loosely speaking, in RealLib the interface is split up into two parts. The first one matches those of iRRAM, while the second one specifically is aimed to hide all the complexity of dealing with representations and multi-valued functions on the costs of a worse performance.

A model is said to be *type-2* if it is align with the theory of computable real functions by machines with oracle access to the function's argument(s) like in Ko and Friedman's model (see Section 2.2.2. Thus, it is a basic requirement for a model to be called type-2 to provide a method of gathering the arguments with arbitrary precision; or, to phrase it differently, the *full information* about them must be present, even if it is not fully utilized in any computation. Therefore, both RealLib and iRRAM are *not* type-2 implementations, but only type-1 (cf. [Lam07, Section 6.3]). This is because in both models neither is full information stored about reals nor is full information present.

2.6 Related Work

As pointed out in former sections, there has been some groundwork done on the question of the computational complexity of maximization, mostly by Ko and Friedman (cf. [KF82, Fri84]). But as Bernard of Chartres famously wrote back in the twelfth century, they also stood *on the shoulders of titans*, with the titans are including Turing, Grzegorczyk, Pour-El,

Richards, Kreitz and Weihrauch. The main result of Ko and Friedman's research (concerning maximization) clearly is the relation to the famous $\mathrm{P} \stackrel{?}{=} \mathrm{NP}$ question in discrete complexity theory; that is, $\mathrm{P} \neq \mathrm{NP}$ is necessary for a *super-polynomial* lower bound on the computation of $\textsc{Max}(f)$. Or, as phrased in Theorem 2.4.2(c), if $\textsc{Max}(f)$ is polynomial-time computable for every polynomial-time computable real function $f : [0,1] \to \mathbb{R}$, then we get $\mathrm{P} = \mathrm{NP}$ out of it. This fact is obtained by examining its proof. In there, Ko and Friedman constructed an NP-set U relative to which a simple binary search suffices to compute $\max(f)$ on $[0,1]$.

Friedman took their work further in [Fri84], but contributed rather a different formulation of real computation and proved it to be equivalent to their former approach than providing new results on the still open question, namely, whether $\mathrm{P} \neq \mathrm{NP}$ also is *sufficient* to prove $\textsc{Max}(f)$ non-polynomial-time computable.

In this Master's Thesis we do not intend to change the whole game by conclusively proving approximate function maximization to be either easy (i. e., computable in polynomial time) or hard (by proving the NP-oracle necessary for Theorem 2.4.2(c)). Instead, we are interested in the analytic properties of computable functions that (presumable) make function maximization such a difficult problem. The gained knowledge is then used to deduce parameterized both lower and upper bounds, whereas the complexity of obtaining the parameters itself can (and is) analyzed independently. Both parts of this two-step approach are examined and discussed in detail in the subsequent chapters 3 and 4.

2.7 Notation

We both summarize the notation seen so far and introduce new notation needed for the subsequent sections. The set of *dyadic rationals* is denoted by $\mathbb{D} = \{q \in \mathbb{Q} \mid (\exists a \in \mathbb{N})(\exists n \in \mathbb{Z})\ q = a/2^n\}$, and its restriction to those of precision n by $\mathbb{D}_n = \{d \in \mathbb{D} \mid d = a/2^n\}$.

To maintain readability of both definitions and results when encoding functions by protocols, we use the following conventions (for one-dimensional functions): (a) On the x-axis, dyadic rationals are denoted by $q \in \mathbb{D}$, precisions by $n \in \mathbb{Z}$ and intervals by I, while (b) along the y-axis, we denote dyadic rational by $p \in \mathbb{D}$, precisions by $m \in \mathbb{Z}$, and intervals by J. While for higher dimensional functions $f :\subseteq \mathbb{R}^k \to \mathbb{R}$, $k \geq 2$, the conventions are still valid, we extend them by using the vector notation. That is, e. g., $\vec{q}$ instead of q.

As on the Ko-model, reals are essentially encoded by infinite sequences of dyadic rational intervals, the following notions are introduced to ease their use. Thus, let $[a,b] \subseteq \mathbb{R}$ be some interval.

- *center* or *midpoint* of $[a,b]$: $\mathrm{mid}([a,b]) = (a+b)/2$;
- *length/size* of $[a,b]$: $\mathrm{len}([a,b]) := b-a$;
- *radius* of $[a,b]$: $\mathrm{rad}([a,b]) := \mathrm{len}([a,b])/2 = (b-a)/2$;
- *precision* of $[a,b]$: $\mathrm{pcs}([a,b]) := \log(1/\mathrm{rad}([a,b])) = -\log(b-a)+1$.

Example: We often talk about intervals $\mathrm{int}(q,n) := [q-2^{-n}, q+2^{-n}]$ with dyadic rational endpoints $q-2^{-n}$, $q+2^{-n} \in \mathbb{D}$. The center of $\mathrm{int}(q,n)$ is q, where the length becomes 2^{-n+1}. The radius $\mathrm{rad}(\mathrm{int}(q,n))$ is 2^{-n}, where the precision is n.

Given a closed interval $I = [a,b]$, we denote its endpoints by $\underline{I}$ and $\overline{I}$, i. e., $I = [a,b] = [\underline{I}, \overline{I}]$.

3 Protocols: Approximate Real Functions

In this chapter we describe the internals of our so-called protocols. To recall, a protocol is a black-box modeling the actual computation of some real-valued function; we introduce eight of them, prove their existence (if not obvious) as well as their computability, computational complexity and simulation complexity (i. e., their relation to the other protocols) as far as possible.

3.1 Introduction to Protocols and Protocol Simulation

Protocols are one way to say, at this instant, we do not care about how to compute the information we need for an algorithm to approximate the function maximization. But since we also focus on the implementational aspects of our approach, at some point we do have to care about our protocols internals. More precisely, we will deal with (a) each protocol's computational complexity, and (b) its simulation complexity. By the second phrase (namely, the simulation complexity) we mean the amount of queries to some protocol A in order to simulate another protocol B. As simple as it sounds, there is a certain aspect in this notion that has to be carefully examined: the *information shared* between both protocols. For protocols A and B there is some information N_A and N_B, respectively, that can be used by any algorithm. But what information is given to an algorithm that simulates B by A? It could be as simple as, say, the summation of both information (i. e., $N_A \cup N_B$), but also $N_A \cap N_B$, or even $(N_A \bullet N_B) \cup N$ is possible for some set operation $\bullet$ and information N that is independent of N_A and N_B (but maybe dependent on both protocols, i. e., the additional information N given to a simulation algorithm for B by protocol A is described by a mapping *sim-inf* : *proto* $\times$ *proto* $\rightarrow$ *information*). The respective choice of information is individually detailed in each result.

In combination with the computational complexity of our protocols, we get a better picture of the overall complexity of approximate function maximization in the following sense: While some protocol A might be useful to determine the maximum of a function more easily

(e. g., for the reader), a second protocol B might have a significantly lower complexity. Then, if the simulation complexity allows it (i. e., by replacing A through a simulation by B, the overall complexity of simulating A by B is still not (significantly) higher than applying A directly), we could simply choose the protocol of the least overall complexity to apply it to the approximate function maximization problem. Therefore, we get a parameterized (in the information used) and quantitative upper bound on its complexity.

Notice that this approach might be applicable also to other problems with partially (or even completely) unknown, or at least very difficult internal structure, to get some insights what (information) causes a high complexity.

A Protocol Overview

In the following sections we are about to examine different kinds of protocols, each of them revealing some particular information about the encoded function.

Protocols 0 and 1 (Section 3.2) are, more or less, small deviations of Definition 2.3.2; as we will prove, their computability is easy to obtain. Protocol 2 (Section 3.3) extends protocol 0 such that it not only returns approximations of f, but also of f'. The additional information about f first derivative might hopefully help to approximate $\textsc{Max}(f)$ more easily, but (at worst) on the cost of its computational complexity; we have already provided and argument for that in Theorem 2.3.8.

Where protocol L (Section 3.5) reveals the function's Lipschitz constant (hence an upper bound of the size of each y-interval), both protocol Y (Section 3.4) and X (Section 3.6) are providing (in some sense) "tight" approximations of y- and x-intervals, respectively; see Table 3.2 for a more precise description.

While both protocol Coeff and SLP (Section 3.7) root from a numerical analysis point of view (i. e., approximation of functions by interpolation polynomials), protocol SLP is particularly motivated to provide a more compact encoding of polynomials as, for example, is used in *algebraic complexity theory*.

Notes on Simulation Complexity

Given two protocols A and B, the simulation of protocol B by A is denoted by $A \longrightarrow B$. With this notion, Figure 3.1 depicts the structural (or, say, quantitative) results obtained in the remainder of this section.

Protocol	*Short description of what it computes*
Protocol 0	Dyadic rational 2^{-n}–approximation of f
Protocol 1	Dyadic rational $(1+L)\cdot 2^{-n}$–approximation of f
Protocol 2	Dyadic rational 2^{-n}–approximation of both f and f'
Protocol Y	Given $(q,n) \in \mathbb{D}_n \times \mathbb{Z}$, compute not-too-large 2^{-m}–approximation $(m \in \mathbb{Z})$ of $f|_{\mathrm{int}(q,n)}$
Protocol L	$4\cdot L\cdot(b-a)$–approximation of any y-interval $f([a,b])$
Protocol X	Given $(q,m_J) \in \mathbb{D}_n \times \mathbb{Z}$, compute a not-too-small 2^{-m}–approximation $(p,n_I) \in \mathbb{D}_{m_J} \times \mathbb{Z}$ such that $f(\mathrm{int}(q,n_I)) \subseteq \mathrm{int}(p,m_J)$
Protocol Coeff	Dyadic rational approximations to coefficients $a_i \in \mathbb{R}$, where polynomial $\sum_i a_i X^i$ approximates f within a prescribed error 2^{-n}
Protocol SLP	Straight line program Π for f so that the polynomial encoded by Π approximates f within a prescribed error 2^{-n}

Table 3.2: Short descriptions (modulo details) of each protocol examined in this thesis.

In contrast, the qualitative results are summarized in Table 1.2. As Figure 3.1 suggests, most entries are not obtained by direct results (i. e., results of direct comparisons), but by use of transitivity. Those results are indicated by entries of form $A \to^* C$, i. e., there is an (intermediate) protocol B so that $A \to B \to^* C$ or $A \to^* B \to C$.

Also, both direct and transitive results on *simulation complexity* help to ease the analysis of protocol's *computational complexity*. That is, let protocol A be polynomial-time computable, and the simulation $A \to^* B$ be polynomially bounded. Then protocol B also is polynomial-

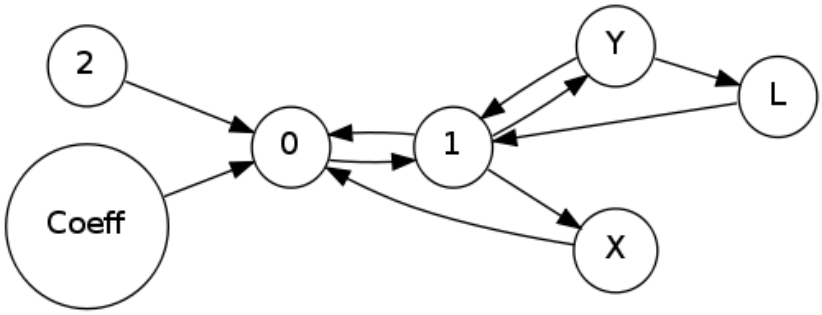


Figure 3.1: Overview of (direct) results regarding protocol's simulation complexity

time computable *in* the information N_A used to compute protocol A as well as in the information $N_{A\to^* B}$ used in the simulation of protocol B.

As part of the simulation of a certain protocol, it often is required to compute a dyadic rational of some fixed precision $m \in \mathbb{Z}$, while in the respective proofs we will often come up with a dyadic rational $\tilde{p} \in \mathbb{D}$ of lower precision $\tilde{m} \in \mathbb{Z}$, $\tilde{m} < m$. We denote the minimal distance between $\tilde{p} \in \mathbb{D}_{\tilde{m}}$ and its most nearby neighbor in $\mathbb{D}_m$ by $d(\tilde{p}, \mathbb{D}_m) = \min_{p\in\mathbb{D}_m} |\,\tilde{p} - p\,|$. This distance is always bounded by $0 \leq d(\tilde{p}, \mathbb{D}_m) \leq 2^{-(m+1)}$.

3.2 Protocols 0 and 1

The first protocols we like to examine are (in a sense) immediate formulations of real function computability: While protocol 0 provides rational dyadic approximations of function values in single dyadic rational points, protocol 1 on the other hand provides close approximations for f's image on a whole interval with dyadic rational endpoints on the cost of a possible higher approximation error measured, amongst others, in f's Lipschitz constant.

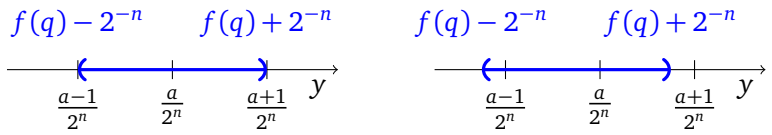
Definition 3.2.1 (protocol 0)**.** Let $f : \mathbb{R} \to \mathbb{R}$ be a continuous function. Given an integer $n \in \mathbb{Z}$ and a dyadic rational $q \in \mathbb{D}_n$ as inputs, protocol 0 returns a dyadic rational $p \in \mathbb{D}_n$ satisfying

$$f(q) - 2^{-n} < p < f(q) + 2^{-n}.$$

Protocol 1 is well-defined in the sense that there always exists at least one point $p \in \mathbb{D}_n$ such that $p \in (f(q) - 2^{-n}, f(q) + 2^{-n})$, because the distance of two consecutive points $p_1, p_2 \in \mathbb{D}_n$ (which is 2^{-n}) does not exceed the length of the open interval $(f(q) - 2^{-n}, f(q) + 2^{-n})$. Moreover, this interval contains at most two dyadic rational points of precision n. To see why, consider the minimum distance between three or more points of precision n: It is at least $3 \cdot 2^{-n}$, which exceeds the length of $(f(q) - 2^{-n}, f(q) + 2^{-n})$ by more than 2^{-n}. Both cases are depicted in Figure 3.2. Protocol 0 easily extends to higher dimensions, where intervals are replaced by the more general term of hypercubes.

Definition 3.2.2 (protocol 0, d-dimensional version, $d \in \mathbb{N}$)**.** Given a continuous function $f : \mathbb{R}^d \to \mathbb{R}$, along with inputs $n \in \mathbb{Z}$ and $\vec{q} \in \mathbb{D}_n^d$. Then, a d-dimensional version of protocol 0 returns a point $p \in \mathbb{D}_n$ satisfying

$$f(\vec{q}) - 2^{-n} < p < f(\vec{q}) + 2^{-n}.$$



(a) one dyadic rational point of $\mathbb{D}_n$ is contained

(b) two dyadic rational points of $\mathbb{D}_n$ are contained

Figure 3.2: Well-definedness of protocol 0: The interval $(f(q)-2^{-n}, f(q)+2^{-n})$ contains either one or two rational dyadic points in $\mathbb{D}_n$.

In contrast to protocol 0, the approximation error for protocol 1 is measured not only in the input precision $n \in \mathbb{Z}$, but also in the (global) Lipschitz constant of f. Since continuity in general does not imply Lipschitz continuity (only the other way around; for an example, consider $\sqrt{x}$ on $[0,1]$), the set of functions to define protocol 1 on is restricted to those functions being Lipschitz continuous with a *given and known*, but arbitrary constant $L > 0$.

The general definition of a d-dimensional version of protocol 1 (in comparison to Definition 3.2.2) requires it specify what we mean by Lipschitz continuity in higher dimensions. Therefore, a function $f : \mathbb{R}^d \to \mathbb{R}$ is L-Lipschitz continuous with constant $L > 0$ if $|f(\vec{x}) - f(\vec{y})| \leq L \cdot \|\vec{x} - \vec{y}\|_\infty$ holds for all real-valued d-tuples $\vec{x} := (x_1, \ldots, x_d)$, $\vec{y} := (y_1, \ldots, y_d) \in \mathbb{R}^d$, where $\|\vec{x}\|_\infty := \max_{1 \leq i \leq d} |x_i|$ denotes the maximum norm.

Definition 3.2.3 (protocol 1)**.** Let $f : \mathbb{R}^d \to \mathbb{R}$ be an L-Lipschitz continuous function. When given an integer $n \in \mathbb{Z}$ and a d-tuple $\vec{q} := (q_1, \ldots, q_d) \in \mathbb{D}_n^d$ as inputs, then protocol 1 returns a single dyadic rational $p \in \mathbb{D}_n$, satisfying

$$f(\vec{x}) - (1+L) \cdot 2^{-n} < p < f(\vec{x}) + (1+L) \cdot 2^{-n} \tag{3.1}$$

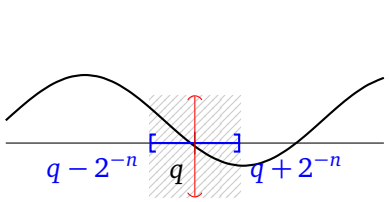
for all $\vec{x} := (x_1, \ldots, x_d) \in \mathbb{R}^d$ with $\vec{q} - 2^{-n} \leq \vec{x} \leq \vec{q} + 2^{-n}$. I. e., $\vec{x}$ lies in the d-dimensional hypercube spanned by the diagonal vertices $\vec{q} - 2^{-n}$ and $\vec{q} + 2^{-n}$.

Here, the following notation is introduced:

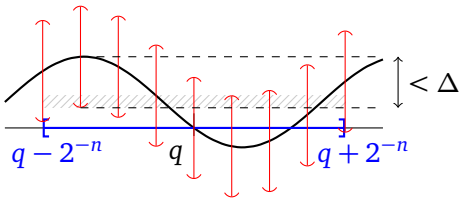
$$\begin{aligned} &\vec{q} - 2^{-n} := (q_1 - 2^{-n}, \ldots, q_d - 2^{-n}), \\ \text{and} \quad &\vec{q} - 2^{-n} \leq \vec{x} \text{ if and only if } q_i - 2^{-n} \leq x_i \text{ for all } i = 1, \ldots, d. \end{aligned}$$

Loosely speaking, the dyadic rational approximation $p \in \mathbb{D}_n$ is taken from a $(d+1)$-dimensional hypercube (a "strip" in case of $d = 1$) of height (in y-direction) $\leq (1+L) \cdot 2^{-n+1}$

so that p's distance to *every point* $\vec{y} \in \mathrm{int}(\vec{q}, n)$ is less than $(1+L)\cdot 2^{-n}$. Also, for a better understanding about the relation to protocol 0, a side-by-side comparison of both protocol is provided in Figure 3.3.



(a) Point p, protocol 0's approximation for $f(q)$, is contained in the shaded area.



(b) Point p, protocol 1's approximation for f in the blue interval, is contained in the shaded area; $\Delta := (1+L)\cdot 2^{-n}$.

Figure 3.3: Comparison of protocols 0 and 1 (for $d = 1$): Protocol 1 does not get a single d-tuple $\vec{q} \in \mathbb{D}_n^d$ as an input (like for protocol 0), but a whole hypercube, spanned by diagonal vertices $\vec{q} - 2^{-n}$ and $\vec{q} + 2^{-n}$.

Below we prove the well-definedness of protocol 1 and its connection (regarding simulatability) to protocol 0.

Lemma 3.2.4. *Protocol 1 exists for arbitrary Lipschitz-continuous functions. More precisely, given an L-Lipschitz continuous function $f : \mathbb{R}^d \to \mathbb{R}$, there is a point $p \in \mathbb{D}_n$ satisfying (3.1) for every integer $n \in \mathbb{Z}$ and every point $\vec{q} \in \mathbb{D}_n^d$.*

Proof. We only have to show that there is a point $p \in \mathbb{D}_n$ lying in the (uncountable) intersection of all intervals $(f(\vec{x}) - (1+L)\cdot 2^{-n}, f(\vec{x}) + (1+L)\cdot 2^{-n})$ for all $\vec{x} \in \mathrm{int}(\vec{q}, n)$. This can be restated as

$$\left(f(\vec{y}) + (1+L)\cdot 2^{-n}\right) - \left(f(\vec{x}) - (1+L)\cdot 2^{-n}\right) \geq 2^{-n+1}$$

for $\vec{x}, \vec{Y} \in \mathrm{int}(\vec{q}, n)$ with $f(\vec{x}) \geq f(\vec{y})$, i. e., the intervals $(f(\cdot) - (1+L)\cdot 2^{-n}, f(\cdot) + (1+L)\cdot 2^{-n})$ are overlapping just enough to contain at least one point $p \in \mathbb{D}_n$. So:

$$\begin{aligned}\left(f(y) + (1+L)\cdot 2^{-n}\right) - \left(f(x) - (1+L)\cdot 2^{-n}\right) &\geq 2\cdot(1+L)\cdot 2^{-n} - L\cdot 2^{-n+1} \\ &= 2^{-n+1}\end{aligned}$$

□

As a consequence of above's proof, we get a relationship between precision n and the Lipschitz constant L on one side, and the difference between f's minima and maxima on any hypercube $\mathrm{int}(\vec{q}, n)$, $\vec{q} \in \mathbb{D}_n^d$, on the other.

Proposition 3.2.5. *Given an L-Lipschitz continuous function $f : \mathbb{R}^d \to \mathbb{R}$, and a hypercube $int(\vec{q}, n)$ where $n \in \mathbb{Z}$, $\vec{q} \in \mathbb{D}_n^d$. Then* $\max f|_{int(\vec{q},n)} - \min f|_{int(\vec{q},n)} \leq (1+L) \cdot 2^{-n+1}$.

