Invitation to Real Complexity Theory: Algorithmic Foundations to Reliable Numerics with Bit-Costs*

Akitoshi Kawamura (The University of Tokyo) and Martin Ziegler (KAIST)

Abstract. While concepts and tools from Theoretical Computer Science are regularly applied to, and significantly support, software development for discrete problems, Numerical Engineering largely employs recipes and methods whose correctness and efficiency is demonstrated empirically. We advertise *Real Complexity Theory*: a resource-oriented foundation to rigorous computations over continuous universes such as real numbers, vectors, sequences, continuous functions, and Euclidean subsets: in the bit-model by approximation up to given absolute error. It offers sound semantics (e.g. of comparisons/tests), closure under composition, realistic runtime predictions, and proofs of algorithmic optimality by relating to known classes like NP, #P, PSPACE.

Numerical methods permit digital computers, operating over sequences of bits, to solve problems involving *a priori* continuous objects such as real numbers, functions, or operators. Partial differential equations for instance are regularly treated by discretizing the domain, thus approximating the infinite-dimensional solution function space by a high but finite dimensional one. Other common (sub)problems include ordinary differential equations, numerical integration and differentiation, or maximizing some objective function subject to certain constraints. We thus record that numerical science has devised a variety of methods working impressively well in practice in terms of an intuitive conception of efficiency.

Formal notions of algorithmic efficiency, on the other hand, are at the core of Computer Science and have led to the well-established complexity theory with famous classes like P and NP. They capture the computational difficulty inherent to a fully specified problem, rather than the cost of some method solving certain instances of it: thus providing a sound framework for comparing any specific algorithm against an (usually unknown) optimal one — in the discrete realm, that is, for problems over integers or graphs encoded as finite sequences of bits. Concerning real problems, we quote from [Linz88, p.412] and from [BCSS98, §1.4]:

> *How do engineers deal with the problem of assigning some measure of reliability to the numbers that the computer produces? Over the years, I have sat on many Ph.D. qualifying examinations or dissertation defenses for engineering students whose work involved a significant amount of numerical computing. In one form or another, I invariably ask two questions: "Why did you choose that particular algorithm?" and "How do you know that your answers are as accurate as you claim?" The first question is usually answered confidently, using such terms as "second-order convergence" or "von Neumann stability criterion". The next question, alas, tends to be embarrassing. After an initial blank or hostile stare, I usually get an answer like "I tested the method with some simple examples and it worked", "I repeated the computation with several values of n and the results agreed to three decimal places", or more lamely, "the answers looked like what I expected". So far, I have not faulted any student for the unsatisfactory nature of such a response. One reason for my reluctance to criticize is that I have really nothing better to offer.* Rigorous analysis is out of the question.

*Partly supported by the Japan Society for the Promotion of Science (JSPS) *Core-to-Core Program*, by the *JSPS Kakenhi* project 26700001, by the *German Research Foundation* (DFG) project Zi 1009/4-1, and by the *EU FP7 IRSES* project 294962.

The developments described in the previous section (and the next) have given a firm foundation to computer science as a subject in its own right. Use of the Turing machines yields a unifying concept of the algorithm well formalized. [...] The situation in numerical analysis is quite the opposite. Algorithms are primarily a means to solve practical problems. There is not even a formal definition of algorithm in the subject. [...] Thus we view numerical analysis as an eclectic subject with weak foundations; this certainly in no way denies its great achievements through the centuries.

However nowadays we do have a sound, realistic, and applicable theory of computations over continuous universes in the bit-cost model approximations up to guaranteed absolute error 2^{-n}: Initiated by Alan Turing in the very same work [Turi37] that introduced the machine now named after him — and before he 'invented' matrix condition numbers in 1948 — *Recursive Analysis* has developed into a sound algorithmic foundation to verified/reliable/rigorous (not necessarily interval) numerics and to computer-assisted proofs [Rump04,PWNT14] in unbounded precision; cmp. [BLWW04,BrCo06,Brav13]. Traditionally focused on the computational contents of existence claims from classical analysis (see Items a,b,e,g,k,m in the below Example), incomputability results often encode the Halting Problem in a clever way; while positive ones usually devise algorithms and establish their correctness. Starting with [KoFr82], however, the more refined view of complexity has received increasing attention and in the last few years accumulated a striking momentum [Ko91,Weih03,Schr04,KaCo10] with quantitative notions of efficiency and optimality of actual numerical computations! Here are some of its main features:

- Including *transcendental calculations* such as the exponential function: no restriction to algebraic numbers
- *Guaranteed output approximations* suitable (among others) for testing inner-mathematical conjectures
- up to *absolute error* 2^{-n} roughly corresponding to n valid binary digits after the radix point: renders addition of real numbers computable in time linear in n.
- *Adaptive precision* for intermediate calculations beyond the paradigmatic chains of hardware floats.
- *Fully specified algorithms* of guaranteed behavior on explicit classes of admissible inputs.
- *Parameterized analyses* asymptotically for $n \to \infty$
- *quantitatively* with respect to *resources* like running time, memory (often the harder constraint), or #processors/cores
- *Relating to standard complexity classes* from Theoretical Computer Science such as $\mathsf{L} \subseteq \mathsf{NC} \subseteq \mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{P}^{\#\mathsf{P}} \subseteq \mathsf{PSPACE} \subseteq \mathsf{EXP}$ etc.
- Proving *optimality* of an algorithm: for instance using adversary arguments in a bounded-precision adaptation of IBC or relative to complexity-theoretic conjectures like "$\mathsf{P} \neq \mathsf{NP}$".
- Based on *Turing machines* for a formal foundation of numerical calculations in the bit-cost model that yield
- both *practical predictions* of the behaviour of actual implementations
- and *closure under composition,* both of computable and polynomial-time computable functions: a prerequisite to *modular software development* relying crucially on both real number outputs and inputs being given by approximations!
- Actual programming in *imperative object-oriented higher-level languages* such as `C++` using libraries that implement a new data type `REAL` which through overloading support 'exact' operations, thus facilitating

– *Rapid numerical prototyping* with accuracy/stability issues taken care of by the system transparently to the user.
– Modified but *consistent semantics* for tests (multivalued) and (partial) branching: because equality "$x = 0$?" is provably not semi-decidable.
– *Interface declarations* of `C++` functions provided by the theory proving certain enrichment [Zieg12] of the continuous data necessary and sufficient for (say, polynomial-time) computability [ASBZ13].

This flourishing field combines real (and complex) analysis with theoretical computer science:

Definition 1. *a) Computing a real number x means printing some infinite sequence a_n of integers (in binary without leading zeros) as mantissae/numerators to dyadic rationals $a_n/2^n$ that approximate x up to absolute error $1/2^n$.*
b) More generally, a real sequence (x_j) means producing an integer double sequence $a_{j,m}$ with $|x_j - a_{j,m}/2^m| \leq 2^{-m}$. Formally, the elements of said sequence occur in order according to the Cantor pairing function

$$\mathbb{N} \times \mathbb{N} \ni (j, m) \ \mapsto \ \langle j, m \rangle := j + (j + m) \cdot (j + m + 1)/2 \in \mathbb{N} := \{0, 1, 2, \ldots\} \qquad (1)$$

that is, the output consists of the single integer sequence (a_n) with $n = \langle j, m \rangle$.
c) Computing a univariate and possibly partial real function $f :\subseteq \mathbb{R} \to \mathbb{R}$ amounts to converting every given sequence $(a_m) \subseteq \mathbb{Z}$ with $|x - a_m/2^m| \leq 1/2^m$ for any $x \in \mathrm{dom}(f)$ into a sequence $(b_n) \subseteq \mathbb{Z}$ with $|f(x) - b_n/2^n| \leq 1/2^n$.
d) For some mapping $t : \mathbb{N} \to \mathbb{N}_+ := \{1, 2, \ldots\}$, the above computations are said to run in time $t(n)$ if the n-th integer output appears within at most $t(n)$ steps. Polynomial time *(or* polytime *for short) means running time bounded by some polynomial $t \in \mathbb{N}[X]$. Similarly for exponential time and polynomial space.*
e) Generalizing (c) and (d), consider a partial real multivalued function $f :\subseteq \mathbb{R}^d \times \mathbb{N} \rightrightarrows \mathbb{R}^e \times \mathbb{N}$. Computing f amounts to converting, for every $(\vec{x}, k) \in \mathrm{dom}(f)$, given $k \in \mathbb{N}$ and any sequence $(\vec{a}_m) \subseteq \mathbb{Z}^d$ with $\|\vec{x} - \vec{a}_m/2^m\| \leq 1/2^m$, into $\ell \in \mathbb{N}$ and some sequence $(\vec{b}_n) \subseteq \mathbb{Z}^e$ with $\|\vec{y} - \vec{b}_n/2^n\| \leq 1/2^n$ for some $(\vec{y}, \ell) \in f(\vec{x}, k)$.
Such a computation is said to run in fully polynomial time *if $\vec{b}_n$ appears within a number of steps at most polynomial in $n + k$, and ℓ is bounded by some polynomial in k only.*

