A THEORY OF COMPLEXITY, CONDITION, AND ROUNDOFF

FELIPE CUCKER

Department of Mathematics, City University of Hong Kong, Hong Kong;
email: macucker@cityu.edu.hk

Abstract

We develop a theory of complexity for numerical computations that takes into account the condition of the input data and allows for roundoff in the computations. We follow the lines of the theory developed by Blum, Shub and Smale for computations over $\mathbb{R}$ (which in turn followed those of the classical, discrete, complexity theory as laid down by Cook, Karp, and Levin, among others). In particular, we focus on complexity classes of decision problems and, paramount among them, on appropriate versions of the classes P, NP, and EXP of polynomial, nondeterministic polynomial, and exponential time, respectively. We prove some basic relationships between these complexity classes, and provide natural NP-complete problems.

2010 Mathematics Subject Classification: 68Q15 (primary); 68Q17 (secondary)

1. Introduction

1.1. Background. A quarter of a century ago, Blum *et al.* published an article [6] developing a theory of complexity over the real numbers. The two previous decades had witnessed a spectacular development of the foundations of discrete computations, and the declared purpose of [6] was to lay down the grounds for a similar development for numerical computations.

The P = NP question along with everything involved in it (notably, a formal machine model upon which a notion of cost can be defined) revealed a gap between the theoretical foundations of discrete computations in the early 1980s and those of numerical computations. It was this gap (indeed, the desire to fill it) that motivated Blum, Shub, and Smale. Among other results, their paper [6]

defined a formal computational model over the real numbers, associated a natural cost measure to computations in this model, and used this cost measure to define complexity classes $P_\mathbb{R}$, $NP_\mathbb{R}$, and $EXP_\mathbb{R}$ satisfying the following properties (all of them mimicking known properties of their discrete counterparts P, NP, and EXP):

(i) the classes $P_\mathbb{R}$ and $EXP_\mathbb{R}$ are closed by complements (that is, if a problem *S* is in the class, so is the problem obtained by exchanging Yes and No answers);

(ii) $P_\mathbb{R} \subset NP_\mathbb{R} \subset EXP_\mathbb{R}$; and

(iii) the class $NP_\mathbb{R}$ has natural complete problems.

We note here that the inclusion $NP_\mathbb{R} \subset EXP_\mathbb{R}$ was not proved in [6] and that it is certainly nontrivial (see, for example, [33, 45]). It is, in addition, strict (that is, $NP_\mathbb{R} \neq EXP_\mathbb{R}$; see [17]), a separation that in the discrete setting remains conjectural as of today.

The ideas in [6] fused algebraic complexity theory and structural complexity and, simultaneously, built a bridge between the theory of computation and numerical analysis. Its influence after a quarter of century can hardly be overestimated.

The above notwithstanding, Blum *et al.* were aware of at least two aspects left out of their exposition: first the consideration of roundoff errors and their effect on computation, and second, the complexity of iterative methods. Both issues are related to the notion of *condition* (the analyses of both are expressed in terms of a *condition number*) and are the warp and woof of numerical analysis. Actually, the last section of [6] is devoted to open problems, the last of which reads as follows.

> Finally, to bring machines over $\mathbb{R}$ closer to the subject of numerical analysis, it would be useful to incorporate roundoff error, condition numbers and approximate solutions into our development.

Our only agenda here is to pursue this proposal.

1.2. Main results and structure of the exposition.

In this paper, we extend the notion of *decision problem* to include the condition of an input. Besides the *length* of such an input there is a notion of *size* that naturally takes into account its condition. Also, for any finite-precision computation we define a notion of *cost* that accommodates precision requirements.

Endowed with these basic notions, the first goal is to define a version of the class P in the context of finite precision. Our version is the class P_{ro} of problems

decidable with *roundoff polynomial cost*. This is a very general class that captures, we believe, the features and uncertainties of finite-precision computations. The complexity class EXP_{ro} is similarly defined. Both P_{ro} and EXP_{ro} are closed by complements. In other words, property (i) above holds in this setting as well.

Thinking about nondeterminism leads to the issue of which candidate solutions *y* are taken into account for a given problem. In this paper, we consider two possible answers to this question according to whether the magnitude of such a *y* (roughly speaking, the lengths of the exponents in a floating-point representation of the components y_i) is to be polynomially bounded on the size of the problem's data. We thus obtain the classes of *bounded* and *unrestricted* nondeterministic polynomial time NP_{ro}^B and NP_{ro}^U, respectively. For both of these two classes we show that a version of the *circuit feasibility problem* is complete (under a notion of reduction that is appropriate to our finite-precision context). The question to be decided is the same in both cases, but the condition numbers are different.

For these classes we therefore show the following:

(ii′) $P_{ro} \subset NP_{ro}^B \subset NP_{ro}^U$, and $NP_{ro}^B \subset EXP_{ro}$, and

(iii′) the classes NP_{ro}^B and NP_{ro}^U have natural complete problems.

Furthermore, we show that the inclusion $NP_{ro}^B \subset EXP_{ro}$ is strict, just as the inclusion $NP_{\mathbb{R}} \subset EXP_{\mathbb{R}}$ is, but the proof now relies on bounds for precision requirements (which we show in the more general form of a hierarchy theorem). The inclusion $NP_{ro}^U \subset EXP_{ro}$ remains open; we conjecture that it holds true.

The class P_{ro} is, as we said, very general. A glance at the literature shows the existence of two, more down-to-earth, subclasses of P_{ro}, which we will denote by P_{dir} and P_{iter}. In the first (which roughly coincides with the so-called *direct* algorithms) the running time of the algorithm does not depend on the precision needed. An error-free execution provides the right answer. In the second, iteration is of the essence, and machine precision can be adjusted during the execution. A fundamental feature now is that the algorithm's outputs are guaranteed correct. The corresponding classes for exponential cost naturally follow.

A way in which the class P_{ro} becomes 'closer to numerical analysis' (than the class $P_{\mathbb{R}}$ introduced in [6]) is that subsets of $\mathbb{R}^n$ decided in P_{ro} no longer need to have semialgebraic boundaries. An example is the set $\{(x, y) \mid x \in (0, 1)$ and $y \leqslant e^x\}$ which belongs to P_{ro} (when endowed with a natural condition number).

The classes P_{dir} and P_{iter} naturally introduce four further classes for nondeterministic polynomial cost. For two of them, NP_{dir}^U and NP_{dir}^B, we show a completeness result. A diagram of the main classes studied in this paper, with arrows indicating inclusions shown here, follows.
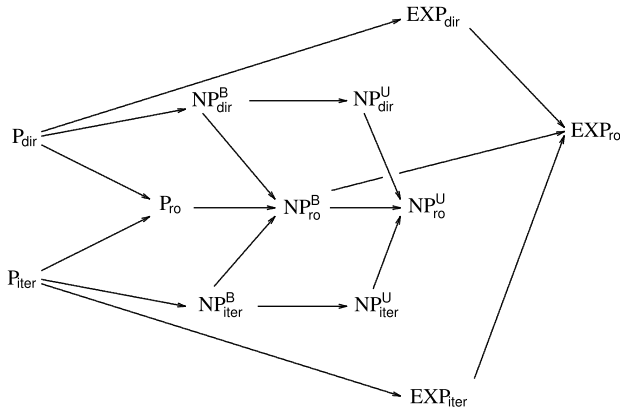
Decision problems, a notion of input size, finite-precision machines, and the cost of their computations, are all introduced in Section 3. The classes P_{ro}, P_{dir}, and P_{iter} are described in Section 4. The classes NP_{ro}^U, NP_{ro}^B, NP_{dir}^U, NP_{dir}^B, NP_{iter}^U, and NP_{iter}^B are examined in Section 5. Different versions of the *circuit feasibility problem* are then shown to be complete in the first four of these nondeterministic classes, under appropriately defined reductions. The inclusion $NP_{ro}^B \subset EXP_{ro}$ is shown in Section 6. Section 7 briefly discusses two issues related to our exposition: a relationship between condition and average complexity which is ubiquitous in the modern approach to condition, and a comparison with a Turing machine-based approach to numerical computations.

Preceding these developments we spend some time overviewing the basic features of finite-precision computations and conditioning. For the latter, and in order to convey a certain *ad hoc* character of condition numbers (see Remark 4 below), we are liberal with examples.

1.3. Previous and related work. To the best of our knowledge, this is the first exposition of a theory of NP-completeness that considers condition and finite precision. Nonetheless, our development was only possible because substantial literature on related ideas was available.

Condition-based accuracy analysis goes back to the work of Turing [59] and von Neumann and Goldstine [60]. This analysis has since been pervasive in numerical linear algebra, and more recently began occurring in other subjects (linear optimization, polynomial computations, and so on).

Then there is the condition-based complexity analysis, which goes back to the analysis of the conjugate gradient method by Hestenes and Stiefel [34], and was more recently championed by Blum [4] and Smale [55, 56]. The work of Renegar on linear optimization [47–49] in the mid 1990s is an outstanding example.

Then there are the theories of complexity that grew in the 1970s for discrete computations (see [30, 36, 43] for text expositions) and in the late 1980s for numerical problems (see [5]), which set the model for the kind of results one should look for in the finite-precision setting.

Finally, there are a few articles that, without attempting to build a theory, delve into the overlap of complexity and accuracy. Notably among them are an old result by Miller [42] establishing a trade-off between complexity and numerical stability for matrix multiplication, and a recent paper by Allender *et al.* [1] that relates a specific discrete problem with various aspects of numerical computations.

We conclude the section by mentioning another stream of research on the complexity of numerical computations. The point of departure now is the adoption of the Turing machine as computational model, together with the procedure of both inputting and outputting real numbers bit by bit. An excellent short account of this viewpoint is in [7]. Comprehensive expositions can be found in [38, 62]. We will briefly describe the differences with our approach in Section 7.2.

2. Finite precision, condition, and stability

This section recounts the main features of finite-precision computations. The idea is not to propose a theory (this will be done in subsequent sections) but to acquaint readers possibly unfamiliar with these features and to motivate ensuing definitions.

2.1. Finite-precision computations.
Numerical computations on a digital computer are supported by a representation of real numbers and their arithmetic. Because of the necessary finiteness of computer data, real numbers are replaced by *approximations*, and the ubiquitous form of these approximations is the so-called *floating-point* numbers. We next briefly describe them, pointing to the reader that a comprehensive exposition of the subject is given in [35, Ch. 2] (from where our short description has been extracted).

A *floating-point number system* $\mathbb{F} \subset \mathbb{R}$ is a set of real numbers y having the form

$$y = \pm m \times \beta^{e-t},$$

where

(1) $\beta \in \mathbb{Z}$, $\beta \geqslant 2$, is the *base* of the system;

(2) $t \in \mathbb{Z}$, $t \geqslant 2$ is its *precision*;

(3) $e \in \mathbb{Z}$ satisfies $\mathsf{e}_{\min} \leqslant e \leqslant \mathsf{e}_{\max}$ (the *exponent range*) with $\mathsf{e}_{\min}, \mathsf{e}_{\max} \in \mathbb{Z}$;

and $m \in \mathbb{Z}$ (the *mantissa*) satisfies $0 \leqslant m \leqslant \beta^t - 1$. We actually impose, to ensure a unique representation of y, that for $y \neq 0$ we have $\beta^{t-1} \leqslant m \leqslant \beta^t - 1$. This implies that

$$y = \pm \beta^e \left(\frac{d_1}{\beta} + \frac{d_2}{\beta^2} + \cdots + \frac{d_t}{\beta^t} \right) = \pm \beta^e \times 0 \cdot d_1 d_2 \ldots d_t \tag{1}$$

with $0 \leqslant d_i \leqslant \beta - 1$ for all i and $d_1 \neq 0$.

The nonzero elements y of $\mathbb{F}$ satisfy

$$\beta^{e_{\min}-1} \leqslant |y| \leqslant \beta^{e_{\max}}(1 - \beta^{-t}). \tag{2}$$

The real numbers in these two intervals, along with $\{0\}$, make the *range* of $\mathbb{F}$, which we denote by $\mathsf{Range}(\mathbb{F})$. That is,

$$\mathsf{Range}(\mathbb{F}) := [-\beta^{e_{\max}}(1 - \beta^{-t}), -\beta^{e_{\min}-1}] \cup \{0\} \cup [\beta^{e_{\min}-1}, \beta^{e_{\max}}(1 - \beta^{-t})].$$

Associated to the system $\mathbb{F}$ there is a *rounding function* $\mathsf{fl} : \mathbb{R} \to \mathbb{F}$ which maps each real x in the range of $\mathbb{F}$ to an element $\mathsf{fl}(x)$ in $\mathbb{F}$ closest to x (there are several ways to break ties whose nature is of no consequence to our development; the interested reader can see the notes and references of [35, Ch. 2]). If x is not in the range of $\mathbb{F}$ then it is either too large in absolute value ($|x| > \beta^{e_{\max}}(1 - \beta^{-t})$) or too small ($0 < |x| < \beta^{e_{\min}-1}$). We talk about *overflow* or *underflow*, respectively. Different implementations of fl treat these cases in different manners, a common one letting $\mathsf{fl}(x)$ map x to the nearest nonzero element in $\mathbb{F}$.

The *unit roundoff* of $\mathbb{F}$ is defined to be $\mathsf{u}_{\mathsf{mach}} := \frac{1}{2}\beta^{1-t}$. 'It is the most useful quantity associated with $\mathbb{F}$ and is ubiquitous in the world of rounding error analysis' [35, page 42]. It satisfies the following:

$$\text{for all } x \in \mathsf{Range}(\mathbb{F}), \ \mathsf{fl}(x) = x(1 + \delta) \text{ for some } \delta \text{ with } |\delta| < \mathsf{u}_{\mathsf{mach}}. \tag{3}$$

Arithmetic in $\mathbb{F}$, in the *standard model*, is performed by first computing on $\mathbb{Q}$ the exact result of an operation and then applying fl to it. This defines, for any operation $\circ \in \{+, -, \times, /\}$, a corresponding operation

$$\tilde{\circ} : \mathbb{F} \times \mathbb{F} \to \mathbb{F}$$

which satisfies, for all $x, y \in \mathbb{F}$,

$$x \circ y \in \mathsf{Range}(\mathbb{F}) \Rightarrow x\tilde{\circ}y = (x \circ y)(1 + \delta) \quad \text{for some } \delta \text{ with } |\delta| < \mathsf{u}_{\mathsf{mach}}. \tag{4}$$

REMARK 1. There is a qualitative difference between the bounds required for mantissas ($t < \infty$) and exponents ($|e_{\min}|, e_{\max} < \infty$) in a floating-point number

system. For all real numbers $x > 0$ the exponent e needed to represent x in the interval $(\beta^{t-1}, \beta^t - 1)\beta^{e-t}$ is finite. In contrast with this, for almost all real numbers x one must have $t = \infty$ if one wants that $\mathsf{fl}(x) = x$. This feature, together with the fact that overflow and underflow are rare when compared with the all-pervasive presence of rounding errors, is at the origin of the fact that, almost invariably, theoretical analyses of roundoff assume a floating-point system without bounds for the exponents, that is, a system where (3) and (4) hold true without requiring $x \in \mathsf{Range}(\mathbb{F})$ (or, equivalently, where $\mathsf{Range}(\mathbb{F}) = \mathbb{R}$). We will refer to such a system as having *unrestricted exponents*. Most accuracy analyses in the literature (for example, all those cited in this paper) assume unrestricted exponents.

2.2. A helpful equality.

The sequencing of arithmetic operations in the execution of an algorithm entails the accumulation of errors given by equation (4), and, with it, the occurrence of products of quantities of the form $(1 + \delta)$ with $|\delta| \leqslant \mathsf{u}_{\mathsf{mach}}$. The following result (see [35, Lemma 3.1]) deals with these products.

LEMMA 1. *If $|\delta_i| \leqslant \mathsf{u}_{\mathsf{mach}}$ and $\rho_i = \pm 1$ for $i = 1, \ldots, n$, and $n\mathsf{u}_{\mathsf{mach}} < 1$, then*

$$\prod_{i=1}^{n}(1 + \delta_i)^{\rho_i} = 1 + \theta_n,$$

where θ_n is a real number satisfying

$$|\theta_n| \leqslant \frac{nu}{1 - nu} =: \gamma_n. \qquad \square$$

2.3. Stability and condition.

The possible effect of roundoff errors on a computation caught the attention of the founding figures of modern numerical analysis. Both Turing in the UK and von Neumann and Goldstine in the USA considered this effect for the case of linear equation solving and attempted a quantitative explanation [59, 60]. The story of the former is nicely described by Wilkinson in his 1970's Turing Lecture [63]. What follows is a brief exposition of the ideas introduced by them and their subsequent extensions.

Errors in a finite-precision algorithm $\mathcal{A}$ computing a function $\varphi : U \subset \mathbb{R}^n \to \mathbb{R}^m$ will accumulate, and $\mathcal{A}$ will return, on input $x \in \mathbb{R}^n$, a point $\varphi^{\mathcal{A}}(x)$ different from $\varphi(x)$. The extent of this difference can be measured by the *normwise relative error*

$$\mathsf{RelError}(\varphi^{\mathcal{A}}(x)) := \frac{\|\varphi^{\mathcal{A}}(x) - \varphi(x)\|}{\|\varphi(x)\|}. \tag{5}$$

An approach that has often worked to estimate this error passes through showing that

$$\varphi^{\mathcal{A}}(x) = \varphi(\tilde{x}) \tag{6}$$

for a point $\tilde{x} \in \mathbb{R}^n$ sufficiently close to x, say, satisfying that

$$\mathsf{RelError}(\tilde{x}) := \frac{\|\tilde{x} - x\|}{\|x\|} \leqslant \mathsf{u}_{\mathsf{mach}} g(n, m), \tag{7}$$

where $g(n, m)$ grows slowly with n and m (for example, as a low-degree polynomial). If such a result—known as *backward-error analysis*—is possible, the relative error in (5) can be estimated from the knowledge of the *(relative, normwise) condition number* of x,

$$\mathsf{cond}^{\varphi}(x) := \lim_{\delta \to 0} \sup_{\mathsf{RelError}(\tilde{x}) \leqslant \delta} \frac{\mathsf{RelError}(\varphi(\tilde{x}))}{\delta}. \tag{8}$$

Equation (8) shows the usual understanding of a condition number as 'the worst possible magnification of the error, in the value of $\varphi(x)$, produced by a small perturbation of the data x.' It follows from (5)–(8) that

$$\mathsf{RelError}(\varphi^{\mathcal{A}}(x)) \leqslant \mathsf{u}_{\mathsf{mach}} g(n, m) \mathsf{cond}^{\varphi}(x) + o(\mathsf{u}_{\mathsf{mach}}).$$

It is important to note here that a backward-error analysis is not always possible. In these cases, one needs to obtain bounds for $\mathsf{RelError}(\varphi^{\mathcal{A}}(x))$ with a more direct (and usually laborious) approach referred to as *forward-error analysis*.