The proof actually is a consequence of f being L-Lipschitz continuous. Instead, we argue using the definition of protocol 1. First define

$$\mathscr{J} := \bigcap_{\vec{x} \in \mathrm{int}(\vec{q},n)} (f(\vec{x}) - (1+L) \cdot 2^{-n}, f(\vec{x}) + (1+L) \cdot 2^{-n})$$

and note that by Lemma 3.2.4 the set $\mathscr{J} \cap \mathbb{D}_n$ is non-empty, thus implying the existence of a dyadic rational $p \in \mathbb{D}_n$ coherent with protocol 1. Again, using the definition of protocol 1, this fact can be restated as

$$\mathscr{J} \neq \emptyset \Rightarrow 0 \leq \left(\min f|_{\mathrm{int}(\vec{q},n)} + (1+L) \cdot 2^{-n}\right) - \left(\max f|_{\mathrm{int}(\vec{q},n)} - (1+L) \cdot 2^{-n}\right)$$

which concludes the proof of above's proposition.

Theorem 3.2.6. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be an L-Lipschitz continuous function. Then protocols 0 and 1 are asymptotically equivalent when keeping L constant.*

Proof. Simulation $1 \to 0$: Given a precision $n \in \mathbb{Z}$ and a dyadic rational $\vec{q} \in ([0,X] \cap \mathbb{D}_n)^d$, protocol 1 returns a dyadic rational $p \in \mathbb{D}_n$ so that $|f(\vec{x}) - p| < (1+L) \cdot 2^{-n}$ holds in all $\vec{x} \in \mathrm{int}(\vec{q}, n)$, thus also for $\vec{x} = \vec{q}$. Query protocol 1 with precision $\tilde{n} := n + 1 + \lceil \log(1+L) \rceil$ and dyadic rational $\vec{q}$, then it returns a dyadic rational $\tilde{p} \in \mathbb{D}_{\tilde{n}}$ such that

$$|f(\vec{q}) - p| \leq |f(\vec{q}) - \tilde{p}| + |p - \tilde{p}| < (1+L) \cdot 2^{-\tilde{n}} + 2^{-(n+1)} \leq 2^{-n}$$

for $p \in \mathbb{D}_n$ with $|p - \tilde{p}| = d(\tilde{p}, \mathbb{D}_n)$. Returning p now simulates a query of protocol 0 by using only one call to protocol 1.

Simulation $0 \to 1$: For the converse simulation, we have to ensure that for inputs $\vec{q} \in ([0,X] \cap \mathbb{D}_n)^d$ and $n \in \mathbb{Z}$, the simulation algorithm returns a dyadic rational $p \in \mathbb{D}$ such that (a) it is of precision n and (b) it satisfies $|f(\vec{x}) - p| < (1+L) \cdot 2^{-n}$ for all $\vec{x} \in \mathrm{int}(\vec{q}, n)$.

For that, set $\underline{y} = \min f|_{\mathrm{int}(\vec{q},n)}$ and $\overline{y} = \max f|_{\mathrm{int}(\vec{q},n)}$. Denote the dyadic rational returned by protocol 0 when queried with precision n and dyadic rational $\vec{q} \in \mathbb{D}_n$ by p. It satisfies $|f(\vec{q}) - p| < 2^{-n}$ and also is of precision n. We instantly observe

$$|\overline{y} - p| \leq |\overline{y} - f(\vec{q})| + |f(\vec{q}) - p| < L \cdot 2^{-n} + 2^{-n} = (1+L) \cdot 2^{-n}.$$

The same holds for $|\underline{y} - p|$. Figure 3.4 also gives a graphical proof. □

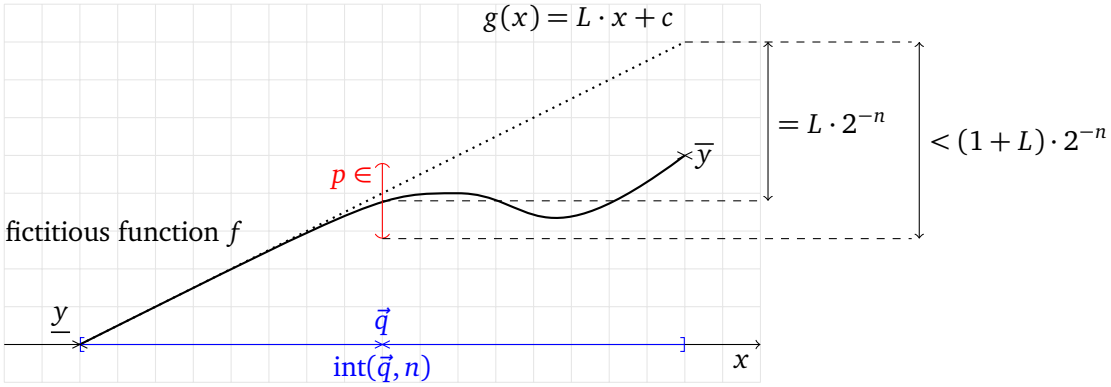


Figure 3.4: Schematic for simulation of protocol 1 by 0. There is no need to split up $x-$interval $\text{int}(\vec{q}, n)$ because the dyadic rational returned by protocol 0 already obeys the definition of protocol 1.

Computational Complexity

Protocol 0. If f is a (polynomial-time) computable function, then protocol 0 also is (polynomial-time) computable because, loosely speaking, protocol 0 is only a protocol formulation of a (polynomial-time) computable real function; its definition can be restated as

$$(\forall x \in [0,X])(\forall \phi)(\forall n \in \mathbb{Z}) \quad |x - \phi(n)| \leq 2^{-n} \\ \Rightarrow \left[M^{\phi}(n) \in \mathbb{D}_n \text{ and } |M^{\phi}(n) - f(x)| \leq 2^{-n}\right]$$

with $\phi(n) \in \mathbb{D}_n$ encoding the input $q \in \mathbb{D}_n$.

Protocol 1. As shown in the proof of Theorem 3.2.6, it only takes a constant amount of queries to protocol 0 in order to approximate $f(x)$ for all $x \in \text{int}(\vec{q}, n)$ within a prescribed error 2^{-n} as long as both the dimension d and the Lipschitz constant L is fixed. Thus, in combination with the (polynomial-time) computability of protocol 0, protocol 1 also is (polynomial-time) computable.

3.3 Protocol 2

The idea behind protocol 2 goes as follows: Equip protocol 2 with the ability to return approximations of finitely many derivatives of f in a given dyadic rational point. As described in Section 2.4, the computability of f' in combination with the ability to compute its roots could (in some situations) actually help to compute functional $\textsc{Max}(f)$. In this section, we lay out the ground work, before we use it in Section 4.1.2 to analyze the query complexity of approximating $\textsc{Max}(f)$ when given access to protocol 2.

Definition 3.3.1 (protocol 2)**.** Let $f : \mathbb{R} \to \mathbb{R}$ be a continuously differentiable function. Given an integer $n \in \mathbb{Z}$ and a point $q \in \mathbb{D}_n$ as inputs, protocol 2 returns points $p, p' \in \mathbb{D}_n$ satisfying

$$\begin{aligned} f(q) - 2^{-n} < p &< f(q) + 2^{-n}, \text{ and} \\ f'(q) - 2^{-n} < p' &< f'(q) + 2^{-n}. \end{aligned}$$

The existence of such dyadic rationals $p, p' \in \mathbb{D}_n$ directly follows from earlier discussions on protocol 0.

Simulation Complexity

It is obvious that, for example, protocol 0 can be simulated by only using a constant number (here: exactly one) of queries to protocol 2, thus yielding a whole chain of further simulation results; cf. Table 1.2 or Figure 3.1. The hard problem about simulating protocol 2 by another protocol is the need to come up with an (at least local) approximation of f'. A naive approach would go like this (when attempting to simulate protocol 2 by queries to protocol 0):

> Protocol 0 evaluates f on some rational open ball around q, say on $B_\delta(q)$ for some to-be-determined $\delta > 0$, in 2^{n^*} many points ($n^* \in \mathbb{N}$ for the sake of simplicity), computes an interpolation polynomial h for f based on them, and symbolically differentiates and finally evaluates h in q.

Several questions are coming up using this method: (a) What degree does such a polynomial has to have? (the evaluation time highly depends on the number of coefficients); (b) How do we have to choose δ and n^*?; (c) How to circumvent the typical adversary argument for interpolation polynomials? Those and related questions will be postponed until Section 3.7

where we explicitly analyze the simulation of protocol 2 by (the yet undefined) protocol Coeff that returns dyadic rational coefficients of an approximation polynomial for real-valued function f.

As we will see in the following: Unless we restrict our analysis to continuously differentiable computable real functions f with compact domain and the extension of f' having a computable modulus of continuity on $\mathrm{dom}(f)$, the first derivative f' is not even computable.

Computability and Computational Complexity

Theorem 2.3.8 states that, roughly speaking, if we restrict the set of functions $f : S \to \mathbb{R}$ (with compact domain $S \subseteq \mathbb{R}$) to those being (polynomial-time) computable, then protocol 2 is (polynomial-time) computable if and only if f' has a (polynomially bounded) recursive modulus of continuity over the whole domain. While this is an upper bound result on the (polynomial-time) computability for protocol 2, it also has been shown in [KF82, Theorem 5.1] that there is a polynomial-time computable function f which is *nowhere* differentiable on $[0,1]$. Moreover, Ko proved (in [Ko91, Theorem 6.4], extending a result from [Myh71]) that there even is a polynomial-time computable differentiable real function $f : [0,1] \to \mathbb{R}$ such that $f'(d)$ is *not* computable for any dyadic rational $d \in \mathbb{D}$. Thus, without the restriction on f's modulus of continuity being computable, protocol 2 in general is not even computable.

At least one further positive result is known (cf. [Ko91, Corollary 6.3]): If $f : [0,1] \to \mathbb{R}$ polynomial-time computable and *at least two times continuously differentiable*, then f' also is polynomial-time computable. In general, the first $k-1$ derivatives of f are polynomial-time computable if $f \in C^k[0,1]$ and polynomial-time computable itself.

3.4 Protocol Y

Protocol Y will be formulated in a way such that it implicitly delivers information about the *local* Lipschitz constant[1] of a function f. To actually define protocol Y, we only consider locally non-constant functions. Loosely speaking, the image of locally non-constant functions varies in its function values on arbitrarily small open ball on the whole domain. The reasons for the restriction to such functions are explained right after their definition.

[1] A function $f :\subseteq \mathbb{R} \to \mathbb{R}$ is said to have a local Lipschitz constant $L > 0$ at $x \in \mathrm{dom}(f)$ if there is a $\delta > 0$ so that $(\forall y \in \mathrm{dom}(f)) \, |x - y| \leq \delta \Rightarrow |f(x) - f(y)| \leq L \cdot |x - y|$.

Definition 3.4.1 (locally non-constant functions)**.** A continuous function $f : \mathbb{R} \to \mathbb{R}$ is said to be *locally non-constant* if f is non-constant on every open ball around x where $x \in \mathbb{R}$; i. e.,

$$(\forall x \in \mathbb{R})(\forall \varepsilon > 0)(\exists y \in B_\varepsilon(x)) \text{ such that } |f(x) - f(y)| > 0$$

where $B_\varepsilon(x) = \{z \in \mathbb{R} \mid |x - z| < \varepsilon\}$ denotes an open ball around x of radius $< \varepsilon$.

The restriction to locally non-constant functions is motivated by the following fact: Given an arbitrary continuous function $f : \mathbb{R} \to \mathbb{R}$ and an interval $[a, b] \subseteq \mathbb{R}$, then in general there is no notion of a *smallest* dyadic rational interval $\text{int}(p, m)$ containing $f([a, b])$, e. g., in case of $f|_{[a,b]} = 0$. In this situation, the precision m would have to go up to infinity in order to justify the notation of $\text{int}(p, m)$ being the smallest dyadic rational interval of that kind. Respecting the restriction to locally non-constant functions, the following definition now describes in details what we want to understand by "smallest".

Definition 3.4.2 (most suitable interval)**.** Let $f : \mathbb{R} \to \mathbb{R}$ be a locally non-constant L-Lipschitz continuous function, and $\text{int}(q, n)$ be a sub-interval of $\mathbb{R}$, where $n \in \mathbb{Z}$ and $q \in \mathbb{D}_n$. Furthermore, let $\text{int}(p, m)$ be an interval with $m \in \mathbb{Z}$ and $p \in \mathbb{D}_m$. We say, $\text{int}(p, m)$ is *most suitable for f on int(q, n)* if and only if the following conditions hold:

(a) The image of f on $\text{int}(q, n)$ is contained in $\text{int}(p, m)$, i. e., $f(\text{int}(q, n)) \subseteq \text{int}(p, m)$;

(b) The interval $\text{int}(p, m)$ can not be refined, i. e., $f(\text{int}(q, n)) \not\subseteq \text{int}(\tilde{p}, m + k)$ for all $k \geq 1$ and all $\tilde{p} \in \mathbb{D}_{m+k}$.

Condition (b) of Definition 3.4.2 provides an upper bound on the precision of a most suitable interval for continuous functions $f : \mathbb{R} \to \mathbb{R}$ and input intervals $\text{int}(q, n)$ along the x-axis. Besides that, condition (b) also requires a most suitable interval (denoted as J^{suit} to be located on the grid of dyadic rationals; that is, its endpoints have to be of the same precision, say m. Moreover, interval J^{suit} is required to be of length 2^{-m+1}, so the endpoints can not be arbitrarily far away while, by contrast, providing them with high accuracy.

Since $f(\text{int}(q, n))$ does *not* have to lie perfectly on the grid, there are situations in which J^{suit} might nearly have twice the size of $f(\text{int}(q, n))$. As an example, let $f : [0, 1] \to \mathbb{R}$ be an arbitrary locally non-constant function. Further assume that $2^{-(m+2)} < \max f|_{[a,b]} - \min f|_{[a,b]} < 2^{-(m+1)}$ for some $m \in \mathbb{Z}$ and some sub-interval $[a, b]$ of $[0, 1]$. This assumption does not imply that some y-interval of length $2^{-(m+1)}$ (i. e., $\text{int}(p, m+2)$ for some $p \in \mathbb{D}_{m+2}$) is most suitable for f on $[a, b]$. This situation is depicted in Figure 3.5.

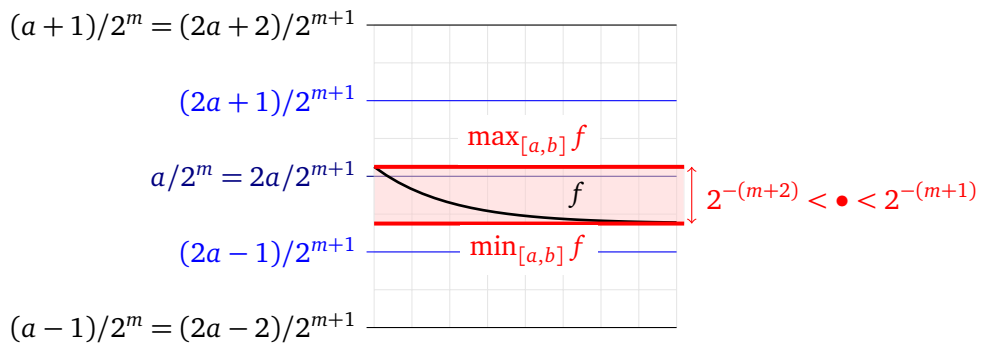


Figure 3.5: Rasterization around $f([a,b])$

There, $\mathrm{int}(a/2^m, m+1)$ would be a most suitable interval because

(a) $f(\mathrm{int}(q,n))$'s length exceeds $2^{-(m+2)}$, but it is bounded from above by $2^{-(m+1)}$;

(b) no interval of precision $m+2$ contains $f(\mathrm{int}(q,n))$.

Note that the second property is not true in general. For a slightly modified example where $f(\mathrm{int}(q,n))$ actually *is* contained in an interval of precision $m+2$, see Figure 3.6.

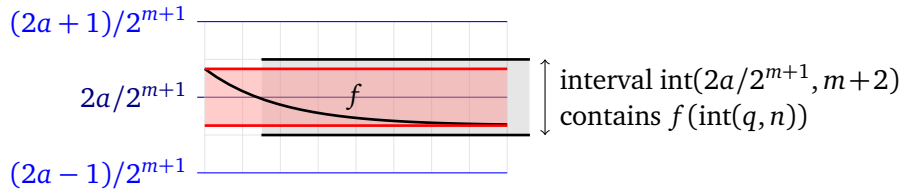


Figure 3.6: Moving $f(\mathrm{int}(q,n))$ may yield a more precise most suitable interval

The following two lemmata justify the definition of a most suitable interval by proving its existence and uniqueness.

Lemma 3.4.3 (existance of J^{suit})**.** *Given a locally non-constant continuous function $f : \mathbb{R} \to \mathbb{R}$, and an x-interval $int(q,n)$ with $n \in \mathbb{Z}$ and $q \in \mathbb{D}_n$. Then there exists a non-refinable y-interval $J^{suit} := int(p,m)$ containing $f(int(q,n))$.*

Proof. We state a proof by contradiction. Assume that for some $\tilde{m} \in \mathbb{Z}$ (chosen as $\tilde{m} \geq \mathrm{pcs}(f(\mathrm{int}(q,n))) - 1$) there is a sequence $(\mathrm{int}(p^{(k)}, \tilde{m}+k-1))_k$ of nested intervals (i. e., $\mathrm{int}(p^{(k+1)}, \tilde{m}+k) \subseteq \mathrm{int}(p^{(k)}, \tilde{m}+k-1)$ for all $k \geq 1$) such that $f(\mathrm{int}(q,n)) \subseteq \mathrm{int}(p^{(k)}, \tilde{m}+k-1)$ for all $k \geq 1$. This is equivalent to say that for our situation, there is *no* most suitable interval.

Note that such midpoints $p^{(k)}$ do exist; simply choose them to have minimal distance to the center of $f(\mathrm{int}(q,n))$, i. e., $|\,p^{(k)} - \mathrm{mid}(f(\mathrm{int}(q,n)))\,| = d(\mathrm{mid}(f(\mathrm{int}(q,n))), \mathbb{D}_{\tilde{m}+k-1})$. Hence,

$$f(\mathrm{int}(q,n)) \subseteq \big(\bigcap_{k\geq 1} \mathrm{int}(p^{(k)}, \tilde{m}+k-1)\big) =: \mathscr{J}\,.$$

Independent of how the sequence of nested intervals evolves, $\mathscr{J}$ becomes a degenerate interval. This leads to an immediate contradiction since we assumed that f is locally non-constant. So there must be a number $k_0 \in \mathbb{N}$ such that $f(\mathrm{int}(q,n)) \subseteq \mathrm{int}(p^{(k_0-1)}, \tilde{m}+k_0-2)$, but $f(\mathrm{int}(q,n)) \not\subseteq \mathrm{int}(p^{(k)}, \tilde{m}+k-1)$ for all $k \geq k_0$. Then $\mathrm{int}(p,m)$ with $m := \tilde{m}+k_0-2$ and $p := p^{(k_0-1)} \in \mathbb{D}_{\tilde{m}+k_0-2}$ is a candidate for a most suitable interval.

It remains to prove that $\mathrm{int}(p,m)$ (with $p = p^{(k_0-1)}$) is non-refinable, i. e., that there are no other values $\mathbb{D}_{m'} \ni p' \neq p$ with $m' = m+1$ so that $f(\mathrm{int}(q,n)) \subseteq \mathrm{int}(p',m')$. However, this follows directly by our choice of midpoints of having the minimal distance to the center of $f(\mathrm{int}(q,n))$. □

Lemma 3.4.4 (uniqueness of J^{suit})**.** *Given a function $f : \mathbb{R} \to \mathbb{R}$, and an x-interval $int(q,n)$ with $n \in \mathbb{Z}$ and $q \in \mathbb{D}_n$. Furthermore, let $int(p,m)$ be a y-interval with $f(int(q,n)) \subseteq int(p,m)$ that cannot be refined. Then $int(p,m)$ is unique with this property.*

Proof. First, let $p \in \mathbb{D}$ be of the form $p = a/2^m$ for some $a \in \mathbb{Z}$. Assume that there is another non-refinable y-interval $\mathrm{int}(\tilde{p},m)$ different from $\mathrm{int}(p,m)$, where $f(\mathrm{int}(q,n)) \subseteq \mathrm{int}(\tilde{p},m)$ and $\tilde{p} := (a \pm 1)/2^m$.[2] W. l. o. g., let $\tilde{p} = (a+1)/2^m$; the other case follows by the same argument. Since, by our assumption, both y-intervals $\mathrm{int}(p,m)$ and $\mathrm{int}(\tilde{p},m)$ are containing $f(\mathrm{int}(q,n))$, we get

$$f(\mathrm{int}(q,n)) \subseteq \big(\mathrm{int}(p,m) \cap \mathrm{int}(\tilde{p},m)\big)\,,$$

which implies

$$f(\mathrm{int}(q,n)) \subseteq [p, p+2^{-m}] = \mathrm{int}((2a+1)/2^{m+1}, m+1) =: \mathscr{J}\,.$$

The gained interval $\mathscr{J}$ is of precision $\mathrm{pcs}(\mathscr{J}) = m+1 < m = \mathrm{pcs}(\mathrm{int}(p,m))$, but $\mathrm{int}(p,m)$ was assumed to be non-refinable. We get a contradiction to our assumption, proving $\mathrm{int}(p,m)$ to be unique. □

[2] In (the fictitious) case of $|\,p-\tilde{p}\,| > 2^{-m}$, the intersection of their respective intervals would be either empty or a degenerate interval. Either way, this cannot happen because $f(\mathrm{int}(q,n))$ is supposed to be contained in this intersection. Therefore, $\tilde{p} := (a \pm 1)/2^m$ are the only possible values for $\tilde{p}$.

