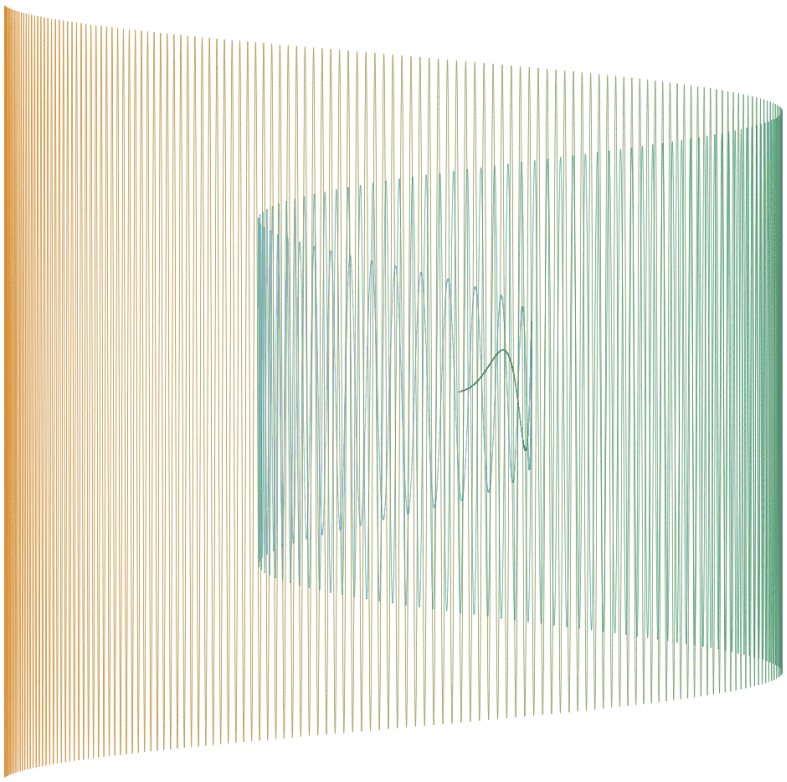
Komputerowe rozwiązywanie problemów matematycznych — English edition

Wydział Matematyki i Informatyki UAM · winter semester 2026/27

Richardson's theorem

A complete proof step by step — with definitions, appendices and problems



The curve $t \mapsto (t \sin t,\ t \sin t^3)$ for $0 \le t \le 11$; the image of the whole half-line is dense in the plane (Theorem 4.4).

Bartosz Naskręcki

Version of October 26, 2026

About this document

Printable version of the chapter "Richardson's theorem" from the "Deep dives" part of the English edition of the course book *Komputerowe rozwiązywanie problemów matematycznych* (WMI UAM, winter semester 2026/27; the course is taught in Polish). Online version: https://bnaskrecki.faculty.wmi.amu.edu.pl/math-comp/en/book/richardson/richardson.html

The text, the numbering of definitions, lemmas and theorems, the formulas and the results of computations are the same as in the online version: the document is assembled automatically from the same sources, and the results and plots come from the notebooks executed when the book is built. The Python code can be run on your own — the notebooks `r2_dowod.ipynb` and `r2_przyklad.ipynb` can be downloaded from the pages of the chapters "The complete proof, step by step" and "Example: an expression whose sign cannot be seen".

As of October 26, 2026. © 2026 Bartosz Naskręcki (co-created with Claude). Translated from the Polish original.

Notation

	theorems, lemmas, corollaries and propositions
	definitions, remarks and examples
	hints, comments on the structure of the proof, claims
	warnings: places where it is easy to go wrong
	references to the literature
`output`	result of a computation (text with a gray bar on the left)
★ … ★★★	difficulty of a problem: routine … challenge; 💻 — the problem requires computation

Contents

Richardson's theorem: why a computer cannot recognize every zero **5**
The theorem . . . 5
Proof sketch in six steps . . . 5
How to read this chapter . . . 7

Preliminaries **8**
1. Algorithms and decidability . . . 8
2. Diophantine equations and Hilbert's tenth problem . . . 9
3. Expressions, functions, languages . . . 10
4. Tools from analysis . . . 11
5. Differential algebra (only for the integration part) . . . 12

The complete proof, step by step **14**
0. The theorem and the plan of the proof . . . 14
1. The arithmetic input . . . 15
2. Local majorants . . . 16
3. The weighted penalty and the rounding theorem . . . 18
4. One variable suffices: a dense curve . . . 21
5. Transfer to one variable . . . 25
6. The sign problem . . . 27
7. The identity (zero) problem . . . 27
8. The integration problem . . . 28
9. Richardson's original, general formulation . . . 29
10. What exactly we have proved . . . 30
11. The modern version: polynomials and sine suffice . . . 31

Example: an expression whose sign cannot be seen **32**
1. The expression $A_a(t)$ . . . 32
2. Four equations . . . 34
3. What numerics sees . . . 35
4. The floating-point trap . . . 36
5. A certificate for $a = 3$ . . . 37
6. A certificate for $a = 42$ . . . 40
7. $a = 4$: a proof instead of numerics . . . 40
8. $a = 114$: we know the formula, not the answer . . . 40
9. The same question as a zero problem . . . 42
10. The moral . . . 42
11. Questions and exercises . . . 43

Appendices: powers of two and the integral of e^{x^2} **44**
A. Powers of two suffice . . . 44
B. e^{x^2} has no elementary antiderivative — not even locally . . . 44

Context, consequences and further results **48**
1. What the theorem says about computer algebra systems . . . 48
2. Why the sine? The boundary of decidability . . . 48

3. The constant problem and Schanuel's conjecture 49
4. Further results 49
5. Connection with formal proofs 49
6. Timeline 50
7. Annotated bibliography 50

Review questions and problems **52**
Review questions 52
Problems 53
Topics for further exploration 54
Answers to the review questions 54

Richardson's theorem: why a computer cannot recognize every zero

> *"No method is known for deciding whether a subelementary expression with no variables in it represents a number less than zero."* — Daniel Richardson (1968)

In Lecture 1 (§1.3) we stated that *for expressions built from x, rational numbers, π, $\ln 2$, sin, exp and the absolute value, there is no algorithm that decides whether an expression is identically zero.* This chapter is a **deep dive**: it contains a complete proof of this theorem (Daniel Richardson, *Journal of Symbolic Logic*, 1968), all the necessary concepts from outside the course, context, and exercises.

Materials

- Applet: one variable suffices — the dense curve, the lifting lemma, the role of the weights, the absolute-value trick.
- "Preliminaries" — computability, Hilbert's tenth problem, expression languages, tools from analysis, differential algebra.
- "The complete proof, step by step" — the complete proof with illustrations in Python.
- "Example: an expression whose sign cannot be seen" — numerics says "positive everywhere" for $x^3 + y^3 + z^3 = a$ with $a = 3, 4, 42, 114$, but the truth varies (for $a = 114$ nobody knows it).
- "Appendices: powers of two and the integral of e^{x^2}" — powers of two; why e^{x^2} has no elementary antiderivative.
- "Context, consequences and further results" — consequences for CAS, the boundary of decidability, the constant problem, literature.
- "Review questions and problems" — 20 review questions and 17 problems.

The theorem

Let $\mathcal{A}_1$ denote the expressions in one variable x built from rational numbers, π, $\log 2$, the operations $+, -, \cdot$ and the functions exp and sin (with arbitrary nesting), and let $\mathcal{B}_1$ denote the same with the absolute value $|\cdot|$.

Richardson's theorem (1968)

1. There is no algorithm that, given $A \in \mathcal{A}_1$, decides whether $A(x) < 0$ for some x.
2. There is no algorithm that, given $B \in \mathcal{B}_1$, decides whether $B(x) = 0$ for all x.
3. There is no algorithm that decides whether a given expression in $\mathcal{B}_1$ has an antiderivative expressible in $\mathcal{B}_1$.

The practical consequence: **there is no perfect `simplify`**. Every computer algebra system must sometimes answer "I don't know", get it wrong, or compute forever.

Proof sketch in six steps

The idea: we show that if such an algorithm existed, it would solve **Hilbert's tenth problem** — which is unsolvable. Each step transforms one question into another, equivalent one.

Step 1. The starting point: an undecidable equation in integers. There is a fixed polynomial $P(y, x_1, \dots, x_n)$ (in Richardson's version, with powers 2^{x_i}) such that no algorithm decides whether a given number $a \in \mathbb{N}$ belongs to the set U of those a for which the equation $P(a, \mathbf{x}) = 0$ has a solution in natural numbers (Davis–Putnam–Robinson 1961, Matiyasevich 1970).

Step 2. From integers to reals: the sine detects integrality. Since $\sin(\pi x) = 0 \iff x \in \mathbb{Z}$, the integer solutions are the zeros of the function

$$P^2 + \sum_i \sin^2(\pi x_i).$$

But later we will only be able to *approach* zeros, so we need more.

Step 3. The gap: small values force an exact zero. Richardson multiplies the sine terms by rapidly growing **weights** k_i — expressions that bound the derivatives of P^2 on a neighborhood of the point:

$$f = (n+1)^4 \Big[P^2 + \sum_i \sin^2(\pi x_i)\, k_i^4 \Big].$$

If $f(a, \mathbf{x}) \le 1$ for some $\mathbf{x} \in [0, \infty)^n$, then $\mathbf{x}$ lies so close to a point $\mathbf{m} \in \mathbb{N}^n$ that $P(a, \mathbf{m})^2 < 1$, and since this is an integer, $P(a, \mathbf{m}) = 0$. Hence: **either f has a zero, or $f > 1$ everywhere**. (Without the weights this fails — see the example $P(x) = Nx - 1$.)

Step 4. One variable suffices. The functions $h(t) = t \sin t$ and $g(t) = t \sin(t^3)$ run at different "speeds": on a short interval near a large t, the function h barely moves, while g performs full oscillations with amplitude $\approx t$. Thanks to this, the curve

$$\Gamma_n(t) = \big(h(t),\, h(g(t)),\, \dots,\, h(g^{[n-2]}(t)),\, g^{[n-1]}(t)\big)$$

has a **dense** image in $\mathbb{R}^n$. Let $F(a, \mathbf{u}) = f(a, u_1^2, \dots, u_n^2)$ (the squares remove the constraint $x_i \ge 0$). Since the inequality $F < 1$ is strict and holds on a whole neighborhood, and the curve enters every neighborhood, the function of **one** variable $G_a(t) = F(a, \Gamma_n(t))$ satisfies: $a \in U \iff G_a(t) < 1$ for some t.

Step 5. The sign problem. The expression $A_a = G_a - 1$ is negative somewhere exactly when the Diophantine equation has a solution. An algorithm for the sign would therefore solve Hilbert's problem.

Step 6. The zero problem and the integration problem. The absolute value turns a sign question into an identity question: $B_a = |A_a| - A_a$ is identically zero exactly when $A_a \ge 0$ everywhere. For integration, we multiply e^{x^2} (which has no elementary antiderivative) by a switch S_a that is identically zero for $a \notin U$ and, for $a \in U$, equals 1 on some interval.

$$\underbrace{\exists\, \mathbf{m} \in \mathbb{N}^n \;\; P(a, \mathbf{m}) = 0}_{\text{Hilbert}} \iff \exists\, \mathbf{x} \in [0, \infty)^n \;\; f(a, \mathbf{x}) \le 1 \iff \exists\, t \;\; G_a(t) < 1$$
$$\iff A_a \text{ somewhere} < 0 \iff B_a \not\equiv 0.$$

What is an algorithm, and what merely exists?

The transformation $a \mapsto G_a \mapsto A_a \mapsto B_a$ consists of **operations on expression trees** — it is an algorithm. The points where $G_a < 1$ exist only in the proof; nobody has to find them (and in practice they lie astronomically far away — in the example in §5 of the proof, at $t \approx 10^{11}$).

How to read this chapter

- To understand the sketch, Lecture 1 and basic analysis suffice.
- The complete proof ("The complete proof, step by step") requires the concepts from "Preliminaries": decidability, reductions, Diophantine sets, the mean value theorem, density.
- Facts taken without proof: the theorems of Davis–Putnam–Robinson and of Matiyasevich (arithmetic), and Liouville's theorem (only in the integration part). Apart from the standard theorems of analysis ("Preliminaries", §4), everything else is proved.
- Notation and the numbering of lemmas in "The complete proof, step by step" are matched to the original: Lemma 2.1 = *Lemma One*, Theorem 3.4 = *Theorem One* §1, Lemma 4.2 = *Lemmas Two* and *Three*, Theorem 4.4 = *Theorem Two*, §6–§8 = *Theorems One–Three* §2.

This chapter is based on Richardson's original paper and on expositions prepared by the instructor; the computations in it are run every time the book is built.

Preliminaries

The proof of Richardson's theorem combines three areas of mathematics that appear in our course only marginally: **computability theory**, **number theory** (Diophantine equations) and — in the integration part — **differential algebra**. This chapter collects all the definitions and theorems needed. Facts marked as *external* are taken without proof (with a reference to the source); apart from the standard theorems of analysis in §4, everything else is proved in this book.

Conventions: $\mathbb{N} = \{0, 1, 2, \dots\}$ (zero belongs to $\mathbb{N}$), and for a vector $\boldsymbol{\eta} = (\eta_1, \dots, \eta_d) \in \mathbb{R}^d$ we write $\|\boldsymbol{\eta}\|_\infty = \max_j |\eta_j|$ (the maximum norm).

1. Algorithms and decidability

Definition 1.1 (algorithm — informally)

An **algorithm** is a finite, unambiguous recipe that processes finite input data (a string over a finite alphabet) step by step and — if it halts — returns a finite result. There is no bound on time or memory, but every step is mechanical.

To talk about what an algorithm *cannot* do, we need a precise model.

Definition 1.2 (Turing machine)

A **Turing machine** is a quintuple $(S, \Sigma, \delta, s_0, F)$: a finite set of states S with an initial state $s_0 \in S$ and a set of final states $F \subseteq S$, a finite tape alphabet Σ (with a blank symbol), and a transition function $\delta\colon S \times \Sigma \to S \times \Sigma \times \{\leftarrow, \rightarrow\}$. The machine works on an infinite tape with a head: at each step it reads a symbol, writes a new one, moves the head and changes its state. The computation on a given input may **halt** (reach a final state) or run forever.