EXAMPLE 1. For the case of matrix inversion, we have the function $A \overset{\varphi}{\mapsto} A^{-1}$ from the set of invertible matrices to $\mathbb{R}^{n \times n}$. Under the assumption of unrestricted exponents (see Remark 1 above), the corresponding condition number $\mathsf{cond}^{\varphi}(A)$ is exactly $\|A\| \|A^{-1}\|$, a quantity usually denoted by $\kappa(A)$. Similar bounds hold for the problem of linear equation solving, $(A, b) \overset{\psi}{\mapsto} x = A^{-1}b$, except that now we only have the bounds $\kappa(A) \leqslant \mathsf{cond}^{\psi}(A, b) \leqslant 2\kappa(A)$. Also, in this case, a backward-error analysis shows that the computed (using Householder QR decomposition) solution $\tilde{x}$ satisfies, for some constant C,

$$\frac{\|\tilde{x} - x\|}{\|x\|} \leqslant C n^3 \mathsf{u}_{\mathsf{mach}} \mathsf{cond}^{\varphi}(A) + o(\mathsf{u}_{\mathsf{mach}}). \tag{9}$$

REMARK 2. Soon after the introduction of $\kappa(A)$ for error analysis, Hestenes and Stiefel [**34**] showed that this quantity also played a role in complexity analyses. More precisely, they showed that the number of iterations of the conjugate

gradient method (assuming infinite precision) needed to ensure that the current approximation to the solution of a linear system attained a given accuracy is proportional to $\sqrt{\kappa(A)}$.

A goal of this section is to show that, in spite of the rigor of the definition in (8), the idea of condition has a bit of an *ad hoc* character (we will return to this idea in Remark 4 below). The next two examples show a first gradual departure of condition as given in (8).

EXAMPLE 2. Given a matrix $A \in \mathbb{R}^{m \times n}$ such that $K(A) := \{y \in \mathbb{R}^n \mid Ay \geqslant 0\} \neq \{0\}$, we want to find a point in $K(A) \setminus \{0\}$.

This problem does not fit the framework above in the sense that the function φ is not well defined: any point y in $K(A)$ would do. In 1980, Goffin [29] analyzed the cost of a procedure to find one such y in terms of the 'best conditioned' point in $K(A)$. For $y \in \mathbb{R}^m$, one defines

$$\rho(A, y) := \min_{i \leqslant m} \frac{a_i \cdot y}{\|a_i\| \|y\|},$$

then

$$\rho(A) := \sup_{y \in K(A)} \rho(A, y),$$

and finally

$$\mathscr{C}(A) := \frac{1}{\rho(A)}.$$

Goffin's complexity analysis is in terms of $\mathscr{C}(A)$ (in addition to n and m).

EXAMPLE 3. Let $d_1, \ldots, d_n$ be positive integers, and let $\mathbf{d} = (d_1, \ldots, d_n)$. We denote by $\mathcal{H}_\mathbf{d}$ the complex vector space of systems $f = (f_1, \ldots, f_n)$ with $f_i \in \mathbb{C}[X_0, \ldots, X_n]$ homogeneous of degree d_i. The problem is to compute (that is, to approximate) a zero of $f \in \mathcal{H}_\mathbf{d}$. Generically, such a system f has $\mathcal{D} := d_1 \cdot \ldots \cdot d_n$ different zeros in complex projective space $\mathbb{P}^n$. For each one of them, Shub and Smale have characterized the value of $\mathsf{cond}^\zeta(f)$ (here, ζ is the selected zero) to be

$$\mu(f, \zeta) := \|f\| \left\| Df(\zeta)_{|T_\zeta}^{-1} \mathsf{diag}(\|\zeta\|^{d_i - 1}) \right\|,$$

where T_ζ is the tangent space at ζ and the inverse is of the restriction of $Df(\zeta)$ to this tangent space. Also, here, $\mathsf{diag}(x_i)$ denotes the diagonal matrix with entries $\{x_i\}$, and $\|f\|$ is the norm induced by Weyl's Hermitian product on $\mathcal{H}_\mathbf{d}$. A variation of this quantity,

$$\mu_{\mathsf{norm}}(f, \zeta) := \|f\| \left\| Df(\zeta)_{|T_\zeta}^{-1} \mathsf{diag}(\sqrt{d_i} \|\zeta\|^{d_i - 1}) \right\|,$$

has the advantage of being unitarily invariant, and has been used in several analyses of algorithms for approximating zeros [**2**, **3**, **9**, **50**–**54**]. For this problem, and as in the previous example, since the data is the system f and no zero ζ is specified (any one would do), a condition number $\mu(f)$ is defined in terms of the collection $\{\mu_{\mathsf{norm}}(f, \zeta)\}_{\zeta \mid f(\zeta)=0}$.

EXAMPLE 4. Let $a > 0$. We want to approximate $\sqrt{a}$ by using Hero's method (from Hero of Alexandria, although the method was already known to the Babylonians).

Assume, for the time being, that $a \in [1, 4)$, and take $x_0 = 5/2$ so that $x_0 > \sqrt{a}$. Let $x_{k+1} = H_a(x_k)$, where

$$H_a(x) := \frac{1}{2}\left(x + \frac{a}{x}\right).$$

Because of roundoff errors we actually compute a sequence $\tilde{x}_{k+1}$, and it is not difficult to show that, for some small constant C,

$$0 < \frac{\tilde{x}_k - \sqrt{a}}{\sqrt{a}} \leqslant \frac{3}{2^{k+1}} + C\mathsf{u}_{\mathsf{mach}}.$$

It follows that to ensure that $(\tilde{x}_k - \sqrt{a})/\sqrt{a} < \varepsilon$ it is enough to have both

$$k \geqslant |\log_2 \varepsilon| + 2 \quad \text{and} \quad \mathsf{u}_{\mathsf{mach}} \leqslant \frac{\varepsilon}{2C}. \tag{10}$$

To compute $\sqrt{a}$ with $a \geqslant 1$ arbitrary, one computes $b \in [1, 4)$ and $q \in \mathbb{N}$ such that $a = b \cdot 4^q$, and then $\sqrt{a} = \sqrt{b}\, 2^q$. The case $a \in (0, 1)$ is dealt with using that $\sqrt{a^{-1}} = (\sqrt{a})^{-1}$.

When $a \in [1, 4)$, the requirements (10) that ensure a relative error bounded by ε are independent of a. This is consistent with the fact that, for all $x > 0$, the condition number $\mathsf{cond}^{\sqrt{}}(x)$ given by (8) is constant (and equal to $\frac{1}{2}$). For arbitrary $a > 0$, in contrast, the scaling process, that is, the computation of b and q above, depends on the magnitude of a, in terms of both complexity (the value of k grows linearly with $\log q$) and accuracy (the log of $\mathsf{u}_{\mathsf{mach}}^{-1}$ also grows linearly with $\log q$).

2.4. The condition number theorem.

A common feature of the problems in Examples 1 to 3 is the existence of a subset of data instances at which the condition number is not well defined (or, more precisely, takes the value ∞). These data are considered *ill posed* with respect to the problem at hand in the sense that no amount of (finite) precision will guarantee a given accuracy

in the output. An old result [26] related to Example 1, nowadays known as the *condition number theorem* (CNT in what follows), shows that, if Σ denotes the set of noninvertible matrices (which are the ill-posed matrices for the problem of matrix inversion or linear equation solving), then

$$\kappa(A) = \frac{\|A\|}{d(A, \Sigma)}.$$

Here, $\|A\|$ denotes the spectral norm, and $d(A, \Sigma)$ refers to the distance induced by this norm. A systematic search for relations between condition and distance to ill-posedness was championed by Demmel [24]. These relations are nowadays frequently established.

2.5. Finite-valued problems. For a finite-valued problem, that is, one given by a function $\varphi : \mathbb{R}^n \to F$, where F is a finite set (say $F = \{y_1, \ldots, y_k\}$), the quantity $\mathsf{cond}^\varphi(x)$ is of little use. It is immediate to check that $\mathsf{cond}^\varphi(x) = \infty$ when x is in the boundary of some $S_j := \{x \in \mathbb{R}^n \mid \varphi(x) = y_j\}$, for $j = 1, \ldots, k$, and that $\mathsf{cond}^\varphi(x) = 0$ otherwise.

The family of boundaries between the sets S_j is composed of data x for which there is no hope that a finite-precision computation with input x will yield a reliable output. Elements in these boundaries are considered to be *ill posed*.

An idea championed by Renegar [47–49] is to define condition, for finite-valued problems, as the (relativized) inverse to the distance to ill-posedness (that is, to impose a CNT). But other approaches to define condition for these problems have been used as well.

EXAMPLE 5. One can turn the problem in Example 2 into a feasibility problem: given $A \in \mathbb{R}^{m \times n}$, decide whether $K(A) \neq \{0\}$. This problem can be solved with finite precision, and both the accuracy and the complexity of the algorithm are (nicely) bounded in terms of a simple extension of $\mathscr{C}(A)$ (for not necessarily feasible matrices A). One takes

$$\rho(A) := \sup_{\|y\|=1} \rho(A, y)$$

and

$$\mathscr{C}(A) := \frac{1}{|\rho(A)|}.$$

Note that $\rho(A) \geqslant 0$ if and only if A is feasible and A is ill posed precisely when $\rho(A) = 0$. This extension was done in [12], where it was proved that $\mathscr{C}(A)$ satisfies a CNT, namely, that $\mathscr{C}(A) = \|A\|_{12}/d_{12}(A, \Sigma)$ (here, $\| \ \|_{12}$ is the 1–2 operator norm, d_{12} its associated distance, and Σ the boundary between the sets

of feasible and infeasible matrices A). This extension was then used to analyze interior-point methods for the feasibility problem (see [21] and [10, Chs. 6, 7, 9, and 10]).

A variety of other condition measures have been proposed for this feasibility problem. A description of a few of them (with some comparisons) appears in [13].

EXAMPLE 6. The space $\mathcal{H}_{\mathbf{d}}^{\mathbb{R}}$ is the same as $\mathcal{H}_{\mathbf{d}}$, but with real coefficients. The problem now is, given $f \in \mathcal{H}_{\mathbf{d}}^{\mathbb{R}}$, to count the number of real projective zeros of f. This is a finite-valued problem. A finite-precision algorithm that solves it is described in [19], and both its complexity and accuracy are analyzed in terms of the condition number

$$\kappa(f) := \max_{x \in \mathbb{S}^n} \frac{\|f\|}{(\|f\|^2 \mu_{\mathsf{norm}}(f, x)^{-2} + \|f(x)\|_2^2)^{1/2}}.$$

Here, $\|f\|$ is the norm induced by Weyl's inner product on $\mathcal{H}_{\mathbf{d}}^{\mathbb{R}}$. In [20] it is shown that $\kappa(f)$ satisfies a CNT. Indeed, a system f is ill posed when arbitrary small perturbations can change the number of its real zeros. Denote by Σ the set of ill-posed systems. Then $\kappa(f) = \|f\|/d(f, \Sigma)$.

EXAMPLE 7. Let $\mathcal{H}_{[\mathbf{d}, m]}^{\mathbb{R}}$ denote the space of systems of m homogeneous polynomials in $n + 1$ variables with real coefficients. We want to determine whether one such system f is feasible, that is, whether there exists $x \in \mathbb{S}^n$ such that $f(x) = 0$. Here, $\mathbb{S}^n$ denotes the unit sphere in $\mathbb{R}^{n+1}$. An algorithm for solving this problem was given in [22] and analyzed in terms of

$$\kappa_{\mathsf{feas}}(f) := \begin{cases} \min_{\zeta \in Z_{\mathbb{S}}(f)} \mu_\dagger(f, \zeta) & \text{if } Z_{\mathbb{S}}(f) \neq \emptyset \\ \max_{\zeta \in \mathbb{S}^n} \dfrac{\|f\|}{\|f(x)\|} & \text{otherwise.} \end{cases}$$

Here, $Z_{\mathbb{S}}(f)$ denotes the zero set of f on $\mathbb{S}^n$ and $\mu_\dagger(f, \zeta)$ a version of $\mu_{\mathsf{norm}}(f, \zeta)$ for overdetermined systems defined with the help of the Moore–Penrose inverse.

REMARK 3. For the condition numbers $\mathscr{C}(A)$ and $\kappa(f)$ in Examples 5 and 6 we have that the condition is ∞ if and only if the data is in the boundary between sets of the form $S_j = \{x \in \mathbb{R}^n \mid \varphi(x) = y_j\}$ (inputs with a specific output).

This is not the case for $\kappa_{\mathsf{feas}}(f)$ in Example 7. This condition number takes the value infinity at the ill-posed systems f, but it does so at other systems as well (for instance on the case $m = 1$, $n = 1$, $d = 3$ and the polynomial X_1^3).

We still want to say that inputs f for which $\kappa_{\mathsf{feas}}(f) = \infty$ are ill posed. To distinguish between them and those in the boundaries between sets S_j we will, using an expression due to Renegar [46], call the latter *definitely ill posed*.

REMARK 4. We close this section by returning to the nature of condition numbers. Already the definition in (8) depends on a number of choices, for instance, the selection of a particular norm, or the way of measuring errors, which needs not be normwise (as in (5) and (7)) but may be componentwise instead. On top of this, we just saw that for finite-valued problems the quantity defined in (8) is of no practical use, and other forms of condition need to be defined. Examples 5–7 show that there is not a single, all-purpose, choice here either.

On top of all this, the result of Hestenes and Stiefel mentioned in Remark 2 triggered the emergence of an assortment of measures associated to data for various problems which were used for the complexity analysis of iterative algorithms. In the recent literature, these measures are almost invariably referred to as condition numbers.

3.   Decision problems and finite-precision machines

We formally define in this section both the class of problems we will deal with and the model for the machines solving them.

3.1. Decision problems.   Among the finite-valued problems the class of decision problems deserves emphasis. These are problems with only two outputs and are usually stated as a question on the input data which may have as answer either Yes or No. In the rest of this paper we will focus on decision problems.

Before giving a formal definition of decision problems, we note that natural objects occurring in the theory (circuits, polynomials, machines, . . .) have discrete and continuous components. Consistently with this division, algorithms perform computations both with real and discrete data (and computer languages such as C, Fortran, or Mathlab, distinguish between floating-point and integer numbers, and implement arithmetics for both). To reflect this situation, we will consider data in the product

$$\mathcal{I} := \{0, 1\}^\infty \times \mathbb{R}^\infty,$$

and define decision problems to appropriately take into account this structure. Here,

$$\mathbb{R}^\infty := \bigsqcup_{i=0}^{\infty} \mathbb{R}^i,$$

where the union is disjoint and $\mathbb{R}^0$ is an empty symbol so that $\{0, 1\}^\infty \times \mathbb{R}^0 \simeq \{0, 1\}^\infty$, and similarly for $\{0, 1\}^\infty$.

DEFINITION 1. A *decision problem* is a pair $(A, \boldsymbol{\mu})$ where $A \subset \mathcal{I}$ and $\boldsymbol{\mu} : \mathcal{I} \to [1, \infty]$. Here, $\boldsymbol{\mu}$ is the *condition number*.

We denote by Σ the set $\{(u, x) \in \mathcal{I} \mid \boldsymbol{\mu}(u, x) = \infty\}$, and we say that elements in Σ are *ill-posed*.

Examples 5 and 7 are decision problems.

REMARK 5. Different condition numbers for the same subset $A \subset \mathcal{I}$ define different decision problems. This is akin to the situation in classical (that is, both discrete and infinite-precision BSS) complexity theory where different encodings of the intended input data define (sometimes radically) different problems.

For instance, to specify a univariate real polynomial in *dense encoding*, we provide both its coefficients (the continuous data) and its degree (the discrete data). That is, we describe a polynomial $f = a_0 + a_1 X + \cdots + a_d X^d$ by using the array $[d, a_0, \ldots, a_d]$. Instead, if the encoding is *sparse*, we describe f by the list $\{(i, a_i) \mid a_i \neq 0\}$. The size of the dense encoding may be exponentially larger than the size of the sparse one, and consequently the complexity of a problem (for example, decide whether f has a real root) may dramatically depend on which of these two encodings the input is given.

We will return to the issue of the arbitrariness of condition numbers in Section 7.1.

3.2. Finite-precision machines, input size, and computational cost.

Briefly, a finite-precision machine is a BSS machine whose arithmetic is not exact but obeys the laws described in Section 2.1. We will not give a completely formal definition of them to avoid repeating a definition that nowadays is well known (readers wishing to read such definition will find it in [6] or [5, Section 3.2]).

DEFINITION 2. A *finite-precision BSS machine* is a BSS machine performing finite-precision computations. To precisely define the latter, we fix a number $\mathsf{u}_{\mathsf{mach}} \in (0, 1)$ (the *unit roundoff*), and let

$$\mathsf{k}_{\mathsf{mach}} := \left\lceil \log_2 \frac{1}{\mathsf{u}_{\mathsf{mach}}} \right\rceil.$$

In a $\mathsf{u}_{\mathsf{mach}}$-*computation*, built-in constants, input values, and the result of arithmetic operations (call any such number z) are systematically replaced by another real number $\mathsf{fl}(z)$ satisfying

$$\mathsf{fl}(z) = z(1 + \delta) \quad \text{for some } |\delta| < \mathsf{u}_{\mathsf{mach}}. \tag{11}$$

For all $(u, x) \in \mathcal{I}$, we denote by $\mathsf{Comp}(M, \mathsf{u}_{\mathsf{mach}}, u, x)$ the set of all possible $\mathsf{u}_{\mathsf{mach}}$-computations of M with input (u, x).