For the restricted case of $f : [0,1] \to \mathbb{R}$ being 1-Lipschitz continuous, the image of f on some interval $I \subseteq [0,1]$ can be nicely bounded:

Proposition 3.4.5. *Given a dyadic rational interval $int(q,n) \subseteq [0,1]$ with $n \in \mathbb{Z}$ and $q \in \mathbb{D}_n$. Then, for every 1-Lipschitz continuous function $f : [0,1] \to \mathbb{R}$, there exists a dyadic rational $p \in \mathbb{D}_{n-1}$ such that $f(int(q,n)) \subseteq int(p,n-1)$. Furthermore, there are functions for whom such a y-interval $int(p,n-1)$ cannot be refined.*

Now we have everything together to actually define protocol Y.

Definition 3.4.6 (protocol Y)**.** Let $f : \mathbb{R} \to \mathbb{R}$ be a locally non-constant continuous function. Given an integer $n \in \mathbb{Z}$ and a dyadic rational $q \in \mathbb{D}_n$, protocol Y returns an integer $m \in \mathbb{Z}$ and a dyadic rational $p \in \mathbb{D}_m$. Let $J := \text{int}(p,m)$ be the interval described by the outputs, and $J^{\text{suit}} = \text{int}(p^{\text{suit}}, m^{\text{suit}})$ the most suitable interval for f on $\text{int}(q,n)$. Then interval J (a) has to be "not-too-small", i. e., $m^{\text{suit}} - 1 \leq m \leq m^{\text{suit}}$, and (b) $p \in \mathbb{D}_m$ has to be chosen such that $J^{\text{suit}} \subseteq J$.

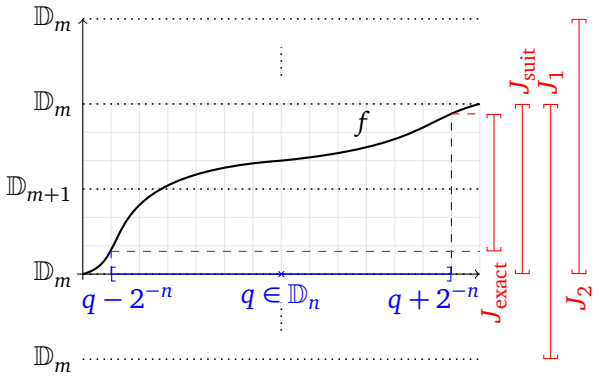


Figure 3.7: Schematic of protocol Y; possible choices J_i of the to-be-returned y-interval and their relation to J^{suit}.

Figure 3.7 depicts the choice protocol Y has about the returned y-interval and its relation to J^{suit}. As a side note, having an interval J of less than twice the length as of the most suitable interval J^{suit} for f on an x-interval I implies $\text{len}(J) < 4 \cdot \text{len}(f(I))$. As opposed to the computation of the most suitable interval, this relaxation is introduced to enable protocol Y to be at least computable. We treat this problem and choice at the end of this section.

Simulating Protocol Y by 1 and vice versa

There are two choices: We could either measure the simulation complexity in some shared information about both protocols, or the one we will now discuss. Assume we want to simulate protocol Y by an appropriate (and yet to be determined) number of queries to protocol 1. From protocol 1, the function's Lipschitz constant L can be retrieved and used by the "simulation algorithm". When exchanging both protocols, a simulation algorithm for protocol 1 with access to protocol Y does *not* have this information about L; in fact, now there is no shared information to measure the simulation complexity in; in short, $(N_{1\to Y} \cap N_{Y\to 1}) = \emptyset$.

To circumvent this problem, we equip the simulation algorithm with additional information about the function (e. g., about its moduli), and study the simulation complexity in those additions.

Theorem 3.4.7. *Let $f : [0,X] \to \mathbb{R}$ (for some arbitrary, but fixed $X > 0$) be a Lipschitz continuous computable real function with Lipschitz constant $L > 0$. Then the following bounds hold for the mutual simulation of protocols 1 and Y.*

(a) For the additional restriction on f being locally non-constant, simulating protocol 1 takes $\mathcal{O}(1+L)$ queries to protocol Y.

(b) In addition, let f have both a modulus of continuity $\overline{m}(\cdot)$ and a modulus of strong unicity $\underline{m}(\cdot)$. Also, let $\mathrm{int}(q,n)$, $q \in \mathbb{D}_n \cap [0,X]$, be some x-interval. Then it takes $\mathcal{O}(1+2^{\underline{m}(n)-n})$ queries to simulate a modified version *of protocol Y on $\mathrm{int}(q,n)$ by protocol 1 where it is allowed to return a y-interval which has at most* four times *the size of the most suitable interval.*

Proof of Theorem 3.4.7, simulation of protocol 1 by Y. The simulation algorithm for protocol 1 receives as inputs a precision $n \in \mathbb{Z}$ and a dyadic rational $q \in \mathbb{D}_n$, where it has to compute a dyadic rational $p \in \mathbb{D}_n$ such that $|f(x) - p| < (1+L)\cdot 2^{-n}$ for all $x \in \mathrm{int}(q,n)$. On the other side, protocol Y (which the simulation algorithm has given access to) requires an x-interval (say, $\mathrm{int}(\tilde{q},\tilde{n})$) as an input, and returns values $\tilde{m} \in \mathbb{Z}$ and $\tilde{p} \in \mathbb{D}_{\tilde{m}}$ that satisfy (a) $J^{\mathrm{suit}} \subseteq \mathrm{int}(\tilde{p},\tilde{m})$ and (b) $m^{\mathrm{suit}} - 1 \le \tilde{m} \le m^{\mathrm{suit}}$. Recall that m^{suit} was defined in protocol Y as $m^{\mathrm{suit}} = \max\{k \in \mathbb{Z} \mid \exists p \in \mathbb{D}_k \ : \ f(\mathrm{int}(\tilde{q},\tilde{n})) \subseteq \mathrm{int}(p,k)\}$.

Denote $\min f$ and $\max f$ on $\mathrm{int}(q,n)$ by $\underline{y}$ and $\overline{y}$, respectively. As mentioned, protocol 1 imposes a restriction on its output $p \in \mathbb{D}_n$ which has to be checked for all $x \in \mathrm{int}(q,n)$. Using f's continuity, this can be rephrased as (1) $|\overline{y} - p| < (1+L)\cdot 2^{-n}$ plus (2) $|\underline{y} - p| <$

$(1+L)\cdot 2^{-n}$. If both conditions are met, then they also hold for all intermediate points $\underline{y} \le y \le \overline{y}$.

It is the challenge of this simulation to determine a suitable $p \in \mathbb{D}$ of the *right* precision. This task is split up into two parts: First, partition interval $\mathrm{int}(q,n)$ into $2^{\tilde{n}-n}$ equally-sized intervals $\mathrm{int}(\tilde{q}_i,\tilde{n})$, where $\tilde{n}$ has to be suitably large. Second, recombine the resulting y-intervals to get a dyadic rational $p \in \mathbb{D}_n$ coherent with protocol 1.

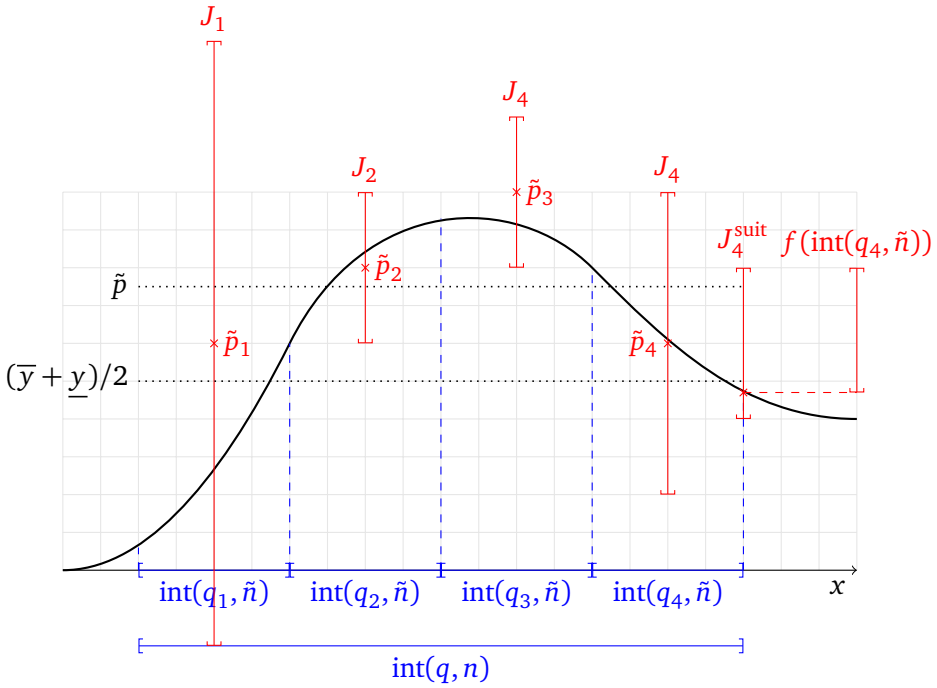


Figure 3.8: Connection between introduced variables for proving the simulation complexity of protocol 1 by access to protocol Y. As an example, the given interval $\mathrm{int}(q,n)$ is partitioned into four sub-intervals $\mathrm{int}(q_i,\tilde{n})$ whose respective approximative y-intervals (obtained by protocol Y) are denoted by J_i.

We start by conceiving both an upper and lower bound for $\tilde{n}$ based on parameters known to the simulation algorithm. For that, let $\tilde{p}_i$ and $\tilde{m}_i$ be values obtained from protocol Y when queried with intervals $\mathrm{int}(\tilde{q}_i,\tilde{n})$, and denote the resulting interval $\mathrm{int}(\tilde{p}_i,\tilde{m}_i)$ by J_i. Also set $\overline{\tilde{p}} := \max_i \tilde{p}_i$, $\underline{\tilde{p}} := \min_i \tilde{p}_i$ and $\tilde{p} := (\overline{\tilde{p}}+\underline{\tilde{p}})/2 \in \mathbb{D}_{\overline{\tilde{m}}+1}$. Accordingly, let $\underline{\tilde{m}} := \min_i \tilde{m}_i$ and $\overline{\tilde{m}} := \max_i \tilde{m}_i$. Figure 3.9 depicts the connection between these variables.

Now $\tilde{n}$ has to be chosen large enough so that $\tilde{p}$ has a close dyadic rational neighbor of precision n whose distance to each $y \in f(\mathrm{int}(q,n))$ differs at most by the tolerance allowed

in protocol 1, i. e., $\tilde{p}$ is guaranteed to satisfy $\max\{|\overline{y}-\tilde{p}|,|\underline{y}-\tilde{p}|\}+d(\tilde{p},\mathbb{D}_n)<(1+L)\cdot 2^{-n}$. Combining f's Lipschitz continuity with the definition of protocol Y implies

$$\operatorname{len}(J_i)=2^{-\tilde{m}_i+1}<4\cdot\left(\max f|_{\operatorname{int}(\tilde{q}_i,\tilde{n})}-\min f|_{\operatorname{int}(\tilde{q}_i,\tilde{n})}\right)\leq 4\cdot L\cdot 2^{-\tilde{n}+1}. \tag{3.2}$$

Also, we have $\max\{|\overline{y}-\tilde{p}|,|\underline{y}-\tilde{p}|\}<L\cdot 2^{-n}+2^{-\underline{\tilde{m}}+1}$. This is due to $\underline{y}\leq\underline{\tilde{p}}+2^{-\underline{\tilde{m}}+1}$ and $\overline{y}\leq\overline{\tilde{p}}+2^{-\underline{\tilde{m}}+1}$ implying

$$|(\underline{y}+\overline{y})/2-\tilde{p}|\leq(\underline{y}+\overline{y})/2-(\overline{y}+\underline{y}-2^{-\underline{\tilde{m}}+2})/2\leq 2^{-\underline{\tilde{m}}+1}.$$

Set $\tilde{n}:=n+3+\lceil\log(1+L)\rceil$. Then by Equation (3.2) we get

$$\begin{aligned}\max\{|\overline{y}-\tilde{p}|,|\underline{y}-\tilde{p}|+d(\tilde{p},\mathbb{D}_n)&<L\cdot 2^{-n}+4\cdot L\cdot 2^{-\tilde{n}+1}+2^{-(n+1)}\\&<(1+L)\cdot 2^{-n}.\end{aligned}$$

To conclude, when the function's Lipschitz constant L is known by the algorithm, then $\tilde{n}$ can be chosen as above and therefore a provably suitable dyadic rational $p\in\mathbb{D}_n$ is computable that is coherent with the definition of protocol 1. □

In fact, the second part of Theorem 3.4.7 is more complicated to prove than the converse situation because now, we have to compute a "close enough" y-interval without actually knowing about the exact one (i. e., about $f(\operatorname{int}(q,n))$). So, at first, it seems unclear what "close enough" might mean to a simulation algorithm when the subject itself is not given (i. e., the exact interval). In the following proof, we will resolve this problem by heavily relying on both the modulus of continuity $\overline{m}(\cdot)$ and the modulus of strong unicity $\underline{m}(\cdot)$. To use both concepts, the set of functions for which such a simulation is provably possible is restricted to computable functions having (at least) computable moduli $\overline{m}(\cdot)$ and $\underline{m}(\cdot)$.

Proof of Theorem 3.4.7, simulation of protocol Y by 1. The simulation algorithm (we will devise here) receives as inputs both a precision $n\in\mathbb{Z}$ and a dyadic rational $q\in\mathbb{D}_n$, and computes a precision $m\in\mathbb{Z}$ as well as a dyadic rational $p\in\mathbb{D}_m$ so that $\operatorname{int}(p,m)$ is a y-interval consistent with protocol Y. I. e., $m^{\text{suit}}-1\leq m\leq m^{\text{suit}}$ for $m^{\text{suit}}=\max\{k\in\mathbb{Z}\mid\exists p\in\mathbb{D}_k: f(\operatorname{int}(q,n))\subseteq\operatorname{int}(p,k)\}$. As proposed, the algorithm has access to protocol 1 so its simulation of protocol Y can rely on dyadic rational approximations of f in a finite amount of breakpoints.

The basic idea behind the proof goes as follows:

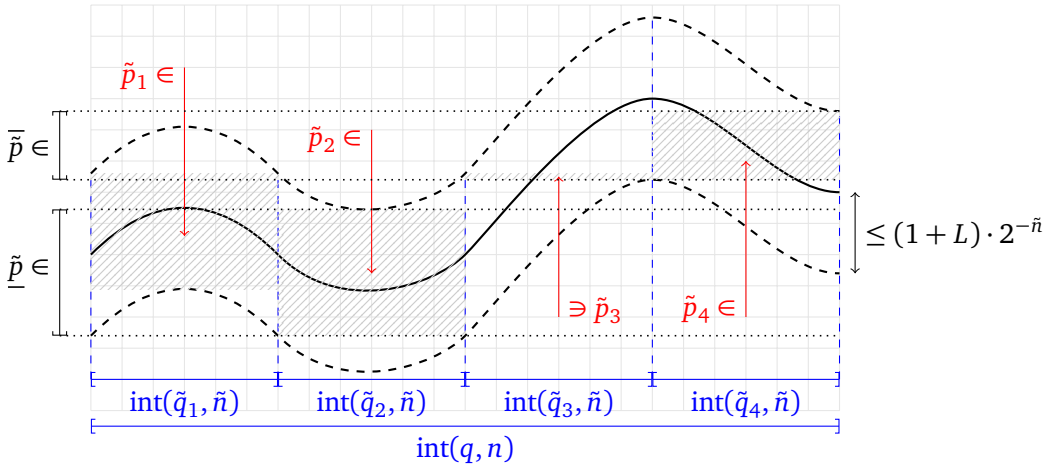


Figure 3.9: Information given by protocol 1 (when partitioning $\mathrm{int}(q,n)$ into pairwise distinct sub-intervals $\mathrm{int}(\tilde{q}_i, \tilde{n})$) that can be used to simulate a protocol Y query. The shaded areas are the ranges of possible dyadic rationals $\tilde{p}_i \in \mathbb{D}_{\tilde{n}}$ which approximate $f(\mathrm{int}(\tilde{q}_i, \tilde{n}))$ coherent with protocol 1.

(a) Split the given x-interval $\mathrm{int}(q,n)$ into $2^{\tilde{n}-n}$ equally sized intervals $\mathrm{int}(\tilde{q}_i, \tilde{n})$. (For future reference, $\tilde{n}$ will depend on n as well as the moduli $\underline{m}(\cdot)$ and $\overline{m}(\cdot)$, i. e., on f's "steepness".)

(b) Apply protocol 1 to each interval $\mathrm{int}(\tilde{q}_i, \tilde{n})$; the respective results are denoted by $\tilde{p}_i \in \mathbb{D}_{\tilde{n}}$.[3]

(c) Devise non-refinable $m \in \mathbb{Z}$ and $p \in \mathbb{D}_m$ so that (i) $\mathscr{J} \subseteq \mathrm{int}(p,m)$ with $\mathscr{J} := [\underline{\tilde{p}} - (1+L)\cdot 2^{-\tilde{n}}, \overline{\tilde{p}} + (1+L)\cdot 2^{-\tilde{n}}]$, (ii) $[\underline{y}, \overline{y}] \subseteq \mathscr{J}$, but also (iii) $1.5 \cdot \mathrm{len}(\mathrm{int}(p,m)) < 2 \cdot (\overline{y} - \underline{y})$.[4]

Note that the last point (c) is the hardest one for reasons explained prior to this proof.

Notation: $\underline{y} := \min f|_{\mathrm{int}(q,n)}$, $\overline{y} := \max f|_{\mathrm{int}(q,n)}$, $\underline{\tilde{p}} := \min_i \tilde{p}_i$, $\overline{\tilde{p}} := \max_i \tilde{p}_i$. Also, we express the Lipschitz constant as special modulus of continuity. For that, set $\overline{m}(n) := n + \log(L)$. Then, by definition of the modulus of continuity,

$$|x-y| \leq 2^{-\overline{m}(\tilde{n})} = 2^{-(\tilde{n}+\log(L))} \Rightarrow |f(x)-f(y)| \leq L \cdot 2^{-\overline{m}(\tilde{n})} = 2^{-\tilde{n}}$$

for all $x, y \in [0, X]$ and $n \in \mathbb{Z}$

[3] Remember that by definition of protocol 1 we get $|f(x) - \tilde{p}_i| < (1+L)\cdot 2^{-\tilde{n}}$ for all $x \in \mathrm{int}(\tilde{q}_i, \tilde{n})$ and all $i = 1, \ldots, 2^{\tilde{n}-n}$.

[4] The interval J returned by protocol Y is allowed to be at most two times larger than the most suitable interval J^{suit}, whereas J^{suit} itself has (in every situation) less than two times the size of $[\underline{y}, \overline{y}]$. Relating J and $[\underline{y}, \overline{y}]$ leads to $\mathrm{len}(J) < 2 \cdot (\overline{y} - \underline{y})$.

If $\tilde{n}$ is chosen so that

$$1.5 \cdot (1+L) \cdot 2^{-\tilde{n}+2} \leq 0.5 \cdot 2^{-\underline{m}(n)}, \tag{3.3}$$

then condition (c)(iii) also is satisfied. Moreover, Equation (3.3) makes sure that the approximation error (the difference between the exact y-interval's length $\overline{y} - \underline{y}$ and the length of our computed approximation $\mathscr{J}$) is small enough (...). This is, on the one hand, due to

$$\mathrm{len}(\mathscr{J}) = \overline{\tilde{p}} - \underline{\tilde{p}} + (1+L) \cdot 2^{-\tilde{n}+1} < \overline{y} - \underline{y} + (1+L) \cdot 2^{-\tilde{n}+2}, \tag{3.4}$$

inferred by the weak relationship $|(\overline{y} - \underline{y}) - (\overline{\tilde{p}} - \underline{\tilde{p}})| < (1+L) \cdot 2^{-\tilde{n}+1}$ between $\mathrm{len}([\underline{y}, \overline{y}])$ and $\mathrm{len}([\underline{\tilde{p}}, \overline{\tilde{p}}])$, and on the other hand a result of the modulus of strong unicity

$$(\forall x, z \in \mathrm{int}(q,n)) \quad 2^{-\underline{m}(n)} < |f(x) - f(z)| \leq \overline{y} - \underline{y}. \tag{3.5}$$

More precisely, Equation (3.4) provides the lower estimation for Equation (3.3), while Equation (3.5) provides the upper one:

$$\begin{aligned} 1.5 \cdot \mathrm{len}(\mathscr{J}) &\overset{(3.4)}{<} 1.5 \cdot (\overline{y} - \underline{y} + (1+L) \cdot 2^{-\tilde{n}+2}) \overset{(3.3)}{\leq} 2 \cdot 2^{-\underline{m}(n)} \\ &\overset{(3.5)}{<} 2 \cdot (\overline{y} - \underline{y}). \end{aligned}$$

By definition of $\overline{m}(\cdot)$, term $(1+L) \cdot 2^{-\tilde{n}+2}$ can be rewritten as $(1+2^{\overline{m}(\tilde{n})-\tilde{n}}) \cdot 2^{-\tilde{n}+2}$. So, bottom line, for $\tilde{n}$ to satisfy (3.3) it has to be chosen so that the following inequality holds:[5]

$$\min\{\tilde{n},\, 2 \cdot \tilde{n} - \overline{m}(\tilde{n})\} \geq \underline{m}(n) + \log(12). \tag{3.6}$$

Let $\mathrm{int}(t,k)$, $t \in \mathbb{D}_k$, be a non-refinable interval, i. e., there are *no* other values $k' > k$ and $t' \in \mathbb{D}_{k'}$ so that $[\underline{y}, \overline{y}] \subseteq \mathrm{int}(t',k')$, which implies $\mathrm{int}(t,k+1) \subset [\underline{y}, \overline{y}] \subseteq \mathrm{int}(t,k)$. We differentiate between two cases. First, assume $[\underline{y}, \overline{y}] = \mathrm{int}(t,k)$. This implies $J^{\mathrm{suit}} = \mathrm{int}(t,k)$. Combined with $1.5 \cdot \mathrm{len}(\mathscr{J}) < 2 \cdot \mathrm{len}(J^{\mathrm{suit}})$ we get $\mathscr{J} \subset \mathrm{int}(t,k-1)$. Since $t \in \mathbb{D}_k$ might not be of precision $k-1$ we have to choose a close neighbor $\tilde{t} \in \mathbb{D}_{k-1}$ so that $\mathscr{J} \subset \mathrm{int}(\tilde{t}, k-1)$, but then we finally have found a dyadic rational interval which is coherent with the *modified* definition of protocol Y, allowing an interval which has at most *four times* the size of the most suitable interval.