The Church–Turing thesis

Every algorithm in the informal sense can be implemented by a Turing machine (equivalently: by a Python program with unlimited memory, in the lambda calculus, by recursive functions…). This is not a mathematical theorem but a thesis about the adequacy of the definition — so far no "reasonable" model of computation has refuted it. Thanks to it, in proofs we may write "there is an algorithm" instead of constructing a Turing machine.

Definition 1.3 (decision problem, decidable set)

A **decision problem** is a set W of finite strings (e.g., codes of natural numbers or of expressions). The problem W is **decidable** (the set W is *recursive*) if there is an algorithm that, for *every* string w, halts and correctly answers whether $w \in W$. Otherwise W is **undecidable**.

Definition 1.4 (semi-decidable set)

A set W is **semi-decidable** (*computably enumerable*, *recursively enumerable*) if there is an algorithm that halts exactly on the elements of W (on the remaining strings it runs forever). Equivalently: there is an algorithm that lists all the elements of W one after another.

Theorem 1.5 (Post)

A set W is decidable if and only if both W and its complement $\overline{W}$ are semi-decidable.

Proof. If W is decidable, the same algorithm (forced to loop forever on the answer "no", respectively "yes") semi-decides W and $\overline{W}$. Conversely: for a given w we run both semi-decision procedures alternately, step by step. Exactly one of them halts, since w belongs to exactly one of $W, \overline{W}$; then we know the answer. ■

The most important example of an undecidable problem concerns programs themselves.

Theorem 1.6 (Turing, 1936: the halting problem)

The set $H = \{(e, w) : \text{the program with code } e \text{ halts on input } w\}$ is semi-decidable but undecidable.

Proof. Semi-decidability: we simulate the program e on w and halt when it halts. Now suppose that some algorithm $\mathsf{STOP}(e, w)$ decides H. Construct a program D that on input e does the following: if $\mathsf{STOP}(e, e)$ answers "yes", it enters an infinite loop; if "no", it halts. Let d be the code of the program D. Does D halt on input d? If so, then $\mathsf{STOP}(d, d) =$ "yes", so D loops forever — a contradiction. If not, then $\mathsf{STOP}(d, d) =$ "no", so D halts — a contradiction. ■

Post's theorem implies that the **complement** of the halting problem is not even semi-decidable.

Undecidability is transferred to other problems by means of *reductions*.

Definition 1.7 (many-one reduction)

A problem V **reduces** to a problem W (we write $V \leq_m W$) if there is a computable function r (an algorithm transforming strings into strings) such that

$$w \in V \iff r(w) \in W \qquad \text{for every string } w.$$

Proposition 1.8

If $V \leq_m W$ and V is undecidable, then W is undecidable. Moreover, W is decidable if and only if $\overline{W}$ is decidable, so a reduction $V \leq_m \overline{W}$ also proves that W is undecidable.

Proof. An algorithm deciding W, composed with r, would decide V. For the complement it suffices to swap the answers "yes" and "no". ■

Important: what a reduction does and what it need not do

The function r must be **computable** — in our proof it will be a transformation of *strings* (expression trees). The justification that $w \in V \iff r(w) \in W$, on the other hand, may be arbitrarily non-effective: it may use continuity, the intermediate value theorem, the existence of points that we cannot compute. The algorithm does not have to find these points.

2. Diophantine equations and Hilbert's tenth problem

Definition 2.1 (Diophantine equation, Diophantine set)

A **Diophantine equation** is an equation $p(x_1, \dots, x_n) = 0$, where $p \in \mathbb{Z}[x_1, \dots, x_n]$, for which we seek integer solutions (here: in $\mathbb{N}$). A set $D \subseteq \mathbb{N}$ is **Diophantine** if there is a polynomial $p(y, x_1, \dots, x_n)$ with integer coefficients such that

$$y \in D \iff \exists\, x_1, \dots, x_n \in \mathbb{N} \quad p(y, x_1, \dots, x_n) = 0.$$

For example, the set of composite numbers is Diophantine: $y \in D \iff \exists a, b \;\; y = (a+2)(b+2)$.

Hilbert's tenth problem (1900) asked for an algorithm that decides, for an arbitrary Diophantine equation, whether it has a solution in integers. The answer is negative, and the road to it led through equations with exponentiation.

Definition 2.2 (exponential Diophantine equation)

An **exponential Diophantine equation** is an equation $E_1 = E_2$ in which E_1, E_2 are built from variables and natural-number constants by addition, multiplication and exponentiation a^b (also with a variable exponent). Sets defined by such equations (as in Definition 2.1) are called **exponential Diophantine**.

Theorem 2.3 (Davis, Putnam, Robinson, 1961; external)

Every semi-decidable set $W \subseteq \mathbb{N}$ is exponential Diophantine. In particular, there is no algorithm deciding the solvability of exponential Diophantine equations.

Theorem 2.4 (Matiyasevich, 1970; DPRM; external)

Every semi-decidable set $W \subseteq \mathbb{N}$ is Diophantine. In particular, Hilbert's tenth problem is unsolvable.

Matiyasevich showed that the relation $c = a^b$ is itself Diophantine. The key was finding a Diophantine relation of exponential growth — Matiyasevich used Fibonacci numbers ($v = F_{2u}$); modern expositions usually use the Pell equation. Hence Theorem 2.4 follows from 2.3. It is even known that nine unknowns suffice (Matiyasevich; see J. P. Jones, 1982).

Corollary 2.5 (a fixed undecidable polynomial)

There exist $n \geq 1$ and a polynomial $p(y, x_1, \dots, x_n)$ with integer coefficients such that the set

$$U = \{\, a \in \mathbb{N} : \; \exists\, \mathbf{m} \in \mathbb{N}^n \;\; p(a, \mathbf{m}) = 0 \,\}$$

is undecidable.

Proof. The set $\{e : \text{program } e \text{ halts on input } e\}$ is semi-decidable (by simulation) but undecidable (the proof of Theorem 1.6 shows exactly this). Theorem 2.4 gives a Diophantine definition of it. ■

Important: the polynomial p is **a single, fixed one**. Only the parameter a varies. Richardson (1968), before Matiyasevich's result, used the analogous corollary of Theorem 2.3 in a form in which exponentiation occurs only as 2^{x_i} — see "The complete proof, step by step", §1, and "Appendices: powers of two and the integral of e^{x^2}", Appendix A.

3. Expressions, functions, languages

An algorithm is not given a "function", only a **finite description** of it.

Definition 3.1 (expression language, expression tree)

An **expression language** is specified by a grammar: a set of symbols for constants, variables and operations. For example

$$A \;::=\; q \mid \pi \mid \log 2 \mid x_1 \mid \cdots \mid x_d \mid (A+A) \mid (A-A) \mid (A\cdot A) \mid \exp(A) \mid \sin(A), \qquad q \in \mathbb{Q}.$$

Every expression is a finite **tree**: constants and variables at the leaves, operations at the nodes (see the section "How does a computer algebra system work?"). Trees are finite strings, so they can serve as input to an algorithm.

Definition 3.2 (the classes $\mathcal{A}_d$ and $\mathcal{B}_d$)

$\mathcal{A}_d$ — the expressions of the above grammar in the variables $x_1, \dots, x_d$ (the *analytic core*); each defines a function $\mathbb{R}^d \to \mathbb{R}$ that is defined everywhere. $\mathcal{B}_d$ — the same, with the additional operation $A \mapsto |A|$. Substituting expressions for variables is an operation on trees, so both classes are **effectively closed under composition**.

Different expressions may denote the same function: $\sin^2 x + \sin^2(x+\pi/2) - 1$ and 0 are different trees but the same function (since $\sin(x+\pi/2) = \cos x$). The questions we study concern **functions**, while the inputs are **trees**. Undecidability lives in the space between the two.

Definition 3.3 (partial functions and equality with domains)

A **partial** function f has a domain $\operatorname{dom} f \subseteq \mathbb{R}$. For a sum, difference and product, the domain is the intersection of the domains, and $\operatorname{dom}(f \circ g) = \{x \in \operatorname{dom} g : \; g(x) \in \operatorname{dom} f\}$. We write $f \equiv g$ when $\operatorname{dom} f = \operatorname{dom} g$ and $f = g$ on this domain. In particular, $f \equiv 0$ means that f is defined **everywhere** and equal to zero everywhere.

4. Tools from analysis

Definition 4.1 (real analytic function)

A function $f\colon \mathbb{R}^d \to \mathbb{R}$ is **analytic** if in a neighborhood of every point it is the sum of a convergent power series. Constants, coordinates, exp and sin are analytic, and sums, products and compositions of analytic functions are analytic. Hence **every expression in $\mathcal{A}_d$ defines an analytic function**. Expressions in $\mathcal{B}_d$ define continuous functions, but not always differentiable ones (e.g., $|x|$ at zero).

Theorem 4.2 (identity theorem)

If a function analytic on an interval $I \subseteq \mathbb{R}$ vanishes on some nonempty open interval $J \subseteq I$, then it vanishes on all of I.

Functions involving the absolute value lack this property: $|x| - x$ vanishes on $[0, \infty)$ but not on $\mathbb{R}$. This is precisely why the absolute value is needed in the identity part.

Theorem 4.3 (Darboux — intermediate value theorem)

If φ is continuous on $[\alpha, \beta]$ and c lies between $\varphi(\alpha)$ and $\varphi(\beta)$, then $\varphi(\tau) = c$ for some $\tau \in [\alpha, \beta]$.

Theorem 4.4 (mean value theorem; integral version)

If $\varphi\colon [\alpha,\beta]\to\mathbb{R}$ is differentiable, then $|\varphi(\beta)-\varphi(\alpha)|\le(\beta-\alpha)\sup_{[\alpha,\beta]}|\varphi'|$. For $Q\colon \mathbb{R}^n\to\mathbb{R}$ of class C^1 and the segment $\gamma(t)=\mathbf{x}+t(\mathbf{m}-\mathbf{x})$:

$$Q(\mathbf{m})-Q(\mathbf{x})=\int_0^1\frac{d}{dt}Q(\gamma(t))\,dt=\int_0^1\sum_{i=1}^{n}(m_i-x_i)\,\frac{\partial Q}{\partial x_i}(\gamma(t))\,dt.$$

Definition 4.5 (density)

A set $D\subseteq\mathbb{R}^n$ is **dense** if it meets every nonempty open set; equivalently: for every $\mathbf{a}\in\mathbb{R}^n$ and $\varepsilon>0$ there is $\mathbf{d}\in D$ with $\|\mathbf{d}-\mathbf{a}\|_\infty<\varepsilon$.

Fact 4.6 (strict inequalities are robust)

If $H\colon \mathbb{R}^n\to\mathbb{R}$ is continuous and $H(\mathbf{u}_0)<c$, then $H<c$ on some neighborhood of $\mathbf{u}_0$. For the non-strict inequality $H(\mathbf{u}_0)\le c$ the analogous conclusion need not hold.

This small observation is the crux of the whole proof: a dense curve need not pass through any particular point, but it enters every neighborhood — so it suffices to have a **strict** inequality with some room to spare.

Definition 4.7 (infimum versus minimum)

$\inf_t G(t)=0$ means that $G\ge 0$ and the values of G come arbitrarily close to zero; it does not mean that G takes the value 0 somewhere (e.g., $G(t)=\frac{1}{1+t^2}$ has infimum 0 on $\mathbb{R}$, which is not attained).

5. Differential algebra (only for the integration part)

Definition 5.1 (differential field)

A **differential field** is a field K with a **derivation** $D\colon K\to K$: an additive map satisfying the Leibniz rule $D(uv)=(Du)v+u(Dv)$. The **constants** are the elements c with $Dc=0$. Example: $\mathbb{C}(x)$ (rational functions) with $D=\frac{d}{dx}$; its constants are $\mathbb{C}$.

Definition 5.2 (elementary extension)

An extension $K\subseteq L$ is **elementary** if $L=K(\theta_1,\dots,\theta_s)$, where each θ_j is, over $K(\theta_1,\dots,\theta_{j-1})$, either (i) algebraic, or (ii) **exponential**: $D\theta_j/\theta_j=Db$ (i.e., "$\theta_j=e^b$"), or (iii) **logarithmic**: $D\theta_j=Db/b$ (i.e., "$\theta_j=\log b$"), for some b in the preceding field. An **elementary function** is an element of such an extension of $\mathbb{C}(x)$. Trigonometric functions are elementary, since $\sin z=(e^{iz}-e^{-iz})/(2i)$.

Theorem 5.3 (Liouville; Rosenlicht's form; external)

Let K be a differential field of characteristic 0 with field of constants $\mathbb{C}$, and let $a\in K$ have an antiderivative in some elementary extension of K with the same constants. Then there exist $v\in K$, nonzero $u_1,\dots,u_s\in K$ and constants $c_1,\dots,c_s\in\mathbb{C}$ such that

$$a=Dv+\sum_{j=1}^{s}c_j\,\frac{Du_j}{u_j}.$$

In other words: if the integral is elementary, it has the form "a function in K plus a combination of logarithms of functions in K". From this theorem we will derive in "Appendices: powers of two and the integral of e^{x^2}" that e^{x^2} has no elementary antiderivative on any interval.

Where to find proofs of the external facts

- Theorem 1.6 and reductions: M. Sipser, *Introduction to the Theory of Computation*, Chapters 3–5.
- Theorems 2.3–2.4: M. Davis, *Hilbert's Tenth Problem is Unsolvable*, Amer. Math. Monthly 80 (1973) — an accessible, complete exposition (Chauvenet and Steele Prizes); Yu. Matiyasevich, *Hilbert's Tenth Problem*, MIT Press 1993.
- Theorem 5.3: M. Rosenlicht, Pacific J. Math. 24 (1968); B. Conrad, *Impossibility theorems for elementary integration* (notes, Thm. 4.1).