Parameterized complexity theory as in (e) relaxes Condition (d) requiring the running time to be bounded in terms of the output precision only; see Example 2(f) and (s) below. Fully polynomial-time computable functions are closed under composition.

Example 2 a) Monotone Convergence Theorem: *There exists a computable, monotonically increasing sequence $(x_j) \subseteq [0; 1]$ with incomputable limit $\sup_j x_j$, that is, (x_j) admits no recursively bounded rate of convergence* [Spec49].
b) Bolzano–Weierstraß: *There exists a computable sequence $(x_j) \subseteq [0; 1]$ such that no accumulation point can be computed even relative to the Halting Problem as oracle* [LRZi08, Theorem 3.6].
c) Real-Closed Fields: *Sum, product, and inverse of (polynomial-time) computable reals are again (polynomial-time) computable. For every complex polynomial with computable coefficients, all its real roots are again computable* [Spec69]. *In fact, if the polynomial's coefficients are polynomial-time computable, then so are its real roots* [Scho82].

d) Matrix Diagonalization: *Every real symmetric $d\times d$–matrix M with computable entries admits a computable basis of eigenvectors. However such a basis cannot in general be computed from approximations to M only; whereas restricted to non-degenerate M and, more generally, when providing in addition to approximations to M the integer* $\operatorname{Card}\sigma(M)\in\{1,\ldots,d\}$ *of distinct eigenvalues, it can* [ZiBr04, §3.5] *– and this amount of so-called* discrete enrichment *is optimal* [Zieg12].

e) Power Series: *There exists a computable real sequence $\bar{c}=(c_j)$ whose radius of convergence $R(\bar{c}):=1/\limsup_j|c_j|^{1/j}$ is not computable even relative to the Halting Problem as oracle* [ZhWe01, Theorem 6.2]. *Moreover power series evaluation $x\mapsto\sum_j c_j x^j$ is in general not computable on $(-R,R)$* [Weih00, Example 6.5.1]*; whereas, for every fixed $0<r<R$, it is computable on $[-r;r]$* [Weih00, Theorem 4.3.12].

f) Some Polynomial-Time Computable Functions: *Addition $(x,y)\mapsto x+y$ and multiplication $(x,y)\mapsto x\cdot y$ on the real interval $[-2^k;2^k]$ are computable in time polynomial in $k+n$, where n denotes the output precision in the sense of guaranteed approximation up to absolute error $1/2^n$. The exponential function (family) on $[-2^k;2^k]$ is computable in time polynomial in 2^k+n but not in time depending only on n. Reciprocals $[2^{-k};\infty)\ni x\mapsto 1/x$ are computable in time polynomial in $k+n$. The square root is computable on $[0;2^k]$ in time polynomial in $k+n$. The following explicit function is computable in exponential time, but not in polynomial time — even relative to any oracle:*

$$h_{\exp}:[0;1]\to\mathbb{R},\qquad h_{\exp}(0):=0,\quad 0<x\mapsto h_{\exp}(x):=1/\ln(e/x)\ . \tag{2}$$

g) Maxima: *For every computable $f:[0;1]\to\mathbb{R}$, the parametric maximum function* $\mathrm{MAX}(f):[0;1]\ni x\mapsto\max\{f(t):0\le 1\le x\}$ *is again computable. However there exists a computable smooth f which attains its maximum at (many, but) no computable x* [Spec59].