We will refer to $\mathsf{k}_{\mathsf{mach}} \in \mathbb{N}$ as the *precision* of M.

Finally, we will assume that the machine M has some special-purpose, read-only, registers storing the values of k_{mach} (as an element in $\{0, 1\}^\infty$) and u_{mach}.

REMARK 6. The definition above does not distinguish between real and discrete data. Making such a distinction is unnecessary (and we have therefore proceeded with simplicity as a goal). Indeed, discrete data can be encoded by sequences of real numbers in $[0, +\infty)$. A number x encodes 1 if $x > 0$ and encodes 0 otherwise. Furthermore, Turing machine computations can be simulated by finite-precision computations, and these simulations are both robust—they are always correct, independently of the precision at hand—and efficient—the cost of the simulation is linear in the cost of the Turing machine computation. We will assume both these encoding and simulations when talking about finite-precision machines taking inputs in $\{0, 1\}^\infty \times \mathbb{R}^\infty$, refer to the data thus encoded as *discrete data*, and indicate as *discrete computations* those (simulations) performed on discrete data. It is crucial to keep in mind, and we repeat it here, that these computations are error free.

REMARK 7. (i) Some of the details in the definition above are meant to allow for generality, whereas some others are chosen to simplify the exposition. The notion of u_{mach}-computation mimics the finite-precision computations described in Section 2.1 under the assumption of unrestricted exponents. But we do not impose a finite set $\mathbb{F}$ and actually allow for all possible outputs of fl in $\mathbb{R}$ as long as (11) is satisfied.

 (ii) The inclusion of divisions as a basic arithmetic operation in BSS machines makes possible the occurrence of divisions by zero. We will assume (as done in [6]) that all divisions are preceded by a test eliminating this possibility.

(iii) It is useful to think of the 'machine program' and the constant u_{mach} as separate entities. This allows one to consider the computations of a machine M for different values of u_{mach}. Recall, however, that the values of k_{mach} and u_{mach} are available to the program, and that the machine may use these values during the computation.

(iv) A (somehow peculiar) consequence of the availability of u_{mach} to the machine program is the fact that some u_{mach}-computations may not be u'_{mach}-computations for some $u'_{mach} > u_{mach}$. The obvious example is a computation that tests an inequality of the form $z < u_{mach}$.

 (v) We adopt the usual conventions encoding Yes and No that allow us to say that a u_{mach}-computation returns one such tag. In particular, we may talk about a computation *deciding* whether an input belongs to a set.

DEFINITION 3. We will say that a computation is *exact* when all its arithmetic operations are performed error free; that is, $\mathsf{fl}(z) = z$ in Definition 2.

Obviously, for every $\varepsilon > 0$ an exact computation is a possible ε-computation.

REMARK 8. It will occasionally be useful to talk about *infinite precision*. This amounts to setting $\mathsf{k}_{mach} = \infty$ and $\mathsf{u}_{mach} = 0$. In this case, it is easy to see that we recover the standard BSS machines.

To deal with complexity we need to fix two yardsticks: a measure of size (for the inputs of a problem) and a measure of cost (for the computations solving this problem). Complexity is the dependence of the latter on the former.

We define the *length* of $(u, x) \in \{0, 1\}^s \times \mathbb{R}^n \subset \mathcal{I}$, which we write as $\mathsf{length}(u, x)$, to be $s + n$. We define the *size* of (u, x) as

$$\mathsf{size}(u, x) := \mathsf{length}(u, x) + \lceil \log_2 \boldsymbol{\mu}(u, x) \rceil.$$

Note that if (u, x) is ill posed then $\mathsf{size}(u, x) = \infty$, and that otherwise $\mathsf{size}(u, x) \in \mathbb{N}$. Also, note that for pairs (u, x) with $\boldsymbol{\mu}(u, x) = 1$ (in particular, for elements $u \in \{0, 1\}^\infty$) we have $\mathsf{size}(u, x) = \mathsf{length}(u, x)$.

The computation of a BSS machine has a cost associated to it which is the number of steps performed before halting. We call this the *arithmetic cost* and, for the computation of a machine M on input $(u, x) \in \mathcal{I}$, we denote it by $\mathsf{ar\_cost}_M(u, x)$.

In addition to the arithmetic cost, we define the *accuracy cost* of a computation (of a machine M on input (u, x)) to be the smallest value of k_{mach} guaranteeing a correct answer. This is commonly associated with the cost in practice (measured in number of bit operations) of operating with floating-point numbers since, assuming the exponents of such numbers are moderately bounded, this cost is at most quadratic on k_{mach} for all the common implementations of floating-point arithmetic.

We can now deal with complexity.

3.3. Clocked computations.

Complexity classes are usually defined by putting restrictions on computation resources (notably, running time) as a function of input length. Our situation demands for a more involved approach due to a number of features proper to it: size depends on condition as well as on length (and condition is not known *a priori*), output correctness depends on the machine precision, and total cost must depend on this machine precision as well (since the cost of arithmetic operations in practice does so). Definition 4 below intends

to capture these features. It uses the common notion of time constructibility, which we next recall.

A function $T : \mathbb{N} \to \mathbb{N}$ is *time constructible* when there exists a Turing machine that with input n returns $T(n)$ in time $\mathcal{O}(T(n))$. Most of the functions used in complexity theory (for example, polynomial and exponential functions) are time constructible.

DEFINITION 4. Let $\mathsf{Arith} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ and $\mathsf{Prec} : \mathbb{N} \to \mathbb{N}$ be time-constructible functions. We say that a decision problem $(S, \boldsymbol{\mu})$ is *solved with cost* $(\mathsf{Arith}, \mathsf{Prec})$ when there exists a finite-precision BSS machine M satisfying the following. For every $(u, x) \in \mathcal{I}$ with $\boldsymbol{\mu}(u, x) < \infty$, the computation of M with input (u, x) satisfies

$$\mathsf{ar\_cost}_M(u, x) \leqslant \mathsf{Arith}(\mathrm{length}(u, x), \mathsf{k}_{\mathsf{mach}}),$$

and, if

$$\mathsf{k}_{\mathsf{mach}} \geqslant \mathsf{Prec}(\mathrm{size}(u, x)),$$

then all computations of M correctly decide whether $(u, x) \in S$.

We observe that the machine M in Definition 4 would need an infinite amount of time to correctly decide ill-posed inputs. In addition, we highlight two important features.

 (i) Computations are clocked; that is, their arithmetic cost is bounded by a function on two parameters immediately available: length of the input data and machine precision.

(ii) Computations are unreliable in the sense that there is no guarantee that the precision used is enough to ensure a correct output. Actually, correctness is not guaranteed even for exact computations.

Our basic deterministic complexity classes, P_{ro} and $\mathsf{EXP}_{\mathsf{ro}}$, will be obtained by appropriately bounding Arith and Prec in Definition 4.

The evaluation of many common functions (for example, a determinant) is done with clocked computations whose arithmetic cost, in general, depends only on the length of the input. The following example shows a general situation where, in contrast, this cost depends on $\mathsf{k}_{\mathsf{mach}}$ as well.

EXAMPLE 8. We want to decide whether a continuous function (for example, a polynomial) $f : [a, b] \to \mathbb{R}$ has a zero in the interval $[a, b]$. We consider as ill posed any pair $(f, [a, b])$ on which all the zeros of f are either at the endpoints of $[a, b]$ or are extrema of f. That is, such a pair is ill posed if f has a zero but does not change sign on $[a, b]$.

A simple scheme to solve this problem is to evaluate f in a set $X = \{a = x_0, x_1, \ldots, x_n = b\}$, say of equally spaced points, and to reject the input if all the obtained values have the same sign. Common sense suggests that it is useless to choose X to have too many points when $\mathsf{k}_{\mathsf{mach}}$ is small. It also suggests that it is useless to have a large $\mathsf{k}_{\mathsf{mach}}$ if the set X has few points. The values of $\mathsf{k}_{\mathsf{mach}}$ and n will have to grow, as it were, in tandem (and here the fact that a machine can read its current $\mathsf{k}_{\mathsf{mach}}$ is of the essence). When both values are low, we do not expect our scheme to correctly decide the existence of a zero of f. How large do they need to be for the scheme to do so? This of course depends on the characteristics of f, and should be measured by a condition number. We will see a detailed version of this scheme in Theorem 4.

3.4. A hierarchy theorem.
Early in the development of complexity theory it was proved that given more resources a Turing machine could solve more problems. These results were referred to as *hierarchy theorems*, and the two best known are for time [32] and space [31].

Here, we show a hierarchy theorem for precision.

PROPOSITION 1 (Precision hierarchy theorem). *Let $T : \mathbb{N} \to \mathbb{N}$ be time constructible, and let $P_1, P_2 : \mathbb{R}_+ \to \mathbb{R}_+$ be such that P_2 is continuous and increasing and $P_1 < P_2/2$. There exists a decision problem (B, μ) which can be decided with $\mathsf{ar\_cost}(u, x) \leqslant \mathcal{O}(T(\mathsf{length}(u, x)))$ and $\mathsf{k}_{\mathsf{mach}} = P_2(\mathsf{size}(u, x)) + 3$, without reading the values of $\mathsf{k}_{\mathsf{mach}}$ or $\mathsf{u}_{\mathsf{mach}}$, but cannot be decided with $\mathsf{k}_{\mathsf{mach}} = P_1(\mathsf{size}(u, x))$ (regardless of the arithmetic cost).*

Proof. Let (B, μ) given by the set

$$B := \Big\{ (n, x) \in \mathbb{N} \times \mathbb{R} \mid x \geqslant 0 \text{ and } x^{2^{T(\mathsf{length}(n, x))}} \geqslant \tfrac{1}{2} \Big\},$$

and the condition number

$$\mu(n, x) := 2^{P_2^{-1}(\log(1/\xi(n, x)))},$$

where

$$\xi(n, x) := \inf\{1, \varepsilon > 0 \mid \exists \delta, |\delta| \leqslant \varepsilon \text{ s.t. } (n, x(1 + \delta)) \in B \iff (n, x) \notin B\}.$$

The fact that P_2 is continuous and increasing allows us to use its inverse.

The definition of ξ implies that, for all (n, x), if $\mathsf{u}_{\mathsf{mach}} > \xi(n, x)$, then for any possible machine deciding B there are $\mathsf{u}_{\mathsf{mach}}$-computations yielding a wrong answer for input (n, x). Indeed, assume that $(n, x) \in B$, and let δ be such that

$|\delta| < \mathsf{u}_{\mathrm{mach}}$ and $x(1+\delta) \notin B$. Then the computation that first rounds x to $x(1+\delta)$ and then proceeds error free is a $\mathsf{u}_{\mathrm{mach}}$-computation and it returns that $(n, x) \notin B$ since this is the case for $(n, x(1+\delta))$; likewise, for the case $(n, x) \notin B$. It follows that the precision needed by any machine deciding B satisfies $\mathsf{u}_{\mathrm{mach}} \leqslant \xi(n, x)$ for all input (n, x). That is, we must have

$$\mathsf{k}_{\mathrm{mach}} = \left\lceil \log \frac{1}{\mathsf{u}_{\mathrm{mach}}} \right\rceil \geqslant \log \frac{1}{\xi(n, x)} = P_2(\log \boldsymbol{\mu}(n, x)).$$

Now consider any pair (n, x) with $\mathsf{length}(n, x) \leqslant \log \boldsymbol{\mu}(n, x)$. Then

$$\mathsf{size}(n, x) \leqslant 2 \log \boldsymbol{\mu}(n, x),$$

and therefore to decide such a pair we must have

$$\mathsf{k}_{\mathrm{mach}} \geqslant P_2(\log \boldsymbol{\mu}(n, x)) \geqslant P_2\!\left(\frac{\mathsf{size}(n, x)}{2} \right) > P_1(\mathsf{size}(n, x)).$$

This shows that (independently of arithmetic cost considerations) $(B, \boldsymbol{\mu})$ cannot be decided with $\mathsf{k}_{\mathrm{mach}} \leqslant P_1(\mathsf{size}(n, x))$.

To conclude, we will show that $(B, \boldsymbol{\mu})$ can be solved with the claimed cost bounds. To do so, we consider the algorithm that computes $x^{2^{T(\mathsf{length}(n,x))}}$ by repeated squaring, and use a simple backward-error argument.

The algorithm first computes $t := T(\mathsf{length}(n, x))$ (note that this is a discrete computation whose cost is $\mathcal{O}(T(\mathsf{length}(n, x))$ since T is time constructible) and then performs t multiplications. Its arithmetic cost is therefore $\mathcal{O}(T(\mathsf{length}(n, x))$. In addition, using Lemma 1, it is easy to see that the computed value q is of the form

$$x^{2^t}(1 + \theta_{2^{t+1}-1}),$$

and therefore, of the form

$$x^{2^t}(1 + \theta_2)^{2^t} = (x(1 + \theta_2))^{2^t},$$

where, we recall, $\theta_2 \in \mathbb{R}$ satisfies $|\theta_2| \leqslant \gamma_2 = (2\mathsf{u}_{\mathrm{mach}})/(1 - 2\mathsf{u}_{\mathrm{mach}}) \leqslant 3\mathsf{u}_{\mathrm{mach}}$ if $\mathsf{u}_{\mathrm{mach}} \leqslant \frac{1}{6}$.

Take $\mathsf{k}_{\mathrm{mach}} := P_2(\mathsf{size}(n, x)) + 3$. Then,

$$\mathsf{u}_{\mathrm{mach}} = 2^{-P_2(\mathsf{size}(n,x))+3} \leqslant \frac{1}{8} 2^{-P_2(\log \boldsymbol{\mu}(n,x))} = \frac{\xi(n, x)}{8}.$$

Our choice of $\mathsf{k}_{\mathrm{mach}}$ also implies that $\mathsf{u}_{\mathrm{mach}} \leqslant \frac{1}{6}$, and therefore that $|\theta_2| \leqslant 3\mathsf{u}_{\mathrm{mach}} < \xi(n, x)$. It follows from the definition of $\xi(n, x)$ that $(x(1 + \theta_2))^{2^t} \geqslant \frac{1}{2}$ if and only if $x^{2^t} \geqslant \frac{1}{2}$, and therefore that the machine correctly decides the pair (n, x). $\qquad \square$

4. Polynomial cost

We focus in this section on polynomial cost. We first define the general class capturing this notion and then proceed to describe the subclasses $\mathsf{P}_{\mathsf{dir}}$ and $\mathsf{P}_{\mathsf{iter}}$.

4.1. General polynomial time: the class P_{ro}.

DEFINITION 5. A decision problem $(S, \boldsymbol{\mu})$ belongs to P_{ro} (*roundoff polynomial cost*) when there exists a finite-precision BSS machine M solving S with cost $(\mathsf{Arith}, \mathsf{Prec})$ and such that

(i) Prec is bounded by a polynomial function; and

(ii) the function $\mathsf{Arith}(\mathsf{length}(u, x), \mathsf{Prec}(\mathsf{size}(u, x)))$ is bounded by a polynomial in $\mathsf{size}(u, x)$, for all $(u, x) \in \mathcal{I}$.

We note that the polynomial bound on Prec is satisfied whenever a bound of the form

$$\mathsf{u}_{\mathsf{mach}} \leqslant \frac{E}{2^{(\mathsf{length}(u,x))^r} \boldsymbol{\mu}(u, x)^s},$$

for some constants $E, r, s > 0$, ensures that M correctly decides whether $(u, x) \in S$. It is this kind of expression which is found in the literature.

REMARK 9.   (i) Definition 5 is somehow tortuous, and a few remarks may help to understand the issues at hand.

The fact that Prec is bounded by a polynomial guarantees that the precision required by the algorithm is at most polynomial in the input size, and hence that the (Turing) cost of each arithmetic operation is so.

Furthermore, the bound in (ii) implies that the arithmetic cost with this required precision is also polynomially bounded in the size of the input.

But the definition allows for various ways to achieve this combination. In the simplest, both Prec and Arith are polynomials in their arguments. A different possibility would allow a smaller bound for Prec, say logarithmic, and in this case the dependence of Arith on its second variable could be exponential and still have (i) holding true. In this case we say that $(S, \boldsymbol{\mu})$ can be solved with *logarithmic precision*.

(ii) It is clear that the quality of being in P_{ro}, for the problem of deciding a subset $S \subset \mathcal{I}$, depends on the condition number $\boldsymbol{\mu}$ associated to S. As we mentioned in Remark 5, this is akin to the situation in classical complexity theory where different encodings of the intended input may affect the membership of the problem to the class P (over $\{0, 1\}$) or $\mathsf{P}_{\mathbb{R}}$ (over $\mathbb{R}$).

The choice of a particular condition number (just as the choice of a particular encoding) is outside the theory. In the case of condition numbers, a probabilistic analysis allows one to compare complexity bounds (for possibly different algorithms, in terms of possibly different condition numbers) by taking expectations and expressing expected values of functions of condition numbers in terms of the length of the input. We will return to this theme in Section 7.1.

4.2. Fixed and variable precision.

A cursory look at various textbooks in numerical analysis shows the existence of two categories of algorithms, referred to as *direct* and *iterative* (the table of contents of [25] is a case at hand). For instance, for linear equation solving, Gauss elimination is a direct method whereas Jacobi's method is iterative, and for linear programing, the same distinction applies to the simplex and interior-point methods, respectively.

This grouping is strongly related to another one, based on whether the precision u_{mach} an algorithm works with needs to be considered as fixed or can be increased during the execution of the algorithm. In the first case one can expect correct outputs only for sufficiently well-conditioned inputs, whereas the goal in the second is to ensure correct outputs for all well-posed inputs, at the price of an indeterminate halting time.

For the case of functional (as opposed to decisional) problems, the difference can be discerned on examples we have already seen. In Example 1, we solve a system $Ax = b$ with Householder QR decomposition, which requires $\mathcal{O}(n^3)$ operations, a bound depending on the dimension of A only. The computed solution $\tilde{x}$ satisfies the relative error bound (9), whose right-hand side cannot be estimated without knowledge of the condition number $\kappa(A)$. Hence, we are uncertain about the magnitude of this error.