The complete proof, step by step

This chapter contains a complete proof of Richardson's theorem in the form given in the original paper (D. Richardson, *Some undecidable problems involving elementary functions of a real variable*, J. Symbolic Logic **33** (1968), 514–520), with the details filled in. Apart from standard theorems of analysis, the only facts taken without proof are the Davis–Putnam–Robinson theorem (and, in §11, Matiyasevich's theorem) and — only in the integration part — Liouville's theorem; see "Preliminaries". We have illustrated every step of the construction with a computation in Python.

```
import math
import random
import numpy as np
import sympy as sp
import mpmath as mp
import matplotlib.pyplot as plt
```

0. The theorem and the plan of the proof

Recall (Definition 3.2 in "Preliminaries"): $\mathcal{A}_1$ consists of the expressions in one variable built from rational numbers, π, $\log 2$, the variable x, the operations $+, -, \cdot$ and the functions exp and sin (with arbitrary nesting); $\mathcal{B}_1$ is the same with the absolute value $|\cdot|$.

Theorem R (Richardson, 1968 — concrete form)

1. **(A — the sign problem.)** There is no algorithm that, for an arbitrary expression $A \in \mathcal{A}_1$, decides whether $A(x) < 0$ for some $x \in \mathbb{R}$.
2. **(B — the zero/identity problem.)** There is no algorithm that, for an arbitrary expression $B \in \mathcal{B}_1$, decides whether $B(x) = 0$ for all $x \in \mathbb{R}$.
3. **(C — the integration problem.)** There is no algorithm that, for an arbitrary $J \in \mathcal{B}_1$, decides whether there exists $H \in \mathcal{B}_1$, differentiable on $\mathbb{R}$, with $H' = J$.

In §9 we give Richardson's original, more general formulation (arbitrary effectively given classes of functions, including partial ones). All three parts follow from a single construction: for a fixed undecidable set $U \subseteq \mathbb{N}$ we algorithmically build, for each $a \in \mathbb{N}$, an expression $G_a \in \mathcal{A}_1$ satisfying a **dichotomy with a gap**

$$\begin{aligned} a \in U &: \quad G_a \geq 0 \text{ everywhere and } \inf_t G_a(t) = 0, \\ a \notin U &: \quad G_a(t) > 1 \text{ for every } t \in \mathbb{R}. \end{aligned}$$

Step	Construction	Purpose
§1	polynomial P with powers 2^{x_i}	encode $a \in U$ as the existence of an **integer** zero
§2	local majorants k_i	control the change of P^2 under rounding, everywhere in space
§3	weighted penalty f, squared coordinates F	"$f \leq 1$" forces an integer zero — a **gap** between 0 and 1

Step	Construction	Purpose
§4	curve $\Gamma_n(t)$ from $t\sin t$ and $t\sin t^3$	one variable, image dense in $\mathbb{R}^n$
§5	$G_a(t) = F(a, \Gamma_n(t))$	dichotomy in one variable
§6–8	$G_a - 1$, absolute value, cutoff	sign, identity and integration problems

The whole proof is a chain of equivalences, valid for every $a \in \mathbb{N}$:

$$\begin{aligned} a \in U &\iff \exists\, \mathbf{m} \in \mathbb{N}^n \quad P(a, \mathbf{m}) = 0 && (\S1)\\ &\iff \exists\, \mathbf{u} \in \mathbb{R}^n \quad F(a, \mathbf{u}) = 0 && (\S3)\\ &\iff \exists\, t \in \mathbb{R} \quad G_a(t) < 1 && (\S4\text{–}5)\\ &\iff A_a = G_a - 1 \text{ is negative somewhere} && (\S6)\\ &\iff B_a = |A_a| - A_a \not\equiv 0 && (\S7)\\ &\iff J_a = e^{x^2} S_a \text{ has no antiderivative in } \mathcal{B}_1 && (\S8). \end{aligned}$$

1. The arithmetic input

Theorem 1.1 (Davis–Putnam–Robinson in the form used by Richardson; external)

There exist $n \geq 2$ and a polynomial $p(Y, X_1, \dots, X_n, Z_1, \dots, Z_n)$ with integer coefficients such that for

$$P(y, \mathbf{x}) = p(y,\, x_1, \dots, x_n,\, 2^{x_1}, \dots, 2^{x_n})$$

the set

$$U = \{\, a \in \mathbb{N} : \ \exists\, \mathbf{m} \in \mathbb{N}^n \quad P(a, \mathbf{m}) = 0 \,\}$$

is undecidable.

This follows from Theorem 2.3 in "Preliminaries": exponentiation with an arbitrary base, $c = a^b$, can be expressed through powers of two (proof in "Appendices: powers of two and the integral of e^{x^2}", Appendix A), and a system of equations $E_1 = \dots = E_r = 0$ is equivalent to the single equation $\sum E_j^2 = 0$. "Dummy" variables allow us to assume $n \geq 2$. The polynomial p is **fixed once and for all**; only a varies.

We use the polynomial P only through two properties:

- **(P1)** $P \in \mathcal{A}_{n+1}$, since $2^x = \exp(x \log 2)$ — hence the constant $\log 2$ in the language;
- **(P2)** $P(a, \mathbf{m}) \in \mathbb{Z}$ for $a \in \mathbb{N}$, $\mathbf{m} \in \mathbb{N}^n$; in particular

$$0 \leq P(a, \mathbf{m})^2 < 1 \implies P(a, \mathbf{m}) = 0.$$

Property (P2) is the only place where the *discreteness* of the integers comes into play: a small approximate zero becomes an exact zero.

Remark 1.2 (on the constant π)

In his definition of "subelementary" expressions, Richardson (p. 515) does not list π, although he immediately uses $\sin(\pi x_i)$, and π appears in the hypotheses of the main theorem. We include π from the start. Thanks to this, $\cos u = \sin(u + \pi/2)$ and derivatives remain in the language (Lemma 2.3).

Remark 1.3 (where $\log 2$ and $\exp$ come from)

The function exp and the constant $\log 2$ are needed **only** to write 2^{x_i}, that is, only because in 1968 the undecidability of *exponential* Diophantine equations was known. After Matiyasevich's theorem (1970) one can take an ordinary polynomial, and then the whole construction does without exp and $\log 2$ — see §11.

2. Local majorants

We are going to round real points to integer points, and we must control how P^2 changes in the process. The points may lie arbitrarily far away, so a bound on the derivative on a single compact set is not enough: we need a **formula** — an expression that, at every point, bounds the function on a whole neighborhood of that point.

Lemma 2.1 (local majorants; Richardson, *Lemma One*)

For every expression $A \in \mathcal{A}_d$ one can algorithmically construct $M_A \in \mathcal{A}_d$ such that for every $\mathbf{z} \in \mathbb{R}^d$

$$M_A(\mathbf{z}) > 1 \qquad \text{and} \qquad |A(\mathbf{z}+\boldsymbol{\eta})| < M_A(\mathbf{z}) \quad \text{for every } \boldsymbol{\eta} \text{ with } \|\boldsymbol{\eta}\|_\infty \le 1.$$

Proof. We define M_A recursively on the expression tree of A:

$$\begin{aligned} M_c &= c^2+2 \quad (c \text{ — a constant: } q\in\mathbb{Q},\ \pi,\ \log 2), & M_{x_j} &= x_j^2+2,\\ M_{A\pm B} &= M_A + M_B, & M_{A\cdot B} &= M_A\, M_B,\\ M_{\exp A} &= \exp(M_A), & M_{\sin A} &= 2. \end{aligned}$$

Each right-hand side belongs to $\mathcal{A}_d$, and the recursion terminates because the tree is finite. We verify both properties by induction on the structure of the expression; fix $\mathbf{z}$ and $\boldsymbol{\eta}$ with $\|\boldsymbol{\eta}\|_\infty \le 1$.

- *A constant c.* The value does not depend on the point, and $c^2+2-|c| = \left(|c|-\tfrac12\right)^2 + \tfrac74 > 0$, so $|c| < c^2+2$; obviously $c^2+2 \ge 2 > 1$.
- *A variable x_j.* $|z_j+\eta_j| \le |z_j|+1 < z_j^2+2$, since $z_j^2+2-(|z_j|+1) = \left(|z_j|-\tfrac12\right)^2 + \tfrac34 > 0$.
- *Sum and difference.* By the triangle inequality, $|A\pm B| \le |A|+|B| < M_A + M_B$ (values at $\mathbf{z}+\boldsymbol{\eta}$, majorants at $\mathbf{z}$); the sum of the majorants exceeds 2.
- *Product.* $|AB| = |A|\,|B| < M_A M_B$, since both majorants are positive; a product of numbers >1 is >1.
- *Exponential.* $e^A \le e^{|A|} < e^{M_A}$ by monotonicity, and $e^{M_A} > e > 1$.
- *Sine.* $|\sin A| \le 1 < 2$. ∎

The construction never asks for the sign or the value of any expression. For the constants of the grammar (q, π, $\log 2$) the sign is known, so $|c|+2$ would also work, as in the original. We take c^2+2 because this rule also works without change for any variable-free subexpression treated as a constant (this is what the function `majoranta` below does), e.g., $\pi - \frac{22}{7}$, whose sign the algorithm need not be able to decide. Here is an implementation on SymPy trees:

```
def majoranta(e):
    """Majoranta lokalna M_e (Lemat 2.1) zbudowana rekurencyjnie po drzewie wyrażenia
     ↪ SymPy."""
    if e.is_number:                      # stała: liczba wymierna, π, log 2, ...
        return e**2 + 2
    if e.is_Symbol:                      # zmienna
        return e**2 + 2
```

```
    if e.is_Add:
        return sp.Add(*[majoranta(a) for a in e.args])
    if e.is_Mul:                          # SymPy zapisuje A - B jako A + (-1)·B
        return sp.Mul(*[majoranta(a) for a in e.args])
    if e.is_Pow and e.exp.is_Integer and e.exp > 0:
        return majoranta(e.base) ** e.exp  # potęga = wielokrotny iloczyn
    if isinstance(e, sp.exp):
        return sp.exp(majoranta(e.args[0]))
    if isinstance(e, (sp.sin, sp.cos)):   # cos u = sin(u + π/2)
        return sp.Integer(2)
    raise ValueError(f"węzeł spoza języka: {e.func}")

u, v = sp.symbols('u v')
A = sp.exp(u*v) - sp.sin(u)
M = majoranta(A)
M
```

$$e^{(u^2+2)(v^2+2)} + 6$$

Lemma 2.1 promises the inequality $|A(\mathbf{z}+\boldsymbol{\eta})| < M_A(\mathbf{z})$ for every shift with $\|\boldsymbol{\eta}\|_\infty \le 1$. Let us check it on 20,000 random pairs $(\mathbf{z}, \boldsymbol{\eta})$ and count the violations:

```
fA = sp.lambdify((u, v), A, 'math')
fM = sp.lambdify((u, v), M, 'math')
random.seed(0)
naruszenia = 0
for _ in range(20_000):
    z = (random.uniform(-1.5, 1.5), random.uniform(-1.5, 1.5))
    eta = (random.uniform(-1, 1), random.uniform(-1, 1))
    if not abs(fA(z[0] + eta[0], z[1] + eta[1])) < fM(*z):
        naruszenia += 1
naruszenia
```

```
0
```

No violations — as it should be: the lemma has been proved, and the test merely illustrates it. The result $\exp((u^2+2)(v^2+2)) + 6$ (rather than $+2$) comes from the fact that SymPy stores $-\sin u$ as the product $(-1)\cdot\sin u$, and the product rule gives $3 \cdot 2$. This is still a valid majorant. More generally:

Remark 2.2 (the majorant depends on the notation)

M_A is a function of the **tree** A, not of the function that A denotes: the trees $x - x$ and 0 get the majorants $2(x^2+2)$ and 2 (SymPy would simplify $x - x$ to 0 already when building the tree). This does no harm — the reduction needs *some* proven bound given by a formula, not the best one.

Lemma 2.3 (effective differentiation)

For $A \in \mathcal{A}_d$, an expression for $\partial A/\partial x_j$ in the language $\mathcal{A}_d$ can be constructed algorithmically.

Proof. Recursively: $\partial_j c = 0$, $\partial_j x_i \in \{0, 1\}$, the sum and product rules, $\partial_j e^A = e^A \partial_j A$ and $\partial_j \sin A = \sin\!\left(A + \frac{\pi}{2}\right) \partial_j A$. ■

Definition 2.4 (weights)

Let $Q(y, \mathbf{x}) = P(y, \mathbf{x})^2$. For $i = 1, \dots, n$ we set $k_i = M_{\partial Q/\partial x_i}$ (Lemma 2.1 applied to the expression from Lemma 2.3, in the variables $(y, x_1, \dots, x_n)$, with zero perturbation of y). Then $k_i \in \mathcal{A}_{n+1}$, $k_i > 1$, and

$$\left|\frac{\partial Q}{\partial x_i}(y, \mathbf{z})\right| < k_i(y, \mathbf{x}) \qquad \text{whenever } \|\mathbf{z} - \mathbf{x}\|_\infty \le 1. \tag{2.1}$$

Note: the derivative is evaluated at the point $\mathbf{z}$, while the weight is evaluated at the **center** $\mathbf{x}$. A single weight controls the derivative on the whole cube around $\mathbf{x}$.