h) Integration: *For every computable $f:[0;1]\to\mathbb{R}$, the indefinite Riemann integral $\int f:[0;1]\ni x\mapsto\int_0^x f(t)\,dt$ is again computable.*

j) Derivatives: *There exists a computable $f\in C^1[0;1]$ with f' non-computable* [Myhi71]. *Every computable $f\in C^2[0;1]$ has a computable derivative* [Weih00, Corollary 6.4.8].

k) Ordinary Differential Equations: *There exists a computable (and thus continuous) $f:[0;1]\times[-1;1]\to[-1;1]$ such that the initial value problem*

$$\dot{y}(t)\ =\ f\big(t,y(t)\big),\quad y(0)=0 \tag{3}$$

has (many, but) no computable solution $y:[0;1]\to[-1;1]$ [PER79]. *However if f is computable* and Lipschitz*, then the (now unique) solution y is again computable.*

m) Partial Differential Equations: *There exists a computable $f\in C^1(\mathbb{R}^3)$ such that the strong solution $u=u(t,x,y,z)$ to the linear Wave Equation in 3D*

$$\tfrac{\partial^2}{\partial t^2}u\ =\ \tfrac{\partial^2}{\partial x^2}u\ +\ \tfrac{\partial^2}{\partial y^2}u\ +\ \tfrac{\partial^2}{\partial z^2}u,\quad u(0,\cdot)=f,\quad \tfrac{\partial}{\partial t}u(0,\cdot)=0 \tag{4}$$

is incomputable at $(1,0,0,0)$ [PER81]. *Its Sobolev solution however is computable* [WeZh02].

n) Euclidean subsets: *There exist compact subsets $A,B\subseteq\mathbb{R}$ with computable distance function whose intersection $A\cap B\neq\emptyset$ has a non-computable distance function* [Weih00, Exercise 5.1.15]. *There exists a regular compact subset of the plane whose* weak membership oracle *[GLS93] is polynomial-time computable, while its distance function is so if and only if* $\mathsf{P}=\mathsf{NP}$ *holds* [Brav04]. *For any convex compact Euclidean subset on the other hand, the computational complexity of its distance function and its* weak membership oracle *are parameterized polynomially related* [Roes15].

o) Maximum values, again: *Whenever* $f : [0;1] \to \mathbb{R}$ *is polynomial-time computable,* $\mathrm{MAX}(f)$ *is computable in exponential time and polynomial space. In case* $\mathsf{P} = \mathsf{NP}$*,* $\mathrm{MAX}(f)$ *is even polynomial-time computable. Conversely, there exists a polynomial-time computable* $f \in C^\infty[0;1]$ *such that polynomial-time computability of* $\mathrm{MAX}(f)$ *implies* $\mathsf{P} = \mathsf{NP}$ [Ko91, Theorem 3.7]*. For analytic functions* f*, however,* $\mathrm{MAX}(f)$ *is polynomial-time computable whenever* f *is* [Müll87].

p) Integration, again: *Whenever* $f : [0;1] \to \mathbb{R}$ *is polynomial-time computable,* $\int f$ *is computable in exponential time and polynomial space. In case* $\mathsf{FP} = \#\mathsf{P}$*,* $\int f$ *is even computable in polynomial time. Conversely, there exists a polynomial-time computable* $f \in C^\infty[0;1]$ *such that polynomial-time computability of* $\int f$ *implies* $\mathsf{FP} = \#\mathsf{P}$ [Ko91, Theorem 5.33]*. For analytic functions* f*, however,* $\int f$ *is polynomial-time computable whenever* f *is* [Müll87].

q) ODEs, again: *Whenever* $f : [0;1] \times [-1;1] \to [-1;1]$ *is polynomial-time computable and Lipschitz continuous, then the unique solution* y *to Equation (3) is computable in exponential time and polynomial space. In case* $\mathsf{P} = \mathsf{PSPACE}$*,* y *is even computable in polynomial-time. Conversely, there exists a polynomial-time computable* $f \in C^1$ *such that polynomial-time computability of solution* y *implies* $\mathsf{P} = \mathsf{PSPACE}$ [Kawa10,KORZ12]*. If analytic* f *is polynomial-time computable, then so is* y [Müll95].