[5] Notice that the constant terms $-\log(1.5)$, -2 and $-\log(0.5)$ of (3.6) appearing as $(\ldots) - \log(1.5) - 2 \geq (\ldots) - \log(0.5)$, are combined to $\log(12)$ on the right hand side.

For the second case assume $[\underline{y},\overline{y}] \subset \mathrm{int}(t,k)$. Because of $\mathrm{int}(t,k+1) \subset [\underline{y},\overline{y}]$ we get $J^{\mathrm{suit}} = \mathrm{int}(t,k)$, so the rest goes as analogous to the first case. □

As has been proven above, an at most four times too large y-interval approximation (compared with J^{suit}) is computable under the given conditions. However, the original version of protocol Y only allowed an approximation of at most *twice* the length of J^{suit}. This observation combined with the inability of any simulation algorithm to compute an upper bound on the approximation error so that $\mathscr{J}$ is guaranteed to be not too large gives credit to our following conjecture.

Conjecture 1. No deterministic simulation algorithm for protocol Y relative to protocol 1 is capable to return a y-interval that obeys the original definition of protocol Y.

Computational Complexity

Theorem 2.3.7 *only* provides an upper bound on the length of the encoding of a y-interval J,[6] but no lower bound.

In 1981, Gaganov showed (cf. [KLRK98, Theorem 3.1]) that computing an *absolute* ε–*approximation* of $f(\mathrm{int}(q,n))$ (for some x-interval $\mathrm{int}(q,n)$) is NP–hard. To compute an absolute ε–approximation means to come up with (dyadic) rational endpoints $\underline{y},\overline{y} \in \mathbb{D}$ so that $0 \leq \underline{f(\mathrm{int}(q,n))} - \underline{y} \leq \varepsilon$ and $0 \leq \overline{y} - \overline{f(\mathrm{int}(q,n))} \leq \varepsilon$. However, it does not apply to our problem of computing an approximative y-interval with a *relative* (and unknown explicitly) error that depends on the exact interval's length.

Whether or not the proof of Gaganov's statement (see [KLRK98, p. 47]) can be adapted to also cover protocol Y is open by now. However, we have obtained complexity results for the *modified version* of protocol Y (for now denoted by Y*) while analyzing its simulation complexity, namely: For locally non-constant computable real functions $f : [0,X] \to \mathbb{R}$, protocol Y* is (exponential-time) computable if f has a (polynomially-bounded) computable modulus of strong unicity.

[6] To recall: polynomial-time computability of f implies the existence of a polynomially-bounded modulus of continuity for f.

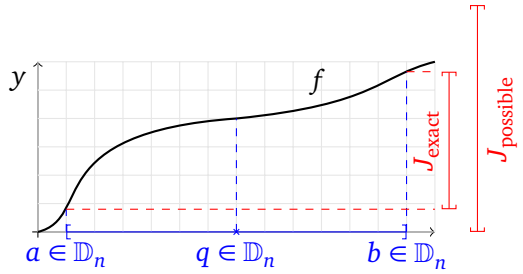


Figure 3.10: Schematic of protocol L. A y-interval J_{possible} returned by protocol L has to satisfy $\text{len}(J_{\text{possible}}) < 4 \cdot L \cdot \text{len}(J_{\text{exact}})$ and $J_{\text{exact}} \subseteq J_{\text{possible}}$.

3.5 Protocol L

For the definition of protocol Y we restricted the set of functions to those being locally non-constant because constant functions would have made it impossible to define a most suitable interval. By definition of protocol L we will trade this restriction for a (significantly) higher worst-case bound on the length of y-intervals returned by protocol L.

Definition 3.5.1 (protocol L)**.** Let $f : \mathbb{R} \to \mathbb{R}$ be an L-Lipschitz continuous function. Given a precision $n \in \mathbb{Z}$ and a dyadic rational $q \in \mathbb{D}_n$, protocol L returns a precision $m \in \mathbb{Z}$ and a dyadic rational $p \in \mathbb{D}_m$ so that (a) $f(\text{int}(q,n)) \subseteq \text{int}(p,m)$, and (b) $\text{len}(\text{int}(p,m)) < 4 \cdot L \cdot \text{len}(\text{int}(q,n))$.

The definition is visualized in Figure 3.10. The existence of protocol L on the other hand is a direct consequence of f's Lipschitz continuity.

Lemma 3.5.2 (existance of protocol L)**.** *Given an L-Lipschitz continuous function $f : \mathbb{R} \to \mathbb{R}$, and a dyadic rational x-interval $\text{int}(q,n)$ where $n \in \mathbb{Z}$ and $q \in \mathbb{D}_n$. Then there is a dyadic rational y-interval $\text{int}(p,m)$ satisfying Definition 3.5.1.*

Proof. Set $m := \text{pcs}(f(\text{int}(q,n)))$, and $\tilde{m} := \lfloor m - 1 \rfloor$. Fact: There is a $p \in \mathbb{D}_{\tilde{m}}$ so that $|\,\text{mid}(f(\text{int}(q,n))) - p\,| \leq 2^{-(\tilde{m}+1)}$. Then $f(\text{int}(q,n))$ is contained in $\text{int}(p,\tilde{m})$ and we get

$$\begin{aligned}\text{len}(\text{int}(p,\tilde{m})) = 2^{-\tilde{m}+1} &< 2^{-m+3} = 4 \cdot \text{len}(f(\text{int}(q,n))) \\ &\leq 4 \cdot L \cdot \text{len}(\text{int}(q,n)).\end{aligned}$$

□

Note that neither the Definition nor this Lemma imply uniqueness regarding such a y-interval $\text{int}(p,m)$.

Simulation Complexity

We introduced Protocol L by motivating it as a relaxed version of protocol Y. The simulation costs of that relaxation are analyzed below.

Theorem 3.5.3. *Let $f : [0,X] \to \mathbb{R}$ (with an arbitrary, but fixed $X > 0$) be a Lipschitz continuous computable real function with Lipschitz constant $L > 0$. Then the following bounds apply.*

(a) One query to protocol Y suffices to simulate protocol L.

(b) The simulation of protocol 1 requires linearly many queries (in L) to protocol L, where for the simulation of protocol L by protocol 1, $\mathcal{O}(1 + 1/L)$ queries are sufficient.

(c) In addition, let f have both a modulus of continuity $\overline{m}(\cdot)$ and a modulus of strong unicity $\underline{m}(\cdot)$. Then, the simulation of the modified (relaxed) version *of protocol Y requires at most polynomially many queries (in $\underline{m}(\cdot)$ and $\overline{m}(\cdot)$) to protocol L.*

Proof of Theorem 3.5.3, simulation of protocol L by Y. Since protocol L is a less restrict version of protocol Y, one query to protocol Y suffices to simulate L. Let $\mathrm{int}(q,n)$ be the input to and $\mathrm{int}(p,m)$ the answer of protocol Y. Then

$$\mathrm{len}(\mathrm{int}(p,m)) \le 2 \cdot \mathrm{len}(J^{\mathrm{suit}}) < 4 \cdot \mathrm{len}(f(\mathrm{int}(q,n))) \le 4 \cdot L \cdot \mathrm{len}(\mathrm{int}(q,n)),$$

so interval $\mathrm{int}(p,m)$ is consistent with protocol L. □

Proof of Theorem 3.5.3, simulation $1 \longleftrightarrow L$. The simulation of protocol 1 by L will be discussed in the subsequent proof. For now we only prove the counter direction. For that, let $\mathrm{int}(q,n)$ be a given x-interval and $L > 0$ be the function's Lipschitz constant. This proof now follows the idea applied before: First, partition $\mathrm{int}(q,n)$ into smaller intervals $\mathrm{int}(\tilde{q}_i,\tilde{n})$ where $\tilde{n}$ is set as $\tilde{n} = n+2+\log((1+L)/L)$. Querying protocol 1 with these intervals results in y-intervals $\mathrm{int}(\tilde{p}_i,\tilde{n})$ so that $|f(x) - \tilde{p}_i| < (1+L) \cdot 2^{-\tilde{n}}$ for all i and $x \in \mathrm{int}(\tilde{q}_i,\tilde{n})$.

Set $\mathscr{J} := [\min_i \tilde{p}_i - (1+L) \cdot 2^{-\tilde{n}}, \max_i \tilde{p}_i + (1+L) \cdot 2^{-\tilde{n}}]$. This immediately results in (a) $f(\mathrm{int}(q,n)) \subseteq \mathscr{J}$ and (b) $\mathrm{len}(\mathscr{J}) < |f(\mathrm{int}(q,n))| + 4 \cdot (1+L) \cdot 2^{-\tilde{n}} \le 2 \cdot L \cdot |\mathrm{int}(q,n)|$. This concludes the proof since the search for close-enough values $m \in \mathbb{Z}$ and $p \in \mathbb{D}_m$ so that $\mathscr{J} \subseteq \mathrm{int}(p,m)$ is not only easy, but also results in an interval $\mathrm{int}(p,m)$ of at most twice the size of $\mathscr{J}$. □

Proof of Theorem 3.5.3, simulation of protocol Y by L. Instead of proving $L \to Y$ directly, we prove $L \to 1$ and use $1 \to Y$. To simulate protocol 1 we have to compute a dyadic rational $p \in \mathbb{D}_n$ so that $|f(x) - p| < (1+L) \cdot 2^{-n}$ for all $x \in \text{int}(q,n)$.

We define $\underline{\delta}$ and $\overline{\delta}$ as in Figure 3.11 as the maximum distance between $\underline{f(\text{int}(q,n))}$ and $\overline{f(\text{int}(q,n))}$, and the midpoint of the hypothetical y-interval $\text{int}(p,m)$, respectively. Both $\underline{\delta}$ and $\overline{\delta}$ are bounded by

$$(\overline{y} + \underline{y})/2 \leq \max\{\underline{\delta}, \overline{\delta}\} \leq 2^{-m} < 4 \cdot L \cdot 2^{-n}. \tag{3.7}$$

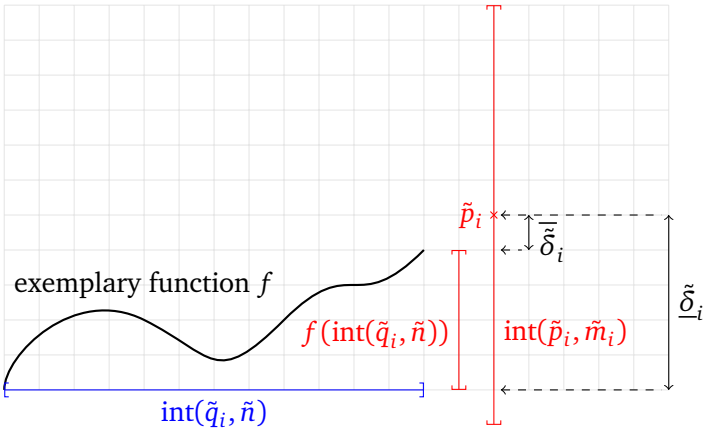


Figure 3.11: Simulating protocol 1 by L: Relation of $\underline{\tilde{\delta}}_i$ and $\overline{\tilde{\delta}}_i$ to a y-interval $\text{int}(\tilde{p}_i, \tilde{n})$, returned by protocol 1.

Consequence: Partition $\text{int}(q,n)$ into pairwise distinct intervals $\text{int}(\tilde{q}_i, \tilde{n})$, where we choose $\tilde{n}$ as $\tilde{n} := \lfloor n + 3 + \log(1+L) \rfloor$. Same problem as with most of the other protocols: Choose interval $\text{int}(p,m)$ so that (a) $p \in \mathbb{D}_m$, (b) $[\min_i \tilde{p}_i - 2^{-\tilde{m}_i}, \max_i \tilde{p}_i + 2^{-\tilde{m}_i}] \subseteq \text{int}(p,m)$, and (c) $\text{int}(p,m)$ satisfies condition (b) of Definition 3.5.1. Combining Equation (3.7) with the choice of $\tilde{n}$ bounds $\underline{\tilde{\delta}}_i$ and $\overline{\tilde{\delta}}_i$ (the worst-case distances between a midpoint $\tilde{p}_i$ and both endpoints of the exact y-interval $f(\text{int}(\tilde{q}_i, \tilde{n}))$) from above by

$$\max\{\underline{\tilde{\delta}}_i, \overline{\tilde{\delta}}_i\} \overset{(3.7)}{\leq} 2^{-\tilde{m}_i} < 4 \cdot L \cdot 2^{-\tilde{n}} \leq 4 \cdot L \cdot 2^{-(n+3+\log(1+L))} < 2^{-(n+1)} \tag{3.8}$$

for all i. It is easy for an algorithm to choose p and m based on some interval $\tilde{J} := [\min_i \tilde{p}_i - 2^{-\tilde{m}_i}, \max_i \tilde{p}_i + 2^{-\tilde{m}_i}]$ so that (a) $\text{len}(\text{int}(p,m)) \leq 2 \cdot \text{len}(\tilde{J})$ and (b) $J \subseteq \text{int}(p,m)$.

As mentioned in (a), when choosing $\mathrm{int}(p, m)$, the size of this interval at most doubles compared to $\tilde{J}$. Thus, Equation (3.8) implies

$$\max\{\underline{\delta}, \overline{\delta}\} \leq 2 \cdot \mathrm{len}(\tilde{J}) < 2^{-n} < (1 + L) \cdot 2^{-n}$$

for all Lipschitz constants $L > 0$. Hence, it takes $\mathcal{O}(1 + L)$ many queries to protocol L to simulate one protocol 1 query. □

Computational Complexity

By setting $\overline{m}(n) = n + \log(L)$, the modulus of continuity is nothing less than the generalization of f's Lipschitz constant. As we know from Theorem 2.3.5, computability of f also implies computability of $\overline{m}(\cdot)$. Consequently, by definition of computable functions, protocol L also is computable for all L-Lipschitz continuous computable functions when $\overline{m}(n)$ is set as proposed above.

3.6 Protocol X

Roughly speaking, when given a continuous function $f : \mathbb{R} \to \mathbb{R}$, along with the center of a (yet to be determined) x-interval I and the length of a y-interval J, protocol X returns the position of J and a not-too-small interval I so that f's image on I is contained in J, i. e., $f(I) \subseteq J$. Figure 3.12 depicts this situation.

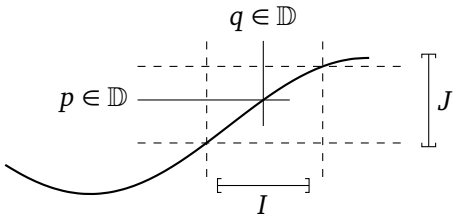


Figure 3.12: Schematic of Protocol X

Analogous to Protocol Y, the requirement on I being "not-too-small" is defined by comparison with the most-suitable interval $I^{\mathrm{suit}} = \mathrm{int}(q^{\mathrm{suit}}, n^{\mathrm{suit}})$: We say, interval I is "not-too-small" if and only if (a) $I \subseteq I^{\mathrm{suit}}$ and (b) $\mathrm{pcs}(I) \leq \mathrm{pcs}(I^{\mathrm{suit}}) + 1$. However, when adapting the definition of the most-suitable interval from protocol Y, it essentially collapses to the maximum-

sized rational interval centered at $q \in \mathbb{D}_n$ whose image under f is contained in J, i. e., there is a unique precision $n_I \in \mathbb{Z}$ to that $f(\text{int}(q, n_I)) \subseteq J$.

Definition 3.6.1 (protocol X)**.** Let $f : \mathbb{R} \to \mathbb{R}$ be a continuous function. Given a dyadic rational $q \in \mathbb{D}_n$ and an integer $m_J \in \mathbb{Z}$, protocol X returns a dyadic rational $p \in \mathbb{D}_{m_J}$ and an integer $n_I \in \mathbb{Z}$. Let $I := \text{int}(q, n_I)$ and $J := \text{int}(p, m_J)$ be those two intervals described by above's parameters. Then (a) $f(I) \subseteq J$, and (b) interval I has to be "not-too-small", i. e., $n^{\text{suit}} \leq n_I \leq n^{\text{suit}} + 1$ with $n^{\text{suit}} := \min\{\tilde{n} \in \mathbb{Z} \mid (\exists \tilde{p} \in \mathbb{D}_{m_J})\, f(\text{int}(q, \tilde{n})) \subseteq \text{int}(\tilde{p}, m_J)\}$.

It is crucial for the understanding of protocol X to recognize the tight relation between the midpoint $p \in \mathbb{D}_{m_J}$ and length of I^{suit}. Even if p is exchanged by one of its closest neighbors (i. e., $p \to p \pm 2^{-m_J}$), $\text{len}(I)$ might be significantly smaller (e. g., in case of steep functions) compared to the most suitable x-interval I^{suit}, based on the original p.

Lemma 3.6.2 (protocol X, existance and uniqueness)**.** *For every parameterization as in the definition of protocol X, there is an x-interval I^{suit}, and it is unique.*

Proof. (a) *Existence*. More precisely, requiring $p \in \mathbb{D}$ to be at least of precision m_J is both sufficient and necessary for I^{suit} to exist. Set

$$f^{-1}(\text{int}(p, m_J)) := \bigcup_{\substack{[a,b] \subseteq [0,X] \\ f([a,b]) \subseteq \text{int}(p, m_J)}} [a, b]$$

for some $p \in \mathbb{D}$. So the statement can be rephrased as the search for a precision $n_I \in \mathbb{Z}$ so that $\text{int}(q, n_I) \subseteq f^{-1}(\text{int}(p, m_J))$. If p is chosen to be at least in $\mathbb{D}_{m_J}$, then the union of all intervals $\bigcup_{p \in \mathbb{D}_{m_J}} \text{int}(p, m_j)$ entirely covers the y-axis. Furthermore,

$$\text{len}(\text{int}(p, m_J) \cap \text{int}(p \pm 2^{-m_J}, m_J)) = 2^{-m_J} \tag{3.9}$$

is no degenerate interval.[7] Thus, in combination with f being a continuous function, there is a precision n_I so that $\text{int}(q, n_I) \subseteq f^{-1}(\text{int}(p, m_J))$.

As for the necessity of p being of precision m for some $m \geq m_J$, it is easy to check that for any other choice $m < m_J$ and $p \in \mathbb{D}_m$, the intersection in (3.9) becomes degenerate or even empty.[7]

[7] This is an important fact: If the interval in (3.9) were degenerate, i. e., a single real, then the precision n_I would go up to infinity – which is neither allowed, nor representable.

(b) *Uniqueness of I^{suit}.* Decrease n_I (i. e., enlarge x-interval I) then we get $n_I - 1 < \min\{k \in \mathbb{Z} \mid f(\mathrm{int}(q,k)) \subseteq \mathrm{int}(p,m_J)\}$, or in other words: $f(\mathrm{int}(q,n_I)) \subseteq \mathrm{int}(p,m_J) \subsetneq f(\mathrm{int}(q,n_I-1))$. Thus, this n_I is unique since the midpoint q is fixed. □

Simulation Complexity

As we prove below, protocol X is powerful enough to simulate the "easy" protocol 0, but also not too hard itself so there exists a simulation algorithm relative to protocol 1 for X.

Theorem 3.6.3. *Let $f : [0,X] \to \mathbb{R}$ (for an arbitrary, but fixed $X > 0$) be a computable real function.*

(a) The simulation of protocol 0 requires only one query to protocol X.

(b) Assume that f is L-Lipschitz continuous and also has a computable modulus of strong unicity $\underline{m}(\cdot)$. Then the number of queries to protocol 1 in order to simulate a call to protocol X with precision $m_J \in \mathbb{Z}$ and dyadic rational $q \in \mathbb{D}_n$ is bounded by $\mathcal{O}((1+L) \cdot X \cdot 2^{\underline{m}(m_J + \log(1+L))})$.

Proof of Theorem 3.6.3, simulation of protocol 0 by X. Given a continuous function f and inputs $n \in \mathbb{Z}$, $q \in \mathbb{D}_n$, a dyadic rational $p \in \mathbb{D}_n$ has to be determined so that $|f(q) - p| < 2^{-n}$ (coherent with protocol 0). On the other side, protocol X requires as an input the size of a y-interval J of precision m_J, and in return we get the size of an x-interval around q that is contained in J.

Set $m_J := n + k$, $k \in \mathbb{N}$, where the exact value (or range) for k is yet to be determined. Providing protocol X with this information gives us a dyadic rational $p \in \mathbb{D}_{n+k}$. Observe that $|f(q) - p| \leq 2^{-(n+k)}$ because $f(q) \in \mathrm{int}(p, n+k)$ and $\mathrm{rad}(\mathrm{int}(p, n+k)) = 2^{-(n+k)}$.