3. The weighted penalty and the rounding theorem

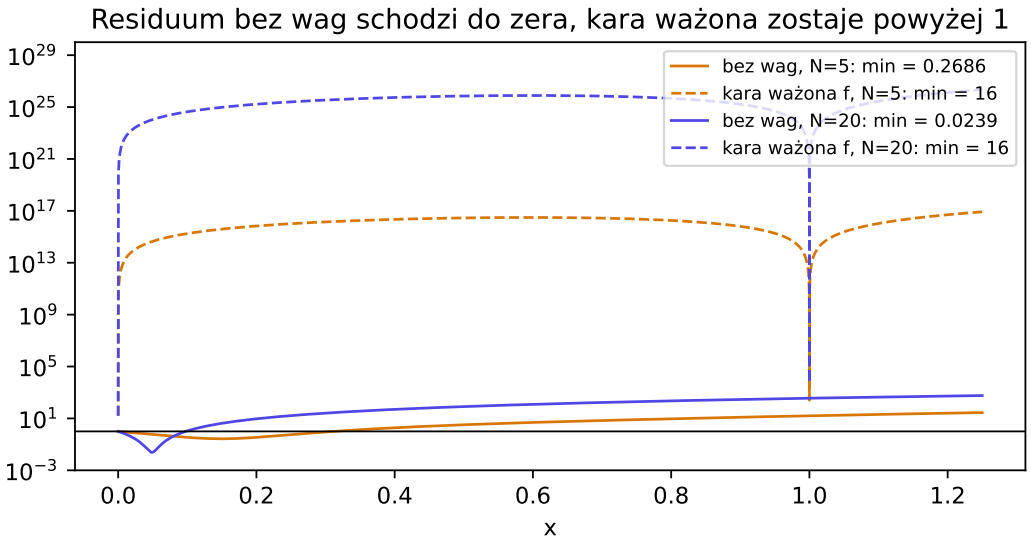
On the set $[0, \infty)^n$, the zeros of the expression $P(y, \mathbf{x})^2 + \sum_i \sin^2(\pi x_i)$ are exactly the integer solutions (since $\sin(\pi x) = 0 \iff x \in \mathbb{Z}$). Later, however, we will replace the variables by the points of a dense curve, which **comes arbitrarily close** to a zero but need not hit it. So we need a theorem about **small values**, not only about zeros. The naive residual does not provide one:

Example 3.1 (no gap without weights)

For $N \ge 2$ let $P_N(x) = Nx - 1$: there is no root in $\mathbb{N}$. And yet $P_N(1/N)^2 + \sin^2(\pi/N) = \sin^2(\pi/N) \to 0$ as $N \to \infty$. No fixed threshold separates "yes" from "no" across the whole family: the polynomial becomes more and more sensitive to perturbations.

Let us compare, in a plot, the unweighted residual with the weighted penalty f (Definition 3.2 below, here with $n = 1$) for $N = 5$ and $N = 20$:

```
x = sp.symbols('x')
xs = np.linspace(0, 1.25, 40_001)
fig, ax = plt.subplots(figsize=(7.5, 3.4))
for N, kolor in ((5, '#d97706'), (20, '#4f46e5')):
    P = N*x - 1
    k = majoranta(sp.factor(sp.diff(P**2, x)))          # waga z Lematu 2.1
    r = sp.lambdify(x, P**2 + sp.sin(sp.pi*x)**2, 'numpy')
    f = sp.lambdify(x, 2**4 * (P**2 + sp.sin(sp.pi*x)**2 * k**4), 'numpy')
    ax.semilogy(xs, r(xs), color=kolor, lw=1.2, label=f"bez wag, N={N}: min =
↪   {r(xs).min():.4f}")
    ax.semilogy(xs, f(xs), color=kolor, lw=1.2, ls='--', label=f"kara ważona f, N={N}:
↪   min = {f(xs).min():.0f}")
ax.axhline(1, color='black', lw=0.8)
ax.set_xlabel('x'); ax.legend(fontsize=8, loc='upper right'); ax.set_ylim(1e-3, 1e30)
ax.set_title('Residuum bez wag schodzi do zera, kara ważona zostaje powyżej 1')
plt.show()
```



Without weights, the minimum of the residual drops from 0.27 ($N = 5$) to 0.024 ($N = 20$), while in both cases the weighted penalty has minimum 16 (at $x = 0$, where $P_N(0)^2 = 1$), far above the threshold 1.

Definition 3.2 (weighted penalty; Richardson, p. 516)

$$f(y, \mathbf{x}) = (n+1)^4 \Big[P(y, \mathbf{x})^2 + \sum_{i=1}^{n} \sin^2(\pi x_i)\, k_i(y, \mathbf{x})^4 \Big].$$

This is an expression in $\mathcal{A}_{n+1}$, nonnegative everywhere. The term P^2 measures the error in the equation, the sine terms measure the distance from the lattice $\mathbb{Z}^n$; the larger the derivatives of P^2 at a given place, the closer to the lattice the weights require the point to be.

Lemma 3.3 (the sine measures the distance to an integer)

Let $x \geq 0$, let $m \in \mathbb{N}$ be the natural number nearest to x (either one in case of a tie), and $d = |x - m|$. Then $0 \leq d \leq \frac{1}{2}$, $|\sin(\pi x)| = \sin(\pi d)$, and

$$2d \leq |\sin(\pi x)| \qquad \text{and consequently} \qquad d \leq \big(\sin^2(\pi x)\big)^{1/4}.$$

Proof. The periodicity and symmetry of the sine give $|\sin(\pi x)| = |\sin(\pi(x-m))| = \sin(\pi d)$, since $0 \leq \pi d \leq \pi/2$. On $[0, \frac{1}{2}]$ the function $t \mapsto \sin(\pi t)$ is concave (its second derivative is $-\pi^2 \sin(\pi t) \leq 0$), so it lies above the chord joining $(0, 0)$ and $(\frac{1}{2}, 1)$: $\sin(\pi d) \geq 2d$. Since $d \leq \frac{1}{2} \leq 2$, we have $2d \geq d^2$; hence $d^2 \leq \sin(\pi d) = |\sin(\pi x)|$, and taking square roots, $d \leq |\sin \pi x|^{1/2} = (\sin^2 \pi x)^{1/4}$. ■

(Richardson proves $t^2 \leq \sin(\pi t)$ on $[0, \frac{1}{2}]$ by studying the function $\sin(\pi t) - t^2$; the concavity argument is shorter.)

Theorem 3.4 (rounding theorem; Richardson, Theorem One)

For every $a \in \mathbb{N}$ the following conditions are equivalent:

1. $\exists\, \mathbf{m} \in \mathbb{N}^n \quad P(a, \mathbf{m}) = 0$ (i.e., $a \in U$);
2. $\exists\, \mathbf{x} \in [0, \infty)^n \quad f(a, \mathbf{x}) = 0$;
3. $\exists\, \mathbf{x} \in [0, \infty)^n \quad f(a, \mathbf{x}) \leq 1$.

In particular, if $a \notin U$, then $f(a, \mathbf{x}) > 1$ at every point $\mathbf{x} \geq 0$.

Proof. (1)⇒(2): at an integer zero $\mathbf{m}$, both P and all the $\sin(\pi m_i)$ vanish. (2)⇒(3) is obvious. It remains to prove (3)⇒(1). Let $\mathbf{x} \geq 0$ and $f(a, \mathbf{x}) \leq 1$.

Step 1 (each term is small). All the terms in the brackets are nonnegative, so

$$Q(a, \mathbf{x}) \leq \frac{1}{(n+1)^4}, \qquad \sin^2(\pi x_i)\, k_i(a, \mathbf{x})^4 \leq \frac{1}{(n+1)^4} \quad (i = 1, \dots, n).$$

Taking fourth roots (the weights are positive) gives $\left(\sin^2(\pi x_i)\right)^{1/4} k_i(a, \mathbf{x}) \leq \frac{1}{n+1}$.

Step 2 (rounding). Let m_i be the natural number nearest to x_i, and $d_i = |m_i - x_i| \leq \frac{1}{2}$. By Lemma 3.3,

$$d_i\, k_i(a, \mathbf{x}) \leq \frac{1}{n+1} \qquad (i = 1, \dots, n). \tag{3.1}$$

Step 3 (estimating along the whole segment). Let $\gamma(t) = \mathbf{x} + t(\mathbf{m} - \mathbf{x})$, $t \in [0, 1]$. Every point of the segment lies at distance $\leq \frac{1}{2}$ from $\mathbf{x}$ in the maximum norm, so (2.1) applies. By Theorem 4.4 in "Preliminaries":

$$Q(a, \mathbf{m}) - Q(a, \mathbf{x}) = \int_0^1 \sum_{i=1}^{n} (m_i - x_i)\, \frac{\partial Q}{\partial x_i}(a, \gamma(t))\, dt \leq \sum_{i=1}^{n} d_i\, k_i(a, \mathbf{x}).$$

Hence, by Step 1 and (3.1),

$$0 \leq Q(a, \mathbf{m}) \leq \frac{1}{(n+1)^4} + \frac{n}{n+1} < 1, \tag{3.2}$$

where the last inequality is strict because $(n+1)^{-4} < (n+1)^{-1}$.

Step 4 (integrality). Since $a \in \mathbb{N}$ and $\mathbf{m} \in \mathbb{N}^n$, by (P2) the number $Q(a, \mathbf{m}) = P(a, \mathbf{m})^2$ is a nonnegative integer less than 1, i.e., 0. Hence $P(a, \mathbf{m}) = 0$. ■

Division of labor in the proof

Nonnegative coordinates round to **natural** numbers (this matters, since 2^x is not an integer for negative integers). The weights control the whole segment, not just its endpoints. The factor $(n+1)^4$ leaves a margin below 1. Integrality enters only in the last step, where an approximate zero becomes an exact one.

Remark 3.5 (a variant with squares)

The same weights work in the simpler penalty $f_{\text{alt}} = (n+1)^2[P^2 + \sum_i \sin^2(\pi x_i) k_i^2]$: from $f_{\text{alt}} \leq 1$ and the stronger inequality $2d_i \leq |\sin \pi x_i|$ we get $d_i k_i \leq \frac{1}{2(n+1)}$, so $Q(a, \mathbf{m}) \leq (n+1)^{-2} + \frac{n}{2(n+1)} < 1$. In what follows we stick to Richardson's fourth powers.

Remark 3.6 (the gap is even uniform)

If $f(a, \mathbf{x}) \leq s$, the same computation gives $Q(a, \mathbf{m}) \leq \varphi(s) = \frac{s}{(n+1)^4} + \frac{n\, s^{1/4}}{n+1}$. Since $\varphi(1) < 1$ and φ is continuous, there is $s_0 > 1$ (depending only on n) with $\varphi(s_0) < 1$. Then for $a \notin U$ we have $f(a, \cdot) > s_0$, i.e., $\inf f \geq s_0 > 1$. This strengthened version is not needed later.

The last step of this part removes the constraint $\mathbf{x} \geq 0$: squares range over the entire set $[0, \infty)^n$.

Corollary 3.7 (the gap on all of $\mathbb{R}^n$; Richardson, Corollary)

Let $F(y, u_1, \dots, u_n) = f(y, u_1^2, \dots, u_n^2)$. Then $F \in \mathcal{A}_{n+1}$, $F \geq 0$, and for $a \in \mathbb{N}$:

- if $a \in U$, then $F(a, \mathbf{u}) = 0$ for some $\mathbf{u} \in \mathbb{R}^n$;
- if $a \notin U$, then $F(a, \mathbf{u}) > 1$ for **every** $\mathbf{u} \in \mathbb{R}^n$.

Equivalently: $a \in U \iff \exists \mathbf{u}\ F(a, \mathbf{u}) = 0 \iff \exists \mathbf{u}\ F(a, \mathbf{u}) < 1 \iff \exists \mathbf{u}\ F(a, \mathbf{u}) \leq 1$.

Proof. The map $\mathbf{u} \mapsto (u_1^2, \dots, u_n^2)$ maps $\mathbb{R}^n$ **onto** $[0, \infty)^n$ (a witness for $\mathbf{m}$ is $u_i = \sqrt{m_i}$). The claim follows from Theorem 3.4. Square roots appear only in the proof; the expression F contains only squares. ■

4. One variable suffices: a dense curve

We now have an undecidable inequality in n variables. Richardson replaces all the variables by functions of **one** variable, using two oscillations at different scales:

$$h(t) = t \sin t, \qquad g(t) = t \sin(t^3).$$

The idea: near a large v, on an interval of length of order δ/v, the function h changes by at most δ (since $|h'(t)| \leq 1 + t$), while the phase t^3 changes by about $3v^2 \cdot \delta/v = 3\delta v$ — arbitrarily much. So h barely moves, while g (with amplitude $\approx v$) sweeps the whole interval $[-v, v]$.

Lemma 4.1 (arbitrarily distant solutions of $h(v) = a$)

For every $a \in \mathbb{R}$ and $L > 0$ there is $v > L$ with $h(v) = a$.

Proof. Choose $k \in \mathbb{N}$ so large that $p = 2\pi k + \frac{\pi}{2} > \max\{L, |a|\}$, and let $q = 2\pi k + \frac{3\pi}{2}$. Then $h(p) = p > a$ and $h(q) = -q < a$, so by the intermediate value theorem $h(v) = a$ for some $v \in (p, q)$; clearly $v > L$. ■

Lemma 4.2 (lifting: one coordinate approximate, the other exact; Richardson, *Lemmas Two* and *Three*)

For any $a, b \in \mathbb{R}$, $\delta > 0$ and $M \geq 0$ there is $w > M$ such that

$$|h(w) - a| < \delta \qquad \text{and} \qquad g(w) = b.$$

Proof. Choose $R > \max\{M,\ |b|,\ 1,\ 4\pi/\delta\}$. By Lemma 4.1 we take $v > \sqrt{R^2 + \delta/2}$ with $h(v) = a$ and set $u = \sqrt{v^2 - \delta/2}$. Then $v > u > R$ and $v^2 - u^2 = \delta/2$.