In Example 4, instead, given $\varepsilon > 0$, we can guarantee a relative error at most ε for the computed approximation $\tilde{x}_k$ of $\sqrt{a}$ provided that $k \geqslant |\log_2 \varepsilon| + 2$ and $u_{mach} \leqslant D\varepsilon$ for a constant D. Now both the arithmetic cost and the required precision will increase with a decrease on ε but, if we are allowed to adjust u_{mach}, we can ensure that the relative error is bounded by this ε.

The distinction between fixed and variable precision, even though less common in the literature, will serve us to define two natural subclasses of P_{ro}.

4.3. Fixed precision: the class P_{dir}.

DEFINITION 6. A decision problem (S, μ) belongs to P_{dir} (*direct polynomial cost*) when there exists a finite-precision BSS machine M satisfying the following.

For every $(u, x) \in \mathcal{I}$, the computation of M with input (u, x) never reads the values of $\mathsf{k}_{\mathsf{mach}}$ or $\mathsf{u}_{\mathsf{mach}}$ and satisfies that

$$\mathsf{ar\_cost}_M(u, x) \leqslant (\mathsf{length}(u, x))^{\mathcal{O}(1)},$$

and that, if

$$\mathsf{k}_{\mathsf{mach}} \geqslant (\mathsf{size}(u, x))^{\mathcal{O}(1)},$$

then all computations of M correctly decide whether $(u, x) \in S$.

If correctness is ensured as soon as $\mathsf{k}_{\mathsf{mach}} \geqslant (\log \mathsf{size}(u, x))^{\mathcal{O}(1)}$, we say that $(S, \boldsymbol{\mu})$ can be solved with *logarithmic precision*.

REMARK 10. (i) Computations of machines in $\mathsf{P}_{\mathsf{dir}}$ always halt within a bounded number of steps since their time bound depends only on the input length. The output of such computation, however, may be incorrect, and this will happen when the machine precision is insufficient.

(ii) If both $\mathsf{k}_{\mathsf{mach}} = \infty$ and $\mathsf{size}(u, x) = \infty$, we consider the second bound in Definition 6 to hold. This means that we can decide ill-posed inputs as long as we compute with infinite precision.

The following result is trivial.

PROPOSITION 2. *We have* $\mathsf{P}_{\mathsf{dir}} \subset \mathsf{P}_{\mathsf{ro}}$. □

The fact that a $\mathsf{P}_{\mathsf{dir}}$ machine never reads the values of $\mathsf{k}_{\mathsf{mach}}$ or $\mathsf{u}_{\mathsf{mach}}$ immediately yields the following result (compare with Remark 7(iv)).

PROPOSITION 3. *Let M be a $\mathsf{P}_{\mathsf{dir}}$ machine, and let $\varepsilon < \delta$. Then every ε-computation of M is a δ-computation as well.* □

The notion of Boolean circuits plays a fundamental role in discrete complexity theory (see, for example, [**43**, Section 4.3]). The same is true for algebraic circuits in algebraic complexity [**5**, Section 18.4], whose definition we recall next.

DEFINITION 7. An *algebraic circuit* is a connected, directed acyclic graph whose nodes have in-degree either 0, 2, or 3. Nodes with in-degree 0 are labeled either with a variable (we call them *input nodes*), or with a real constant (*constant nodes*). Nodes with in-degree 2 (called *arithmetic*) are labeled with an arithmetic operation in $\{+, -, \times, /\}$. Nodes with in-degree 3 are called *selection nodes*. Nodes with out-degree 0 are called *output nodes*.
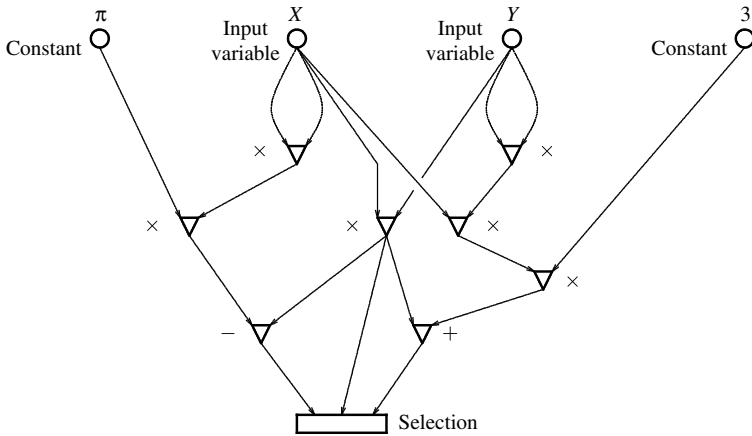
Figure 1. An algebraic circuit.

Figure 1 gives an example of an algebraic circuit.

REMARK 11. As in Remark 7(ii), we will assume that all division nodes are preceded by a test making sure that denominators are nonzero.

An algebraic circuit $\mathcal{C}$ with input variables $X_1, \ldots, X_n$ and m output nodes has naturally associated to it the computation of a function $f_{\mathcal{C}} : \mathbb{R}^n \to \mathbb{R}^m$. Given a point $x \in \mathbb{R}^n$, this computation proceeds from input and constant nodes to output nodes by performing the arithmetic operations and the selections. For the latter, if the selection node has three parents ξ, y, z, the selection returns y if $\xi < 0$ and z if $\xi \geqslant 0$. We call this computation the *canonical procedure*.

Figure 2 shows how the canonical evaluation is performed on the circuit of Figure 1 for an input $(x, y) \in \mathbb{R}^2$.

For $\varepsilon \in (0, 1)$, we define an ε-*evaluation* of $\mathcal{C}$ on input x to be any finite-precision computation (as described in Definition 2) of the canonical procedure with $\mathsf{u}_{\mathsf{mach}} = \varepsilon$, that is, any perturbation of this procedure in which the values of input variables, constants, or arithmetic nodes are multiplied by quantities of the form $(1 + \delta)$ with $|\delta| < \varepsilon$ (selections are performed error free). The notion of an *exact* evaluation is similarly defined.

The following result is straightforward.

PROPOSITION 4. *If* $\varepsilon < \delta$, *then every* ε-*evaluation of* $\mathcal{C}$ *at* $x \in \mathbb{R}^n$ *is a* δ-*evaluation as well.* □

$$\begin{cases} xy & \text{if } \pi x^2 - xy < 0 \\ xy + 3xy^2 & \text{otherwise.} \end{cases}$$
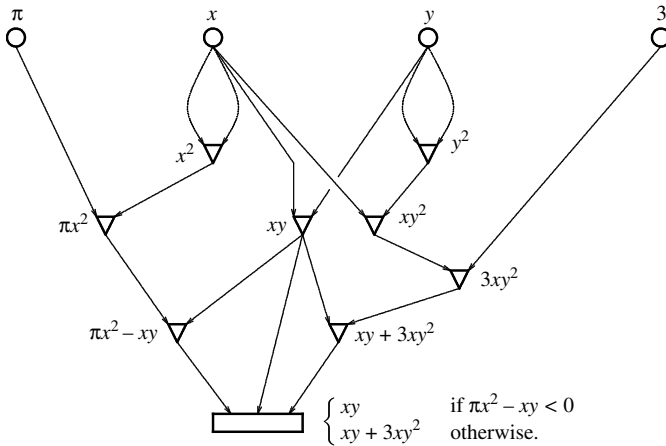
Figure 2. The canonical evaluation.

In all that follows we will assume that circuits have a single output node or that, if this is not the case, we have singled out one of them. In this way, we will be interested only in associated functions of the form $f_{\mathcal{C}} : \mathbb{R}^n \to \mathbb{R}$. In addition, we will write

$$S_{\mathcal{C}} := \{x \in \mathbb{R}^n \mid f_{\mathcal{C}}(x) \geqslant 0\}.$$

Algebraic circuits are easily encoded as points in $\mathcal{I}$ and can therefore be passed as input data to finite-precision BSS machines. We will define two decision problems based on this fact shortly. But before doing so we want to state a fundamental property of the simulation of the canonical procedure by BSS machines.

LEMMA 2. *There exists a finite-precision BSS machine that, with input a circuit $\mathcal{C}$ with n input nodes and a point $x \in \mathbb{R}^n$, computes $f_{\mathcal{C}}(x)$ following the canonical procedure. The machine does not read the values of $\mathsf{k}_{\mathrm{mach}}$ or $\mathsf{u}_{\mathrm{mach}}$. The arithmetic cost of this computation is linear in $\mathsf{length}(\mathcal{C})$. Furthermore, for every $\varepsilon \in (0, 1)$, the set $\mathsf{Comp}(M, \varepsilon, \mathcal{C}, x)$ of M bijects with the set of possible ε-evaluations of $\mathcal{C}$ on input x.*

Proof. The existence of a machine M evaluating circuits on points via the canonical procedure is clear. Furthermore, we note that all the data management of such M is performed with discrete data and is therefore error free. The only real-number arithmetic performed by M corresponds to the operations of $\mathcal{C}$, and the fact that M follows the canonical procedure means that both M and $\mathcal{C}$ evaluate

$f_{\mathcal{C}}$ following the same sequence of arithmetic operations. It follows from this that to each ε-computation of M with input $(\mathcal{C}, x)$ there corresponds an ε-evaluation of $\mathcal{C}$ on x, and conversely. $\qquad\square$

The next decision problem will be essential in what follows.

EXAMPLE 9. Instances for CircEval are algebraic circuits $\mathcal{C}$ (with input variables $X_1, \ldots, X_n$) together with a point $x \in \mathbb{R}^n$. The problem is to decide whether $x \in S_{\mathcal{C}}$.

To specify a condition number, we first define

$$\varrho_{\text{eval}}(\mathcal{C}, x)$$
$$:= \begin{cases} \sup\{\varepsilon < 1 \mid \text{all } \varepsilon\text{-evaluations of } \mathcal{C} \text{ at } x \text{ yield } x \in S_{\mathcal{C}}\} & \text{if } x \in S_{\mathcal{C}} \\ \sup\{\varepsilon < 1 \mid \text{all } \varepsilon\text{-evaluations of } \mathcal{C} \text{ at } x \text{ yield } x \notin S_{\mathcal{C}}\} & \text{otherwise.} \end{cases}$$

We then take as condition number

$$\mu_{\text{eval}}(\mathcal{C}, x) := \frac{1}{\varrho_{\text{eval}}(\mathcal{C}, x)}.$$

In case $(\mathcal{C}, x)$ is syntactically incorrect (for example, $\mathcal{C}$ is not properly encoded, $x \in \mathbb{R}^s$ with $s \neq n$) we set $(\mathcal{C}, x) \notin$ CircEval and take $\mu_{\text{eval}}(\mathcal{C}, x) := 1$.

PROPOSITION 5. *We have* CircEval $\in \mathsf{P}_{\text{dir}}$.

Proof. We consider the machine, given by Lemma 2, that with input $(\mathcal{C}, x)$ computes $f_{\mathcal{C}}(x)$ and accepts if the result of this computation is greater than or equal to zero.

The arithmetic cost of this computation is linear in length$(\mathcal{C})$. Hence, we only need to check that the machine decides correctly as long as its precision is polynomially bounded on size$(\mathcal{C}, x)$.

For a well-posed input $(\mathcal{C}, x)$, let $\mathsf{u}_{\text{mach}} := 2^{-\text{size}(\mathcal{C}, x)}$. Then

$$\mathsf{u}_{\text{mach}} = \frac{1}{2^{\text{length}(\mathcal{C}, x) + \lceil \log \mu(\mathcal{C}, x) \rceil}} < \frac{1}{\mu(\mathcal{C}, x)} = \varrho_{\text{eval}}(\mathcal{C}, x),$$

and the definition of $\varrho_{\text{eval}}(\mathcal{C}, x)$ ensures that every u_{mach}-evaluation of $\mathcal{C}$ at x yields $f_{\mathcal{C}(x)} \geqslant 0$ if and only if $x \in S_{\mathcal{C}}$. That is, the u_{mach}-evaluation yields the same sign ($\geqslant 0$ or < 0) as the infinite-precision evaluation. In other words, this u_{mach}-evaluation correctly decides the input. Since the choice of u_{mach} is equivalent to setting $\mathsf{k}_{\text{mach}} := \text{size}(\mathcal{C}, x)$, we are done. $\qquad\square$

The (discrete) class P is included in P_{dir} via the simulation of Turing machines mentioned in Remark 6 (we take condition one for all inputs in a discrete problem). We will see in Section 4.5 that P_{dir} is also closely related to $P_{\mathbb{R}}$.

4.4.   Variable precision: the class P_{iter}.   The definition of P_{ro} requires that, for a given input (u, x), the machine correctly decides the membership of (u, x) to S as soon as k_{mach} is large enough. But it does not impose any specific output otherwise. If the precision is insufficient, an output in {Yes, No} may be wrong. In general, this possibility cannot be ruled out *a priori* as we do not know the condition $\mu(u, x)$ of the input and, consequently, cannot estimate whether the available precision is sufficient or not for the input at hand. The same can be said of P_{dir}.

For some decision problems, however, this uncertainty can be avoided. This is the case when there is a simple procedure to guarantee, given (u, x) and k_{mach}, that the computed output is correct. Such a situation would therefore allow three possible outputs: Yes, No, and Unsure, the first two being guaranteed correct, and the last meaning 'I need more resources to decide this input'. Availability of such a procedure naturally introduces the consideration of *variable-precision* algorithms.

These are iterative algorithms which can modify the value of their precision k_{mach} and always return the correct output (Yes or No) to the decision question. They adaptively increase their precision (in theory rereading their input each time they do so) and only halt when the current precision, together with the computations done, guarantee a correct answer. That is, their form typically follows the following general scheme:

```
input (u, x)
initialize k_mach
repeat
  attempt to decide (u, x) and halt                (GS)
    if the outcome is either Yes or No
  if the outcome is Unsure then increase k_mach
```

One can therefore define a subclass P_{iter} of P_{ro} that contains the problems for which such an error-free algorithm exists (with the appropriate polynomial cost bounds).

DEFINITION 8.   A decision problem (S, μ) belongs to P_{iter} (*iterative polynomial cost*) when there exists a BSS machine M and functions Arith, Prec satisfying the bounds (i) and (ii) in Definition 5 and such that, for all (u, x) with $\mu(u, x) < \infty$,

(i) all computations of M return an element in {Yes, No, Unsure} and in the first two cases this output is always correct,

(ii)
$$\mathsf{ar\_cost}_M(u, x) \leqslant \mathsf{Arith}(\mathsf{length}(u, x), \mathsf{k_{mach}}),$$

and

(iii) if
$$\mathsf{k_{mach}} \geqslant \mathsf{Prec}(\mathsf{size}(u, x))$$

then all computations of M return an element in $\{\texttt{Yes}, \texttt{No}\}$.

REMARK 12. Unlike machines in $\mathsf{P_{dir}}$, the halting time of the scheme (GS) (endowed with a machine as in Definition 8) is not bounded. It increases with the input size and may be infinite (that is, the machine may loop forever) for ill-posed inputs. On the other hand, outputs of this scheme are always correct.

We have already described (without saying so) a problem in $\mathsf{P_{iter}}$.

EXAMPLE 5 (CONTINUED). It was shown in [**21**] (see also [**10**, Section 9.4]) that the feasibility of a system $Ay \geqslant 0$, $y \neq 0$, can be decided using an iterative algorithm that follows the general scheme (GS). The algorithm carries out

$$\mathcal{O}\big(\sqrt{n}(\log n + \log \mathscr{C}(A))\big)$$

iterations, each of them performing $\mathcal{O}(n^3)$ arithmetic operations. The value of $\mathsf{u_{mach}}$ is refined at each iteration, and the finest value used by the algorithm satisfies
$$\mathsf{u_{mach}} = \frac{1}{\mathcal{O}(n^{12}\mathscr{C}(A)^2)}.$$

These bounds show that the problem is in $\mathsf{P_{iter}}$.

PROPOSITION 6. *We have* $\mathsf{P_{iter}} \subset \mathsf{P_{ro}}$.

Proof. The only difference between the machine M in Definition 8 and one as requested in Definition 5 is that the former may return the tag $\texttt{Unsure}$. And it is straightforward to check that if an output $\texttt{No}$ is imposed on these cases the resulting machine shows membership to $\mathsf{P_{ro}}$ (the same holds true on imposing $\texttt{Yes}$). □

4.5. Some remarks on infinite precision. Most of the literature in numerical computation describes algorithms in a context of infinite precision. Finite-precision analyses are tedious and, more often than not, avoided. It is therefore worthwhile pondering on what the classes $\mathsf{P_{dir}}$ and $\mathsf{P_{iter}}$ become under the presence

of infinite precision. For this, in the definition of these two classes, one replaces the occurences of k_{mach} in the bound for the arithmetic cost by the bound for k_{mach} (so that the former will be in terms of the input's size only) and then disregards the requirement on k_{mach} altogether. Because the assumption of infinite precision puts us on the standard BSS realm, the classes thus obtained will be either new or already existing classes in the standard theory.

By definition, the arithmetic cost of machines in P_{dir} is independent of the precision at hand or the input's condition. Also, since we are assuming infinite precision, the issue of how large k_{mach} needs to be to guarantee a correct answer becomes irrelevant. All in all, condition plays no role, and the following straightforward result shows that, under the presence of infinite precision, the class P_{dir} is, essentially, $P_{\mathbb{R}}$.

PROPOSITION 7. *Let $S \subset \mathcal{I}$. If $(S, \boldsymbol{\mu}) \in P_{dir}$ then $S \in P_{\mathbb{R}}$. Conversely, if $S \in P_{\mathbb{R}}$ then $(S, \mu_\infty) \in P_{dir}$. Here, μ_∞ is the constant function with value ∞.* $\qquad\square$

Consider now a computation in P_{iter} endowed with infinite precision. Because the computations are error free, the only role played by k_{mach} is in allowing more computing time at each iteration. And part (ii) in Definition 8 puts a bound on the total running time in terms of the input's condition (or size). The complexity class emerging captures many algorithms described in the literature.

We say that a problem $(S, \boldsymbol{\mu})$ is in P_∞ when there exists a (standard) BSS machine deciding $(S, \boldsymbol{\mu})$ whose running time on input $(u, x) \in \mathcal{I}$ is bounded by a polynomial in $\mathsf{size}(u, x)$.