r) PDEs, again: *Recall the Dirichlet Problem for Poisson's linear partial differential equation:*

$$f \;=\; \Delta u \quad \text{on } \Omega, \qquad u = g \quad \text{on } \partial\Omega \;. \tag{5}$$

On the Euclidean unit ball $\Omega \subseteq \mathbb{R}^d$ *for polynomial-time computable* $f : \Omega \to \mathbb{R}$ *and* $g : \partial\Omega \to \mathbb{R}$*, the strong solution* u *exists and is computable in exponential time and polynomial space. In case* $\mathsf{FP} = \#\mathsf{P}$*, it is even computable in polynomial time. Conversely, there exist polynomial-time computable smooth* f, g *such that polynomial-time computability of* u *implies* $\mathsf{FP} = \#\mathsf{P}$ [KSZ15].

s) Analytic and Gevrey Functions: *For* $\gamma, B, \ell \in \mathbb{N}$ *let* $G^\gamma_{B,\ell}[0;1]$ *denote the class of all infinitely differentiable functions* $f : [0;1] \to \mathbb{R}$ *satisfying* $|f^{(j)}(x)| \;\leq\; B \cdot \ell^j \cdot j^{j\gamma}$*. Then* $\bigcup_{B,\ell} G^1_{B,\ell}[0;1]$ *coincides with the class of real analytic functions. Moreover (i) evaluation, (ii) addition, (iii) multiplication, (iv) differentiation, (v) integration, and (vi) maximization on* $G^\gamma_{B,\ell}[0;1]$ *is uniformly computable in time polynomial in* $(n + \ell + \log B)^\gamma$*; and this is asymptotically best possible* [KMRZ15].

Items (n) to (r) suggest numerical approaches to famous open problems in discrete complexity theory. From a different perspective, they demonstrate exponential-time behaviour of algorithms for several numerical problems to be optimal subject to said conjectures [KaOt14]. Item (s) 'explains' for the phase transition between analytic to smooth functions indicated in Items (o) and (p). It also exhibits parameterized preductions to the actual behaviour of fully-specified algorithms — as opposed to 'methods' or 'recipes' for a vaguely-defined task such as in the following quote from the NAG library's documentation:

> `nag_opt_one_var_deriv(e04bbc)` *normally computes a sequence of* x *values which tend in the limit to a minimum of* $F(x)$ *subject to the given bounds.*

We particularly promote three concepts from logic that have turned out essential in *Real Complexity Theory* with consequences for computational practice:

Nonextensional/multivalued Functions: A (total) relation $f \subseteq X \times Y$ can alternatively be regarded as a partial set-valued mapping $f :\subseteq X \to 2^Y \setminus \{\emptyset\}$, sometimes also written as $f : \mathrm{dom}(f) \subseteq X \rightrightarrows Y$ with $\mathrm{dom}(f) := \{x \in X : \exists y \in Y : (x,y) \in f\}$, via $x \mapsto \{y \in Y : (x,y) \in f\}$. It corresponds to a computational semantics where an algorithm is permitted, given x encoded in one way, to output some $y \in f(x)$ but some possibly different $y' \in f(x)$ when given the same x encoded in another way. Such *non-extensionality* models computational search problems well-known in the discrete case. However in the real setting this effect emerges naturally also in subproblems even when computing only single-valued functions; see [Brat96, §2.3.6], [Weih00, Exercise 5.1.13], or [Luck77].