The interval $[u, v]$ is short for h. Since $u > 1$,

$$(v - u)(v + 1) < (v - u)(v + u) = \tfrac{\delta}{2} < \delta. \tag{4.1}$$

On it, the phase t^3 changes by more than a period. Using $v^2 = u^2 + \delta/2$:

$$v^3 - u^3 = v\,(u^2 + \tfrac{\delta}{2}) - u^3 = u^2(v - u) + \tfrac{\delta}{2}\, v > \tfrac{\delta}{2}\, u > \tfrac{\delta}{2}\, R > 2\pi. \tag{4.2}$$

g passes through b. The open interval (u^3, v^3) has length $> 2\pi$, so it contains a number $\theta_+ \equiv \frac{\pi}{2}$ and a number $\theta_- \equiv \frac{3\pi}{2} \pmod{2\pi}$ (both progressions have step 2π). Their cube roots $t_\pm = \theta_\pm^{1/3}$ lie in (u, v) and

$$g(t_+) = t_+ > u > |b| \geq b, \qquad g(t_-) = -t_- < -u < -|b| \leq b.$$

By the intermediate value theorem there is w between t_+ and t_- (so $w \in (u, v)$, $w > M$) with $g(w) = b$.

h stays close to a. For $s \in [u, v]$: $|h'(s)| = |\sin s + s\cos s| \le 1 + s \le 1 + v$. By the mean value theorem and (4.1),

$$|h(w) - a| = |h(w) - h(v)| \le (v - w)(1 + v) \le (v - u)(1 + v) < \delta. \qquad \blacksquare$$

This proof is constructive and can be carried out numerically. Since the phase t^3 is huge, we compute in high-precision arithmetic (`mpmath`, 80 digits):

```
mp.mp.dps = 80
h = lambda t: t * mp.sin(t)
g = lambda t: t * mp.sin(t**3)

def bisekcja(F, lo, hi, kroki=400):
    """Miejsce zerowe F na [lo, hi], gdy F(lo), F(hi) mają różne znaki (tw. Darboux)."""
    flo = F(lo)
    for _ in range(kroki):
        mid = (lo + hi) / 2
        if (F(mid) > 0) == (flo > 0):
            lo, flo = mid, F(mid)
        else:
            hi = mid
    return (lo + hi) / 2

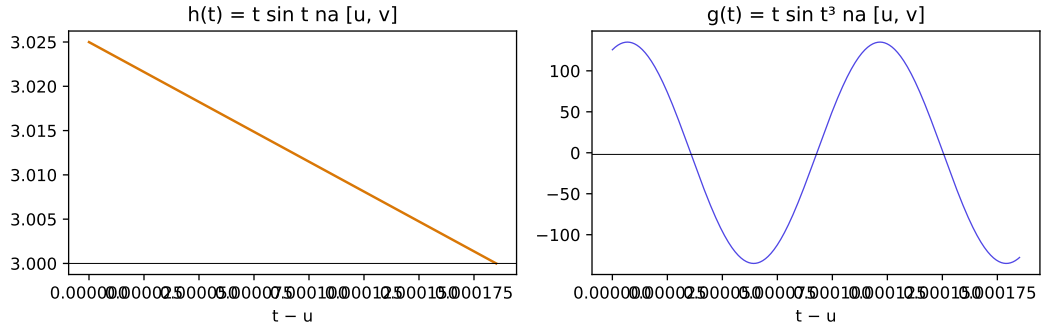
def podnies(a, b, delta, M=0):
    """Lemat 4.2: zwraca w > M z |h(w) - a| < delta i g(w) = b — dokładnie według
     ↪ dowodu."""
    a, b, delta = mp.mpf(a), mp.mpf(b), mp.mpf(delta)
    R = max(mp.mpf(M), abs(b), mp.mpf(1), 4 * mp.pi / delta) * (1 + mp.mpf('1e-9'))
    dolna = mp.sqrt(R**2 + delta / 2)                          # chcemy v > sqrt(R^2 + δ/2)
    k = int(mp.ceil((max(dolna, abs(a)) - mp.pi / 2) / (2 * mp.pi))) + 1
    v = bisekcja(lambda t: h(t) - a, 2*mp.pi*k + mp.pi/2, 2*mp.pi*k + 3*mp.pi/2)   #
 ↪ Lemat 4.1
    u = mp.sqrt(v**2 - delta / 2)
    assert (v - u) * (v + 1) < delta / 2 and v**3 - u**3 > 2 * mp.pi          # (4.1),
     ↪ (4.2)
    nastepna = lambda c: c + 2 * mp.pi * mp.ceil((u**3 - c) / (2 * mp.pi))     # pierwsza
 ↪ ≡ c (mod 2π) > u^3
    t_plus, t_minus = mp.cbrt(nastepna(mp.pi / 2)), mp.cbrt(nastepna(3 * mp.pi / 2))
    w = bisekcja(lambda t: g(t) - b, min(t_plus, t_minus), max(t_plus, t_minus))
    return w, dict(R=R, u=u, v=v)

w, info = podnies(3, -2, 0.1)
print(f"w = {mp.nstr(w, 20)}")
print(f"|h(w) - 3| = {mp.nstr(abs(h(w) - 3), 6)}  (< δ = 0.1)")
print(f"g(w) - (-2) = {mp.nstr(g(w) + 2, 6)}")
print(f"długość przedziału v - u = {mp.nstr(info['v'] - info['u'], 6)}")
```

```
w = 135.06612167797138163
|h(w) - 3| = 0.0201545  (< δ = 0.1)
g(w) - (-2) = 3.88149e-73
długość przedziału v - u = 0.000185094
```

We obtained $w \approx 135.07$: $|h(w) - 3| \approx 0.02 < \delta$, while $g(w)$ differs from -2 by about $4 \cdot 10^{-73}$, i.e., at the limit of the working precision. On the interval $[u, v]$, of length about $1.85 \cdot 10^{-4}$, the function h barely moves, while g performs more than one and a half oscillations with amplitude about v:

```
ts = [info['u'] + (info['v'] - info['u']) * j / 2000 for j in range(2001)]
fig, (a1, a2) = plt.subplots(1, 2, figsize=(9, 3))
a1.plot([float(t - info['u']) for t in ts], [float(h(t)) for t in ts], color='#d97706')
a1.axhline(3, color='black', lw=0.6); a1.set_title('h(t) = t sin t na [u, v]');
 ↪  a1.set_xlabel('t − u')
a2.plot([float(t - info['u']) for t in ts], [float(g(t)) for t in ts], color='#4f46e5',
 ↪  lw=0.8)
a2.axhline(-2, color='black', lw=0.6); a2.set_title('g(t) = t sin t³ na [u, v]');
 ↪  a2.set_xlabel('t − u')
plt.tight_layout(); plt.show()
```



Now we assemble the curve in $\mathbb{R}^n$. We write $g^{[j]}$ for the j-fold **composition** of g (not a power), $g^{[0]}(t) = t$.

Definition 4.3 (the Richardson curve)

For $n \geq 2$:

$$\Gamma_n(t) = \Big(h(t),\ h(g(t)),\ \ldots,\ h(g^{[n-2]}(t)),\ g^{[n-1]}(t)\Big).$$

The first $n-1$ coordinates begin with h, the last one is an iterate of g alone; e.g., $\Gamma_2 = (h, g)$, $\Gamma_4 = (h(t), h(g(t)), h(g(g(t))), g(g(g(t))))$. The following recursion holds:

$$\Gamma_{n+1}(t) = (h(t),\ \Gamma_n(g(t))). \tag{4.3}$$

Theorem 4.4 (density with an exact last coordinate; Richardson, Theorem Two)

For $n \geq 2$, $\mathbf{a} \in \mathbb{R}^n$, $\delta > 0$ and $M \geq 0$ there is $w > M$ such that

$$\big|h(g^{[j-1]}(w)) - a_j\big| < \delta \quad (1 \leq j \leq n-1), \qquad g^{[n-1]}(w) = a_n.$$

In particular, the set $\Gamma_n((M, \infty))$ is dense in $\mathbb{R}^n$.

Proof. Induction on n. For $n = 2$ this is Lemma 4.2. Assume the claim for n and take a target $(a_1, \ldots, a_{n+1})$. The induction hypothesis for the "tail" $(a_2, \ldots, a_{n+1})$ (with $M = 0$) gives $w^* > 0$ for which the first $n-1$ coordinates of $\Gamma_n(w^*)$ approximate $a_2, \ldots, a_n$ to within δ, and the last one equals a_{n+1}. Lemma 4.2 with targets a_1 and w^* gives $w > M$ with $|h(w) - a_1| < \delta$ and $g(w) = w^*$. By (4.3), $\Gamma_{n+1}(w) = (h(w), \Gamma_n(w^*))$ — exactly the required approximation. Density: every nonempty open set contains a cube $\{\mathbf{u} : \|\mathbf{u} - \mathbf{a}\|_\infty < \delta\}$. ∎

Remark 4.5 (why the last coordinate is exact)

The induction step requires $g(w) = w^*$ **exactly**. If we only had $g(w) \approx w^*$, the error would be amplified by the very steeply varying iterates of g; it could be controlled only by a separate

argument based on the continuity of Γ_n at the point w^*, with an astronomically small tolerance. The exact "lift" of the last coordinate ensures that the whole tail is reproduced without error.

Let us check this numerically in $\mathbb{R}^3$: we aim at $(1.5, -4, 7)$ with $\delta = 0.05$.

```
cel, delta = (1.5, -4, 7), 0.05
w_gwiazdka, _ = podnies(cel[1], cel[2], delta)      # ogon: |h(w*) - a2| < δ, g(w*) = a3
w, _ = podnies(cel[0], w_gwiazdka, delta)            # |h(w) - a1| < δ, g(w) = w*
Gamma3 = (h(w), h(g(w)), g(g(w)))
print("w =", mp.nstr(w, 15))
for j, (c, a) in enumerate(zip(Gamma3, cel), 1):
    print(f"współrzędna {j}: {mp.nstr(c, 12):>16}   błąd = {mp.nstr(abs(c - a), 3)}")
```

```
w = 273.313041756929
współrzędna 1:    1.50843582206   błąd = 0.00844
współrzędna 2:   -3.98980322229   błąd = 0.0102
współrzędna 3:              7.0   błąd = 1.1e-64
```

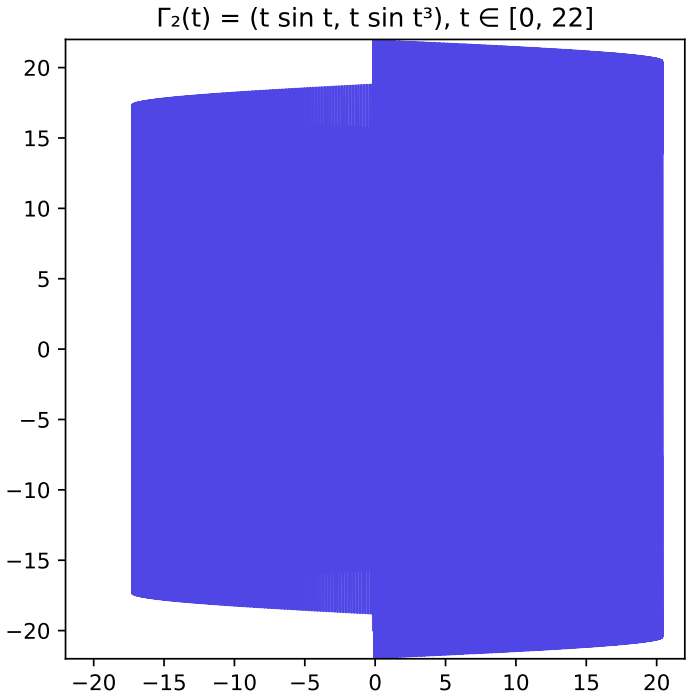
The first two coordinates differ from the target by 0.008 and 0.010, i.e., by less than $\delta = 0.05$, and the third equals 7 to within the working precision, as Theorem 4.4 predicts.

Remark 4.6 (density, not surjectivity)

The image of Γ_n is dense, but it has **measure zero** in $\mathbb{R}^n$ for $n \geq 2$: on a compact parameter interval of length ℓ the curve is Lipschitz with some constant L, so its image can be covered by N cubes of side $2L\ell/N$, of total volume $(2L\ell)^n N^{1-n} \to 0$; the whole parameter line is a countable union of such intervals. That is why hitting all coordinates exactly is the wrong goal — and it is not needed: it suffices to enter a neighborhood of the point (Fact 4.6 in "Preliminaries").

Let us plot an initial portion of the curve Γ_2, for $t \in [0, 22]$:

```
t = np.linspace(0, 22, 400_000)
fig, ax = plt.subplots(figsize=(5.2, 5.2))
ax.plot(t*np.sin(t), t*np.sin(t**3), lw=0.15, color='#4f46e5')
ax.set_xlim(-22, 22); ax.set_ylim(-22, 22); ax.set_aspect('equal')
ax.set_title('Γ₂(t) = (t sin t, t sin t³), t ∈ [0, 22]')
plt.show()
```



In the plot, the curve merges into a solid patch: for $t \le 22$ the phase t^3 makes about 1700 turns, so successive sweeps of the second coordinate lie very close to one another. This is an artifact of the line width, since the image of the curve itself has measure zero (Remark 4.6).

5. Transfer to one variable

Lemma 5.1 (what substituting a dense curve preserves)

Let $H\colon \mathbb{R}^n \to \mathbb{R}$ be continuous, $c \in \mathbb{R}$, $M \ge 0$. Then

$$\exists\, \mathbf{u} \in \mathbb{R}^n \;\; H(\mathbf{u}) < c \quad \Longleftrightarrow \quad \exists\, t > M \;\; H(\Gamma_n(t)) < c,$$

and also $\inf_{t>M} H(\Gamma_n(t)) = \inf_{\mathbf{u}\in\mathbb{R}^n} H(\mathbf{u})$.

Proof. "⇐": take $\mathbf{u} = \Gamma_n(t)$. "⇒": if $H(\mathbf{u}_0) < c$, then by continuity $H < c$ on a cube $\|\mathbf{u} - \mathbf{u}_0\|_\infty < \eta$, and by Theorem 4.4 the curve $\Gamma_n((M,\infty))$ enters it. Equality of the infima: the curve lies in $\mathbb{R}^n$, so the inf over the curve $\ge$ inf over $\mathbb{R}^n$; and for every $c > \inf_{\mathbb{R}^n} H$ the first part gives a point of the curve with $H < c$. ■

For a **non-strict** inequality the analogous equivalence for an arbitrary dense set is false: $D = \mathbb{R}\setminus\{0\}$ is dense in $\mathbb{R}$, and $H(z) = z^2$ has the same infimum 0 on D and on $\mathbb{R}$, and yet $H(z) \le 0$ has a solution only in $\mathbb{R}$. That is why we built the gap first (§3).