Remaining problem: For $k > 0$, the midpoint p of J might not be in $\mathbb{D}_n$, so instead of returning p, the simulation algorithm has to choose a close neighbor $\tilde{p} \in \mathbb{D}_n$ with $|p - \tilde{p}| \leq 2^{-(n+1)}$. This sums up to

$$\begin{aligned} |f(q) - \tilde{p}| &\leq |f(q) - p| + |p - \tilde{p}| \leq 2^{-(n+k)} + 2^{-(n+1)} \\ &< 2^{-n} \text{ for } k \geq 2. \end{aligned}$$

To conclude, it requires only one query to protocol X to come up with a dyadic rational $\tilde{p} \in \mathbb{D}_n$ as above that simulates the answer of protocol 0. □

The second conjectured simulation, namely the one of protocol X by protocol 1, takes a little more effort to be proven because we have both to find a dyadic rational on the y-axis and not-too-small interval on the x-axis, whereas any choice of the former affects the latter.

Proof of Theorem 3.6.3, simulation of protocol X by 1. Given $m_J \in \mathbb{Z}$ and $q \in \mathbb{D}_n$, we have to describe a procedure that computes a dyadic rational $p \in \mathbb{D}_{m_J}$ and a precision $n_I \in \mathbb{Z}$ by only using protocol 1 and arithmetical operations such that p and n_I obey the definition of protocol X.

The proof goes as follows: First, determine a bound on the size of the smallest x-interval we have to consider as being one protocol X might return. Second, determine the smallest slope possible for f on $[0,X] \setminus \bigcup_{(a,b)\subset I}(a,b)$ (where I is the x-interval from step 1); this bound depends on f's modulus of strong unicity. Third, partition $[0,X]$ into smaller sub-intervals of length $2^{-\tilde{n}+1}$. Here we have to take care that $\tilde{n}$ is chosen reasonably large, making the error introduced by protocol 1 on y-interval approximations smaller than the minimum slope (as computed in the second step). At last, check for the sequence of increasingly smaller y-interval approximations (returned by protocol 1) for the first y-interval that both "lies on the grid $\mathbb{D}_{m_J}$" and is of size $\leq 2^{-m_J+1}$.

The whole approach is iterative. Thus, let $n^{(0)}$ be the precision of the smallest x-interval around q that covers $[0,X]$; that is (at worst), $n^{(0)} := -\lceil \log(X) \rceil$. For all subsequent iterations we set $n^{(i+1)} := n^{(i)} + 1$ for $i \geq 0$. By convention, let $I^{(i)} := \mathrm{int}(q, n^{(i)})$ with associated approximative y-interval $J^{(i)}$ (obtained from protocol 1), where the exact y-interval is denoted by $J^{(i)}_{\mathrm{exact}}$ (that is, $J^{(i)}_{\mathrm{exact}} = f(I^{(i)})$). More precisely, let $I^{(i)} = \dot{\bigcup}_{j=1,\dots,\ell_i} \mathrm{int}(q^{(i)}_j, \tilde{n})$. Query protocol 1 with every x-interval $\mathrm{int}(q^{(i)}_j, \tilde{n})$ and denote the resulting dyadic rationals by $p^{(i)}_j$. Then $J^{(i)} := [\min_{j=1,\dots,\ell_i} p^{(i)}_j - (1+L)\cdot 2^{-\tilde{n}}, \max_{j=1,\dots,\ell_i} p^{(i)}_j + (1+L)\cdot 2^{-\tilde{n}}]$.

As outlined in the beginning, we start by determining a bound on the smallest x-interval that could possibly obey the definition of protocol X. The *Intermediate Value Theorem* (IVT) in combination with f being L-Lipschitz continuous implies the existence of an index $\bar{\imath} \in \mathbb{N}$ such that $\mathrm{len}(f(\mathrm{int}(q, n^{(\bar{\imath}-k)}))) \geq 2^{-m_J+1}$, but $\mathrm{len}(f(\mathrm{int}(q, n^{(\bar{\imath})}))) < 2^{-m_J+1}$ for every $q \in \mathbb{D}$ and $k = 1, \dots, \bar{\imath}$. It holds $L \cdot 2^{-n^{(\bar{\imath})}+1} < 2^{-m_J+1}$ if and only if $n^{(\bar{\imath})} > m_J + \log(L)$. The iterative construction of $n^{(\cdot)}$ yields $\bar{\imath} > m_J + \lceil \log(1 + L \cdot X) \rceil$, an *upper bound* on $\bar{\imath}$. Consequently, I^{suit} *cannot* be smaller than $\mathrm{int}(q, n^{(\bar{\imath})})$. Moreover, $\bar{\imath}$ is computable.

Now that we have a found a bound for $\bar{\imath}$, we can also bound the minimum slope for f on $\mathscr{X} := [0,X] \setminus \bigcup_{(a,b)\subset I^{(\bar{\imath})}}(a,b) = [0, q - 2^{-n^{(\bar{\imath})}}] \cup [q + 2^{-n^{(\bar{\imath})}}, X]$. We denote this bound by $\underline{M} := c \cdot \max_{i=0,\dots,\bar{\imath}} \underline{m}(n^{(i)})$ for some constant $c > 1$. Having both $\bar{\imath}$ and $\underline{M}$ allows us to

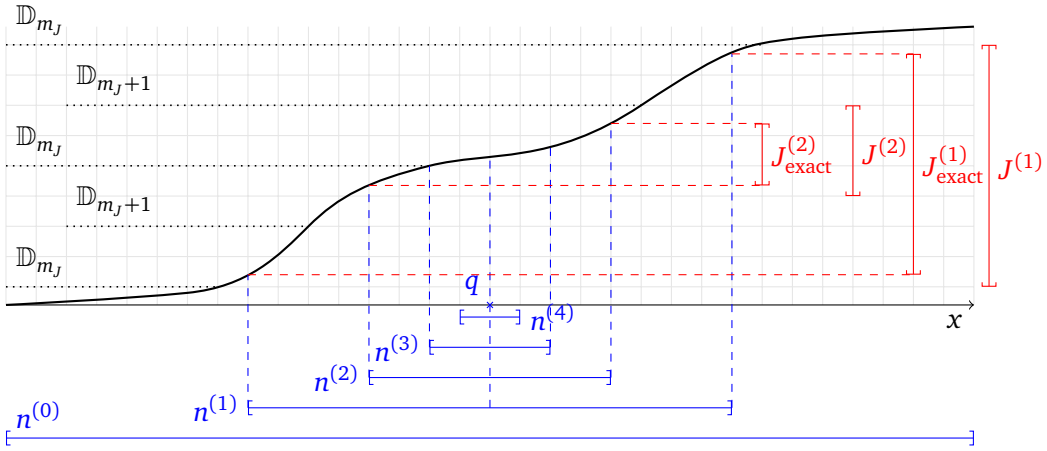


Figure 3.13: Basic obstacle to overcome when simulating protocol X by protocol 1: A small change in the length of the y-interval approximation may result (e. g., for very flat functions) in a significant change in the length of the approximated x-interval I. Thus, we have to provide very tight approximations $J^{(i)}$ for $J^{(i)}_{\text{exact}}$ so that $|\operatorname{len}(J^{(i)}) - \operatorname{len}(J^{(i)}_{\text{exact}})| << 2^{-\underline{m}(\cdot)}$.

define the precision $\tilde{n}$ (which, in turn, specifies in how many sub-intervals we partition $\mathscr{X}$). Set $\tilde{n} > \underline{M} + \lceil \log(1+L) \rceil$, then it satisfies $(1+L) \cdot 2^{-\tilde{n}} < 2^{-\underline{M}}$ (i. e., the error introduced by protocol 1 is neglectable compared to the minimal slope of f on $\mathscr{X}$; this will be of importance later).

The procedure to compute p and n_I has two phases: First, determine a "small-enough" y-interval approximation; second, shrink that approximation so that is "lies on the grid $\mathbb{D}_{m_J}$". In detail, perform the following steps for $i = 1, \ldots, \bar{\iota}$ until the procedure stops.

(a) Query protocol 1 with x-interval $J^{(i)}$ and assume that $J^{(i-1)}$ is already given.

(b) Check if $\operatorname{len}(J^{(i-1)}) > 2^{-m_J+1} \geq \operatorname{len}(J^{(i)})$.[8] If so, go to the step (c); otherwise, go back to step (a).

(c) Search for the smallest $k \geq 0$, and a dyadic rational $p \in \mathbb{D}_{m_J}$ such that $J^{(i+k)} \subseteq \operatorname{int}(p, m_J)$, but $J^{(i+k-1)} \not\subseteq \operatorname{int}(p, m_J)$.

The value found for k satisfies $J^{(i+k-2)}_{\text{exact}} \not\subseteq \operatorname{int}(p, m_J)$. Then, p is the dyadic rational along the y-axis and $\operatorname{int}(q, n^{(i+k)})$ the not-too-small x-interval that obey protocol X. Reasons:

[8] Note that the y-interval approximations are nested because $\overline{J^{(i)}} \leq \overline{J^{(i)}_{\text{exact}}} + (1+L) \cdot 2^{-\tilde{n}} < \overline{J^{(i)}_{\text{exact}}} + 2^{-\underline{M}} < \overline{J^{(i-1)}_{\text{exact}}} < \overline{J^{(i-1)}}$; similar holds for the respective lower endpoints.

- $J_{\text{exact}}^{(i+k-2)} \not\subseteq \text{int}(p, m_J)$ is due to

$$\begin{aligned}\text{len}(J_{\text{exact}}^{(i+k-2)}) &> \text{len}(J_{\text{exact}}^{(i+k-1)}) + 2^{-\underline{M}+1} > \text{len}(J_{\text{exact}}^{(i+k-1)}) + (1+L)\cdot 2^{-\tilde{n}+1} \\ &> \text{len}(J^{(i+k-1)}) > 2^{-m_J+1};\end{aligned}$$

- The first *exact* y-interval possibly contained in $\text{int}(p, m_J)$ is $J_{\text{exact}}^{(i+k-1)}$. This fact uses the former point plus the choice of k as being minimal for the conditions in step (c). Hence, either $I^{(i+k-1)}$ or $I^{(i+k)}$ is the most suitable x-interval. By choosing $n_I := i+k$ we are guaranteed to have captured an x-interval of at least half the size of the most suitable one.
- The dyadic rational p is the one we searched for (i. e., the one protocol X itself would have come up with) because it satisfies $f(I^{(i+k)}) = J_{\text{exact}}^{(i+k)} \subseteq J^{(i+k)} \subseteq \text{int}(p, m_J)$ due to step (c).

This concludes our proof. □

Computability and Computational Complexity

As we know, protocol 1 is (polynomial-time) computable whenever $f : [0,X] \to \mathbb{R}$ is (polynomial-time) computable due to the result of simulation $0 \to 1$. Hence, a similar result holds for protocol X: If $f : [0,X] \to \mathbb{R}$ is (polynomial-time) computable *and* has a computable (polynomially bounded) modulus of strong unicity, then protocol X is (polynomial-time) computable. Whether the computability of $\underline{m}(\cdot)$ is necessary or not for protocol X to be at least computable is, so far, unknown. However, above's proof (regarding the simulation $1 \to X$) gave some good arguments that this might be true: Without the knowledge of $\underline{M}$ it is unclear how to choose $\tilde{n}$ so that the approximation error is still insignificantly small, much smaller than the grid $\mathbb{D}_{m_J}$. Recall that it does not suffice to determine $\underline{M}$ solely based on m_J because for "almost constant" (i. e., very flat) functions, the most suitable interval I^{suit} might vary vastly in length for even seemingly insignificant changes on the approximation error. The following conjecture now formulates our belief.

Conjecture 2. Protocol X is not (polynomial-time) computable unless f has a (polynomially-bounded) computable modulus of both continuity and strong unicity.

3.7 Protocols Coeff and SLP

In algebraic complexity theory, a *straight line program* (SLP) is a directed acyclic graph (DAG) whose inner nodes labeled with the arithmetical operations *addition* and *multiplication*, while the leaves have attached either scalars of some field, or variables (cf. [AB09, Section 6.1, Note 6.4]).

It is clear how to evaluate such an SLP. Moreover, an SLP directly allows us in a natural way to evaluate a polynomial by an in-order tree-walk. Since we are also interested in keeping the length of our representations small (i. e., at most polynomial in the input size; here: precision of $q \in \mathbb{D}_n$, namely $n \in \mathbb{Z}$), an SLP gives us nice features like the existence of polynomial-sized SLPs for some polynomials when actually dealing with exponentially many coefficients. As an example, consider $(((X + c_1)^2 + c_2)^2 + \ldots)^2$ or (even simpler) X^{2^n}.

We get this nice feature of SLPs for the cost of easy accessibility of the encoded polynomial's coefficients. As one implication, we are inable to differentiate the encoded polynomial symbolically, since this would require to construct it first. Given an arbitrary SLP of polynomial size, this might cause the construction algorithm to take exponentially long, as explained above. As we have seen in Section 3.3: In general, providing a good approximation even only to f's first derivative is hard. So gaining back the ability of easy differentiability (i. e., symbolic differentiability) on costs of a large(r) representation is the motivation for protocol Coeff. Informally speaking, it returns dyadic rational approximations to the coefficients of the approximating polynomial.

Now, we first give a brief discussion on both protocol SLP and Coeff, followed by their actual definition. For protocol SLP, demanding that it always returns an SLP of smallest size amongst SLPs being representations of approximation polynomials for f would presumably be too restrictive (cf., e. g., protocol Y for a similar argument). Therefore, the computed SLP is allowed to be larger in its representation by a constant factor, compared to the minimum sized SLP. For protocol Coeff, we do not need such a requirement since a polynomial is uniquely determined by its coefficients (at least for fields of characteristic 0). For the remainder of this section, any norm without subscript denotes the sup-norm over a function space, i. e., $\|f - g\| := \sup_x |f(x) - g(x)|$.

Definition 3.7.1 (protocols Coeff and SLP)**.** Given a continuous function $f : \mathbb{R} \to \mathbb{R}$, aside with an integer $n \in \mathbb{Z}$.

(a) Let $\varphi_n = \sum_{i=0}^{d} a_i X^i \in \mathbb{R}[X]$ (for some $d \in \mathbb{N}_0$ and real coefficients $a_i \in \mathbb{R}$) be a 2^{-n}–approximation of f, that is, $\|f - \varphi_n\| \leq 2^{-n}$. Further, let be given a precision

$k \in \mathbb{Z}$. Then protocol Coeff returns dyadic rational approximations $b_i \in \mathbb{D}$ so that $|a_i - b_i| \leq 2^{-k}$ for all $i = 0, \ldots, d$.[9]

(b) Let Π_n be a (not necessarily minimum-sized) straight-line program for f, i. e., Π_n encodes[10] some real polynomial $\varphi_n \in \mathbb{R}[X]$ so that $\|f - \varphi_n\| \leq 2^{-n}$. Then protocol SLP returns Π_n.

Protocol Coeff is visualized in Figure 3.14. The existence of both protocols is treated in the computability subsection.

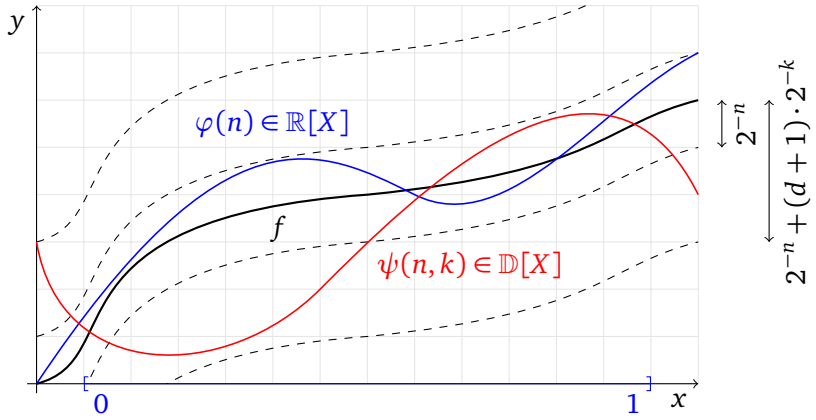


Figure 3.14: Schematic for protocol Coeff. It returns a polynomial $\psi(n,k)$ of some degree d with rational coefficients. The exact coefficients are coming from a polynomial $\varphi(n)$ with *real* coefficients where $\|f - \varphi(n)\| \leq 2^{-n}$.

Simulation Complexity

First we state the obvious.

Lemma 3.7.2. *(a) One query to protocol Coeff suffices to simulate protocol 0.*

(b) Simulating protocol Coeff takes one query to protocol SLP, but at worst exponential time to construct the polynomial's coefficients.

Statement 3.7.2(a) requires any simulation algorithm to evaluate the approximation polynomial φ_{n+1} once in q, whereas both $n \in \mathbb{Z}$ and $q \in \mathbb{D}_n$ are inputs. Let $\tilde{p} = \varphi_{n+1}(q)$ and

[9] The requirement $|a_i - b_i| \leq 2^{-k}$ on each $b_i \in \mathbb{D}$ implies that requiring $b_i \in \mathbb{D}_k$ is not a restriction, but an alternative definition of protocol Coeff.

[10] An SLP is a special kind of a *directed acyclic graph* (DAG), so encoding an SLP with n nodes has space complexity $\mathcal{O}(n^2)$.

choose $p \in \mathbb{D}_n$ so that $|p - \tilde{p}| = d(\tilde{p}, \mathbb{D}_n)$. Then $|p - f(q)| \leq |p - \tilde{p}| + |\tilde{p} - f(q)| = 2^{-(n+1)} + 2^{-(n+1)} = 2^{-n}$.

Statement 3.7.2(b) holds because a straight-line program of polynomial length has (in worst-case) an exponential degree. Thus, computing (i. e., actually writing down) the coefficients of φ_n for protocol Coeff consumes at worst exponential many steps.

Now let us take a look at a presumably much harder problem: Approximating differentiation relative to protocol Coeff. A naive attempt could go as follows. Like in [Ko91, Definition 8.1(a)], protocol Coeff is nothing more than a computable function $\psi : \mathbb{Z} \times \mathbb{Z} \to \mathbb{D}^*$ that on integers n and k returns dyadic rational coefficients $b_0, \ldots, b_d \in \mathbb{D}$ with $|b_i - a_i| \leq 2^{-k}$. Thus, the approximation error is bounded from above by

$$\|f - \psi(n,k)\| \leq 2^{-n} + 2^{-k} \cdot (d+1),$$

assuming $\mathrm{dom}(f) = [0,1]$. To bound the approximation error of f', let $h_n \in \mathbb{R}[X]$ be the exact approximation polynomial (i. e., the interpolation polynomial with real coefficients: $h_n(X) = \sum_{i=0}^{d} a_i \cdot X^i$). Furthermore, denote by $\psi'(n,k)$ and h_n' the symbolic differentiation of $\psi(n,k)$ and h_n, respectively. Then the approximation error of f's first derivative can be bounded by

$$\begin{aligned}\|f' - \psi'(n,k)\| &\leq \|f' - h_n'\| + \|h_n' - \psi'(n,k)\| \leq \|f' - h_n'\| + \sum_{i=1}^{d} i \cdot 2^{-k} \cdot |X^{i-1}| \\ &< \|f' - h_n'\| + (d+1)^2 \cdot 2^{-(k+1)};\end{aligned}$$

sadly, $\|f' - h_n'\|$ is unknown. As we know from Section 3.3, requiring f' to have a polynomially bounded modulus is equivalent to say that f' itself is polynomial-time computable for all polynomial-time computable real functions f. Thus, for instance, requiring f' to be L'-Lipschitz continuous would be too restrictive.

Open Question. Is there a less restrictive property than a polynomially bounded modulus so that differentiation is computable relative to protocol Coeff?

Computability and Computational Complexity

So far, we said nothing about the feasibility of both protocols. For example, given some continuous function $f : \mathbb{R} \to \mathbb{R}$, does a polynomial $\varphi_n \in \mathbb{R}[X]$ exist such that $\|f - \varphi_n\| \leq$

2^{-n}? As a first step, the *Weierstraß Approximation Theorem* (WAT) proves the existence of such an approximative polynomial with *real* coefficients.

The gap between the "classical" (real) version of the WAT and, say, a recursive version of it, will be closed by approximating the coefficients for φ_n itself (as done in the definition of protocol Coeff); that is, we propose the existence of a *computable* function $\psi(n,k)$ that computes dyadic rational approximations for the coefficients of each polynomial φ_n within a prescribed error 2^{-k}. The following definition states that in detail.

Definition 3.7.3 (computable sequence of polynomials)**.** A sequence of real-valued polynomials $\{\varphi_n\}_n$ (commonly represented by $\varphi_n(X) = \sum_{i=0}^{d(n)} a_{n,i}X^i$) is *computable* if there is a computable function ψ such that $\psi(n,k) = (b_{n,1}, \ldots, b_{n,d(n)})$ is a sequence of dyadic rationals $b_{n,i} \in \mathbb{D}$ with $|a_{n,i} - b_{n,i}| \leq 2^{-k}$ for all $n \in \mathbb{Z}$ and $1 \leq i \leq d(n)$.

In [Ko91, Section 8], computable sequences of real-valued polynomials as in Definition 3.7.3 are called *strongly computable*. Consequently, *weak computability* was also introduced, where the difference corresponds to our protocols Coeff and SLP: Where for strong computability of a polynomial sequence we had to find a computable function approximating the coefficients, for weak computability it suffices to come up with a straight-ine program Π_n for each input $n \in \mathbb{Z}$ so that Π_n is a 2^{-n}-approximation to its respective function.

Now having established the foundation for both protocol Coeff and SLP, their well-definedness (i. e., existence) is established by the recursive version of WAT.