This is a new class in the standard BSS setting, the first one (to the best of our knowledge) to consider condition as a complexity parameter. One clearly has $P_{iter} \subset P_\infty$.

5. Nondeterministic polynomial cost

Problems in (all versions of) NP are sets S for which membership of an element x to S can be established through an 'easy' proof y. All definitions of the discrete class NP translate that y is easy by imposing that the length of y must be polynomially bounded in the length of x (in addition of the fact that one can verify that y is a proof of $x \in S$ in polynomial time in the length of (x, y)), and similarly for the class $NP_{\mathbb{R}}$.

The finite-precision context introduces additional considerations. Not only will the length of a proof y have to be appropriately bounded; it is clear that the condition of the pair (x, y) will have to be so as well. The class NP_{ro}^U is obtained by imposing these restrictions.

Even more so, we might want the magnitude of a proof y, understood as how large or small the absolute value of the components of y can be, to be bounded as well. This gives rise to the class $\mathsf{NP}_{\mathsf{ro}}^{\mathsf{B}}$.

5.1. The class $\mathsf{NP}_{\mathsf{ro}}^{\mathsf{U}}$.

DEFINITION 9. A decision problem $(W, \boldsymbol{\mu}_W)$ belongs to $\mathsf{NP}_{\mathsf{ro}}^{\mathsf{U}}$ (*nondeterministic roundoff polynomial cost*) when there exist a decision problem $(B, \boldsymbol{\mu}_B)$, a finite-precision BSS machine M deciding $(B, \boldsymbol{\mu}_B)$ in P_{ro}, and polynomials p, Q, such that, for $(u, x) \in \mathcal{I}$,

(i) if $(u, x) \in W$, then there exists $y^* \in \mathbb{R}^m$, such that $(u, x, y^*) \in B$, and $\log \boldsymbol{\mu}_B(u, x, y^*) \leqslant Q(\log \boldsymbol{\mu}_W(u, x))$; and

(ii) if $(u, x) \notin W$ then, for all $y \in \mathbb{R}^m$ we have $(u, x, y) \notin B$ and $\log \boldsymbol{\mu}_B(u, x, y) \leqslant Q(\log \boldsymbol{\mu}_W(u, x))$.

Here, $m = p(\mathsf{length}(u, x))$.

EXAMPLE 10. Instances for $\mathsf{CircFeas}$ are algebraic circuits $\mathcal{C}$ (with input variables $Y_1, \ldots, Y_m$). The problem is to decide whether there exists $y \in \mathbb{R}^m$ such that $y \in S_{\mathcal{C}}$ (in which case, we say that $\mathcal{C}$ is *feasible*). We take as condition number

$$\mu_{\mathsf{feas}}(\mathcal{C}) := \frac{1}{\varrho_{\mathsf{feas}}(\mathcal{C})},$$

where

$$\varrho_{\mathsf{feas}}(\mathcal{C}) := \begin{cases} \sup_{y \in S_{\mathcal{C}}} \varrho_{\mathsf{eval}}(\mathcal{C}, y) & \text{if } \mathcal{C} \text{ is feasible,} \\ \inf_{y \in \mathbb{R}^m} \varrho_{\mathsf{eval}}(\mathcal{C}, y) & \text{otherwise.} \end{cases}$$

Note that, in the feasible case, $\mu_{\mathsf{feas}}(\mathcal{C})$ is the condition of its best-conditioned solution, and, in the infeasible case, it is the condition of the worst-conditioned point in $\mathbb{R}^m$.

PROPOSITION 8. *We have* $\mathsf{CircFeas} \in \mathsf{NP}_{\mathsf{ro}}^{\mathsf{U}}$.

Proof. We need to exhibit a problem $(B, \boldsymbol{\mu}_B)$ and a machine M as in Definition 9. For $(B, \boldsymbol{\mu}_B)$, we take the problem $\mathsf{CircEval}$, which we know is in P_{ro} by Proposition 5. For M, we take the machine in the proof of that proposition (which actually shows that $\mathsf{CircEval} \in \mathsf{P}_{\mathsf{dir}}$).

Let $\mathcal{C}$ be a circuit having m input variables. If $\mathcal{C} \in \mathsf{CircFeas}$ then there exist points $y \in S_{\mathcal{C}}$. Choose y^* among those, that additionally satisfies.

$$\varrho_{\mathsf{feas}}(\mathcal{C}) = \varrho_{\mathsf{eval}}(\mathcal{C}, y^*).$$

(Such a y^* may not exist since the sup in the definition of $\varrho_{\mathsf{feas}}(\mathcal{C})$ may not be attained. As the modification to the proof in such a case is straightforward, for the sake of clarity and conciseness, we will write the proof for the case that y^* exists.)

Since $m \leqslant \mathsf{length}(\mathcal{C})$, the first requirement in Definition 9(i) is trivially true. In addition, $\mu_{\mathsf{feas}}(\mathcal{C}) = \mu_{\mathsf{eval}}(\mathcal{C}, y^*)$, which implies that $\mu_{\mathsf{eval}}(\mathcal{C}, y^*) \leqslant \mu_{\mathsf{feas}}(\mathcal{C})$, and hence the second requirement.

Now assume that $\mathcal{C} \notin \mathsf{CircFeas}$. Then, for all $y \in \mathbb{R}^m$, $(\mathcal{C}, y) \notin B$. The second requirement in Definition 9(ii) is immediate, since the definition of $\mu_{\mathsf{feas}}(\mathcal{C})$ implies that, in the infeasible case, $\varrho_{\mathsf{feas}}(\mathcal{C}) \leqslant \varrho_{\mathsf{eval}}(\mathcal{C}, y)$ for all $y \in \mathbb{R}^m$, and hence that $\mu_{\mathsf{eval}}(\mathcal{C}, y) \leqslant \mu_{\mathsf{feas}}(\mathcal{C})$. $\square$

PROPOSITION 9. *We have* $\mathrm{P_{ro}} \subset \mathsf{NP_{ro}^U}$.

Proof. Let $(W, \boldsymbol{\mu}_W) \in \mathrm{P_{ro}}$. Let $(B, \boldsymbol{\mu}_B)$ where $B = W \times \mathbb{R}$ and $\boldsymbol{\mu}_B(u, x, y) = \boldsymbol{\mu}_W(u, x)$. Let M' be any machine deciding $(W, \boldsymbol{\mu}_W)$ in $\mathrm{P_{ro}}$, and M the machine that with input (u, x, y) runs M' on (u, x). Then M shows that $(B, \boldsymbol{\mu}_B) \in \mathrm{P_{ro}}$, and the pair $((B, \boldsymbol{\mu}_B), M)$ establishes that $(W, \boldsymbol{\mu}_W) \in \mathsf{NP_{ro}^U}$ (with $p = 1$ and $Q(n) = n$). $\square$

DEFINITION 10. A $\mathrm{P_{ro}}$-*reduction* from $(W, \boldsymbol{\mu}_W)$ to $(S, \boldsymbol{\mu}_S)$ is a finite-precision machine $\overline{M}$ which, given a point $(u, x) \in \mathcal{I}$ and a number $k \in \mathbb{N}$, performs a discrete computation and returns a pair $(v, z) \in \mathcal{I}$ with $\mathsf{ar\_cost}_{\overline{M}}(u, x)$ polynomially bounded on $\mathsf{length}(u, x)$ and k.

In addition, we require the existence of some $D, p > 0$ such that, for all $k \geqslant D \, \mathsf{size}(u, x)^p$, one has

(i) $(u, x) \in W \iff (v, z) \in S$; and

(ii) $\log \boldsymbol{\mu}_S(v, z)$ is polynomially bounded in $\mathsf{size}_W(u, x)$.

If all of the above hold, we write $(W, \boldsymbol{\mu}_W) \preceq_{\mathsf{ro}} (S, \boldsymbol{\mu}_S)$.

The notion of $\mathrm{P_{ro}}$-reduction is tailored to capture the basic property any polynomial-time reduction must have (Proposition 10 below) together with the peculiarities of finite-precision computations. We observe that, because the computation of $\overline{M}$ is discrete, the components of z must be either components of x or machine constants of $\overline{M}$.

PROPOSITION 10. *If $(W, \boldsymbol{\mu}_W) \preceq_{\mathsf{ro}} (S, \boldsymbol{\mu}_S)$ and $(S, \boldsymbol{\mu}_S) \in \mathsf{P}_{\mathsf{ro}}$, then $(W, \boldsymbol{\mu}_W) \in \mathsf{P}_{\mathsf{ro}}$.*

Proof. Let $\overline{M}$ be as in the definition above, and let N_S be a finite-precision machine solving S in P_{ro}.

By hypothesis, the arithmetic cost of $\overline{M}$ on input (u, x) is bounded by a polynomial in $\mathsf{length}(u, x)$ and k, and therefore such a bound also holds for $\mathsf{length}(v, z)$ (since the cost of writing the output is smaller than the total arithmetic cost). That is, there exist constants $A, t \in \mathbb{N}$, such that

$$\mathsf{length}(v, z) \leqslant A(\mathsf{length}(u, x)k)^t. \tag{12}$$

Also, there exist $D, E, p, s \in \mathbb{N}$ such that, for any input (u, x), if

$$k \geqslant D\,\mathsf{size}(u, x)^p \tag{13}$$

then

$$\log \boldsymbol{\mu}_S(v, z) \leqslant E\,\mathsf{size}_W(u, x)^s \quad \text{and} \quad (v, z) \in S \iff (u, x) \in W. \tag{14}$$

Finally, we know that, for some $C, q \in \mathbb{N}$, and for any input (v, z), if

$$\mathsf{k}_{\mathsf{mach}}(N_S) \geqslant C\,\mathsf{size}(v, z)^q \tag{15}$$

then N_S correctly decides whether $(v, z) \in S$.

Let $\mathcal{M}$ be the machine given by the following code:

```
input (u, x)
compute k := ⌊k_mach^(1/2qt)⌋
run M̄ on input ((u, x), k)
let (v, z) be the returned point
run N_S on input (v, z) and accept iff N_S accepts
```

We will prove that this machine decides $(W, \boldsymbol{\mu}_W)$ in P_{ro}. We begin by observing that, by our choice of k, we have

$$k^{2qt} \leqslant \mathsf{k}_{\mathsf{mach}} \leqslant (k + 1)^{2qt}. \tag{16}$$

We first deal with the complexity. We know that the arithmetic cost of $\overline{M}$ is polynomial in $\mathsf{length}(u, x)$ and k. Since the latter is bounded by $\mathsf{k}_{\mathsf{mach}}(\mathcal{M})$, we are done, with the cost of $\overline{M}$. And the cost of N_S is also polynomially bounded in $\mathsf{length}(u, x)$ and $\mathsf{k}_{\mathsf{mach}}(\mathcal{M})$, since such a bound holds on $\mathsf{length}(v, z)$ and $\mathsf{k}_{\mathsf{mach}}(\mathcal{M})$ and, by (12), the first term is polynomially bounded on $\mathsf{length}(u, x)$ and k.

We next deal with the precision needed. Set

$$\mathsf{k_{mach}} := \lceil \max\{C^2(2A)^{2q}\,\mathsf{size}(u,x)^{2qt}, C(2E)^q(\mathsf{size}(u,x)+1)^{qs},$$
$$(D\,\mathsf{size}(u,x)^p + 1)^{2qt}\}\rceil.$$

This value of $\mathsf{k_{mach}}$ is clearly polynomially bounded on $\mathsf{size}(u,x)$. Now, because of the last term within the brackets (and (16)), we have

$$k \geqslant \mathsf{k}_{\mathsf{mach}}^{1/2qt} - 1 \geqslant D\,\mathsf{size}(u,x)^p;$$

that is, (13) holds. It follows that the simulation of $\overline{M}$ on input $((u,x),k)$ returns (v,z) satisfying (14).

In addition, we have

$$\mathsf{size}(v,z) \leqslant 2\max\{\log \boldsymbol{\mu}_S(v,z) + 1, \mathsf{length}(v,z)\}.$$

We divide by cases.

(1) Assume first that $\mathsf{size}(v,z) \leqslant 2\,\mathsf{length}(v,z)$. Then, using (12),

$$\begin{aligned}
C\,\mathsf{size}(v,z)^q &\leqslant C(2A)^q\,\mathsf{length}(u,x)^{qt}k^{qt} \\
&\leqslant C(2A)^q\,\mathsf{size}(u,x)^{qt}\sqrt{\mathsf{k_{mach}}} \\
&\leqslant \mathsf{k_{mach}}
\end{aligned}$$

the last inequality since $\mathsf{k_{mach}} \geqslant C^2(2A)^{2q}\,\mathsf{size}(u,x)^{2qt}$.

(2) Assume instead that $\mathsf{size}(v,z) \leqslant 2(\log \boldsymbol{\mu}_S(v,z)+1)$. Then, and here we use the first statement in (14),

$$\begin{aligned}
C\,\mathsf{size}(v,z)^q &\leqslant C\,2^q(\log \boldsymbol{\mu}_S(v,z)+1)^q \\
&\leqslant C\,2^q(E(\mathsf{size}_W(u,x))^s + 1)^q \\
&\leqslant C\,(2E)^q(\mathsf{size}(u,x)+1)^{qs} \\
&\leqslant \mathsf{k_{mach}}.
\end{aligned}$$

In both cases the precision on the simulation of N on input (v,z) satisfies (15), and therefore we have both that $(v,z) \in S$ if and only if $(u,x) \in W$ and that N_S correctly decides whether $(v,z) \in S$. It follows that $\mathcal{M}$ correctly decides whether $(u,x) \in W$. □

DEFINITION 11. We say that a decision problem $(S, \boldsymbol{\mu}_S)$ is $\mathrm{NP}_{\mathsf{ro}}^{\mathsf{U}}$-*hard* when for any problem $(W, \boldsymbol{\mu}_W) \in \mathrm{NP}_{\mathsf{ro}}^{\mathsf{U}}$ we have $(W, \boldsymbol{\mu}_W) \preceq_{\mathsf{ro}} (S, \boldsymbol{\mu}_S)$. We say that it is $\mathrm{NP}_{\mathsf{ro}}^{\mathsf{U}}$-*complete* when it is $\mathrm{NP}_{\mathsf{ro}}^{\mathsf{U}}$-hard and it belongs to $\mathrm{NP}_{\mathsf{ro}}^{\mathsf{U}}$.

THEOREM 1. *The problem* CircFeas *is* $\mathsf{NP}_{\mathsf{ro}}^{\mathsf{U}}$-*complete.*

Proof. We have already seen in Proposition 8 that CircFeas is in $\mathsf{NP}_{\mathsf{ro}}^{\mathsf{U}}$. The hardness of CircFeas applies arguments that have been used once and again, adapted to our context.

Consider a problem $(W, \boldsymbol{\mu}_W)$ and a pair $((B, \boldsymbol{\mu}_B), M)$ as in Definition 9 certifying this problem in $\mathsf{NP}_{\mathsf{ro}}^{\mathsf{U}}$. Let p, Q and m be as in that definition. Also, let Arith_M and Prec_M be the functions bounding the arithmetic cost and necessary precision of M, as in Definition 5. Without loss of generality, we will assume that Prec_M is monotonically increasing.

Next, fix an input $(u, x) \in \mathcal{I}$, $k \in \mathbb{N}$, and let $\ell := \mathsf{length}(u, x)$, $m = p(\ell)$, and $T := \mathsf{Arith}_M(\ell + m, k)$. Then, for all $y \in \mathbb{R}^m$, all computations of M with input (u, x, y) and precision $\mathsf{k}_{\mathsf{mach}} = k$ halt and return an output in $\{\mathsf{Yes}, \mathsf{No}\}$ within T steps. One can construct a decision circuit $\mathcal{C}$ of depth T in the variables $Y_1, \ldots, Y_m$ (having the numbers $x_1, \ldots, x_n$ associated to constant nodes) which replicates these computations. More precisely, $\mathcal{C}$ satisfies the following conditions.

(a) The number of nodes of $\mathcal{C}$ is polynomial in ℓ and k.

(b) For all $y \in \mathbb{R}^m$, there exist accepting $\mathsf{u}_{\mathsf{mach}}$-computations of M with input (u, x, y) if and only if there exist $\mathsf{u}_{\mathsf{mach}}$-evaluations of $\mathcal{C}$ at y yielding $y \in S_{\mathcal{C}}$, and similarly for rejecting computations and $y \notin S_{\mathcal{C}}$.

(c) The circuit $\mathcal{C}$ is computed with cost polynomial in ℓ and k, and this computation is discrete.

The construction of this circuit is given with details in [23], so we will not repeat these details here. We may nonetheless summarize the main idea (but note that this requires familiarity with BSS machines as presented in [6] on the part of the reader).

At any time during the computation of M the internal state of the machine can be described by the current node $\eta \in \{1, \ldots, N\}$ of M together with an element in its state space $\mathbb{N} \times \mathbb{N} \times \mathbb{R}^{\infty}$. If the computation performs T steps, then the values of the first two components are themselves bounded by T, and the only components of $\mathbb{R}^{\infty}$ that ever play a role in it are the first T. It follows that the relevant variable taking values during the computation are the following.

$i_t, j_t \in \{0, \ldots, T\}$: for the values of the two integer components of the state space at time $t = 0, \ldots, T$.

$w_t \in \{1, \ldots, N\}$: for the value of the current node at time $t = 0, \ldots, T$.

$z_{s,t} \in \mathbb{R}$: for the value of the sth component of the state space at time t, s, $t = 0, \ldots, T$.

The values of these variables at time $t = 0$ are given by the initialization of the machine. For $t \geqslant 1$, these values depend on the values of a few (at most eight; see [**23**]) variables at time $t - 1$. In addition, this dependence is simple in the sense that it can be computed by circuits $\mathcal{I}_t$, $\mathcal{J}_t$, $\mathcal{W}_t$, and $\mathcal{Z}_{s,t}$, each of them with a small, bounded number of nodes. The circuit $\mathcal{C}$ above is obtained by appropriately connecting these four families plus a series of constant nodes corresponding to the constants of M, the components of x, and the numbers $\mathsf{u}_{\mathrm{mach}}$ and $\mathsf{k}_{\mathrm{mach}}$. A further small subcircuit is required, which returns 1 if the computation of M accepts and returns -1 otherwise. It is straightforward to see that it satisfies conditions (a)–(c) above.