Definition 5.2

$G(y,t) = F(y, \Gamma_n(t))$ (substitution of n expressions for n variables) and $G_a(t) = G(a,t)$ for $a \in \mathbb{N}$. Thus $G \in \mathcal{A}_2$, $G_a \in \mathcal{A}_1$, and G_a is obtained from a algorithmically (by substituting the number a for y).

Theorem 5.3 (dichotomy in one variable; Richardson, Corollary to Theorem Two)

Every function G_a is nonnegative. Moreover:

1. if $a \in U$, then for every $\varepsilon > 0$ and $M \geq 0$ there is $t > M$ with $G_a(t) < \varepsilon$;
2. if $a \notin U$, then $G_a(t) > 1$ for every $t \in \mathbb{R}$.

In particular, $a \in U \iff \exists t\, G_a(t) < 1 \iff \exists t\, G_a(t) \leq 1 \iff \inf_t G_a(t) = 0$.

Proof. Nonnegativity: $F \geq 0$. If $a \in U$, Corollary 3.7 gives $\mathbf{u}_0$ with $F(a, \mathbf{u}_0) = 0$; Lemma 5.1 for $H = F(a, \cdot)$ and $c = \varepsilon$ gives the claim. If $a \notin U$, then $F(a, \mathbf{u}) > 1$ at **every** point, in particular at the points $\Gamma_n(t)$. ■

Remark 5.4 (why we do not claim that G_a has a zero)

A zero of the function $F(a, \cdot)$ yields arbitrarily small values of G_a (density + continuity), but not a zero of G_a — the curve may miss that point. The arithmetic information is carried by the sharp threshold 1, protected by the rounding theorem. The order of the construction matters: first the gap (§3), then the curve (§4). In the direction "$G_a(t) \leq 1 \Rightarrow a \in U$" we use not density but only the gap: the point $\Gamma_n(t)$ is itself a witness of $F \leq 1$.

The whole chain on a toy example

The actual polynomial from Theorem 1.1 is huge. We can, however, trace **the same construction** on a "Diophantine equation" in two variables: $x_1^2 + x_2^2 = 5$ (which has the solution $(1, 2)$) and $x_1^2 + x_2^2 = 3$ (which has none). We build the weights, the penalty f, F and $G(t) = F(\Gamma_2(t))$, and then — in the "yes" case — we look for a point with $G(t) < 1$ by aiming the curve at the zero of F, i.e., at $\mathbf{u} = (1, \sqrt{2})$ (since f vanishes at $\mathbf{x} = (1, 2) = (u_1^2, u_2^2)$).

```
mp.mp.dps = 120
x1, x2, u1, u2 = sp.symbols('x1 x2 u1 u2')

def lancuch(P, n=2):
    """Wagi k_i, kara f i funkcja F(u) = f(u1², u2²) — dokładnie według §2–§3."""
    Q = P**2
    k = [majoranta(sp.diff(Q, xi)) for xi in (x1, x2)]
    f = (n + 1)**4 * (P**2 + sum(sp.sin(sp.pi*xi)**2 * ki**4 for xi, ki in zip((x1, x2),
 ↪  k)))
    F = f.subs({x1: u1**2, x2: u2**2}, simultaneous=True)
    return k, sp.lambdify((u1, u2), F, 'mpmath')

for opis, P, cel in [("x1²+x2²=5 (TAK)", x1**2 + x2**2 - 5, (mp.mpf(1), mp.sqrt(2))),
                     ("x1²+x2²=3 (NIE)", x1**2 + x2**2 - 3, (mp.mpf(1), mp.mpf(1)))]:
    k, F = lancuch(P)
    print(opis, "   waga k1 =", sp.factor(k[0]))
    for delta in ('1e-3', '1e-8', '1e-10'):
        t, _ = podnies(cel[0], cel[1], mp.mpf(delta))
        print(f"   δ = {delta:>6}:  t ≈ {mp.nstr(t, 12):>16}   G(t) = {mp.nstr(F(h(t),
         ↪  g(t)), 8)}")
```

```
x1²+x2²=5 (TAK)    waga k1 = 18*(x1**2 + 2)*(x1**4 + 4*x1**2 + x2**4 + 4*x2**2 + 35)
   δ =   1e-3:  t ≈    12575.7953128   G(t) = 2.524194e+10
   δ =   1e-8:  t ≈    1256637070.86   G(t) = 2.8217838
   δ =  1e-10:  t ≈   125663706279.0   G(t) = 0.00023293165
x1²+x2²=3 (NIE)    waga k1 = 18*(x1**2 + 2)*(x1**4 + 4*x1**2 + x2**4 + 4*x2**2 + 19)
   δ =   1e-3:  t ≈    12575.7953128   G(t) = 6.6458173e+8
   δ =   1e-8:  t ≈    1256637070.86   G(t) = 81.074264
```

```
δ =  1e-10:  t ≈   125663706279.0   G(t) = 81.000006
```

For the solvable equation a point with $G(t) < 1$ exists, but our construction finds one only at $t \approx 1.26 \cdot 10^{11}$, and it has to be computed with more than a hundred digits of precision (the phase is $t^3 \approx 2 \cdot 10^{33}$). For the unsolvable equation, even the "best attempt" gives $G \approx 81 > 1$ — in accordance with the theorem, which guarantees $G > 1$ **everywhere**. The reduction is effective as a transformation of expressions, but the witnesses can be astronomically far away: yet another reason why numerical "checking" cannot replace deciding.

See also

A larger example — the family of expressions from the equation $x^3 + y^3 + z^3 = a$, including a case in which the witness has about 2770 digits and a case in which nobody knows the answer — is in "Example: an expression whose sign cannot be seen".

6. The sign problem

Theorem 6.1 (Theorem R, part A; Richardson, §2 Theorem One)

There is no algorithm that, given $A \in \mathcal{A}_1$, decides whether $\exists t\ A(t) < 0$.

Proof. For $a \in \mathbb{N}$ we algorithmically build $A_a(t) = G_a(t) - 1$. By Theorem 5.3, $a \in U \iff \exists t\ A_a(t) < 0$. An algorithm for the sign problem would decide U, contrary to Theorem 1.1. ■

The hard instances contain no absolute value: every A_a is analytic and defined everywhere. By Theorem 5.3, the reduction also works when the witness is sought only among $t > 0$ or $t > M$. Replacing A by $-A$ gives the undecidability of the question "is A positive somewhere?", and swapping the answers, that of the question "is $A \geq 0$ everywhere?".

7. The identity (zero) problem

The absolute value turns a question about the **sign** into a question about **identity**:

$$|s| - s = \begin{cases} 0, & s \geq 0, \\ -2s > 0, & s < 0. \end{cases}$$

Theorem 7.1 (Theorem R, part B; Richardson, §2 Theorem Two)

There is no algorithm that, given $B \in \mathcal{B}_1$, decides whether $B \equiv 0$.

Proof. Let $B_a(t) = |G_a(t) - 1| - (G_a(t) - 1) = |A_a(t)| - A_a(t)$. If $a \notin U$, then $G_a > 1$ everywhere, so $B_a \equiv 0$. If $a \in U$, then $G_a(t_0) < 1$ at some t_0, so $B_a(t_0) = 2(1 - G_a(t_0)) > 0$. Hence $B_a \equiv 0 \iff a \notin U$; an algorithm for the identity problem (with the answer swapped) would decide U. ■

Corollary 7.2 (equality and canonical forms)

Equality of the functions defined by two expressions in $\mathcal{B}_1$ is undecidable. In particular, there is **no** algorithm assigning to every expression in $\mathcal{B}_1$ a finite canonical form (such that the forms are equal if and only if the functions are equal).

Proof. $B \equiv C \iff B - C \equiv 0$. A canonical-form algorithm would decide equality by comparing two strings. ■

This is the precise content of the conclusion in Lecture 1 that every `simplify` function must fail on some expressions: it fails to recognize zero, gives a wrong answer, or does not terminate (see "How does a computer algebra system work?").

Remark 7.3 (the role of the absolute value)

The absolute value is the bridge from sign to identity. For the purely analytic language $\mathcal{A}_1$ this proof does **not** give undecidability of identity: an analytic function vanishing on an interval vanishes everywhere (Theorem 4.2 in "Preliminaries"), so a "plateau of zeros" like $|s| - s$ is impossible. Unlike analytic functions, functions with absolute value have zero divisors: $(|x| + x)(|x| - x) \equiv 0$, although neither factor is identically zero.

8. The integration problem

We ask whether, for a given $J \in \mathcal{B}_1$, there exists an **expression** $H \in \mathcal{B}_1$, differentiable on $\mathbb{R}$, with $H' = J$ — a question about a *representable* antiderivative, not about the existence of an antiderivative at all (for continuous J, one always exists: $\int_0^x J$).

Definition 8.1 (local obstruction to integration)

A function Δ defined on $\mathbb{R}$ and represented in a language $\mathcal{E}$ is a **local obstruction** if there exist no $H \in \mathcal{E}$ and no nonempty open interval I on which H is differentiable and $H' = \Delta$.

In "Appendices: powers of two and the integral of e^{x^2}" (Appendix B) we prove that $\Delta(x) = e^{x^2}$ is a local obstruction for $\mathcal{B}_1$ (Liouville's theorem plus local removal of absolute values).

We need an expression that in the "no" case is identically zero, and in the "yes" case equals **exactly** 1 **on some interval**. Richardson builds it from truncated subtraction:

$$r \dot{-} s = \frac{|r - s| + (r - s)}{2} = \max\{r - s, 0\}, \qquad m(r, s) = r - (r \dot{-} s) = \min\{r, s\},$$

$$S_a(t) = m(1,\ 2 \dot{-} 2G_a(t)) = \min\{1, \max\{0,\ 2 - 2G_a(t)\}\} = \begin{cases} 1, & G_a(t) \le \frac{1}{2}, \\ 2 - 2G_a(t), & \frac{1}{2} \le G_a(t) \le 1, \\ 0, & G_a(t) \ge 1. \end{cases}$$

$S_a \in \mathcal{B}_1$ is continuous and takes values in $[0, 1]$.

Lemma 8.2 (the switch)

If $a \notin U$, then $S_a \equiv 0$. If $a \in U$, then $S_a = 1$ on some nonempty open interval.

Proof. In the first case $G_a > 1$ everywhere. In the second, Theorem 5.3 gives t_0 with $G_a(t_0) < \frac{1}{2}$ (here we need the full strength of "$\inf G_a = 0$", not just "$G_a < 1$ somewhere"), and by continuity $G_a < \frac{1}{2}$ on a neighborhood of t_0. ■

Theorem 8.3 (Theorem R, part C; Richardson, §2 Theorem Three)

Let a language $\mathcal{E}$ of everywhere-defined functions effectively contain $\mathcal{B}_1$, be effectively closed under multiplication, and contain a local obstruction Δ. Then there is no algorithm deciding whether a given $J \in \mathcal{E}$ has an antiderivative in $\mathcal{E}$. In particular, this applies to $\mathcal{B}_1$ with $\Delta = e^{x^2}$.

Proof. Let $J_a = \Delta \cdot S_a$. If $a \notin U$, then $J_a \equiv 0$, and the constant function 0 is an antiderivative of it in $\mathcal{E}$. If $a \in U$, then on some interval I we have $S_a = 1$, i.e., $J_a = \Delta$; an antiderivative $H \in \mathcal{E}$

of J_a would satisfy $H' = \Delta$ on I, contrary to the definition of an obstruction. Hence J_a has an antiderivative in $\mathcal{E}$ if and only if $a \notin U$. ■

Remark 8.4 (the problem is global)

For $a \in U$ the function J_a may vanish on some other interval — there the antiderivative 0 is elementary. The obstruction lies on the interval where $S_a = 1$. We ask for **one** expression that works on the whole line.

9. Richardson's original, general formulation

Richardson works with an arbitrary set E of expressions representing **partial** functions, and with the set of functions E^*. He assumes that E^* contains the identity and the rational constants, and that for $A, B \in E$ one can **algorithmically** build expressions for $A \pm B$, $A \cdot B$, $A(B(x))$. He understands the equality $f \equiv g$ with domains taken into account (Definition 3.3 in "Preliminaries"). The conditions:

- **(R1)** E^* contains the constants π, $\log 2$ and the functions e^x, $\sin x$;
- **(R2)** there is $\mu \in E^*$ with $\mu(x) = |x|$ for $x \neq 0$ (the value $\mu(0)$ is arbitrary or undefined);
- **(R3)** there is an everywhere-defined $\Delta \in E^*$ such that for no $f \in E^*$ and no interval I does $f' \equiv \Delta$ hold on I.

Theorem 9.1 (Richardson, 1968)

Under the above effectiveness assumptions:

1. (R1) ⇒ the question whether an expression takes a negative value at some point of its domain is undecidable;
2. (R1)+(R2) ⇒ the identity problem $A \equiv 0$ is undecidable;
3. (R1)+(R2)+(R3) ⇒ the integration problem $\exists f \in E^*\ f' \equiv A$ is undecidable.

Proof. The expressions G_a from §5 have representatives in E: we fix, once and for all, expressions for the finitely many constants and basic functions used in the construction (x, e^x, $\sin x$), and apply the effective constructors along the tree; the number a is built from 1 by addition. This gives (1) via $G_a - 1$.

(2): let $\widetilde{B}_a = \mu(G_a - 1) - (G_a - 1)$. If $a \notin U$, the argument of μ is never zero, so $\widetilde{B}_a$ is defined everywhere and equal to 0. If $a \in U$, then at the point t_0 with $G_a(t_0) < 1$ the argument is nonzero and $\widetilde{B}_a(t_0) = 2(1 - G_a(t_0)) > 0$; hence $\widetilde{B}_a \not\equiv 0$ regardless of the behavior at other points.