Theorem 3.7.4 (recursive version of WAT)**.** *(Pour-El & Caldwell, 1975.) Given a computable real function $f : \mathbb{R} \to \mathbb{R}$, there is a computable sequence of real-valued polynomials $\{\varphi_n\}_n$ such that $\|f - \varphi_n\| \leq 2^{-n}$ for all $n \in \mathbb{Z}$.*

Note that because strong computability implies weak computability, the recursive version of WAT proves the existence of protocol SLP as well.

Having established that a computable sequence $\{\varphi_n\}_n$ always exists for every error bound 2^{-n}, are there also relations to the complexity of f? E. g., does polynomial-time computability of f implies polynomial-time computability of $\{\varphi_n\}$ (and vice versa)? Ko has answered, if you will, both questions with a positive as well as a negative result (cf. [Ko91, Section 8.1]); they summarize as follows: For every exponential-time (polynomial-time) computable function $f : [0,1] \to \mathbb{R}$ there is a strongly exponential-time (weakly polynomial-time) computable sequence of real-valued polynomials $\{\varphi_n\}_n$ such that $\|f - \varphi_n\| \leq 2^{-n}$ for all $n \in \mathbb{N}_0$.

In contrast to those positive results, it also has been proven that the statement when given a polynomial-time computable function (in general) does *not* hold when switching from weakly to strongly polynomial-time computability. Or to say it differently: For functions $f \in \mathrm{PF}$, protocol SLP is always polynomial-time computable, while protocol Coeff in general is *not*.

4 Query Complexity: Approximate Real Function Maximization

So far, we have stated existing results for approximate function maximization and also introduced protocols aimed to provide access to different analytic properties of the functions they encode. In this chapter we now analyze approximate function maximization for real type-2 functions, i. e., for algorithms with access to (certain) function oracles, namely to the formerly introduced protocols. In detail, especially lower and upper bounds on the query complexity relative to various protocols are examined.

This chapter is organized as follows. First, the information (i. e., the choice of protocol queries) is restricted to be non-adaptive. As it turns out in Section 4.1, this restriction corresponds to an absent of certain analytic properties of the input like the Lipschitz constant. We analyze it for both protocol 0 (Section 4.1.1) and protocol 2 (Section 4.1.2). Secondly, information is again allowed to be adaptive in Section 4.2. To exemplify how adaption helps when no analytic information about the input itself is present (like, as mentioned before, the Lipschitz constant) or not even existent, we first examine an adaptive algorithm relative to protocol Y (Section 4.2.1), followed by an example how this algorithm handles a typical adversary function (Section 4.2.2). At last, the example is generalized to functions with a bounded (but unknown to the algorithm) number of maxima in Section 4.2.3.

4.1 Query Complexity w. r. t. Non-Adaptive Information

To recall, information is said to be *non-adaptive* if the same information operations $L_i(\cdot)$ are used throughout all functions $f \in F$ (cf. Section 2.1 for the notation). This means in context of protocol 0 and 2: an algorithm makes a deterministic choice of dyadic rational points q_i (independent of the input function f) in which it queries for approximations of $f(q_i)$ (for protocols 0 and 2) and $f'(q_i)$ (only for protocol 2). As a result of this section we will see that, given non-adaptive information, no algorithm is capable to approximate $\max f$ up to

an error $\varepsilon > 0$ *unless* it is presented with further information (like f's Lipschitz constant) in addition to approximations of $f(q_i)$ (and $f'(q_i)$).

4.1.1 Bounding Real Function Maximization Using Protocol 0

We start by proving the proposed necessity of information in addition to that provided by protocol 0.

Theorem 4.1.1. *No deterministic algorithm relative to protocol 0, presented with a computable real function $f : \mathbb{R} \to \mathbb{R}$, is capable of approximating $\max f|_{[0,X]}$ on any interval $[0,X] \subseteq \mathbb{R}$ $(X > 0)$ up to an error $\varepsilon > 0$ if no further information about f is available.*

Proof. This fact basically relies on the inability of any algorithm to decide whether an approximation for f in some (rational dyadic) point q is close enough to the function's actual maximum (i. e., if an approximation p satisfies $|p - \max f|_{[0,X]}| < \varepsilon$). To bound the error between computed values and f's actual maximum, an algorithm must at least be able to bound the slope of f restricted to some closed interval with dyadic rational endpoints.

For this, let $f \equiv 0$, and construct a piecewise linear function g as follows: For whatever dyadic rational points $q_1, \ldots, q_\ell$ ($q_i < q_{i+1}$, and assume that $q_0 := 0$, $q_{\ell+1} := 1$) an algorithm $\mathscr{A}$ chooses to query protocol 0 with, set

$$g((q_i + q_{i+1})/2) := \varepsilon \text{ for all } i = 0, \ldots, \ell,$$
$$g(q_i) := 0 \text{ for all } i = 0, \ldots, \ell + 1.$$

This construction is illustrated in Figure 4.1. Then, for algorithm $\mathscr{A}$, the functions f and

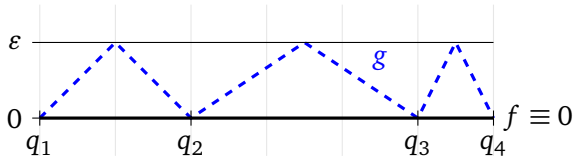


Figure 4.1: Adversary argument: indistinguishable functions f and g, constructed based on the midpoints chosen by algorithm $\mathscr{A}$

g are indistinguishable, but the difference between $g(q_i)$ and $\max g$ is *not* less than ε, i. e., $|g(q_i) - \max g| = \varepsilon$ for all $i = 0, \ldots, \ell + 1$. □

The "technique" used in above's proof is an *adversary argument*; based on whatever choices an algorithm makes, try to construct a function that specifically for those choices is (a) indistinguishable from the original input function, but (b) differs greatly in some major aspects. For our purposes of approximating a function's maximum, the adversary function will often differ in its maximum value from the original input function.

In contrast to this negative result, the problem of function maximization becomes decidable relative to protocol 0 if (as we show hereafter) the set of functions is restricted to Lipschitz continuous functions whereas the Lipschitz constant $L > 0$ is *known* by the algorithm.

Theorem 4.1.2. *Let $L > 0$ be an arbitrary but given Lipschitz constant. There exists an algorithm $\mathscr{A}$ that computes an approximation of $\max f|_{[0,X]}$ on the grid $\mathbb{D}_n$ for any input $n \in \mathbb{N}$, $X > 0$ and L-Lipschitz continuous function $f : [0,X] \to \mathbb{R}$ up to an error $\varepsilon := 2^{-n}$ by using at most $\mathscr{O}((1 + L \cdot X) \cdot 2^n)$ queries to protocol 0. I. e., such an algorithm $\mathscr{A}$ returns a dyadic rational $p \in \mathbb{D}_n$ such that $|\max f|_{[0,X]} - p| < \varepsilon$.*

Note that Theorem 4.1.1 showed that it is crucial for such an algorithm $\mathscr{A}$ to know about f's Lipschitz constant L. Without knowledge of L, algorithm $\mathscr{A}$ in general is unable to determine when it has reached the given error bound.

Proof. Let 2^ℓ (w. l. o. g., $\ell \in \mathbb{N}$) be the maximum number of queries available, and set $m := \lfloor \operatorname{pcs}([0,X]) \rfloor$ (recall that $\operatorname{pcs}([0,X]) = -\log(X) + 1$). We split the interval $[0,X]$ into 2^ℓ equally-sized intervals and take their midpoints, denoted as $q_1, q_2, \ldots, q_{2^\ell} \in D_{m+\ell+1}$.[1] Note that by having pairwise distinct and equally-sized intervals, covering $[0,X]$ entirely, the distance between each pair q_i, q_{i+1} of consecutive midpoints is minimized. This also implies having the smallest upper bound on $|\max f|_{\operatorname{int}(q_i, m+\ell+1)} - f(q_i)|$ for all $i = 1, \ldots, 2^\ell$.

Since we neither know $\max f|_{\operatorname{int}(q_i, m+\ell+1)}$ nor $f(q_i)$, we have to bound them by using protocol 0 and f's Lipschitz continuity. For that, let p_i be a point returned by protocol 0 when queried with q_i and precision $t := \max\{n+1, m+\ell+1\}$, and set $\overline{y}_i := f(q_i) + L \cdot 2^{-(m+\ell+1)}$. Then,

$$|\max f|_{\operatorname{int}(q_i, m+\ell+1)} - p_i| \leq |\overline{y}_i - p_i| \leq L \cdot 2^{-(m+\ell+1)} + 2^{-t}, \tag{4.1}$$

which gives us an easier-to-handle upper bound we can check on being less than ε. Now we are able to prove the proposed upper bound on the number of queries.

[1] Splitting $[0,X]$ into 2^ℓ intervals, as described, causes each interval to be of length $X/2^\ell = 2^{-(-\log(X)+\ell)}$, i. e., the interval's endpoints are of precision $-\log(X)+\ell$. Hence, the midpoints are of precision $-\log(X)+\ell+1$. In order to avoid real-valued precisions, we replace $-\log(X)+1$ by m.

First assume that $L \cdot X \leq 1$. Then ℓ is given as $\log((1+L\cdot X)\cdot 2^n)$, which can be estimated by $-\log(X)+n+1$ by using our assumption (namely $L\cdot X \leq 1$). Now we get our proposed upper bound:

$$\begin{aligned}\overline{y}_i - p_i &< L\cdot 2^{-(\ell+1)} + 2^{-t} \\ &\leq L\cdot 2^{-(m+n+2)} + 2^{-(n+1)} \leq L\cdot X\cdot 2^{-(n+2)} + 2^{-(n+1)} \\ &\leq 2^{-n}.\end{aligned}$$

It remains to prove the upper bound on the number of queries in case of $L\cdot X > 1$. Therefor, we reduce this case to the first one by splitting the interval $[0,X]$ into smaller parts, each of length $\tilde{X}$, such that $L\cdot\tilde{X}\leq 1$. For that, set $\tilde{X} := X/(L\cdot X)$.

The analysis in case 1 now tells us that, because of $L\cdot\tilde{X}\leq 1$, the error in each interval of length $\tilde{X}$ is less than 2^{-n} whenever we are using up to $(1+L\cdot\tilde{X})\cdot 2^n$ queries to protocol 0. Since we have roughly $L\cdot X$ intervals of length $\tilde{X}$, the amount of queries sums up to be $L\cdot X\cdot(1+L\cdot\tilde{X})\cdot 2^n$, which asymptotically is the same as $(1+L\cdot X)\cdot 2^n$ (simply use how we have set $\tilde{X}$). This concludes the proof. □

Note that above's proof implicitly provides an algorithm satisfying Theorem 4.1.2. Moreover, this algorithm can be proven to be optimal, i. e., the proposed number of queries is optimal (modulo constant factors).

Corollary 4.1.3. *The algorithm of Theorem 4.1.2 is* optimal *for all positive integers $n\in\mathbb{N}$, intervals $[0,X]\subseteq\mathbb{R}$ and L-Lipschitz continuous functions $f:\mathbb{R}\to\mathbb{R}$ in the sense that the upper bound $\mathcal{O}((1+L\cdot X)\cdot 2^n)$ for approximating $\max f|_{[0,X]}$ relative to protocol 0 matches the lower bound.*

Proof. Denote by M the number of queries available (as proposed in Theorem 4.1.2). Any algorithm $\mathscr{A}'$ using M queries, that does *not* distribute its dyadic rational points q_i (used to query protocol 0 with) equally along the x-axis, is vulnerable using the formerly introduced adversary technique. Plausibility argument: Since $\mathscr{A}'$ distributes its queries not equally, there are two subsequent queries q_i, q_{i+1} of distance $> 2^{-(n+\log(M))}$. Now construct an adversary function g with $g(q_i)=g(q_{i+1})$, but $g((q_i+q_{i+1})/2) = g(q_i)+1/2\cdot L\cdot|q_i-q_{i+1}|$. Using the minimality of M, we get $|g(q_i)-\max g|\geq 2^{-n}$. Note that the optimality of M follows by an analysis of the proof for Theorem 4.1.2. □

As it turns out, the result from Theorem 4.1.2 extends quite naturally from $\mathbb{R}$ to the d-dimensional space $\mathbb{R}^d$ attached with the maximum norm $\|\cdot\|_\infty$.

Theorem 4.1.4. *Similar to Theorem 4.1.2, there is an algorithm that needs up to $\mathcal{O}((1+L\cdot X)^d\cdot 2^{nd})$ queries to a d-dimensional version of protocol 0 in order to derive an approximation of $\max f|_{[0,X]^d}$ up to an error $\varepsilon := 2^{-n}$.*

Proof. The argument for proving this Theorem is similar to the one already seen back when we proved Theorem 4.1.2, but with midpoints of intervals being replaced by the more general term of midpoints of d-dimensional hypercubes. Figure 4.2 emphasizes the new situation by comparing it with the one-dimensional situation.

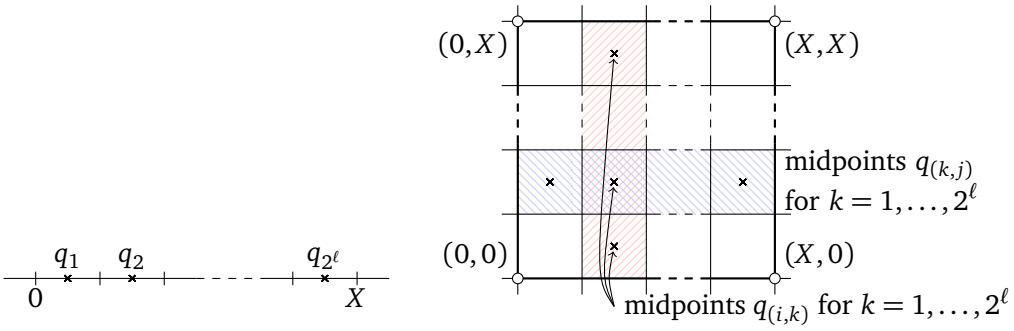


Figure 4.2: Example comparing how we split up the space (and, therefore, where to take the midpoints from to call protocol 0 with) in the original version of protocol 0 (on the left) and in our d-dimensional extension, here for $d=2$ (on the right).

Let $\{\vec{q}_\nu\}_{|\nu|\leq 2^{\ell d}}$ be the set of midpoints of $2^{\ell d}$ equally-sized d-dimensional hypercubes. Note that the indices of our midpoints are written in multi-index notation, with $|\nu| := \nu_1+\cdots+\nu_d$ for every index $\nu=(\nu_1,\dots,\nu_d)\in\mathbb{N}^d$. Then, by choosing ℓ and t as in the proof of Theorem 4.1.2, it holds (similar to statement (4.1)) that

$$\begin{aligned} |f(\vec{q})-p_\nu| &< |f(\vec{q})-f(\vec{q}_\nu)| + 2^{-t} \leq L\cdot\|\vec{q}-\vec{q}_\nu\|_\infty + 2^{-t} \\ &< L\cdot 2^{-(m+\ell+1)} + 2^{-t} \end{aligned} \tag{4.2}$$

for all $\vec{q}\in \mathrm{int}(\vec{q}_\nu, m+\ell+1)$, and p_ν returned by querying protocol 0 with $\vec{q}_\nu$ and precision t. Inequality (4.2) can now be bounded above by 2^{-n} for both $L\cdot X\leq 1$ and $L\cdot X>1$. This Theorem now follows by using that we split $[0,X]^d$ into $2^{\ell d}$ equally-sized hypercubes, and putting in our choice of ℓ. □

4.1.2 Bounding Real Function Maximization Using Protocol 2

So far, we examined both the existence and optimality of algorithms for approximating function maximization relative to protocol 0. Henceforth, we analyze the connection between real function maximization and differentiation. More precisely, an algorithm is allowed oracle access to protocol 2 to approximate $\max f$; this is step one. In the second step we provide the algorithm with further information about its input function f. But first, we state some facts needed for later results.

Lemma 4.1.5. *Some facts concerning differentiability and Lipschitz continuity.*

(a) A differentiable function is L-Lipschitz continuous if and only if its first derivative is bounded by L.

(b) Every continuously differentiable function $f : \mathbb{R} \to \mathbb{R}$ has a bounded first derivative.

(c) Let $f : \mathbb{R} \to \mathbb{R}$ be differentiable, and let f's derivative be L'-Lipschitz continuous. Then, for every pair $x, x' \in \mathbb{R}$ where $x \leq x'$, $f(x')$ is contained in a closed interval with endpoints $f(x) + (x' - x)(f'(x) \mp L'(x' - x))$.

Proof. (a) *Only if.* With f being differentiable and L-Lipschitz continuous on $[0,1]$, it immediately follows that $|\lim_{h\to\infty}(f(x+h) - f(x))/h| \leq L$ for all $x \in (0,1)$. Hence, f' is bounded.

If. Let $L > 0$ be chosen in a way such that $|f'(x)| \leq L$ for all $x \in [0,1]$. Given some points $x, y \in [0,1]$ with $x < y$, the Mean Value Theorem tells us that we can always find a point $\xi \in [x,y]$ satisfying $f(y) - f(x) = f'(\xi)(y - x)$. By taking the absolute value on both sides and using $|f'(\xi)| \leq L$, it follows that f is L-Lipschitz continuous.

(b) Follows by using the Extreme Value Theorem.

(c) With f being continuously differentiable on $[0,1]$, we can use Taylor's Theorem to get a characterization of $f(x')$,

$$f(x') = T_0(x') + R_0(x') = f(x)(x' - x)^0 + \frac{f'(\xi)}{1!}(x' - x), \tag{4.3}$$

where ξ is some point in $[x, x']$. (Note that the remainder term has been taken in Lagrange form.) By using that f' is supposed to be L'-Lipschitz continuous, we can provide lower and upper bounds f' at point ξ:

$$\begin{aligned}|f'(\xi) - f'(x)| \leq |f'(x') - f'(\xi)| + |f'(\xi) - f'(x)| &\leq L'(x' - \xi) + L'(\xi - x)\\ &= L'(x' - x).\end{aligned}$$

Solving above's equation with respect to $f'(\xi)$, we get that

$$f'(\xi) \in [f'(x) - L'(x' - x), f'(x) + L'(x' - x)]. \tag{4.4}$$

Combining Equation (4.3) with (4.4) now proves the statement. □

Similar to the former section (algorithms with non-adaptive information relative to protocol 0) we observe that without additional information about the input function, $\max f$ might not even be computable.

Theorem 4.1.6. *No deterministic algorithm is capable of approximating* $\max f|_{[0,X]}$ *relative to protocol 2 on each interval* $[0,X] \subseteq \mathbb{R}$ *($X > 0$) and for each continuously differentiable function* $f : \mathbb{R} \to \mathbb{R}$ *up to an error* $\varepsilon > 0$ *without having any additional knowledge about the function itself.*

Proof. We prove this theorem by an adversary argument (like we did for Theorem 4.1.1). First, let $q_1, \ldots, q_\ell$ be dyadic rational midpoints of pairwise distinct intervals that entirely cover $[0,X]$. W. l. o. g., assume $q_i < q_{i+1}$ and let $n_i \in \mathbb{N}$ be the smallest integer such that $q_i \in \mathbb{D}_{n_i}$ for all $i = 1, \ldots, \ell$. Now we construct a piecewise linear non-zero adversary function g with $g(b_i \pm 2^{-n_i}) = g'(b_i \pm 2^{-n_i}) = 0$ in breakpoints $b_k := q_k - 2^{-n_k}$, $b_{\ell+1} := 1$ ($i = 1, \ldots, \ell+1$, $k = 1, \ldots, \ell$), and

$$g'(x) := \begin{cases} 0, & x \in \{q_i - 2^{-n_i}, q_i, q_i + 2^{-n_i}\} \\ (-1)^i \cdot L' \cdot 2^{-(n_i+1)}, & x = q_i - 2^{-(n_i+1)} \\ (-1)^{i+1} \cdot L' \cdot 2^{-(n_i+1)}, & x = q_i + 2^{-(n_i+1)} \end{cases}$$

and

$$g(x) := \int_0^x g'(t)dt.$$

Our construction is depicted in Figure 4.3. It implies that $g'|_{[q_i,q_{i+1}]}$ is bounded by $L_i := L' \cdot 2^{-(n_i+1)}$. Note that at this point, L' is still undefined; we will choose it at the end of this proof to complete the adversary argument.

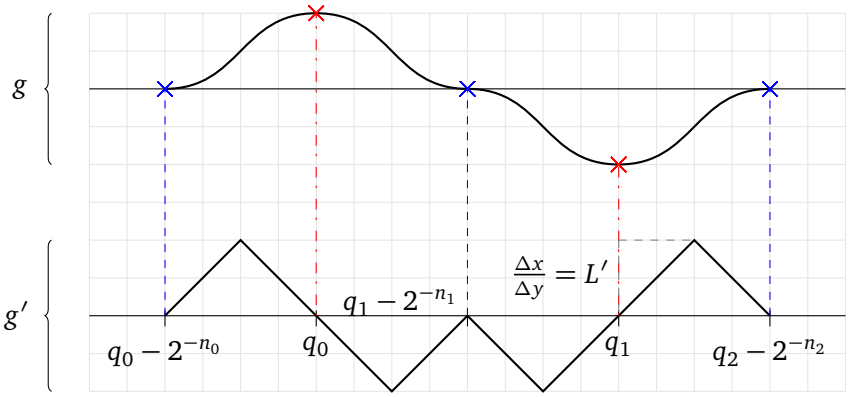


Figure 4.3: Adversary function g, constructed to mislead any approximation algorithm for $\max f|_{[0,X]}$ relative to protocol 2.

Lemma 4.1.5(a) then implies that both g and $(-g)$ are $\max_{1\le i\le \ell}\{L_i\}$-Lipschitz continuous, because

$$g(q_i) = (-1)^i \cdot L_i = (-1)^i \cdot L' \cdot 2^{-(n_i+1)}$$

for all $i = 1,\dots,\ell$.