The P_{ro}-reduction is given by the machine $\overline{M}$ which, given (u, x), returns $\mathcal{C}$.

We will prove that this is indeed a reduction. The first condition in Definition 10, the fact that $\mathsf{ar\_cost}_{\overline{M}}(u, x)$ is polynomially bounded in $\mathsf{length}(u, x)$ and k, is just property (c) above. We therefore focus on the other two conditions in Definition 10, which require us to find an appropriate lower bound for k.

Let Q be the polynomial in Definition 9 and

$$Y := \{y \in \mathbb{R}^m \mid \log \boldsymbol{\mu}_B(u, x, y) \leqslant Q(\log \boldsymbol{\mu}_W(u, x))\}.$$

Then, for all $y \in Y$,

$$\begin{aligned}
\mathsf{size}_B(u, x, y) &= \mathsf{length}(u, x) + m + \lceil \log \boldsymbol{\mu}_B(u, x, y) \rceil \\
&\leqslant \ell + p(\ell) + \lceil Q(\log \boldsymbol{\mu}_W(u, x)) \rceil \leqslant R(\mathsf{size}_W(u, x)) \quad (17)
\end{aligned}$$

for a polynomial R.

Take any k satisfying

$$k \geqslant k_0 := \mathsf{Prec}_M(R(\mathsf{size}(u, x))).$$

Note that k_0 is polynomially bounded in $\mathsf{size}(u, x)$. We claim that requirements (i) and (ii) in Definition 10 hold for these k.

To prove this claim, set the precision of M to be $\mathsf{u}_{\mathrm{mach}} = 2^{-k_0}$ (and, accordingly, $\mathsf{k}_{\mathrm{mach}} := k_0$). The monotonicity of Prec_M, together with (17), implies that, for all $y \in Y$,

$$\mathsf{k}_{\mathrm{mach}} = \mathsf{Prec}_M(R(\mathsf{size}_W(u, x))) \geqslant \mathsf{Prec}_M(\mathsf{size}_B(u, x, y)). \quad (18)$$

We now divide by cases.

Case I: $(u, x) \in W$. In this case, there exists $y^* \in Y$ such that $(u, x, y^*) \in B$. Since $y^* \in Y$, inequality (18) holds true for y^*. Therefore, all $\mathsf{u}_{\mathrm{mach}}$-computations of M with input (u, x, y^*) halt and accept. It follows from property (b) above that all $\mathsf{u}_{\mathrm{mach}}$-evaluations of $\mathcal{C}$ at y^* return $f_{\mathcal{C}}(y^*) \geqslant 0$. Since this occurs, in particular,

for the exact evaluation, we deduce that $\mathcal{C} \in \mathsf{CircFeas}$. This proves (i) in Definition 10. Furthermore, from the fact that all $\mathsf{u}_{\mathsf{mach}}$-evaluations of $\mathcal{C}$ with input y^* yield $y \in S_{\mathcal{C}}$, we deduce that $\varrho_{\mathsf{eval}}(\mathcal{C}, y^*) \geqslant \mathsf{u}_{\mathsf{mach}}$, and therefore that

$$\varrho_{\mathsf{feas}}(\mathcal{C}) \geqslant \varrho_{\mathsf{eval}}(\mathcal{C}, y^*) \geqslant \mathsf{u}_{\mathsf{mach}}.$$

It follows that $\mu_{\mathsf{feas}}(\mathcal{C}) \leqslant 1/\mathsf{u}_{\mathsf{mach}} = 2^{k_0}$, and therefore that

$$\log \mu_{\mathsf{feas}}(\mathcal{C}) \leqslant k_0 = \mathsf{Prec}_M(R(\mathsf{size}_W(u, x))).$$

This bound is polynomial in $\mathsf{size}_W(u, x)$, as we wanted.

Case II: $(u, x) \notin W$. In this case, for every $y \in \mathbb{R}^m$, we have $(u, x, y) \notin B$ and $y \in Y$. Again, for all $y \in \mathbb{R}^m$, inequality (18) holds, and we deduce that all $\mathsf{u}_{\mathsf{mach}}$-computations of M reject (u, x, y). It follows from property (b) that the same result occurs for all $\mathsf{u}_{\mathsf{mach}}$-evaluations of $\mathcal{C}$ at y. And since these evaluations include the exact one, we deduce that $y \notin S_{\mathcal{C}}$. This is true for all $y \in \mathbb{R}^m$. Therefore, $\mathcal{C}$ is infeasible. This proves condition (i) in Definition 10.

To prove condition (ii), we use again the fact that, for all $y \in \mathbb{R}^m$, all $\mathsf{u}_{\mathsf{mach}}$-evaluations of $\mathcal{C}$ at y yield the same output ($y \notin S_{\mathcal{C}}$). This implies that, for all $y \in \mathbb{R}^m$,

$$\varrho_{\mathsf{eval}}(\mathcal{C}, y) \geqslant \mathsf{u}_{\mathsf{mach}} = 2^{-k_0},$$

and therefore that $\varrho_{\mathsf{feas}}(\mathcal{C}) \geqslant 2^{-k_0}$. Consequently, we have $\mu_{\mathsf{feas}}(\mathcal{C}) \leqslant 2^{k_0}$, and we conclude as in Case I. □

The following result is an immediate consequence of Proposition 10 and Theorem 1.

COROLLARY 1. *We have* $\mathsf{P}_{\mathsf{ro}} = \mathsf{NP}_{\mathsf{ro}}^{\mathsf{U}} \iff \mathsf{CircFeas} \in \mathsf{P}_{\mathsf{ro}}.$ □

REMARK 13. The construction of a circuit as in the proof of Theorem 3 has been done in many situations: for discrete computations it is the basis of Ladner's proof [40] of the P-completeness of the circuit evaluation problem (in this case, condition (c) in the proof is strengthened to require that the computation of $\mathcal{C}$ can be done with fewer resources, usually logarithmic space or polylogarithmic parallel time), in the BSS model it is the basis of a similar result over the reals [23], and even in the additive BSS model (where no multiplications are allowed) it is the basis of the proofs of some completeness results [18, 39]. In fact, the universality of this construction has prompted Poizat [44] to define P over an arbitrary structure as the class of sets decidable by families of circuits (with nodes appropriate for the structure) that can be constructed in polynomial time by a (standard) Turing machine.

OPEN QUESTION 1. The main open question in this development is, as one can expect, to decide whether $P_{ro} = NP_{ro}^{U}$. As usual, we believe this is not the case.

5.2. The class NP_{ro}^{B}.
To define this class we must first provide a formal definition of the notion of magnitude.

Given $k \in \mathbb{N}$, we consider the set F_k composed of 0 plus all numbers of the form (1) with

$$\beta = 2, \quad t = k + 1, \quad \text{and} \quad -2^k + 1 \leqslant e \leqslant 2^{(k+1)} - 1.$$

By construction (recall (2)), F_k is a floating-point system whose elements $y \neq 0$ satisfy

$$2^{-2^k} \leqslant |y| \leqslant 2^{2^{k+1}-1}(1 - 2^{-k-1}), \tag{19}$$

and these upper and lower bounds are attained for some elements in F_k. Recall that the real intervals defined by (19), along with {0}, define the range of F_k.

For $x \in \mathbb{R}$, we define the *magnitude* of x to be

$$\mathsf{mgt}(x) := \min\{k \geqslant 1 \mid x \in \mathsf{Range}(F_k)\},$$

and for $x \in \mathbb{R}^n$, $\mathsf{mgt}(x) := \max_{i \leqslant n} \mathsf{mgt}(x_i)$.

DEFINITION 12. A decision problem $(W, \boldsymbol{\mu}_W)$ belongs to NP_{ro}^{B} (*bounded nondeterministic roundoff polynomial cost*) when there exist a decision problem $(B, \boldsymbol{\mu}_B)$, a finite-precision BSS machine M deciding $(B, \boldsymbol{\mu}_B)$ in P_{ro}, and polynomials p, q, Q, such that, for $(u, x) \in \mathcal{I}$,

(i) if $(u, x) \in W$, then there exists $y^* \in \mathbb{R}^m$, such that $(u, x, y^*) \in B$, $\log \boldsymbol{\mu}_B(u, x, y^*) \leqslant Q(\log \boldsymbol{\mu}_W(u, x))$, and $\mathsf{mgt}(y^*) \leqslant q(\mathsf{size}_W(u, x))$; and

(ii) if $(u, x) \notin W$, then, for all $y \in \mathbb{R}^m$, we have $(u, x, y) \notin B$ and $\log \boldsymbol{\mu}_B(u, x, y) \leqslant Q(\log \boldsymbol{\mu}_W(u, x))$.

Here, $m = p(\mathsf{length}(u, x))$.

EXAMPLE 11. Instances for CircBFeas are algebraic circuits $\mathcal{C}$ (with input variables $Y_1, \ldots, Y_m$). The problem is to decide whether there exists $y \in \mathbb{R}^m$ such that $y \in S_{\mathcal{C}}$. What makes this problem different from CircFeas is its condition number. Here, we take

$$\mu_{\mathsf{Bfeas}}(\mathcal{C}) := \frac{1}{\varrho_{\mathsf{Bfeas}}(\mathcal{C})},$$

where

$$\varrho_{\mathsf{Bfeas}}(\mathcal{C}) := \begin{cases} \sup\limits_{y \in S_{\mathcal{C}}} \varrho_{\mathsf{eval}}(\mathcal{C}, y) 2^{-\mathsf{mgt}(y)} & \text{if } \mathcal{C} \text{ is feasible,} \\ \inf\limits_{y \in \mathbb{R}^m} \varrho_{\mathsf{eval}}(\mathcal{C}, y) & \text{otherwise.} \end{cases}$$

Again, in the feasible case, $\mu_{\mathsf{feas}}(\mathcal{C})$ is the condition of its best-conditioned solution, but here we take into account the magnitude of the solution so that feasible circuits having only large-magnitude solutions are poorly conditioned.

PROPOSITION 11. *We have* $\mathsf{CircBFeas} \in \mathsf{NP}_{\mathsf{ro}}^{\mathsf{B}}$.

Proof. The proof is as that of Proposition 11, except that now, in the feasible case, y^* is chosen to satisfy

$$\varrho_{\mathsf{feas}}(\mathcal{C}) = \varrho_{\mathsf{eval}}(\mathcal{C}, y^*) 2^{-\mathsf{mgt}(y^*)}.$$

The two first requirements in Definition 12(i) are shown as in that proposition, and for the third we have

$$\mathsf{size}(\mathcal{C}) \geqslant \log \mu_{\mathsf{feas}}(\mathcal{C}) = |\log \varrho_{\mathsf{feas}}(\mathcal{C})| = |\log \varrho_{\mathsf{eval}}(\mathcal{C}, y^*)| + \mathsf{mgt}(y^*) \geqslant \mathsf{mgt}(y^*),$$

which shows this requirement (with q being the identity). □

In the same manner, the following result is shown as Proposition 9.

PROPOSITION 12. *We have* $\mathsf{P}_{\mathsf{ro}} \subset \mathsf{NP}_{\mathsf{ro}}^{\mathsf{B}} \subset \mathsf{NP}_{\mathsf{ro}}^{\mathsf{U}}$. □

The notions of $\mathsf{NP}_{\mathsf{ro}}^{\mathsf{B}}$-hardness and $\mathsf{NP}_{\mathsf{ro}}^{\mathsf{B}}$-completeness are defined as in Definition 11. Our main result here is the following.

THEOREM 2. *The problem* $\mathsf{CircBFeas}$ *is* $\mathsf{NP}_{\mathsf{ro}}^{\mathsf{B}}$-*complete.*

Proof. The proof follows that of Theorem 2 almost word for word. The only difference is in the arguments for Case I. The point $y^* \in Y$ such that $(u, x, y^*) \in B$ now additionally satisfies that $\mathsf{mgt}(y^*) \leqslant q(\mathsf{size}(u, x))$. This does not affect the proof that all $\mathsf{u}_{\mathsf{mach}}$-evaluations of $\mathcal{C}$ at y^* return $f_{\mathcal{C}}(y^*) \geqslant 0$ and hence that $\mathcal{C}$ is feasible and that $\varrho_{\mathsf{eval}}(\mathcal{C}, y^*) \geqslant \mathsf{u}_{\mathsf{mach}}$. But from the latter we now deduce that

$$\varrho_{\mathsf{feas}}(\mathcal{C}, y^*) \geqslant \mathsf{u}_{\mathsf{mach}} 2^{-\mathsf{mgt}(y^*)} \geqslant \mathsf{u}_{\mathsf{mach}} 2^{-q(\mathsf{size}(u,x))}.$$

It follows that $\mu_{\mathsf{feas}}(\mathcal{C}) \leqslant 2^{-q(\mathsf{size}(u,x))}/\mathsf{u}_{\mathsf{mach}}$, and therefore that

$$\log \mu_{\mathsf{feas}}(\mathcal{C}) \leqslant q(\mathsf{size}(u, x)) + k_0 = q(\mathsf{size}(u, x)) + \mathsf{Prec}_M(R(\mathsf{size}_W(u, x))).$$

This bound is clearly polynomial in $\mathsf{size}_W(u, x)$, as we wanted. □

COROLLARY 2. *We have* $P_{ro} = NP_{ro}^B \iff CircBFeas \in P_{ro}$. ☐

5.3. The classes NP_{dir}^U and NP_{dir}^B. The definitions of NP_{dir}^U and NP_{dir}^B are the obvious variations of those for NP_{ro}^U and NP_{ro}^B.

DEFINITION 13. A decision problem (S, μ_S) belongs to NP_{dir}^U (*nondeterministic direct polynomial cost*) when there exist a decision problem (B, μ_B), a finite-precision BSS machine M deciding (B, μ_B) in P_{dir}, and polynomials p, Q, satisfying properties (i) and (ii) of Definition 9, and similarly for NP_{dir}^B (and Definition 12).

Clearly, $P_{dir} \subset NP_{dir}^B \subset NP_{dir}^U$. Also, our first examples of problems in NP_{dir}^U and NP_{dir}^B follow from a quick look at the proof of Proposition 8.

PROPOSITION 13. *We have* $CircFeas \in NP_{dir}^U$ *and* $CircBFeas \in NP_{dir}^B$. ☐

The fact that the arithmetic cost of P_{dir} machines depends only on the input's length allows for a simpler form of reduction.

DEFINITION 14. A P-*reduction* from (W, μ_W) to (S, μ_S) is a finite-precision machine $\overline{M}$ which, given an input $(u, x) \in \mathcal{I}$, performs a discrete computation and returns a pair $(v, z) \in \mathcal{I}$ satisfying the following:

(i) $(u, x) \in W \iff (v, z) \in S$;

(ii) $ar\_cost_{\overline{M}}(u, x)$ is polynomially bounded on $length(u, x)$; and

(iii) $size_S(v, z)$ is polynomial in $size_W(u, x)$.

If all of the above hold, we write $(W, \mu_W) \preceq_P (S, \mu_S)$.

PROPOSITION 14. *If* $(W, \mu_W) \preceq_P (S, \mu_S)$ *and* $(S, \mu_S) \in P_{dir}$, *then* $(W, \mu_W) \in P_{dir}$.

Proof. The proof is a simpler version of the proof of Proposition 14. ☐

Hardness and completeness with respect to P-reductions are defined as in Definition 11.

THEOREM 3. *The problems* CircFeas *and* CircBFeas *are* NP_{dir}^U-*complete and* NP_{dir}^B-*complete, respectively, with respect to* P-*reductions.*

Proof. Again, Proposition 13 shows that CircFeas is in $\mathsf{NP}^{\mathsf{U}}_{\mathsf{dir}}$, and we only need to prove the hardness. The proof is, essentially, contained in that of Theorem 1. Instead of a family of circuits parameterized by $k \in \mathbb{N}$, we deal with only one circuit whose depth is given by a polynomial in $\mathsf{length}(u, x)$. Property (iii) in Definition 14 is clear. Property (i) is shown word for word as in Theorem 1. Finally, for property (ii), the proof of this theorem shows that $\log \mu_{\mathsf{feas}}(\mathcal{C})$ is polynomially bounded in $\log \boldsymbol{\mu}(u, x)$. And since $\mathsf{length}(\mathcal{C})$ is polynomially bounded in $\mathsf{length}(u, x)$, it follows that $\mathsf{size}(\mathcal{C})$ is polynomially bounded in $\mathsf{size}(u, x)$.

A similar argument holds for CircBFeas and $\mathsf{NP}^{\mathsf{B}}_{\mathsf{dir}}$. □

The following result is an immediate consequence of Proposition 14 and Theorem 3.

COROLLARY 3. *We have* $\mathsf{P}_{\mathsf{dir}} = \mathsf{NP}^{\mathsf{U}}_{\mathsf{dir}} \iff \mathsf{CircFeas} \in \mathsf{P}_{\mathsf{dir}}$ *and* $\mathsf{P}_{\mathsf{dir}} = \mathsf{NP}^{\mathsf{B}}_{\mathsf{dir}}$ $\iff \mathsf{CircBFeas} \in \mathsf{P}_{\mathsf{dir}}$. □

OPEN QUESTION 2. Again, we leave open the truth of the equality $\mathsf{P}_{\mathsf{dir}} = \mathsf{NP}^{\mathsf{U}}_{\mathsf{dir}}$. And again, we believe that equality does not hold.