(3): in the cutoff we replace the absolute value by μ: $q_a = 2 - 2G_a$, $T_a = \frac{\mu(q_a)+q_a}{2}$, $\tilde{S}_a = 1 - \frac{\mu(1-T_a)+(1-T_a)}{2}$. If $a \notin U$: $q_a < 0$, so $T_a = 0$ everywhere, and the outer μ receives 1, so $\tilde{S}_a$ is the everywhere-defined zero function. If $a \in U$: on the interval where $G_a < \frac{1}{2}$ we have $q_a > 1$, $T_a = q_a$, $1 - T_a < 0$, all arguments of μ are nonzero, and $\tilde{S}_a = 1$. The rest is as in Theorem 8.3, with $\tilde{J}_a = \Delta\, \tilde{S}_a$ (the fact that Δ is defined everywhere guarantees that in the "no" case $\tilde{J}_a$ is the everywhere-defined zero function). ■

At the thresholds $G_a = 1$ and $G_a = \frac{1}{2}$ the expressions involving μ may be undefined — the proof avoids them: in the "no" case $G_a > 1$ everywhere, and in the "yes" case we use an interval where $G_a < \frac{1}{2}$.

10. What exactly we have proved

An effectiveness audit. The reduction algorithm performs only operations on expressions: it takes the fixed P, forms $Q = P^2$, differentiates, builds the weights by the rules of Lemma 2.1, assembles f, substitutes the squares, then the coordinates of the curve, and finally the number a for y. It never searches for roots, points of the curve, minima or signs of constants. These objects occur only in the **proof** that the constructed expression has the required property.

> **Proposition 10.1 (negativity is semi-decidable)**
>
> There is an algorithm that, given $A \in \mathcal{B}_1$, halts if and only if $A(t) < 0$ for some t.

Proof. The values of the constants, exp, sin and $|\cdot|$ at rational points can be approximated with a guaranteed error, and finite composition preserves this property. We enumerate the pairs $(q, k) \in \mathbb{Q} \times \mathbb{N}$ and for each compute a rational $r_{q,k}$ with $|r_{q,k} - A(q)| \le 2^{-k}$; we halt when $r_{q,k} + 2^{-k} < 0$. The test never errs. If $A(t_0) < 0$, then by continuity $A < 0$ on a neighborhood of t_0, hence at some rational q; for large k the test succeeds. ■

The same method semi-decides **not being identically zero**. By Post's theorem (Theorem 1.5 in "Preliminaries") and Theorem 7.1, **being identically zero is not even semi-decidable** — otherwise, combining the two procedures, we could decide it. Here is a working semi-decision procedure using interval arithmetic:

```
from fractions import Fraction
from mpmath import iv
iv.dps = 30

def szukaj_ujemnej_wartosci(A, maks_krokow=10_000):
    """Półrozstrzyganie: przegląda wymierne q (coraz gęściej, coraz dalej) i zwraca q
    z certyfikatem A(q) < 0 (górny kraniec przedziału ujemny) albo None po limicie
 ↪  kroków."""
    krok = 0
    for poziom in range(1, 60):                    # siatka o kroku 2^-poziom na
     ↪  [-poziom, poziom]
        mianownik = 2**poziom
        for licznik in range(-poziom * mianownik, poziom * mianownik + 1):
            krok += 1
            if krok > maks_krokow:
                return None
            q = Fraction(licznik, mianownik)
            wartosc = A(iv.mpf(q.numerator) / q.denominator)
            if wartosc.b < 0:                      # cały przedział < 0: certyfikat
                return q, wartosc

print(szukaj_ujemnej_wartosci(lambda t: t**2 - 2*t + iv.mpf('0.99')))   # minimum −0,01 w
 ↪  t = 1
print(szukaj_ujemnej_wartosci(lambda t: t**2 - 2*t + 1))                # (t − 1)² ≥ 0:
 ↪  brak certyfikatu
```

```
(Fraction(1, 1), mpi('-0.010000000000000000000000000000907',
 ↪  '-0.009999999999999999999999999999211'))
None
```

In the first example the procedure found a certificate at $q = 1$: the whole interval of values lies below zero. In the second, the result `None` means only that the step limit was exhausted, not the answer "no": without the limit, the procedure would search for a certificate forever. Richardson's theorem says that no such procedure can be supplemented with an opposite procedure that halts exactly when $A \ge 0$ everywhere (cf. Problem 8 in "Review questions and problems").

Three different questions about zero.

Question	Logical form	What we know
identically zero	$\forall t\ A(t) = 0$	undecidable in $\mathcal{B}_1$ (Theorem 7.1)
existence of a zero	$\exists t\ A(t) = 0$	undecidable (Wang 1974; Laczkovich 2003) — does not follow from the density of the curve alone
whether a constant is zero	$c = 0$, no variable	**open problem**; decidable for exponential-logarithmic constants assuming Schanuel's conjecture (Richardson 1997)

Richardson wrote in 1968 (p. 515): "No method is known for deciding whether a subelementary expression with no variables in it represents a number less than zero. Whether or not such a method exists is an open question." More in "Context, consequences and further results".

11. The modern version: polynomials and sine suffice

Matiyasevich's theorem (Theorem 2.4 in "Preliminaries") allows us to take as P an ordinary polynomial $p(y, \mathbf{x}) \in \mathbb{Z}[y, \mathbf{x}]$ with an undecidable set U. Then the derivatives of $Q = p^2$ are polynomials, and the weights can be taken polynomial: if $\partial Q/\partial x_i = \sum_\alpha c_{i,\alpha}\, y^{\alpha_0} x_1^{\alpha_1} \cdots x_n^{\alpha_n}$, then

$$k_i(y, \mathbf{x}) = 2 + \sum_{\alpha} |c_{i,\alpha}|\,(y^2+2)^{\alpha_0} \prod_{j=1}^{n} (x_j^2+2)^{\alpha_j}$$

satisfies (2.1) (the coefficients are known integers, so we compute $|c_{i,\alpha}|$ right away — this is not the absolute-value operation of the language). The penalty, the squares and the curve h, g use only arithmetic operations and the sine.

Corollary 11.1 (Caviness 1970, after Matiyasevich)

Let $\mathcal{S}_1$ be the language generated by the rational numbers, π, the variable x, the operations $+, -, \cdot$ and the function sin. The sign problem is undecidable for $\mathcal{S}_1$; after adding the absolute value, the identity problem is undecidable too.

Proof. We repeat §3–§5 with the polynomial P and the above weights, and then §6–§7 without change. ■

Laczkovich (2003) also removed π: in the ring generated by the integers and the functions x, $\sin x^n$, $\sin(x \sin x^n)$ $(n = 1, 2, \ldots)$ it is undecidable whether a function takes a positive value and whether it has a root. See "Context, consequences and further results".

Example: an expression whose sign cannot be seen

Richardson's theorem says that **there is no algorithm** deciding whether an expression is negative somewhere. What does this look like in a concrete example? We take the machine from the proof ("The complete proof, step by step", §2–§5) and feed it the equation

$$x_1^3 + x_2^3 + x_3^3 = a \qquad (x_1, x_2, x_3 \in \mathbb{Z}).$$

For each number a the machine produces an explicit expression $A_a(t)$ in one variable, built solely from integers, π, t, the operations $+, -, \cdot$ and the sine. We ask: **is $A_a(t) \geq 0$ for all t?** — for four values: $a = 3,\ 4,\ 42,\ 114$.

Numerics gives the same answer in all four cases. The truth, however, varies:

- $a = 4$: yes, $A_4 > 0$ everywhere — the proof takes three lines of number theory;
- $a = 3$: no — but the place where $A_3 < 0$ lies around $t \approx 10^{13}$, and t has to be given to about a hundred digits;
- $a = 42$: no — this has been known only since 2019, and a suitable t has about 2770 significant digits;
- $a = 114$: **nobody knows**.

```
import random
import signal
import time
from itertools import product
import numpy as np
import sympy as sp
import mpmath as mp
import matplotlib.pyplot as plt
```

1. The expression $A_a(t)$

We follow the proof exactly, with one change: the unknowns are **integers** (of either sign), not natural numbers.

- **Equation:** $P_a(\mathbf{x}) = x_1^3 + x_2^3 + x_3^3 - a$, $n = 3$ unknowns.
- **Weights** (Definition 2.4): $k_i = M_{\partial Q/\partial x_i}$ for $Q = P_a^2$. Since $\partial Q/\partial x_i = 6x_i^2 P_a$, Lemma 2.1 gives

$$k_i(\mathbf{x}) = 38\,(x_i^2 + 2)^2\, M_a(\mathbf{x}), \qquad M_a(\mathbf{x}) = \sum_{j=1}^{3} (x_j^2 + 2)^3 + a^2 + 2.$$

- **Penalty** (Definition 3.2): $f_a(\mathbf{x}) = 4^4[P_a(\mathbf{x})^2 + \sum_{i=1}^{3} \sin^2(\pi x_i)\, k_i(\mathbf{x})^4]$.
- **Curve** (Definition 4.3): $\Gamma_3(t) = (h(t),\, h(g(t)),\, g(g(t)))$, where $h(t) = t \sin t$, $g(t) = t \sin t^3$.
- **Result** (Definition 5.2): $G_a(t) = f_a(\Gamma_3(t))$ and $A_a(t) = G_a(t) - 1$.

Written out: for $s = t\sin t^3$ and $u_1 = t\sin t$, $u_2 = s\sin s$, $u_3 = s\sin s^3$,

$$A_a(t) = 256\Big[\big(u_1^3+u_2^3+u_3^3-a\big)^2 + \sum_{i=1}^{3}\sin^2(\pi u_i)\,\big(38\,(u_i^2+2)^2 M\big)^4\Big] - 1,$$

where $M = \sum_{j=1}^{3}(u_j^2+2)^3 + a^2 + 2$.

Proposition P.1 (Theorem 5.3 for integer unknowns)

Let $a \in \mathbb{Z}$.

1. If $a = m_1^3+m_2^3+m_3^3$ for some $m_i \in \mathbb{Z}$, then for every $\varepsilon > 0$ and $M \geq 0$ there is $t > M$ with $A_a(t) < -1+\varepsilon$; in particular, $A_a(t) < 0$ for some t.
2. If there are no such m_i, then $A_a(t) > 0$ for **every** $t \in \mathbb{R}$.

Proof. The proof of Theorem 3.4 carries over word for word: in Step 2 we round x_i to the nearest integer (of either sign), and in Step 4 the number $P_a(\mathbf{m})$ is an integer because P_a has integer coefficients. We get $f_a(\mathbf{x}) > 1$ for all $\mathbf{x} \in \mathbb{R}^3$ when there are no solutions — so the squares from Corollary 3.7 are not needed. When $P_a(\mathbf{m}) = 0$, we have $f_a(\mathbf{m}) = 0$, and the density of the curve (Theorem 4.4) and the continuity of f_a give points $t > M$ with $G_a(t) < \varepsilon$ — as in the proof of Theorem 5.3. ■

Let us check that the weights really are the majorants from Lemma 2.1 (the same function `majoranta` as in §2 of the proof):

```
def majoranta(e):
    """Majoranta lokalna M_e (Lemat 2.1), rekurencyjnie po drzewie wyrażenia."""
    if e.is_number or e.is_Symbol:
        return e**2 + 2
    if e.is_Add:
        return sp.Add(*[majoranta(c) for c in e.args])
    if e.is_Mul:
        return sp.Mul(*[majoranta(c) for c in e.args])
    if e.is_Pow and e.exp.is_Integer and e.exp > 0:
        return majoranta(e.base) ** e.exp
    if isinstance(e, sp.exp):
        return sp.exp(majoranta(e.args[0]))
    if isinstance(e, (sp.sin, sp.cos)):
        return sp.Integer(2)
    raise ValueError(f"węzeł spoza języka: {e.func}")

x = sp.symbols('x1:4')
for a in (3, 4, 42, 114):
    P = sum(xi**3 for xi in x) - a
    M = sum((xi**2 + 2)**3 for xi in x) + a**2 + 2
    for xi in x:
        assert sp.expand(majoranta(sp.diff(P**2, xi)) - 38 * (xi**2 + 2)**2 * M) == 0
print("wagi k_i = 38 (x_i² + 2)² M_a zgadzają się z Lematem 2.1 (a = 3, 4, 42, 114)")
```

```
wagi k_i = 38 (x_i² + 2)² M_a zgadzają się z Lematem 2.1 (a = 3, 4, 42, 114)
```

The weights agree with Lemma 2.1 identically (the comparison after expansion is symbolic, not numerical). Now let us build the whole expression $A_a(t)$ in SymPy and check that it indeed contains only integers, π, t, arithmetic operations and the sine:

```
t = sp.Symbol('t', real=True)

def A_symb(a):
```

```
    """A_a(t) = f_a(Γ_3(t)) - 1 jako wyrażenie SymPy."""
    s = t * sp.sin(t**3)                                  # s = g(t)
    u = (t * sp.sin(t), s * sp.sin(s), s * sp.sin(s**3))  # Γ_3(t)
    P = sum(ui**3 for ui in u) - a
    M = sum((ui**2 + 2)**3 for ui in u) + a**2 + 2
    k = [38 * (ui**2 + 2)**2 * M for ui in u]
    kara = sum(sp.sin(sp.pi * ui)**2 * ki**4 for ui, ki in zip(u, k))
    return 256 * (P**2 + kara) - 1

A42 = A_symb(42)
print("funkcje w A_42:", {f.func for f in A42.atoms(sp.Function)})
print("liczba operacji w drzewie A_42:", sp.count_ops(A42))
```

```
funkcje w A_42: {sin}
liczba operacji w drzewie A_42: 223
```

The only function in the tree is the sine, and the function `count_ops` counted 223 operations in it — the expression is large, but completely explicit. For the computations we use the same formula written out directly — once for ordinary numbers (`ctx=mp`), once for intervals (`ctx=mp.iv`):

```
def G(t, a, ctx=mp):
    """G_a(t) = f_a(Γ_3(t)); ctx = mp (liczby) albo mp.iv (przedziały)."""
    sin, pi = ctx.sin, ctx.pi
    s = t * sin(t**3)
    u = (t * sin(t), s * sin(s), s * sin(s**3))
    P = u[0]**3 + u[1]**3 + u[2]**3 - a
    M = sum((ui**2 + 2)**3 for ui in u) + a**2 + 2
    k = [38 * (ui**2 + 2)**2 * M for ui in u]
    return 256 * (P**2 + sum(sin(pi * ui)**2 * ki**4 for ui, ki in zip(u, k)))

def A(t, a):
    return G(t, a) - 1

# zgodność z wyrażeniem SymPy w kilku punktach
mp.mp.dps = 50
F42 = sp.lambdify(t, A42, 'mpmath')
for v in ('0.7', '2.5', '13.1'):
    blad = abs(F42(mp.mpf(v)) / A(mp.mpf(v), 42) - 1)
    print(f"t = {v}: błąd względny {mp.nstr(blad, 3)}")
```

```
t = 0.7: błąd względny 8.02e-51
t = 2.5: błąd względny 9.35e-51
t = 13.1: błąd względny 4.94e-50
```

The two versions agree at the three test points with a relative error of order 10^{-50}, i.e., at the limit of the precision we set (50 digits).