Recall that it was part of the Theorem's statement that any algorithm trying to approximate g's maximum up to an error $\varepsilon > 0$ has to do it without having any knowledge about L' or L. Finally, by choosing L' appropriately, we can cause g's or $(-g)$'s maximum to be greater than ε. Hence, the approximation of any algorithm would be insufficient (i. e., too large). For that, let $L' \ge \varepsilon \cdot 2^{n_{i_0}+1}$, where $i_0 \in \{1,\dots,\ell\}$ satisfies $n_{i_0} = \max_{1\le i\le \ell}\{n_i\}$. Then the maximum of either g or $-g$ on $[0,X]$ is bounded from below by ε,

$$\begin{aligned}\max\left\{g|_{[0,X]}, (-g)|_{[0,X]}\right\} &= L' \cdot \max_{1\le i\le \ell}\{L_i\} \ge L' \cdot 2^{-(n_{i_0}+1)} \\ &\ge \varepsilon,\end{aligned}$$

but both g and $-g$ are indistinguishable from the zero function, leading an algorithm to an insufficient approximation of either g's or $(-g)$'s maximum. □

Surprisingly enough, this result even holds for *smooth* functions (i. e., functions that are infinitely differentiable).

Corollary 4.1.7. *Theorem 4.1.6 can be extended to functions $f \in C^\infty(\mathbb{R})$.*

However, to this point, we did not succeed to extend this result also to *analytic* functions, but there appears to be a gap in computability (not only in this setting, but in general) between smooth and analytic functions after all.

Definition 4.1.8 (real analytic functions; cf. [KP02])**.** A function $f : V \to \mathbb{R}$, with $V \subseteq \mathbb{R}$ being an open subset, is said to be *real analytic in* $x' \in \mathbb{R}$ if there is an open neighborhood $U \subseteq V$, such that the power series $\sum_{m=0}^{\infty} a_m (x - x')^m$ converges to $f(x)$ for every $x \in U$. Moreover, f is said to be *real analytic* if it is real analytic in every point $x' \in V$.

This suspicion formulates as follows, but should be interpreted as an open question rather than as a firm belief.

Conjecture 3. There *is* a deterministic algorithm relative to protocol 2, receiving a *real analytic* function $f : \mathbb{R} \to \mathbb{R}$ as its input, that is capable of approximating $\max f|_{[0,X]}$ on any interval $[0, X] \subseteq \mathbb{R}$ up to an arbitrary error $\varepsilon > 0$ without further information about the function f.

Proof of Corollary 4.1.7. We construct a *mollifier* to prove this corollary; concrete, use

$$g : [-1, 1] \ni x \mapsto s_y \cdot \exp\left(-\frac{1}{1 - (s_x \cdot x)^2}\right)$$

with s_x and s_y being scaling factors for the x- and y-axis, respectively. This function is C^∞ and Lipschitz continuous with Lipschitz constant $0.75 \cdot s_y < L < 0.8 \cdot s_y$. Set the x-scaling factor as $s_x := 2^n$ whenever g should be restricted to be $g|_{[q-2^{-n}, q+2^{-n}]}$. Then $f'|_{[q-2^{-n}, q+2^{-n}]} := g \circ (x \mapsto x - q)$ for $x \in [q - 2^{-n}, q + 2^{-n}]$. Note that each interval $[q - 2^{-n}, q + 2^{-n}]$ might have its own scaling factors s_x and s_y, dependent on the interval's length.

The function f we have constructed can now be used as in the proof of Theorem 4.1.6 to prove this Corollary. □

Note that real analyticity implies smoothness, thus in particular continuity.

In contrast to the negative results discussed so far, there are positive results, too, stating that $\max f$ is computable relative to protocol 2 with less queries than needed in the respective result where only oracle access to protocol 0 was given.

Theorem 4.1.9. *Let $L' > 0$ be an arbitrary but given Lipschitz constant. There exists an algorithm $\mathscr{A}$ that computes an approximation of $\max f|_{[0,X]}$ on the grid $\mathbb{D}_n$ for any input $n \in \mathbb{N}$, $X > 0$, and $f \in C^1([0,X])$ with f' being L'-Lipschitz continuous up to an error $\varepsilon := 2^{-n}$ by using both $\mathcal{O}((1+L'\cdot X)\cdot 2^{n/2})$ arithmetical operations and queries to protocol 2.*

Proof. We first describe the idea behind the proof, using a typical adversary argument. By using at most $(1+L'\cdot X)\cdot 2^{n/2}$ queries $\mathrm{int}(q_i,n_i)$ to protocol 2 (modulo constant factors), the worst thing that can happen is that when distributing them (the associated dyadic rationals $q_1,q_2,\dots$) equally along $[0,X]$ (like in the proof of Theorem 4.1.2), then f's first derivative will only differ from the zero function[2] within $(q_i-2^{-n_i},q_i)$ and $(q_i,q_i+2^{-n_i})$. (Note that it is essential for the worst-case analysis to let f' be zero in all mid- and endpoints of our chosen intervals $\mathrm{int}(q_i,n_i)$; otherwise we could discard some intervals in the first place by simply comparing how f' behaves in some points.) Figure 4.3 visualizes this intuition.

Set $m := \mathrm{pcs}([0,X])$ and $\tilde{n} := \lfloor m+\log((1+L'\cdot X)\cdot 2^{n/2})\rfloor = m+n+k$ for some $k\in\mathbb{R}$. So $2^{\lceil \tilde{n}-m\rceil}$ is the maximum number of intervals with dyadic rational endpoints we can divide $[0,X]$ into. The Lipschitz continuity of f's first derivative now tells us that $|f'(\tilde{q}_i)-f'(q_i)|\le L'\cdot 2^{-\tilde{n}}$, where $\tilde{q}_i := (q_i+q_{i+1})/2$. Using this result, statement (c) of Lemma 4.1.5 now implies a bound on $f(\tilde{q}_i)$, i. e.,

$$f(\tilde{q}_i)\le f(q_i)+(\tilde{q}_i-q_i)\cdot(f'(q_i)+L'\cdot(\tilde{q}_i-q_i)). \tag{4.5}$$

We conclude:

$$\begin{aligned}
|f(\tilde{q}_i)-p_i| &\overset{(\star)}{<} |f(\tilde{q}_i)-f(q_i)+2^{-(n+1)}| \\
&\overset{(4.5)}{\le} (\tilde{q}_i-q_i)\cdot(f'(q_i)+L'\cdot(\tilde{q}_i-q_i))+2^{-(n+1)} \\
&= L'\cdot 2^{-\tilde{n}+1}+2^{-(n+1)} \\
&\le 2^{-n}.
\end{aligned}$$

Notice that $f'(q_i)=0$ as explained before. It is another technical (but noteworthy) detail that for each point q_i we use a second call to protocol 2, but this time with precision $n+1$ in order to get $p_i\in\mathbb{D}_{n+1}$. This fact is used in $(\star)$.

[2] We use the term "zero function" for the function on the real line mapping every input to zero.

The theorem now follows by distinguishing the cases $L' \cdot X < 1$ and $L' \cdot X \geq 1$, whereas the latter one can be reduced to the first one. □

Above's result and proof raises the question if knowledge about higher derivatives implies a smaller upper bound on the number of queries needed to protocol 2.

Conjecture 4. In contrast to Corollary 4.1.7, the knowledge about both approximations of higher derivatives and the Lipschitz constant of higher derivatives (assuming $f^{(k)}$ actually is Lipschitz continuous) might help do reduce the upper bound result on the number of queries to protocol 2 (cf. Theorem 4.1.9) even further.

Protocol 2 can be extended to functions $f : \mathbb{R}^d \to \mathbb{R}$ just like we did for protocol 0 (recall Definition 3.2.2). Then Theorem 4.1.9 leads to insights we state below.

Corollary 4.1.10.

(a) As for Theorem 4.1.9, the upper bound on the number of queries extends to $\mathcal{O}((1 + L' \cdot X)^d \cdot 2^{dn/2})$ *when*

considering continuously differentiable functions $f : \mathbb{R}^d \to \mathbb{R}$ *with their first derivative being* L'*-Lipschitz continuous.*

(b) The algorithm imposed by the proof of Theorem 4.1.9 has exponential running time when applied to constant functions.

4.2 Query Complexity w. r. t. Adaptive Information

To recall, information is said to be *adaptive* if the information operations $L_i(\cdot)$ depend on the input function $f \in F$ (cf. Section 2.1). Phrased differently, an approximation algorithm interacts at run-time with its oracle and is able to adapt his questions based former received answers. In this section we analyze how adaption helps to approximate $\max f$ if no other information is available than such provided by the attached oracle.

4.2.1 Using Protocol Y: Convergence and Query Complexity

This section is devoted to the question about the minimal number of queries to protocol Y any approximation algorithm requires to compute a 2^{-n}–approximation for an *arbitrary*

L-Lipschitz continuous functions $f : [0,X] \to \mathbb{R}$, and the number of queries evolves asymptotically, i. e., for $n \to \infty$? Both questions will be examined based on Algorithm 4.1.

Algorithm 4.1: Computing a dyadic rational 2^{-n}–approximation of $\max f$

Input: Locally non-constant L-Lipschitz function $f : \mathbb{R} \to \mathbb{R}$, error bound $n \in \mathbb{Z}$

Output: A dyadic rational $p \in \mathbb{D}_n$ such that $\mathrm{int}(p,n)$ contains $\max f$

1 Set int $\leftarrow (q^{(0)}, n^{(0)})$ with $q^{(0)} := 2^{-n^{(0)}}$, $n^{(0)} = -\lceil \log X \rceil + 1$;
2 **for** $i \leftarrow 1$ **to** ∞ **do**
3 Cut every interval in int exactly in half, and let int then again be the resulting set of intervals;
4 For every interval in int, ask protocol Y for the associated range-approximating y-interval;
5 Let J^* be a y-interval having the greatest lower endpoint amongst all y-intervals;
6 Discard every y-interval with its upper endpoint being strictly less than $\underline{J^*}$;
7 **if** *all remaining y-intervals are at least of precision $n+1$* **then**
8 **return** a y-interval $\mathrm{int}(p,n)$, entirely covering all remaining y-intervals;

We introduce some notation necessary for the following discussions. Given the ith iteration of Algorithm 4.1, the midpoint of the ℓth x-interval of precision $n^{(i,\ell)}$ will be denoted by $q^{(i,\ell)} \in \mathbb{D}$. Since the remaining x-intervals will be cut exactly in half in every iteration, the precision $n^{(i,\ell)}$ becomes $n^{(0)} + i$ for all $i \in \mathbb{N}_0$. As an abbreviation (and to ease the notation a little bit) denote the precision of all x-intervals in the ith iteration by pcs_i (here: $\mathrm{pcs}_i = n^{(0)} + i$).

Let y-intervals $\mathrm{int}(p^{(i,\ell)}, m^{(i,\ell)})$ be defined in the same way as done for the x-intervals. Based on this notation, Figure 4.4 depicts the resulting x- and y-intervals after three iterations of Algorithm 4.1 for some fictitious function $f : [0,1] \to \mathbb{R}$.

Proposition 4.2.1. *Given a locally non-constant L-Lipschitz continuous function $f : [0,X] \to \mathbb{R}$, as well as the y-interval J^* of step 5 satisfying $\underline{J^*} = \max_\ell \{\underline{int(p^{(i,\ell)}, m^{(i,\ell)})}\}$. Then, every y-interval $int(p^{(i,\ell)}, m^{(i,\ell)})$ with $\overline{int(p^{(i,\ell)}, m^{(i,\ell)})} < \underline{J^*}$ does not contain $\max f$.*

Fixed the notation, it now becomes easy to see why step 6 of the algorithm is correct (thus also proving above's proposition). Assume that in the ith iteration there are two y-intervals J_1 and J_2 (those are the exact intervals we do not know about), where their dyadic rational approximations returned by protocol Y are denoted by $\mathrm{int}(p_1, m_1)$ and $\mathrm{int}(p_2, m_2)$, respec-

tively, thus they satisfy $J_i \subseteq \operatorname{int}(p_i, m_i)$ for $i = 1, 2$. Assume that $\underline{\operatorname{int}(p_1, m_1)} > \overline{\operatorname{int}(p_2, m_2)}$, then this implies $\max f \geq \underline{J_1} > \overline{J_2}$. Hence, J_2 can be safely discarded.

Sometimes, when the discussion does not depend on a concrete iteration i, we omit the first index in the two-part superscript of intervals, e. g., $p^{(\ell)}$ instead of $p^{(i,\ell)}$.

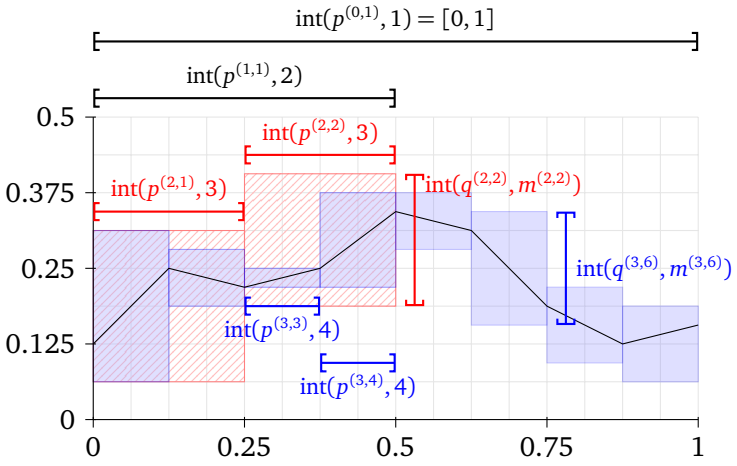


Figure 4.4: Intervals, computed by Algorithm 4.1 after three iterations for some function f

What we are interested in is the number of queries to protocol Y in Algorithm 4.1 given some error bound $n \in \mathbb{Z}$. The following theorem now states that Algorithm 4.1 actually stops after finitely many iterations. This proves in combination with former and also later results the algorithm's correctness.

Theorem 4.2.2. *Algorithm 4.1 converges for every locally non-constant L-Lipschitz continuous function $f : [0, X] \to \mathbb{R}$ and every precision $n \in \mathbb{Z}$, i. e., it stops after finitely many iterations.*

Proof. Let $f \in L\text{–L}[0, X]$ and $n \in \mathbb{Z}$ be given, and assume that Algorithm 4.1 does not converge on this inputs. Then there is a y-interval represented by $(p^{(i,\ell)}, m^{(i,\ell)})$ that will not neither be refined nor discarded in further iterations. W. l. o. g., let us assume that $m^{(i,\ell)}$ does not get worse either[3], i. e., $m^{(i+j,*)} \not> m^{(i,\ell)}$ for all $j \geq 1$, where $m^{(i+j,*)}$ is a precision returned by protocol Y when queried with a sub-interval of $\operatorname{int}(q^{(i,\ell)}, \operatorname{pcs}_i)$. Consequently,

[3] Informally speaking, protocol Y has some freedom to choose the size of the interval $\operatorname{int}(p, m)$ to be returned. Where in the ith iteration interval $\operatorname{int}(p, m)$ might be tight (that is, matching the most suitable interval), the $(i+1)$th iteration might result in an interval which size exceeds the most suitable one's by the (maximal) factor of 2.

$m^{(i,\ell)} = m^{(i+j,*)}$ for all $j \geq 1$. Clearly, there are boundaries (recap that protocol Y provides an interval that is less than four times too large than the exact one)

$$\begin{aligned}\operatorname{len}(f(\operatorname{int}(q^{(i+j,*)}, \operatorname{pcs}_{i+j}))) &\leq \operatorname{len}(\operatorname{int}(p^{(i+j,*)}, m^{(i+j,*)})) \\ &< 4 \cdot \operatorname{len}(f(\operatorname{int}(q^{(i+j,*)}, \operatorname{pcs}_{i+j})))\end{aligned}$$

on the length of $f(\operatorname{int}(q^{(i+j,*)}, \operatorname{pcs}_{i+j}))$, yielding

$$\begin{aligned}1/4 \cdot \operatorname{len}(\operatorname{int}(p^{(i+j,*)}, m^{(i+j,*)})) &= 2^{-(m^{(i,\ell)}+1)} \\ &< \operatorname{len}(f(\operatorname{int}(q^{(i+j,*)}, \operatorname{pcs}_{i+j}))).\end{aligned}$$

Furthermore, $\operatorname{len}(f(\operatorname{int}(q^{(i+j,*)}, \operatorname{pcs}_{i+j}))) \leq L \cdot 2^{-\operatorname{pcs}_{i+j}+1}$ because f is assumed to be Lipschitz continuous with constant $L > 0$. This leads to an immediate contradiction; just choose $j \in \mathbb{N}$ such that $\operatorname{pcs}_{i+j} \geq m^{(i,\ell)} + 2 + \log(L)$. □

The following lemma states additional building blocks, proving in combination the correctness of Algorithm 4.1.

Lemma 4.2.3. *Given a locally non-constant function $f \in L\text{–}\mathrm{L}[0,X]$.*

*(a) There are at most two remaining y-*intervals *of every precision $m \in \mathbb{Z}$ after each iteration of Algorithm 4.1.*

(b) (Existence of the y-interval in step 8) Let there be $\ell \geq 1$ remaining y-intervals after some steps of Algorithm 4.1, represented by $(p^{(1)}, m^{(1)}), \ldots, (p^{(\ell)}, m^{(\ell)})$. Furthermore, let $m^* = \min_{i=1\ldots\ell} m^{(i)}$. Then there is a tuple (p,m) with $m = m^* - 1$ and $p \in \mathbb{D}_m$ such that $\mathit{int}(p,m) \supseteq \bigcup_{i=1}^{\ell} \mathit{int}(p^{(i)}, m^{(i)})$.

Proof. We prove both statements separately.

(a) This follows immediately by the argument already seen in the proof of Lemma 3.4.4: If there were three or more

non-discarded y-intervals, all of precision m with midpoints $p^{(1)}, \ldots, p^{(\ell)}$ where $\ell \geq 3$, then $f(\operatorname{int}(q,n))$ (for some $q \in \mathbb{D}_n$) must be contained in the intersection of all these intervals, i. e.,

$$f(\operatorname{int}(q,n)) \subseteq \bigcap_{i=1}^{\ell} \operatorname{int}(p^{(i)}, m) =: \mathscr{I}.$$

If $\ell > 3$, then $\mathscr{I}$ clearly is empty, where for $\ell = 3$, $\mathscr{I}$ becomes a degenerate interval, yielding f to be constant on $\operatorname{int}(q,n)$; a contradiction since f is supposed to be locally non-constant.

(b) Clearly, $\bigcap_{i=1}^{\ell} \operatorname{int}(p^{(i)}, m^{(i)})$ is neither empty nor a degenerate interval, implying $m^* < \infty$. It is also clear that there are at most two y-intervals with $m^{(i)} = m^{(j)} = m^* > 0$ for $i \neq j$. (The details are stated in case (a).) W. l. o. g., let $p^{(i)} > p^{(j)}$ if there actually are two such intervals. Set $m := m^* - 1$ and $p := p^{(i)} - 2^{-m}$. Then, $\operatorname{int}(p,m) \supseteq \bigcup_{i=1}^{\ell} \operatorname{int}(p^{(i)}, m^{(i)})$. □

Theorem 4.2.4. *Given a locally non-constant L-Lipschitz continuous function $f : [0,1] \to \mathbb{R}$. Then, after finitely many iterations of Algorithm 4.1, at least one interval along the y-axis can be discarded.*

Proof. Given arbitrary $x, y \in [0,1]$ where $x \neq y$, and two x-intervals $\operatorname{int}(q,n), \operatorname{int}(q',n)$ of the same precision with

(a) $x \in \operatorname{int}(q,n)$, $y \in \operatorname{int}(q',n)$ and

(b) $\operatorname{int}(q,n) \cap \operatorname{int}(q',n)$ is either empty or a degenerate interval, i. e., $\operatorname{int}(q,n) \cap \operatorname{int}(q',n) \in \{\emptyset, \{\zeta\}\}$ for some $\zeta \in \mathbb{R}$.

If $|f(x) - f(y)| =: \delta > 0$ then skip the next part. In case of $|f(x) - f(y)| = \delta = 0$, we set $\gamma_x := \min\{|x - \underline{\operatorname{int}(q,n)}|, |x - \overline{\operatorname{int}(q,n)}|\}$ and $\gamma_y := \min\{|y - \underline{\operatorname{int}(q',n)}|, |y - \overline{\operatorname{int}(q',n)}|\}$. Observe that because of $x \neq y$, at most one of both variables can lie in the intersection of $\operatorname{int}(q,n)$ and $\operatorname{int}(q',n)$ (only if it is a degenerate interval, of course). We get three cases:

(a) $\delta = 0$, but $\gamma_* > 0$ for both variables: Remember that f is assumed to be locally non-constant, which can be expressed as

$$(\forall a \in [0,X])(\forall \varepsilon > 0)(\exists b \in B_\varepsilon(a) \cap [0,X]) \text{ such that } |f(a) - f(b)| > 0. \tag{4.6}$$

Choosing $\varepsilon = \gamma_y$ yields a new variable y' with $y' \in \operatorname{int}(q',n)$ and $|f(x) - f(y')| > 0$. For the sake of simplicity, denote the new variable y' by y again.

(b) $\delta = 0$ and $\gamma_* = 0$ for exactly one variable (W. l. o. g., let y be the one with $\gamma_y = 0$): Simply apply the argument of the first case to x.