5.4. The classes $\mathsf{NP}^{\mathsf{U}}_{\mathsf{iter}}$ and $\mathsf{NP}^{\mathsf{B}}_{\mathsf{iter}}$.

Replacing $\mathsf{P}_{\mathsf{dir}}$ by $\mathsf{P}_{\mathsf{iter}}$ in Definition 13 yields the classes $\mathsf{NP}^{\mathsf{U}}_{\mathsf{iter}}$ and $\mathsf{NP}^{\mathsf{B}}_{\mathsf{iter}}$, and we clearly have $\mathsf{P}_{\mathsf{iter}} \subset \mathsf{NP}^{\mathsf{B}}_{\mathsf{iter}} \subset \mathsf{NP}^{\mathsf{U}}_{\mathsf{iter}}$. Similarities with the development in the preceding paragraph, however, appear to stop here, as we see as unlikely the existence of complete problems in either $\mathsf{NP}^{\mathsf{U}}_{\mathsf{iter}}$ or $\mathsf{NP}^{\mathsf{B}}_{\mathsf{iter}}$. This is so because the property characterizing problems in $\mathsf{P}_{\mathsf{iter}}$—the fact that any computation with a given $\mathsf{u}_{\mathsf{mach}}$ 'measures its error' to ensure that outputs in $\{\mathtt{Yes}, \mathtt{No}\}$ are correct—does not appear to be checkable in $\mathsf{P}_{\mathsf{iter}}$. That is, we do not know how to check in $\mathsf{P}_{\mathsf{iter}}$ that, given a circuit $\mathcal{C}$ outputting values in $\{\mathtt{Yes}, \mathtt{No}, \mathtt{Unsure}\}$, a point $x \in \mathbb{R}^n$, and a real $\mathsf{u}_{\mathsf{mach}}$, all $\mathsf{u}_{\mathsf{mach}}$-evaluations of $\mathcal{C}$ with input x return the same value.

The above notwithstanding, the classes $\mathsf{NP}^{\mathsf{U}}_{\mathsf{iter}}$ and $\mathsf{NP}^{\mathsf{B}}_{\mathsf{iter}}$ contain interesting problems such as feasibility for some systems of inequalities involving elementary functions (for example, exponential or trigonometric functions). The basic idea is that a candidate solution can be tested to satisfy the inequalities by computing approximations of the elementary functions involved, and that the error of these approximations can be bounded in terms of $\mathsf{u}_{\mathsf{mach}}$.

6. Deterministic bounds for nondeterministic cost

6.1. Exponential cost.

As we mentioned in the Introduction, a crucial property of NP or $\mathsf{NP}_{\mathbb{R}}$ is that they are subclasses of their corresponding

exponential time classes. In the case of the reals, it is even known that the inclusion $\mathrm{NP}_{\mathbb{R}} \subset \mathrm{EXP}_{\mathbb{R}}$ is strict [**17**]. The main result in this section shows a similar property for NP_{ro}^B. Before stating it, we define the general class EXP_{ro} of exponential cost, along with subclasses extending P_{dir} and P_{iter}.

DEFINITION 15. A decision problem (S, μ) belongs to EXP_{ro} (*roundoff exponential cost*) when there exists a finite-precision BSS machine M deciding S with cost (Arith, Prec) and such that

  (i) Prec is bounded by a exponential function; and

 (ii) the function $\mathsf{Arith}(\mathsf{length}(u, x), \mathsf{Prec}(\mathsf{size}(u, x)))$ is bounded by an exponential in $\mathsf{size}(u, x)$, for all $(u, x) \in \mathcal{I}$.

In both (i) and (ii) by exponential we mean a function of the kind $n \mapsto a^{n^d}$ for some $a > 1$ and $d > 0$.

REMARK 14. What we observed for Definition 5 in Remark 9(i) applies here *mutatis mutandis*. In particular, when Prec in Definition 15 is polynomially bounded, we say that (S, μ) can be solved with *polynomial precision*, and we write $(S, \mu) \in \mathrm{EXP}_{ro}^{[P]}$. It is important to note that in this case the dependence of Arith on k_{mach} may be exponential.

The classes EXP_{dir} and EXP_{iter} are defined with the obvious modifications to Definitions 6 and 8. In both cases, if the required precision satisfies $\mathsf{k}_{mach} = (\mathsf{size}(u, x))^{\mathcal{O}(1)}$, we say that the problem can be solved with *polynomial precision*.

EXAMPLE 7 (CONTINUED). The main result in [**22**] shows that the problem mentioned in Example 7 (feasibility of real homogeneous polynomial systems) is in EXP_{iter}.

PROPOSITION 15. *The inclusion* $\mathrm{EXP}_{ro}^{[P]} \subset \mathrm{EXP}_{ro}$ *is strict. The class* EXP_{dir} *is not included in* $\mathrm{EXP}_{ro}^{[P]}$.

Proof. Proposition 1, with $T(n) = 2^n$, P_1 a polynomial function, and $P_2(n) = 2^n$, proves the first statement. A closer look at its proof reveals that the machine deciding the set (B, μ) there, with the functions above, is in EXP_{dir}. The second statement follows.   □

The following results are proven analogously to Propositions 10 and 14.

PROPOSITION 16. *If* $(W, \boldsymbol{\mu}_W) \preceq_{\mathsf{ro}} (S, \boldsymbol{\mu}_S)$ *and* $(S, \boldsymbol{\mu}_S) \in \mathrm{EXP}_{\mathsf{ro}}$, *then* $(W, \boldsymbol{\mu}_W) \in$ $\mathrm{EXP}_{\mathsf{ro}}$. *A similar statement holds for the class* $\mathrm{EXP}_{\mathsf{ro}}^{[\mathsf{P}]}$. $\square$

PROPOSITION 17. *If* $(W, \boldsymbol{\mu}_W) \preceq_{\mathsf{P}} (S, \boldsymbol{\mu}_S)$ *and* $(S, \boldsymbol{\mu}_S) \in \mathrm{EXP}_{\mathsf{dir}}$, *then* $(W, \boldsymbol{\mu}_W) \in$ $\mathrm{EXP}_{\mathsf{dir}}$. $\square$

6.2. Testing grids. The fact that finite-precision computations need to be robust (that is, they need to result in the same outcome) when the precision is sufficiently large allows us to reduce the search for solutions to the circuit feasibility problem to a search over the points of a sufficiently fine grid. Here, we describe these grids and the cost and accuracy of constructing them.

We begin by recalling the floating-point system F_k defined at the beginning of Section 5.1 and observing that its unit roundoff is $u_k := 2^{-(k+1)}$.

A point in F_k can be given by $2k + 2$ bits ($k + 1$ to write down e along with $k + 1$ more to write $d_1, \ldots, d_t$). We will denote by y_v the element in F_k associated to a point $v \in \{0, 1\}^{2k+2}$.

Consider the set $\mathscr{G}_k := F_k^n \subset \mathbb{R}^n$. For any $\bar{v} = (v_1, \ldots, v_n) \in \{0, 1\}^{n(2k+2)}$, we write $\overline{y_v} = (y_{v_1}, \ldots, y_{v_n}) \in \mathscr{G}_k$.

PROPOSITION 18. *Given* $v \in \{0, 1\}^{(2k+2)}$, *we can compute* $y_v \in \mathbb{R}$ *with* $\mathcal{O}(k)$ *arithmetic operations. If* $k \geqslant 5$ *and* $\mathsf{u}_{\mathsf{mach}} \leqslant 2^{-2k}$, *the computed quantity* $\tilde{y}_v$ *satisfies*

$$\tilde{y}_v = y_v(1 + \theta_{2k+2}).$$

Furthermore, for all $\varepsilon > 0$, *we can ensure that*

$$\tilde{y}_v = y_v(1 + \delta) \quad \text{with } |\delta| \leqslant \varepsilon$$

if, in addition, $2^k < \varepsilon/2$.

Proof. To compute y_v we need to compute both 2^e and $m = 0.d_1 d_2 \ldots d_t$. Assume, without loss of generality, that $e > 0$. Since $e < 2^{k+1}$, it has a binary decomposition $e = \sum_{j=0}^{k} b_j 2^j$ (here, $b_j \in \{0, 1\}$). Hence

$$2^e = \prod_{b_j=1} 2^{2^j}. \tag{20}$$

Since $2^{2^{j+1}} = 2^{2^j} \cdot 2^{2^j}$, we can compute the collection of all the 2^{2^j} with k multiplications, and with at most k additional multiplications we obtain 2^e. The cost of computing 2^e is therefore $\mathcal{O}(k)$.

Also, the mantissa

$$m = 0.d_1 d_2 \dots d_t = \sum_{i=1}^{t} d_i 2^{-i}$$

can be computed with at most $2t$ operations. For $i > 1$, each 2^{-i} is obtained with a single division from $2^{-(i-1)}$ and is added to the partial sum if $d_i = 1$. It follows that the arithmetic cost of computing y_v is $\mathcal{O}(k)$.

We now consider precision issues.

Recall the computation of the mantissa. The quantity 2^{-i} is obtained from $2^{-(i-1)}$ with one division. It is easy to show by induction (using Lemma 1) that the computed quantity $\widetilde{2^{-i}}$ is of the form $2^{-i}(1 + \theta_{2i-1})$. From this bound, using that

$$\left(\left(\sum_{i=1}^{j} d_i 2^{-i} \right)(1 + \theta_{2(j+1)-1}) + d_{j+1} 2^{-(j+1)}(1 + \theta_{2(j+1)-1}) \right)(1 + \theta_1)$$

$$= \left(\left(\sum_{i=1}^{j+1} d_i 2^{-i} \right)(1 + \theta_{2(j+1)-1}) \right)(1 + \theta_1) = \sum_{i=1}^{j+1} d_i 2^{-i}(1 + \theta_{2(j+2)-1}),$$

a further induction argument shows that the computed mantissa is of the form

$$\tilde{m} = \left(\sum_{i=i}^{k+1} d_i 2^{-i} \right)(1 + \theta_{2(k+2)-1}).$$

Recall also the computation of y_v. We compute each of the powers 2^{2^j} with j multiplications, and it is easy to see, using Lemma 1, that we obtain

$$\widetilde{2^{2^j}} = 2^{2^j}(1 + \theta_{2^j - 1}).$$

Continuing using this lemma, we compute 2^e using (20) (multiply the powers with smaller exponent first), and obtain

$$\widetilde{2^e} = 2^e(1 + \theta_{2^{k+1}-2}).$$

An extra multiplication with $\tilde{m}$ yields

$$\tilde{y}_v = y_v(1 + \theta_{2^{k+1}+2k+2}) = y_v(1 + \theta_{2^{k+2}}),$$

as claimed. Assume finally that $2^k < \varepsilon/2$. Then, $\mathsf{u}_{mach} \leqslant 2^{-2k} < \varepsilon^2/4$ and

$$|\theta_{2^{k+2}}| \leqslant \frac{2^{k+2}\mathsf{u}_{mach}}{1 - 2^{k+2}\mathsf{u}_{mach}} \leqslant \frac{2^k \varepsilon^2}{1 - 2^k \varepsilon^2} \leqslant \frac{\varepsilon}{2 - \varepsilon} \leqslant \varepsilon,$$

which finishes the proof. $\qquad\square$

6.3. $\mathsf{NP}_{\mathsf{ro}}^{\mathsf{B}} \subset \mathsf{EXP}_{\mathsf{ro}}$. We can now show a key membership result for $\mathsf{EXP}_{\mathsf{ro}}^{[\mathsf{P}]}$.

THEOREM 4. *We have* CircBFeas $\in \mathsf{EXP}_{\mathsf{ro}}^{[\mathsf{P}]}$.

Proof. The general idea is to evaluate the circuit on the points of a canonical grid.

We consider the machine $\mathcal{M}$ given by the following code (here, n is the number of input gates of the circuit $\mathcal{C}$):

```
input C
compute n
set k := ⌊ k_mach / 2 ⌋
for all v̄ ∈ {0, 1}^{n(2k+2)} do
  compute ȳ_v
  evaluate f_C(ȳ_v)
accept if for one of these evaluations
  we obtain f_C(y) ⩾ 0
```

The arithmetic cost of $\mathcal{M}$ is easily bounded. Since F_k contains $2^{\mathcal{O}(\mathsf{k}_{\mathsf{mach}})}$ numbers, the grid $\mathscr{G}_k$ contains $2^{\mathcal{O}(n\mathsf{k}_{\mathsf{mach}})}$ points. To produce each of these points takes $\mathcal{O}(n\mathsf{k}_{\mathsf{mach}})$ arithmetic operations (Proposition 18), and to evaluate $\mathcal{C}$ at each of them an additional $\mathcal{O}(\mathsf{length}(\mathcal{C}))$ operations. It follows that the arithmetic cost of $\mathcal{M}$ is bounded by $2^{\mathcal{O}(\mathsf{length}(\mathcal{C})\mathsf{k}_{\mathsf{mach}})}(\mathsf{length}(\mathcal{C})\mathsf{k}_{\mathsf{mach}})^{\mathcal{O}(1)}$, a bound we can write in the form $2^{\mathcal{O}(\mathsf{length}(\mathcal{C})\mathsf{k}_{\mathsf{mach}})}$, as we wanted.

To prove the bound on the precision needed, we assume that $\mu_{\mathsf{Bfeas}}(\mathcal{C}) < \infty$ (and hence that the same holds for $\mathsf{size}(\mathcal{C})$), and take $\mathsf{u}_{\mathsf{mach}} := 1/16(\mu_{\mathsf{Bfeas}}(\mathcal{C}))^2$. We want to see that when M works with this precision it correctly decides whether its input $\mathcal{C}$ is in CircBFeas. Note that for $\mathsf{k}_{\mathsf{mach}}$ as above the unit roundoff of F_k satisfies $u_k = 1/4\mu_{\mathsf{Bfeas}}(\mathcal{C})$.

We divide by cases.

Assume first that $\mathcal{C}$ is feasible. In this case, there exist points $x \in S_{\mathcal{C}}$. Let x^* be one such point satisfying

$$\varrho_{\mathsf{Bfeas}}(\mathcal{C}) = \varrho_{\mathsf{eval}}(\mathcal{C}, x^*)2^{-\mathsf{mgt}(x^*)} > 0. \tag{21}$$

This implies that $2^{\mathsf{mgt}(x^*)} < \mu_{\mathsf{Bfeas}}(\mathcal{C}) = 1/4\sqrt{\mathsf{u}_{\mathsf{mach}}}$, and therefore that $\mathsf{mgt}(x^*) < \frac{1}{2}\mathsf{k}_{\mathsf{mach}} - 2 < k$. That is, x^* belongs to $\mathsf{Range}(F_k)$.

This implies the existence of $\bar{v} \in \{0, 1\}^{n(2k+2)}$ such that the corresponding $\overline{y_v} = (y_1, \ldots, y_n) \in \mathscr{G}_k$ satisfies, for $i = 1, \ldots, n$,

$$y_i = x_i^*(1 + \delta) \quad \text{with } |\delta| < u_k = \frac{1}{4\mu_{\mathsf{Bfeas}}(\mathcal{C})}. \tag{22}$$

Since $u_{mach} \leqslant 2^{-2k}$ and $2^k \leqslant 1/4(\mu_{Bfeas}(\mathcal{C}))$, Proposition 18 ensures that the computation of $\overline{y_v}$ done by $\mathcal{M}$ returns a point $\tilde{y}$ satisfying

$$\tilde{y}_i = y_i(1 + \delta) \quad \text{with } |\delta| \leqslant \frac{1}{2\mu_{Bfeas}(\mathcal{C})}. \tag{23}$$

From this inequality, together with (22), we deduce that

$$\tilde{y}_i = x_i^*(1 + \delta) \quad \text{with } |\delta| \leqslant \varrho_{Bfeas}(\mathcal{C}) < \varrho_{eval}(\mathcal{C}, x^*), \tag{24}$$

the last inequality due to (21).

Consider the computation of $\mathcal{M}$ corresponding to the point $\bar{v}$. It first produces the approximation $\tilde{y}$ of $\overline{y_v}$ and then evaluates $f_{\mathcal{C}}(\tilde{y})$ with a u_{mach}-computation γ.

We can associate to this computation the evaluation of $\mathcal{C}$ at x^* that first approximates x^* by $\tilde{y}$ and then approximates $f_{\mathcal{C}}(\tilde{y})$ with γ. We claim that this is a $\varrho_{eval}(\mathcal{C}, x^*)$-evaluation. Indeed, the relative errors in the first process are bounded by $\varrho_{eval}(\mathcal{C}, x^*)$ (because of (24)), and the second process is a $\varrho_{eval}(\mathcal{C}, x^*)$-evaluation of $\mathcal{C}$ at $\tilde{y}$ (we use $u_{mach} < \varrho_{eval}(\mathcal{C}, x^*)$ and Proposition 4). Putting these bounds together, the claim follows.

The definition of $\varrho_{eval}(\mathcal{C}, x^*)$ implies that this evaluation returns $f_{\mathcal{C}}(x^*) \geqslant 0$. But this implies that the u_{mach}-computation of $\mathcal{M}$ above also yields $f_{\mathcal{C}}(x^*) \geqslant 0$. This in turn implies that $\mathcal{M}$ accepts $\mathcal{C}$.

Assume now that $\mathcal{C}$ is not feasible. In this case, for all $x \in \mathbb{R}^n$, $\varrho_{eval}(\mathcal{C}, x) \geqslant \varrho_{Bfeas}(\mathcal{C}) \geqslant u_{mach}$. This implies that, for all $\bar{v} \in \{0, 1\}^{n(2k+2)}$, the u_{mach}-evaluation of $\mathcal{C}$ at the computed approximation $\tilde{y}$ of $\overline{y_v}$ yields $f_{\mathcal{C}}(\tilde{y}) < 0$. This implies that $\mathcal{M}$ rejects $\mathcal{C}$.

We can now conclude, since

$$k_{mach} = \left\lceil \log \frac{1}{u_{mach}} \right\rceil = \lceil \log 16(\mu_{Bfeas}(\mathcal{C}))^2 \rceil \leqslant 2\,\text{size}(\mathcal{C}) + \mathcal{O}(1).$$

This linear bound is well within the polynomial growth required in the definition of $\text{EXP}_{ro}^{[P]}$. $\qquad\qquad\square$

Essential to the fact that the largest precision needed is linear in $\text{size}(\mathcal{C})$ is the circumstance that the values of $f_{\mathcal{C}}(y)$ are computed by $\mathcal{M}$ *independently*, for all the points y in $\mathscr{G}_k$. This fact would also be central in the proof that NP_{ro}^B is actually included in the subclass PAR_{ro} of EXP_{ro} of problems decidable in variable-precision parallel polynomial time. But we do not deal with parallelism in this paper.

COROLLARY 4. *We have* $\text{NP}_{ro}^B \subset \text{EXP}_{ro}^{[P]}$, *and the inclusion is strict.*

Proof. The first statement readily follows from Proposition 16. The second follows from Proposition 15. □

OPEN QUESTION 3. A major open question in this section is whether $\mathsf{NP}_{\mathsf{ro}}^{\mathsf{U}}$ is included in $\mathsf{EXP}_{\mathsf{ro}}$ or, equivalently, whether CircFeas belongs to $\mathsf{EXP}_{\mathsf{ro}}$. We conjecture that this question has a positive answer.