Discrete Enrichment: Many incomputable (single or multivalued) problems $f : X \rightrightarrows Y$ do become computable when providing, in addition to approximations to the arguments $x \in X$ certain additional information, a concept well-known as *enrichment* in logic [KrMa82, p.238/239]. In many practical cases it suffices to enrich the given x with some suitable integer $k = k(x)$, usually even from a bounded range. For instance according to Example 2(d), only the second line in the following `C++` fragment can lead code correctly returning some eigenvector to a given real symmetric 2×2–matrix:

```
void EV(REAL A11, REAL A12, REAL A22,                    REAL &EVx, REAL &EVy)
void EV(REAL A11, REAL A12, REAL A22, int degenerate, REAL &EVx, REAL &EVy)
```

Parameterized Complexity: Theoretical Computer Science traditionally considers the worst-case resource consumption in dependence on the binary input length parameter n. In the real realm inputs are infinite, and n instead denotes the output precision. However in both disciplines additional parameters $\vec{k}$ allow for a finer-grained analysis and more realistic predictions; recall Definition 1(e) with Example 2(f) and (s). In the discrete case this leads to the field of *Parameterized Complexity*. In the real case, condition numbers are (but) one example of a secondary parameter; the complexity analyses of addition and exponential function from the above examples demonstrate also other natural choices. Discrete enrichment often simultaneously serves as a secondary complexity parameter; cmp. Example 2(s).

To conclude, Real Complexity Theory provides a computer-scientific foundation to Numerics bridging from Recursive Analysis to practice. It asserts in a sound setting that common problems with guaranteed precision are surprisingly hard in the worst case, even restricted to smooth functions. On the other hand recipes and methods do work surprisingly well in practice. This gives raise to one among many questions and challenges for future research:

Challenge 3 *a) Formally capture the class of 'practical' functions that renders standard operations (i) to (vi) from Example 2(s) polynomial-time computable.*

b) Devise, similarly to Example 2(s), a parameterized complexity theory and reliable implementation of ODE solutions.

c) Identify a suitable notion of resource-bounded computation on Sobolev spaces, and characterizes the complexity inherent to the last line of Example 2(m).

d) Develop a sound computability and complexity-theoretic foundation of recent approaches in numerical engineering to shape and topology optimization.

We close with some programming examples evolving around iterating the *Logistic Map*:

$$[0;1] \ni x \mapsto r \cdot x \cdot (1-x) \in [0;1], \qquad 1 < r < 4 \ . \tag{6}$$

It is well-known to exhibit chaotic behaviour for many values of the parameter r beyond 3.56995, with the exception of isolated so-called *islands of stability* for example at $r = 1+\sqrt{8} \approx 3.82843$. For $r := 15/4 = 3.75$ and $x_0 := 1/2$ and $m = 30, 40, 85, 100, 200, 500, 1000, 10\,000$, the reader is encouraged to actually run the code fragments below, and to compare the results they produce: in `Matlab` (left box), `Maple` (middle), and in `C++` (right box) with first line alternatingly replaced by `#define REAL double`, by `#define REAL long double`, and by `#include "iRRAM.h"`. The latter refers to the iRRAM library freely available from `http://www.informatik.uni-trier.de/iRRAM/`

```
function y=logistic(m)
x=1/2; r=vpa(15/4);
for j=1:m x=r*x*(1-x);
end; y=x; end
```

```
logistic:=proc(m)
local j,x,r;
x:=1/2; r:=15/4;
for j from 1 to m
do x:=r*x*(1-x)
end do end proc;
```

```
#define REAL float
REAL logistic(int m) {
REAL x = REAL(1)/REAL(2),
    r = REAL(15)/REAL(4);
while (m--) x = r*x*(1-x);
return(x); }
```

References

ASBZ13. K. Ambos-Spies, U. Brandt, M. Ziegler: "Real Benefit of Promises and Advice", pp.1–11 in *Proc. 9th Conf. on Computability in Europe* (CiE'2013), LNCS vol.**7942**.

BCSS98. L. Blum, F. Cucker, M. Shub, S. Smale: "*Complexity and Real Computation*", Springer (1998).

BGPZ13. O. Bournez, D.S. Graça, A. Pouly, N. Zhong: "Computability and computational complexity of the evolution of nonlinear dynamical systems", pp.12–21 in *Proc. 9th Conf. on Computability in Europe* (CiE 2013), Springer LNCS vol.**7921**.

BLWW04. F. Bornemann, D. Laurie, S. Wagon, J. Waldvogel: "The SIAM 100-Digit Challenge: A Study in High-Accuracy Numerical Computing", SIAM (2004).