(c) $\delta = 0$ and $\gamma_* = 0$ for *both* variables x and y: Since both variables match an endpoint of their associated intervals, but at most one of them can lie in the intersection, the distance between x and y is at least 2^{-n+1}, i. e., the length of at least one interval of

precision n. Hence, by choosing $\varepsilon = 2^{-n+1}$ and applying statement (4.6) to variable y we get a new variable y'. This new variable y' is either contained in $\text{int}(q', n)$ or in a direct neighbor whose intersection with $\text{int}(q, n)$ also is empty or a degenerate interval. W. l. o. g., let $\text{int}(q', n)$ again denote this interval containing y'.

Now we have $|f(x) - f(y)| = \delta > 0$. W. l. o. g., let $f(x) > f(y)$. Because of δ being greater than zero we can find a natural number $N \in \mathbb{N}$ where $2 \cdot 2^{-N} < \delta$.[4] Furthermore, there are midpoints $p, p' \in \mathbb{D}_N$ satisfying

(a) $f(\text{int}(q, n)) \subseteq \text{int}(p, N)$ and $f(\text{int}(q', n)) \subseteq \text{int}(p', N)$ (it might be necessary to shrink intervals $\text{int}(q, n)$ and $\text{int}(q', n)$ a little bit);

(b) $(p - 2^{-N}) - (p' + 2^{-N}) = \underline{\text{int}(p, N)} - \underline{\text{int}(p', N)} = \delta - 2^{-N+1} > 0$;

(c) $\text{int}(p, N) \cap \text{int}(p', N) = \emptyset$ as a consequence of point 2.

Hence, finitely many separations along the x-axis yield at least one interval along the y-axis that can be discarded. □

4.2.2 Concrete Query Complexity: Periodic Hat Function

Based on the notions introduced and results stated so far, we will analyze a specific kind of function: the *periodic hat function*. These are interesting functions since we identified them as perfect adversary functions when using non-adaptive information, but more importantly, they work as such for adaptive information as well. In general, a periodic hat function contains k equally sized distinct hats that entirely cover the interval $[0, 1]$. For the sake of simplicity, we restrict our analysis to periodic hat functions having exactly 2^k hats. Denote them by f_k. Figure 4.5 illustrates how such a function f_k evolves.

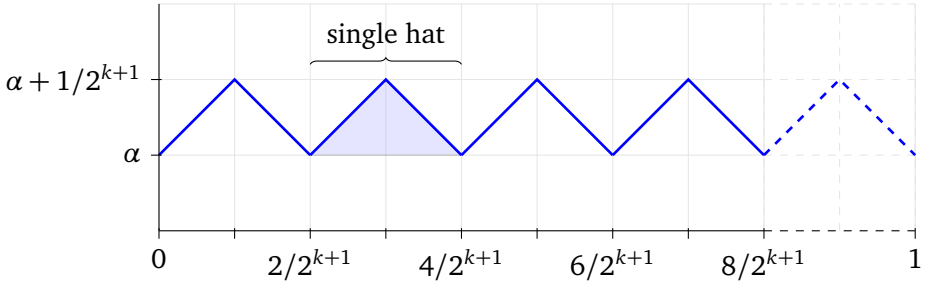


Figure 4.5: Periodic hat function f_k comprised of exactly 2^k hats

[4] This argument only holds for f on $[0, 1]$, but not necessarily on $[0, X]$.

Given an x-interval $\mathrm{int}(q,n)$ and a function $f \in L\text{–}\mathrm{L}[0,X]$, there are only finitely many y-intervals $\mathrm{int}(p,m)$ containing $f(\mathrm{int}(q,n))$ entirely (because the precision m is at least $-(\lceil \log X \rceil + 1)$; see the discussion at the beginning of Section 3.4). Hence, there is a midpoint $p_{\max}$ being the greatest amongst all associated to y-intervals that entirely contain $f(\mathrm{int}(q,n))$, and has less than four times the size of $f(\mathrm{int}(q,n))$. Similarly, define $p_{\min}$ as being the smallest midpoint amongst all such y-intervals.

We will use the existence of such midpoints $p_{\min}$ and $p_{\max}$ to argue about the number of discardable intervals along the y-axis.

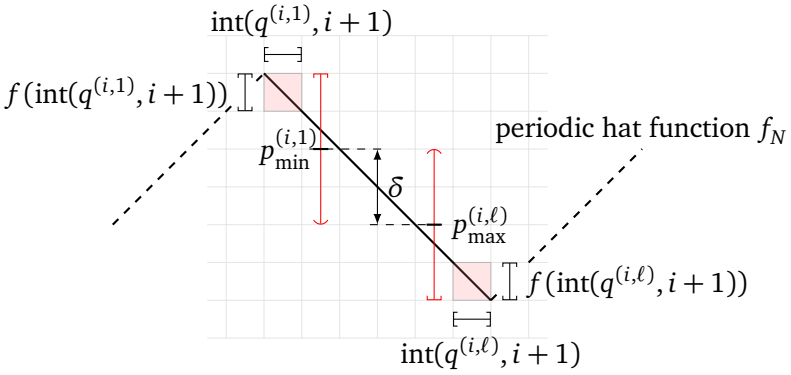


Figure 4.6: Minimal distance δ between two consecutive y-intervals of maximum length

Figure 4.6 gives an example of midpoints $p_{\min}^{(i,1)}$ and $p_{\max}^{(i,\ell)}$. The y-intervals associated to these midpoints are assumed to be of maximal length and to have a maximum overlap with each other (that is, δ gets minimized). Later (in Lemma 4.2.6(a)) we will see that the x-interval $\mathrm{int}(q^{(i,\ell)}, i+1)$ can be safely discarded if $\delta \geq 2^{-(i-2)}$.

Theorem 4.2.5. *Given a precision $n \in \mathbb{Z}$, and the periodic hat function f_k (consisting of 2^k hats). Then Algorithm 4.1 has to query protocol Y at most*

$$\begin{cases} 1, & n \leq 0 \\ 2^{n+1} - 2, & 1 \leq n < k+3 \\ (2^{k+4} - 2) + (2^{k+1} \cdot \ell \cdot 7), & n = k+3+\ell \text{ for } \ell \geq 0. \end{cases}$$

times to compute an 2^{-n}–approximation of $\max f_k$.

Technical detail: Even in case of $n \leq 0$ we must query protocol Y one time to locate $f_k([0,1])$ on the y-axis.

To simplify the proof of Theorem 4.2.5 we scale the right half of a single hat of f_k along the y-axis from $[f(0), f(0)+2^{-(k+1)}]$ to $[0,1]$. (Note that all hats of f_k are equally sized and of length $2^{-(k+1)}$.) Hence, we are actually arguing about the right half of the single hat of f_0. This also allows us to skip the start-up of Algorithm 4.1, i. e., the number of iterations $i' = i + (k+1)$ with $i \geq 1$ reduces to i (because of $k = 0$).

Lemma 4.2.6. *Given a periodic hat function f_k. The following statements pose conditions in which intervals become discardable.*

(a) Let $\mathrm{int}(p^{(i,\ell)}, m^{(i,\ell)})$ be a y-interval after (sufficiently many) iterations $i \in \mathbb{N}$ of Algorithm 4.1, where $\ell \in \{2, \ldots, 2^i\}$. Then, this y-interval can be discarded if $|p_{min}^{(i,1)} - p_{max}^{(i,\ell)}| \geq 2^{-(i-2)}$.

(b) Let $i \geq 3$. Then, $|p_{min}^{(i,1)} - p_{max}^{(i,7)}| < 2^{-(i-2)}$, but $|p_{min}^{(i,1)} - p_{max}^{(i,\ell)}| \geq 2^{-(i-2)}$ for all $\ell = 8, \ldots, 2^i$. I. e., the intervals $\mathrm{int}(q^{(i,\ell)}, i+1)$ for all $\ell = 8, \ldots, 2^i$ can be discarded.

Proof. The first statement merely is a remark, therefore we only prove the second one.

At first we state that $p_{\max}^{(i,8)} > p_{\max}^{(i,\ell)}$ for all $\ell = 9, \ldots, 2^i$. This follows directly with the right half of a single hat being strictly decreasing, and by the definition of $p_{\max}$.

As stated before, we scale a single hat along the y-axis to $[0,1]$. Furthermore, let the right half of such a single hat cover $[0,1]$ entirely. (Note that this is actually no restriction.)

Now, under this assumptions, we can prove Lemma 4.2.6. The first x-interval (namely $\mathrm{int}(q^{(i,1)}, i+1) = [0, 2^{-(i+1)+1}]$) is of precision $i+1$, and the most suitable interval exactly matches $f(\mathrm{int}(q^{(i,1)}, i+1))$. The midpoint $p_{\min}^{(i,1)}$ is of precision $(i+1)-2$, because it is the smallest amongst all midpoints associated to a y-interval both covering $f(\mathrm{int}(q^{(i,1)}, i+1))$ and having at most times the length of the most suitable one (thus, less than four times the length of the exact interval $\mathrm{int}(q^{(i,1)}, i+1)$). Hence, $q_{\min}^{(i,1)}$ can be calculated as

$$p_{\min}^{(i,1)} = 1 - 2^{-(i+1)+2} = (2^i - 2)/2^i.$$

In order to calculate $p_{\max}^{(i,\ell)}$ we use that the lower endpoint of the most suitable interval for $\mathrm{int}(q^{(i,\ell)}, i+1)$ matches $(2^i - \ell)/2^i$. This immediately yields

$$\begin{aligned} q_{\max}^{(i,7)} &= (2^i - 7)/2^i + 2^{-(i+1)+2} = (2^i - 2^3 + 2^1 + 2^0)/2^i \\ \text{and} \quad q_{\max}^{(i,8)} &= (2^i - 8)/2^i + 2^{-(i+1)+2} = (2^i - 2^3 + 2^1)/2^i. \end{aligned}$$

Hence, the distances between $q_{\min}^{(i,1)}$ and $q_{\max}^{(i,\ell)}$ for $\ell \in \{7,8\}$ turn out to be

$$\begin{aligned} |q_{\min}^{(i,1)} - q_{\max}^{(i,7)}| &= 3/2^i < 4/2^i = 1/2^{i-2} \\ \text{and} \quad |q_{\min}^{(i,1)} - q_{\max}^{(i,8)}| &= 4/2^i \geq 4/2^i = 1/2^{i-2}. \end{aligned}$$

Finally, applying the first part of this lemma concludes the proof. □

Findings

Approximating $\max f_k$ for periodic hat functions f_k is linear in the desired precision n (so that $|\max f - p| < 2^{-n}$) if k is kept constant. Otherwise the presented algorithm relative to protocol Y first has an exponential start-up in k before switching over to linear growth in n and k.

4.2.3 Generalization: Query Complexity by Bounded Number of Maxima

Lemma 4.2.7. *It exists a locally non-constant L-Lipschitz continuous computable real function* $f : [0,X] \to \mathbb{R}$ *so that Algorithm 4.1 requires* $\mathcal{O}(1 + L \cdot X \cdot 2^n)$ *queries to protocol Y in order to compute an* 2^{-n}*–approximation of* $\max f$.

Proof. We prove this lemma by constructing a locally non-constant L-Lipschitz continuous computable real function f that evolves closely to $\max f$ (i. e., all function values are contained in $[\max f - 2^{-n}, \max f]$ with $\delta > 0$ but arbitrarily small), and has $M < \infty$ (global) maxima. Figure 4.7 depicts the overall construction which is closely related to the example of periodic hat functions discussed earlier in Section 4.2.2.

Algorithm 4.1 (executed *without* the knowledge about f's Lipschitz constant L) splits the set of x-intervals in half and queries again protocol Y during the search for discardable y-intervals until the union of the remaining y-intervals is of length $< 2^{-n}$, indicating that a 2^{-n}–approximation of $\max f$ is found. This search forms a binary tree of x- and y-intervals. Since the number of nodes (thus, the number of queries to protocol Y) in such a tree is bounded by the number of leaves, we restrict our attention to the latter one. As always, the x-intervals (on the leaf level) are denoted by $\text{int}(\tilde{q}_i, \tilde{n})$ (for a to-be-determined precision $\tilde{n}$), while the respective answers obtained from protocol Y are denoted by $\text{int}(\tilde{p}_i, \tilde{m}_i)$.

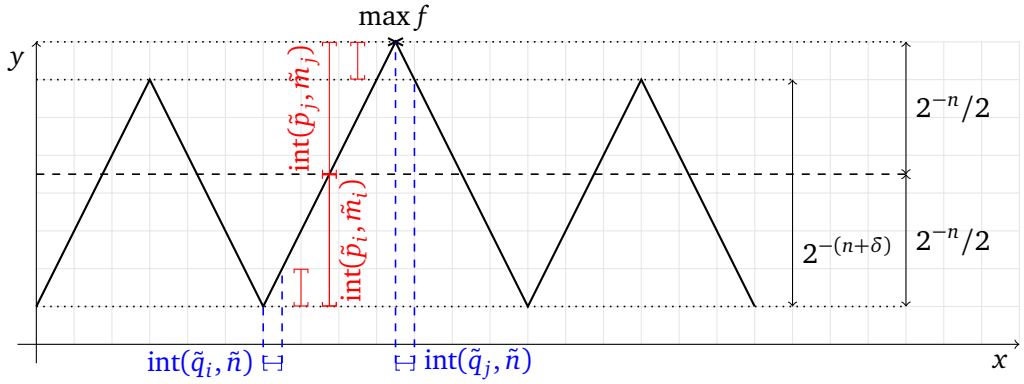


Figure 4.7: At worst, the parameterized complexity of MAX relative to protocol Y is independent of number of maxima $M < \infty$.

The maximum is guaranteed to be reached as soon as $2^{-n}/(8 \cdot L) = 2^{-\tilde{n}}$, resulting in $\tilde{n} \geq n+3+\log(L)$. Thus, $[0,X]$ is partitioned (on leaf level) into $X/2^{-\tilde{n}+1} = L \cdot X \cdot 2^{n+2}$ intervals, resulting in an overall number of queries of $\mathcal{O}(1 + L \cdot X \cdot 2^n)$. □

It is one consequence of Lemma 4.2.7 that the *upper bound* on the number of queries is *independent* of f's number of maxima M. But is the same conclusion correct for the lower bound as well?

Conjecture 5. A lower bound for approximate function maximization relative to protocol Y is *independent* of the number of (global) maxima of a function.

5 Conclusion

In this thesis we used the notion of parameterized complexity to characterize the (simulation and compuational) complexity of real functions and functionals in terms analytic properties. These include the Lipschitz constant, moduli functions $\overline{m}(\cdot)$ and $\underline{m}(\cdot)$ (if existent), and also the number of maxima of a given function. In general, finding those (and other) parameters that influence the parameterized complexity was one of the challenges of our presented approach. In this section we summarize the claims and open issues that surfaced in both the analysis of protocols and the computational complexity of MAX.

(a) *Parameters*. We showed the necessity of, e. g., the function's Lipschitz constant to describe the complexity of MAX more precisely. Nevertheless, are there further parameters influencing its complexity?

(b) *Modulus of strong unicity*. Given a (polynomial-time) computable real function f with compact domain. Then, under what conditions, does a) f has a modulus of strong unicity, and b) when is it computable (and polynomially bounded)? Also there are questions concerning the connection to previously introduced concepts; e. g., how is the modulus of strong unicity related to locally non-constant functions as required for the existence of protocol Y?

(c) *Lower bounds on computational complexity*. While a bounded modulus of continuity is necessary to bound the computation time of protocols Y and X, it is not so clear if the same is true for a modulus of strong unicity. Also the question whether the requirement of a computable modulus of strong unicity is too restrictive demands for an answer.

(d) *Simulation complexity*. (How) Are protocols Coeff and, say, protocol 2 related? As pointed out in the end of Section 3.7, at this point it is unclear if there is a computable function $\nu : \mathbb{Z} \to \mathbb{Z}$ such that for every $n \in \mathbb{Z}$, a $2^{-\nu(n)}$–approximation (obtained from protocol Coeff) suffices to simulate one query to protocol 2 made with input precision

n. As also raised, is there a less restrictive property than a polynomially bounded modulus of continuity so that differentiation is computable relative to protocol Coeff?

(e) *Locally non-constant functions*. Is it a good idea (or even possible) to also apply protocol Y to arbitrary functions f by adding some random (and neglectable) noise g to it such that $f + g$ is locally non-constant?

There is one more concern we like to address separately. As the reader might have noticed, both protocol X and protocol Y impose *no* restriction on precision of their to-be-returned dyadic rationals. So in general, the computation time is *unbounded* in the input precision; not in some artificial and rare cases, but also for extremely simple functions. Just consider functions that become arbitrary flat for protocol Y, and arbitrarily steep for protocol X. Within this thesis we accepted this problem even though it prevents both protocols from being implemented (on, e. g., the iRRAM) unless one restricts himself to only "well-behaved" functions, namely those where there *is* a connection between the input and output precision.

Analysing and answering the problems stated above will not only lead to a more coherent picture, but pave (part of) the way of uniform bounds for real functionals. . . . door opener for numerical analysts.

Bibliography

[AB09] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009. 15, 23, 28, 64

[BCSS98] L. Blum, F. Cucker, M. Shub, and S. Smale. *Complexity and Real Computation*. Springer-Verlag, 1998. 23

[CDS02] C.S. Calude, M.J. Dinneen, and C.K. Shu. Computing a Glimpse of Randomness. *Experimental Mathematics*, 11(3):361–370, 2002. 26

[Fri84] H. Friedman. The computational complexity of maximization and integration. *Advances in Mathematics*, 53(1):80–98, 1984. 9, 30, 32, 33

[Gol91] D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys (CSUR)*, 23(1):5–48, 1991. 8

[KF82] K. Ko and H. Friedman. Computational complexity of real functions. *Theoretical Computer Science*, 20(3):323–352, 1982. 9, 26, 27, 28, 29, 30, 31, 32, 44

[KLRK98] V. Kreinovich, A. Lakeyev, J. Rohn, and P. Kahl. *Computational Complexity and Feasibility of Data Processing and Interval Computations*. Kluwer Academic Publishers, Dordrecht, Netherlands, 1998. 9, 10, 24, 25, 54

[Ko79] K. Ko. *Computational Complexity of Real Functions and Polynomial Time Approximations*. PhD thesis, The Ohio State University, Columbus, 1979. 9

[Ko82] K. Ko. The maximum value problem and NP real numbers. *Journal of Computer and System Sciences*, 24(1):15–35, 1982. 9

[Ko91] K. Ko. *Computational Complexity of Real Functions*. Birkenhäuser, Boston, 1991. 9, 44, 66, 67

[Ko98] K. Ko. Polynomial-time computability in Analysis. *Studies in Logic and the Foundations of Mathematics*, 139:1271–1317, 1998. 9, 15, 30, 31

[KP02] Steven G. Krantz and Harold R. Parks. *A Primer of Real Analytic Functions*. Birkenhäuser, Boston, second edition, 2002. 77

[Lam07] B. Lambov. RealLib: An efficient implementation of exact real arithmetic. *Mathematical Structures in Computer Science*, 17(01):81–98, 2007. 9, 32

[MM05] V. Ménissier-Morain. Arbitrary precision real arithmetic: design and algorithms. *Journal of Logic and Algebraic Programming*, 64(1):13–39, 2005. 8

[Myh71] J. Myhill. A recursive function defined on a compact interval and having a continuous derivative that is not recursive. *Michigan Math. J.*, 18:97–98, 1971. 44

[Mü96] N.Th. Müller. Towards a real Real RAM: a Prototype using C++,(preliminary version). In *Second Workshop on Constructivity and Complexity in Analysis, Forschungsbericht Mathematik-Informatik, Universität Trier 96-44*, volume 44, pages 59–66, 1996. 31

[Mü01] N.Th. Müller. The iRRAM: Exact arithmetic in C++. *Computability and Complexity in Analysis*, pages 222–252, 2001. 9, 10, 31

[Sko08] D. Skordev. $\mathscr{E}^2$-computability of e, π and Other Famous Constants. *Electronic Notes in Theoretical Computer Science*, 202:37–47, 2008. 26

[Tur37] A.M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(1):230, 1937. 23

[TWW83] J.F. Traub, G.W. Wasilkowski, and H. Woźniakowski. *Information, Uncertainty, Complexity*. Addison-Wesley, 1983. 9, 17

[TWW88] J.F. Traub, G.W. Wasilkowski, and H. Woźniakowski. Information-Based Complexity, 1988. 9, 17, 20, 22

[Wei87] K. Weihrauch. *Computability*, volume 9 of *EATCS Monographs on Theoretical Computer Science*. Springer, Berlin, 1987. 23

[Wei00] K. Weihrauch. *Computable Analysis: An Introduction*. Springer Verlag, 2000. 24, 27

[Zie07] M. Ziegler. *Real Computability and Hypercomputation*. Habilitation, University of Paderborn, December 2007. 23

Selbstständigkeitserklärung

Hiermit erkläre ich die vorliegende Masterarbeit selbstständig und nur unter Zuhilfenahme der angegebenen Quellen erstellt zu haben.

Statement of Authorship

I hereby certify that this master's thesis has been composed by me and is based on my own work, unless stated otherwise.

Paderborn, Darmstadt. March 3, 2011. (Carsten Rösnick)

Danksagung

Jeder Versuch die Menschen, denen ich im Folgenden einen Dank widmen möchte, in eine Reihenfolge zu bringen, würde stets einen Beitrag bevorzugen und einen anderen herunterspielen. Daher geht mein tiefster Dank – nun in loser Reihenfolge – an meine Eltern

Ines & Michael Klimczak,

meine Freundin

Julia Neugebauer,

meinen guten Freund

Sebastian Sievert,

und meinen Betreuer

Prof. Dr. Martin Ziegler

für ihre sowohl emotionale Unterstützung (während der Erstellung dieser Arbeit und der Zeit davor), ihren Rat, zeitweilen auch ihre Ablenkung, und nicht zuletzt einfach für ihr Dasein in den wichtigen Momenten.