OPEN QUESTION 4. The question above can be strengthened. We know that CircFeas $\in \mathsf{NP}_{\mathsf{dir}}^{\mathsf{U}}$ and that it is actually complete in this class for P-reductions. This raises the question of whether CircFeas $\in \mathsf{EXP}_{\mathsf{dir}}$ (a membership that would imply $\mathsf{NP}_{\mathsf{dir}}^{\mathsf{U}} \subset \mathsf{EXP}_{\mathsf{dir}}$).

These last open questions are both related to a finite-precision analysis of existing algorithms. The feasibility of an algebraic circuit $\mathcal{C}$ reduces to the feasibility of a Boolean combination of the form

$$\bigvee_{i=1}^{K} \bigwedge_{j=1}^{q_i} \varphi_{ij}(x) *_{ij} 0,$$

where $*_{ij} \in \{\geqslant, >\}$, q_i is polynomially bounded in $\mathsf{length}(\mathcal{C})$, K is exponentially bounded in $\mathsf{length}(\mathcal{C})$, and φ_{ij} are straight-line programs of length polynomially bounded (and hence degree exponentially bounded) in $\mathsf{length}(\mathcal{C})$. There exist a number of algorithms showing that the feasibility of such a system can be done, under the assumption of infinite precision, in time exponential in $\mathsf{length}(\mathcal{C})$ [33, 45]. Hence, an accuracy analysis proving correctness of any of these algorithms under a $\mathsf{k}_{\mathsf{mach}}$ bounded by an exponential function of $\mathsf{length}(\mathcal{C})$ and $\mu_{\mathsf{feas}}(\mathcal{C})$ would imply membership of CircFeas to $\mathsf{NP}_{\mathsf{dir}}^{\mathsf{U}}$ (and hence the inclusions $\mathsf{NP}_{\mathsf{dir}}^{\mathsf{U}} \subset \mathsf{EXP}_{\mathsf{dir}}$ and $\mathsf{NP}_{\mathsf{ro}}^{\mathsf{U}} \subset \mathsf{EXP}_{\mathsf{ro}}$). But these finite-precision analyses are, as of today, not at hand.

7. Final remarks

7.1. Average complexity. In Section 3, we defined decision problems as pairs of subsets and condition numbers, and in doing so, we put essentially no requirement on what a condition number function is. Remark 5 in that section elaborated a little on this generality. We now elaborate more, focusing on a viewpoint that has accompanied the development of condition numbers practically since the origins of the notion of condition.

These origins can be traced back to the condition number $\kappa(A)$ (in Example 1) introduced by Turing [59] and von Neumann and Goldstine [60] to understand the loss of precision in the solution of systems of linear equations. For a system

$Ax = b$, this is the number of correct figures in the entries of A and b minus this number for the computed solution $\tilde{x}$. It follows from (9) that this number is about $3 \log n + \log \kappa(A)$. The fact, however, that $\kappa(A)$ is not known *a priori* (and computing it from A is a process with the same shortcomings as those of solving $Ax = b$) prompted von Neumann and Goldstine to propose studying $\log \kappa(A)$ as a random variable. Their paper [61] provided some results in this direction, assuming A to be Gaussian (that is, having its entries independent and normally distributed). The state of the art for this assumption was shown by Edelman [27] who proved that, for Gaussian real or complex $n \times n$ matrices, we have

$$\mathbb{E} \log \kappa(A) = \log n + C + o(1), \tag{25}$$

where $C = 1.537$ in the real case and $C = 0.982$ in the complex case. As a consequence, the loss of precision in the solution of $Ax = b$ is, on average, of the order of $4 \log n$. A different algorithm, analyzed in terms of a different condition number, would produce, for the same underlying probability measure, a (likely) different dependence on n, and the comparison of these dependences translates into a comparison of the two algorithms' efficiencies.

In 1997, Smale advocated studying the complexity of an algorithm in a similar manner. One would first derive complexity bounds in terms of input length and a condition number and then eliminate the latter by endowing the set of inputs of length n with a probability measure (usually the standard Gaussian) and bounding the expected value of this condition number (or of its logarithm, if appropriate) by a function of n. One thus obtains *average complexity bounds* depending on the input length only. A convenient feature of this procedure is the fact that one is free to choose the condition number. Different analyses for a computational problem may rely on different condition numbers (sometimes for the same algorithm!), and algorithms may be compared by the average complexity bounds shown on these analyses. A case at hand is the polyhedral feasibility problem described in Example 5. We mentioned in this example that algorithmic solutions to this problem have been proposed whose analyses are based on a variety of condition numbers (in addition to $\mathscr{C}(A)$ defined there). Expected values for (the log of) these condition numbers allow for a comparison of the efficiency of these algorithmic solutions. We mention here that in [11] it is proved that for Gaussian matrices $A \in \mathbb{R}^{m \times n}$

$$\mathbb{E} \log \mathscr{C}(A) = 2 \log(m + 1) + 3.31. \tag{26}$$

This means that the contribution of the condition in the complexity bound mentioned in Example 5 (continued) is dominated, on average, by that in terms of the dimension of A. The average complexity of the algorithm is $\mathcal{O}(n^{3.5} \log n)$, and similarly for the average of the required k_{mach}, which turns to be $\mathcal{O}(\log n)$.

Probabilistic analysis is therefore a way to reduce the arbitrariness in the choice of condition numbers. We will not go deeper into these ideas but point instead to the recent monograph [10], where condition numbers are the central character and their probabilistic analysis a recurrent theme.

7.2. Recursive analysis. We mentioned in Section 1.3 a stream of research that approaches numerical computations based on Turing machines. Detailed expositions of this research are in [38, 62]. It is worth pointing out the differences with the theory we have developed in this paper. A first major difference is made clear early on [38, page 4], where the author mentions that, despite the fact that 'the floating-point model is the usual model of computation in numerical analysis', it is not the model used in the theory. Indeed, a look at this theory reveals it to be closer to fixed-point arithmetic. A second major difference is the absence of the notion of conditioning. Again, an examination of [38] reveals that this notion is absent in the book.

None of these differences is in detriment of an approach that has undeniably provided understanding of some numerical computations. But they both row against the agenda cited at the end of Section 1.1, namely 'to incorporate roundoff error [and] condition numbers into [the Blum–Shub–Smale] development'.

Our theory attempts to stay close to the design and analysis of algorithms in current numerical computations (as described in [35]). It is also close to discrete computation in that our finite-precision computations can be simulated by Turing machines. In these simulations real data, both machine constants and input, are replaced by floating-point numbers (with unrestricted exponents). But these simulations do not preserve polynomial time because of the presence of unrestricted exponents (for every polynomial bound, just reading a real number with a sufficiently large magnitude will have a cost beyond this bound).

Our choices for machine model, size, and cost were taken, as wished in [6], 'to bring machines over $\mathbb{R}$ closer to the subject of numerical analysis'. We actually strived to bring them closer to the literature in numerical analysis (where unrestricted exponents dominate).

We nonetheless mention that a modification of our theory that makes it closer to the discrete framework is possible. One defines the size of $(u, x) \in \mathcal{I}$ by

$$\mathsf{size}(u, x) := \mathsf{length}(u, x) + \mathsf{mgt}(x) + \lceil \log_2 \boldsymbol{\mu}(u, x) \rceil.$$

Then, the basic version of polynomial time cost is obtained by writing

$$\mathsf{Arith}(\mathsf{length}(u, x) + \mathsf{mgt}(x), \mathsf{Prec}(\mathsf{size}(u, x)))$$

in Definition 5(ii). We have the feeling, though, that this theory would be more removed from numerical analysts' thinking. We have therefore not pursued this avenue.

Acknowledgements

References

[1] E. Allender, P. Bürgisser, J. Kjeldgaard-Pedersen and P. B. Miltersen, 'On the complexity of numerical analysis', *SIAM J. Comput.* **38**(5) (2008/09), 1987–2006.

[2] C. Beltrán and L. M. Pardo, 'Smale's 17th problem: average polynomial time to compute affine and projective solutions', *J. Amer. Math. Soc.* **22**(2) (2009), 363–385.

[3] C. Beltrán and L. M. Pardo, 'Fast linear homotopy to find approximate zeros of polynomial systems', *Found. Comput. Math.* **11**(1) (2011), 95–129.

[4] L. Blum, 'Lectures on a theory of computation and complexity over the reals (or an arbitrary ring)', in: *Lectures in the Sciences of Complexity II* (ed. E. Jen) (Addison-Wesley, Redwood City, CA, 1990), 1–47.

[5] L. Blum, F. Cucker, M. Shub and S. Smale, *Complexity and Real Computation* (Springer, New York, 1998).

[6] L. Blum, M. Shub and S. Smale, 'On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines', *Bull. Amer. Math. Soc.* **21** (1989), 1–46.

[7] M. Braverman and S. Cook, 'Computing over the reals: foundations for scientific computing', *Notices Amer. Math. Soc.* **53**(3) (2006), 318–329.

[8] P. Bürgisser and F. Cucker, 'Exotic quantifiers, complexity classes, and complete problems', *Found. Comput. Math.* **9** (2009), 135–170.

[9] P. Bürgisser and F. Cucker, 'On a problem posed by Steve Smale', *Ann. Math.* **174** (2011), 1785–1836.

[10] P. Bürgisser and F. Cucker, *Condition*, *Grundlehren der mathematischen Wissenschaften,* Vol. 349, (Springer, Berlin, 2013).

[11] P. Bürgisser, F. Cucker and M. Lotz, 'Coverage processes on spheres and condition numbers for linear programming', *Ann. Probab.* **38** (2010), 570–604.

[12] D. Cheung and F. Cucker, 'A new condition number for linear programming', *Math. Program.* **91** (2001), 163–174.

[13] D. Cheung, F. Cucker and Y. Ye, 'Linear programming and condition numbers under the real number computation model', in: *Handbook of Numerical Analysis,* Vol. XI (eds. Ph. Ciarlet and F. Cucker) (North-Holland, Amsterdam, 2003), 141–207.

[14] A. Cobham, 'The intrinsic computational difficulty of problems', in: *International Congress for Logic, Methodology, and the Philosophy of Science* (ed. Y. Bar-Hillel) (North-Holland, Amsterdam, 1964), 24–30.

[15] S. Cook, 'The complexity of theorem proving procedures', in: *3rd Annual ACM Symposium on the Theory of Computing* (Assoc. Comput. Mach., New York, 1971), 151–158.

[16] S. Cook, 'The P versus NP problem', in: *The Millennium Prize Problems* (Clay Math. Inst., Cambridge, MA, 2006), 87–104.

[17] F. Cucker, '$P_\mathbb{R} \neq NC_\mathbb{R}$', *J. Complexity* **8** (1992), 230–238.

[18] F. Cucker and P. Koiran, 'Computing over the reals with addition and order: higher complexity classes', *J. Complexity* **11** (1995), 358–376.

[19] F. Cucker, T. Krick, G. Malajovich and M. Wschebor, 'A numerical algorithm for zero counting. I. Complexity and accuracy', *J. Complexity* **24** (2008), 582–605.

[20] F. Cucker, T. Krick, G. Malajovich and M. Wschebor, 'A numerical algorithm for zero counting. II. Distance to ill-posedness and smoothed analysis', *J. Fixed Point Theory Appl.* **6** (2009), 285–294.

[21] F. Cucker and J. Peña, 'A primal-dual algorithm for solving polyhedral conic systems with a finite-precision machine', *SIAM J. Optim.* **12** (2002), 522–554.

[22] F. Cucker and S. Smale, 'Complexity estimates depending on condition and round-off error', *J. ACM* **46** (1999), 113–184.

[23] F. Cucker and A. Torrecillas, 'Two P-complete problems in the theory of the reals', *J. Complexity* **8** (1992), 454–466.

[24] J. Demmel, 'On condition numbers and the distance to the nearest ill-posed problem', *Numer. Math.* **51** (1987), 251–289.

[25] J. W. Demmel, *Applied Numerical Linear Algebra* (SIAM, Philadelphia, PA, 1997).

[26] C. Eckart and G. Young, 'The approximation of one matrix by another of lower rank', *Psychometrika* **1** (1936), 211–218.

[27] A. Edelman, 'Eigenvalues and condition numbers of random matrices', *SIAM J. Matrix Anal. Appl.* **9** (1988), 543–556.

[28] J. Edmonds, 'Paths, trees, and flowers', *Canad. J. Math.* **17** (1965), 449–467.

[29] J.-L. Goffin, 'The relaxation method for solving systems of linear inequalities', *Math. Oper. Res.* **5** (1980), 388–414.

[30] O. Goldreich, *Computational Complexity* (Cambridge University Press, Cambridge, 2008), A conceptual perspective.

[31] J. Hartmanis, P. L. Lewis and R. E. Stearns, 'Hierarchies of memory-limited computations', in: *6th IEEE Symposium on Switching Circuit Theory and Logic Design* (IEEE Comput. Soc., Long Beach, CA, 1965), 179–190.

[32] J. Hartmanis and R. E. Stearns, 'On the computational complexity of algorithms', *Trans. Amer. Math. Soc.* **117** (1965), 285–306.

[33] J. Heintz, M.-F. Roy and P. Solerno, 'Sur la complexité du principe de Tarski–Seidenberg', *Bull. Soc. Math. France* **118** (1990), 101–126.

[34] M. R. Hestenes and E. Stiefel, 'Methods of conjugate gradients for solving linear systems', *J. Research Nat. Bur. Standards* **49**(1953) (1952), 409–436.

[35] N. Higham, *Accuracy and Stability of Numerical Algorithms* (SIAM, Philadelphia, PA, 1996).

[36] S. Homer and A. L. Selman, *Computability and complexity theory*, second edn, *Texts in Computer Science*, (Springer, New York, 2011).

[37] R. M. Karp, 'Reducibility among combinatorial problems', in: *Complexity of Computer Computations* (eds. R. Miller and J. Thatcher) (Plenum Press, 1972), 85–103.

[38] K.-I. Ko, *Complexity Theory of Real Functions*, *Progress in Theoretical Computer Science*, (Birkhäuser Boston, Inc., Boston, MA, 1991).

[39] P. Koiran, 'Computing over the reals with addition and order', *Theor. Comput. Sci.* **133** (1994), 35–47.

[40] R. E. Ladner, 'The circuit value problem is log space complete for $\mathbb{P}$', *SIGACT News* **7** (1975), 18–20.

[41] L. Levin, 'Universal sequential search problems', *Probl. Pered. Inform.* **IX 3** (1973), 265–266. (in Russian), (English translation in *Problems of Information Trans.* **9**, 3; corrected translation in [**58**]).

[42] W. Miller, 'Computational complexity and numerical stability', *SIAM J. Comput.* **4** (1975), 97–107.

[43] C. H. Papadimitriou, *Computational Complexity* (Addison-Wesley, Redwood City, CA, 1994).

[44] B. Poizat, *Les Petits Cailloux* (Aléa, Paris, 1995).

[45] J. Renegar, 'On the computational complexity and geometry of the first-order theory of the reals. Part I', *J. Symbolic Comput.* **13** (1992), 255–299.

[46] J. Renegar, 'Is it possible to know a problem instance is ill-posed?', *J. Complexity* **10** (1994), 1–56.

[47] J. Renegar, 'Some perturbation theory for linear programming', *Math. Program.* **65** (1994), 73–91.

[48] J. Renegar, 'Incorporating condition measures into the complexity theory of linear programming', *SIAM J. Optim.* **5** (1995), 506–524.

[49] J. Renegar, 'Linear programming, complexity theory and elementary functional analysis', *Math. Program.* **70** (1995), 279–351.

[50] M. Shub and S. Smale, 'Complexity of Bézout's Theorem I: geometric aspects', *J. Amer. Math. Soc.* **6** (1993), 459–501.

[51] M. Shub and S. Smale, 'Complexity of Bézout's Theorem II: volumes and probabilities', in: *Computational Algebraic Geometry*, (eds. F. Eyssette and A. Galligo) Progress in Mathematics, Vol. 109 (Birkhäuser, Basel, 1993), 267–285.

[52] M. Shub and S. Smale, 'Complexity of Bézout's Theorem III: condition number and packing', *J. Complexity* **9** (1993), 4–14.

[53] M. Shub and S. Smale, 'Complexity of Bézout's Theorem IV: probability of success; extensions', *SIAM J. Numer. Anal.* **33** (1996), 128–148.

[54] M. Shub and S. Smale, 'Complexity of Bézout's Theorem V: polynomial time', *Theor. Comput. Sci.* **133** (1994), 141–164.

[55] S. Smale, 'Some remarks on the foundations of numerical analysis', *SIAM Rev.* **32** (1990), 211–220.

[56] S. Smale, 'Complexity theory and numerical analysis', in: *Acta Numerica* (ed. A. Iserles) (Cambridge University Press, Cambridge, UK, 1997), 523–551.

[57] S. Smale, 'Mathematical problems for the next century', in: *Mathematics: Frontiers and Perspectives* (eds. V. Arnold, M. Atiyah, P. Lax and B. Mazur) (American Mathematical Society, Providence, RI, 2000), 271–294.

[58] B. A. Trakhtenbrot, 'A survey of russian approaches to perebor (brute-force search) algorithms', *Ann. Hist. Comput.* **6** (1984), 384–400.

[59] A. M. Turing, 'Rounding-off errors in matrix processes', *Quart. J. Mech. Appl. Math.* **1** (1948), 287–308.

[60] J. von Neumann and H. H. Goldstine, 'Numerical inverting matrices of high order', *Bull. Amer. Math. Soc.* **53** (1947), 1021–1099.

[61] J. von Neumann and H. H. Goldstine, 'Numerical inverting matrices of high order, II', *Proc. Amer. Math. Soc.* **2** (1951), 188–202.

[62] K. Weihrauch, *Computable Analysis*, *Texts in Theoretical Computer Science. An EATCS Series*, (Springer, Berlin, 2000).

[63] J. Wilkinson, 'Some comments from a numerical analyst', *J. ACM* **18** (1971), 137–147.