Brat96. V. Brattka: "Recursive characterization of computable real-valued functions and relations", pp.45–77 in *Theoretical Computer Science* vol.**162** (1996).

Brav04. M. Braverman: "Computational complexity of Euclidean sets: Hyperbolic julia-sets are poly-time computable", University of Toronto Masters thesis (2004).

Brav13. M. Braverman: "Computing with real numbers, from Archimedes to Turing and beyond", pp.74–83 in *Comm. ACM* vol.**56:9** (2013).

BrCo06. M. Braverman, S.A. Cook: "Computing over the Reals: Foundations for Scientific Computing", pp.318–329 in *Notices of the AMS* vol.**53:3** (2006).

GLS93. M. Grötschel, L. Lovász, A. Schrijver: *Geometric Algorithms and Combinatorial Optimization*, Springer (1993).

vdH*11. J. van der Hoeven, G. Lecerf, B. Mourrain, P. Trbuchet, J. Berthomieu, D. Niang Diatta, A. Mantzaflaris: "Mathemagix, The Quest of Modularity and Efficiency for Symbolic and Certified Numeric Computation", pp.166–188 in *ACM SIGSAM Communications in Computer Algebra* vol.**177:3** (2011).

KaCo10. A. Kawamura, S.A. Cook: "Complexity Theory for Operators in Analysis", pp.495–502 in *Proc. 42nd Ann. ACM Symp. on Theory of Computing* (STOC 2010);
full version in *ACM Transactions in Computation Theory* vol.**4:2** (2012), article 5.

KaOt14. A. Kawamura, H. Ota: "Small Complexity Classes for Computable Analysis", pp. 432–444 in *Proc. 39th Int. Symp. on Math. Found. Computer Science* (MFCS2014), Springer LNCS vol.**8635**.

Kawa10. A. Kawamura: "Lipschitz Continuous Ordinary Differential Equations are Polynomial-Space Complete", pp.305–332 in *Computational Complexity* vol.**19:2** (2010).

Ko91. K.-I. Ko: "*Computational Complexity of Real Functions*", Birkhäuser (1991).

KMRZ15. A. Kawamura, N. Müller, C. Rösnick, M. Ziegler: "Computational Benefit of Smoothness: Parameterized Bit-Complexity of Numerical Operators on Analytic Functions and Gevrey's Hierarchy", to appear in the *Journal of Complexity* (2015); `doi:10.1016/j.jco.2015.05.001`.

KoFr82. K.-I. Ko, H. Friedman: "Computational Complexity of Real Functions", pp.323–352 in *Theoretical Computer Science* vol.**20** (1982).

KORZ12. A. Kawamura, H. Ota, C. Rösnick, M. Ziegler: "Computational Complexity of Smooth Differential Equations", pp.578–589 in *Proc. 37th Int. Symp. on Mathematical Foundations of Computer Science* (MFCS'2012), Springer LNCS vol.**7464**; before presented (in Japanese) at the *10th EATCS/LA Workshop on Theoretical Computer Science* Kyoto, Japan, January 2012.

KrMa82. G. Kreisel, A. Macintyre: "Constructive Logic versus Algebraization I", pp.217-260 in *Proc. L.E.J. Brouwer Centenary Symposium* (Troelstra, van Dalen; Eds.), North-Holland (1982).

KSZ15. A. Kawamura, F. Steinberg, M. Ziegler: "Computationl Complexity of Poisson's Equation", abstract on p.231 in *Bulletin of Symbolic Logic* vol.**20:2** (2014);
full version to appear in *Mathematical Structures in Computer Science* (2015).

Linz88. P. Linz: "Critique of Numerical Analysis", pp.407–416 in the *Bulletin of the American Mathematical Society* vol.**19:2** (1988).

LRZi08. S. Le Roux, M. Ziegler: "Singular Coverings and Non-Uniform Notions of Closed Set Computability", pp.545–560 in *Mathematical Logic Quarterly* vol.**54** (2008).

Luck77. H. Luckhardt: "A Fundamental Effect in Computations on Real Numbers", pp.321–324 in *Theoretical Computer Science* vol.**5** (1977).

Müll87. N.T. Müller: "Uniform Computational Complexity of Taylor Series", pp.435–444 in *Proc. 14th Int Coll. on Automata, Languages, and Programming* (ICALP'87), Springer LNCS vol.**267**.

Müll95. N.T. Müller: "Constructive Aspects of Analytic Functions", pp.105–114 in *Proc. Workshop on Computability and Complexity in Analysis* (CCA), InformatikBerichte FernUniversität Hagen vol.**190** (1995).

Müll01. N.T. Müller: "The `iRRAM`: Exact Arithmetic in `C++`", pp.222–252 in *Proc. 4th Int. Workshop on Computability and Complexity in Analysis* (CCA'00), Springer LNCS vol.**2064** (2001).

Myhi71. J. Myhill: "A recursive function defined on a compact interval and having a continuous derivative that is not recursive", pp.97–98 in *Michigan Math. J.* vol.**18** (1971).

PER79. M.B. Pour-El, J.I. Richards: "A computable ordinary differential equation which possesses no computable solution", pp.61–90 in *Annals of Mathematical Logic* vol.**17** (1979).

PER81. M.B. Pour-El, J.I. Richards: "The wave equation with computable inital data such that its unique solution is not computable", pp.215–239 in *Advances in Math.* vol.**39** (1981).

PWNT14. M. Plum, Y. Watanabe, K. Nagatou, M.T. Nakao: "Verified Computations of Eigenvalue Exclosures for Eigenvalue Problems in Hilbert Spaces", pp.975–992 in *SIAM Journal on Numerical Analysis* vol.**52:2** (2014).

Roes15. C. Rösnick: "Closed Sets and Operators thereon: Representations, Computability and Complexity", to appear in *Logical Methods in Computer Science*.

Rump04. S.M. Rump: "Computer-Assisted Proofs I", pp.2–11 in *Bulletin of the Japan Society for Industrial and Applied Mathematics* vol.**14:3** (2004).

Scho82. A. Schönhage: "The Fundamental Theorem of Algebra in Terms of Computational Complexity", Technical Report, Math. Inst. Univ. Tübingen (1982).

Schr04. M. Schröder: "Spaces Allowing Type-2 Complexity Theory Revisited", pp.443–459 in *Mathematical Logic Quarterly* vol.**50** (2004).

Spec49. E. Specker: "Nicht konstruktiv beweisbare Sätze der Analysis", pp.145–158 in *Journal of Symbolic Logic* vol.**14:3** (1949).

Spec59. E. Specker: "Der Satz vom Maximum in der rekursiven Analysis", pp.254–265 in *Constructivity in Mathematics* (A. Heyting Edt.), Studies in Logic and The Foundations of Mathematics, North-Holland (1959).

Spec69. Specker, E.: "The fundamental theorem of algebra in recursive analysis", pp.321-329 in *Constructive Aspects of the Fundamental Theorem of Algebra*, Wiley-Interscience (1969).

Turi37. Turing, A.M.: "On Computable Numbers, with an Application to the Entscheidungsproblem. A correction", pp.544–546 in *Proc. London Math. Soc.* vol.**43(2)** (1937).

Weih00. K. Weihrauch: "*Computable Analysis*", Springer (2000).

Weih03. K. Weihrauch: "Computational Complexity on Computable Metric Spaces", pp.3–21 in *Mathematical Logic Quarterly* vol.**49:1** (2003).

WeZh02. K. Weihrauch, N. Zhong: "Is Wave Propagation Computable or Can Wave Computers Beat the Turing Machine?", pp.312–332 in *Proc. London Mathematical Society* vol.**85:2** (2002).

ZhWe01. X. Zheng, K. Weihrauch: "The Arithmetical Hierarchy of Real Numbers", pp.51–65 in *Mathematical Logic Quarterly* vol.**47** (2001).

ZiBr04. M. Ziegler, V. Brattka: "Computability in Linear Algebra", pp.187–211 in *Theoretical Computer Science* vol.**326** (2004).

Zieg12. M. Ziegler: "Real Computation with Least Discrete Advice: A Complexity Theory of Nonuniform Computability", pp.1108–1139 in *Annals of Pure and Applied Logic* vol.**163:8** (2012).