2. Four equations

a	solution in integers	known since
3	$1^3+1^3+1^3$ (also $4^3+4^3+(-5)^3$)	always
4	**none** (residues modulo 9)	elementary
42	$(-80538738812075974)^3 + 80435758145817515^3 + 12602123297335631^3$	A. Booker and A. Sutherland, 2019
114	**unknown**	the smallest open case

The solution for 42 was found after 1.3 million hours of computation on the Charity Engine distributed network; all numbers below 114 (except those ruled out by their residue modulo 9) already have known representations, and 114 is the smallest open case (as of September 2026; see the references at the end). Let us check what can be checked:

```
rozwiazania = {3: (1, 1, 1),
               42: (-80538738812075974, 80435758145817515, 12602123297335631)}
for a, m in rozwiazania.items():
    print(f"a = {a}: suma sześcianów {m} = {sum(v**3 for v in m)}")
reszty = {(p**3 + q**3 + r**3) % 9 for p, q, r in product(range(9), repeat=3)}
print("reszty modulo 9 sum trzech sześcianów:", sorted(reszty))
print("a mod 9:", {a: a % 9 for a in (3, 4, 42, 114)})
```

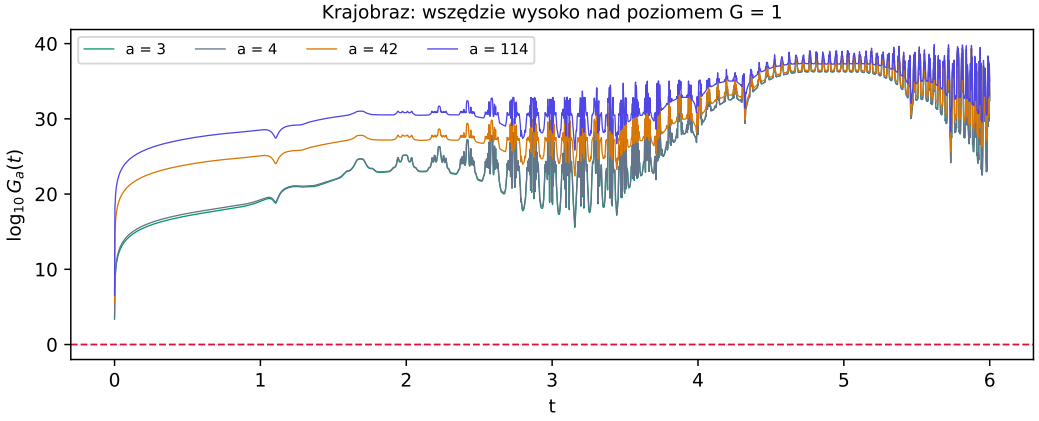
```
a = 3: suma sześcianów (1, 1, 1) = 3
a = 42: suma sześcianów (-80538738812075974, 80435758145817515, 12602123297335631) =
 ↪  42
reszty modulo 9 sum trzech sześcianów: [0, 1, 2, 3, 6, 7, 8]
a mod 9: {3: 3, 4: 4, 42: 6, 114: 6}
```

Cubes leave only the residues $0, \pm 1$ modulo 9, so a sum of three cubes cannot leave the residue 4 or 5. For $a = 4$ there are no solutions; for 42 and 114 the residue modulo 9 rules nothing out.

3. What numerics sees

Let us plot $\log_{10} G_a(t)$ on the interval $[0, 6]$; the dashed line is the level $G_a = 1$, i.e., $A_a = 0$:

```
mp.mp.dps = 40
ts = [mp.mpf(i) / 400 for i in range(2401)]
kolory = {3: '#059669', 4: '#64748b', 42: '#d97706', 114: '#4f46e5'}
fig, ax = plt.subplots(figsize=(8, 3.4))
for a in (3, 4, 42, 114):
    ys = [float(mp.log10(G(v, a))) for v in ts]
    ax.plot([float(v) for v in ts], ys, lw=0.7, color=kolory[a], label=f"a = {a}")
ax.axhline(0, color='crimson', lw=1, ls='--')
ax.set_xlabel('t')
ax.set_ylabel(r'$\log_{10} G_a(t)$')
ax.set_title('Krajobraz: wszędzie wysoko nad poziomem G = 1', fontsize=10)
ax.legend(fontsize=8, ncol=4)
fig.tight_layout()
plt.show()
```



On the whole interval $[0,6]$ all four graphs lie high above the level $G_a = 1$; the lowest value, at $t = 0$, is $G_3(0) = 2304$, i.e., $\log_{10} G_3 \approx 3.4$. Let us also check a dense grid on $[0,20]$ and 3000 random points from $[0,10^6]$, working with a precision of 90 digits so that $\sin t^3$ and $\sin s^3$ are computed correctly:

```
random.seed(2026)
mp.mp.dps = 90
siatka = [mp.mpf(i) / 500 for i in range(10001)]                # [0, 20], krok 0,002
losowe = [mp.mpf(random.uniform(0, 1e6)) for _ in range(3000)]  # z [0, 10^6]
print("   a | najmniejsze G_a na siatce | w punktach losowych")
for a in (3, 4, 42, 114):
    m1, m2 = min(G(v, a) for v in siatka), min(G(v, a) for v in losowe)
    print(f"{a:>4} | {mp.nstr(m1, 4):>25} | {mp.nstr(m2, 4):>19}")
tmin = min(losowe, key=lambda v: G(v, 3))
r4 = G(tmin, 4) / G(tmin, 3) - 1
r114 = G(tmin, 114) / G(tmin, 3) - 1
print("w tym punkcie losowym:  G_4/G_3 − 1 =", mp.nstr(r4, 3),
      "  G_114/G_3 − 1 =", mp.nstr(r114, 3))
```

```
   a | najmniejsze G_a na siatce | w punktach losowych
   3 |                    2304.0 |           1.401e+92
   4 |                    4096.0 |           1.401e+92
  42 |                   4.516e+5 |           1.401e+92
 114 |                   3.327e+6 |           1.401e+92
w tym punkcie losowym:  G_4/G_3 − 1 = 7.64e-12   G_114/G_3 − 1 = 1.42e-8
```

The smallest value on the grid is $G_a(0) = 256a^2$: the curve passes through the lattice point $(0,0,0)$, but there $P_a = -a \neq 0$. At the random points the smallest values agree for all a to many digits — for large t the constant a is negligible compared with u_i^3.

Numerics says the same thing for all four equations: A_a is positive, and hugely so. If we guessed on this basis, we would conclude that none of the four equations has solutions — and we would be wrong at least twice (for $a = 3$ and $a = 42$).

4. The floating-point trap

The same formula in ordinary `float64` arithmetic:

```
def G64(t, a):
    """Ta sama formuła w arytmetyce float64 (NumPy)."""
    with np.errstate(over='ignore', invalid='ignore'):
        s = t * np.sin(t**3)
        u = (t * np.sin(t), s * np.sin(s), s * np.sin(s**3))
        P = sum(ui**3 for ui in u) - a
        M = sum((ui**2 + 2)**3 for ui in u) + a**2 + 2
        k = [38 * (ui**2 + 2)**2 * M for ui in u]
        kara = sum(np.sin(np.pi * ui)**2 * ki**4 for ui, ki in zip(u, k))
        return 256 * (P**2 + kara)

mp.mp.dps = 120
print("       t |     float64 | mpmath, 120 cyfr")
for v in (2.5, 50.0, 1e3, 1e5, 1e7):
    print(f"{v:>8g} | {G64(np.float64(v), 3):.4e} | {mp.nstr(G(mp.mpf(v), 3), 5)}")
```

```
       t |     float64 | mpmath, 120 cyfr
     2.5 | 6.2990e+21 | 6.299e+21
      50 | 1.1048e+70 | 1.1048e+70
    1000 | 5.0285e+124 | 5.0285e+124
```

```
100000 | 2.5697e+202 | 6.1013e+202
 1e+07 | 8.0636e+282 | 5.0512e+280
```

For small t the results agree. For $t = 10^5$ the number $t^3 = 10^{15}$ is still exact in `float64` ($10^{15} < 2^{53} \approx 9 \cdot 10^{15}$), and $\sin t^3$ comes out correct. However, $s = t \sin t^3$ is no longer exact: its error, of order 10^{-12}, produces an error of about 0.05 in $s^3 \approx 6 \cdot 10^{14}$, and $\sin s^3$ depends on s^3 modulo 2π. Beyond that, `float64` computes the sines of "different" numbers than it should.

5. A certificate for $a = 3$

Since $1^3 + 1^3 + 1^3 = 3$, Proposition P.1 guarantees negative values of A_3. Where? The proof is constructive: Theorem 4.4 for $n = 3$ amounts to applying Lemma 4.2 twice — first s with $\Gamma_2(s) \approx (1, 1)$, then w with $h(w) \approx 1$ and $g(w) = s$; then $\Gamma_3(w) = (h(w), \Gamma_2(g(w))) \approx (1, 1, 1)$.

How close do we have to get? At the point $(1, 1, 1)$ the weights are $k_i = 31464$, so $G_3 \approx 256 \sum_i \pi^2 \varepsilon_i^2 k_i^4$ for deviations ε_i — we need $|\varepsilon_i| \lesssim 10^{-11}$. We take $\delta = 10^{-12}$; we will check the result rigorously at the end.

```
h = lambda t: t * mp.sin(t)
g = lambda t: t * mp.sin(t**3)
dh = lambda t: mp.sin(t) + t * mp.cos(t)
dg = lambda t: mp.sin(t**3) + 3 * t**3 * mp.cos(t**3)

def zero(F, dF, lo, hi, tol):
    """Miejsce zerowe F na [lo, hi], gdy F zmienia znak: Newton z bisekcją."""
    flo, t = F(lo), (lo + hi) / 2
    while hi - lo > tol:
        ft = F(t)
        if (ft > 0) == (flo > 0):
            lo, flo = t, ft
        else:
            hi = t
        krok = ft / dF(t)
        nowy = t - krok
        if not (lo < nowy < hi) or abs(krok) > (hi - lo) / 2:
            nowy = (lo + hi) / 2               # krok Newtona niepewny — bisekcja
        elif abs(krok) < tol:
            return nowy
        t = nowy
    return (lo + hi) / 2

def podnies(a, b, delta, tol, M=0):
    """Lemat 4.2: w > M, |h(w) - a| < δ, |g(w) - b| ≤ tol (w dowodzie g(w) = b).
    Przedziały poszukiwań są dokładnie takie jak w dowodzie lematu."""
    a, b, delta, pi = mp.mpf(a), mp.mpf(b), mp.mpf(delta), mp.pi
    R = max(mp.mpf(M), abs(b), mp.mpf(1), 4 * pi / delta) * (1 + mp.mpf('1e-9'))
    dolna = mp.sqrt(R**2 + delta / 2)
    k = int(mp.ceil((max(dolna, abs(a)) - pi / 2) / (2 * pi))) + 1
    tol_v = mp.mpf(10)**(-mp.mp.dps // 2)
    v = zero(lambda t: h(t) - a, dh, 2*pi*k + pi/2, 2*pi*k + 3*pi/2, tol_v)
    u = mp.sqrt(v**2 - delta / 2)
    assert (v - u) * (v + 1) < delta / 2 and v**3 - u**3 > 2 * pi   # (4.1), (4.2)
    nast = lambda c: c + 2 * pi * mp.ceil((u**3 - c) / (2 * pi))
    tp, tm = mp.cbrt(nast(pi / 2)), mp.cbrt(nast(3 * pi / 2))
    lo, hi = min(tp, tm), max(tp, tm)
    w = zero(lambda t: g(t) - b, dg, lo, hi, tol / (3 * hi**3) / 10)
    assert abs(h(w) - a) < delta and abs(g(w) - b) <= tol
```

```
    return w

def podnies3(cel, delta, tol):
    """Twierdzenie 4.4 dla n = 3; błąd |g(w) - s| mnoży się w g(s) przez ≈ 3s³."""
    c1, c2, c3 = map(mp.mpf, cel)
    s = podnies(c2, c3, delta, tol)
    return podnies(c1, s, delta, tol / (3 * abs(s)**3) / 10)

M = sum((v**2 + 2)**3 for v in (1, 1, 1)) + 3**2 + 2
print("wagi w punkcie (1, 1, 1):", [38 * (v**2 + 2)**2 * M for v in (1, 1, 1)])
mp.mp.dps = 260
t0 = podnies3((1, 1, 1), mp.mpf('1e-12'), mp.mpf('1e-14'))
print("t₀ ≈", mp.nstr(t0, 30))
odchylenia = [v - 1 for v in (h(t0), h(g(t0)), g(g(t0)))]
print("Γ_3(t₀) − (1, 1, 1) =", [mp.nstr(d, 3) for d in odchylenia])
print("A_3(t₀) =", mp.nstr(A(t0, 3), 12))
```

```
wagi w punkcie (1, 1, 1): [31464, 31464, 31464]
t₀ ≈ 12566370639513.9053311440479139
Γ_3(t₀) − (1, 1, 1) = ['1.09e-13', '1.41e-13', '-3.53e-81']
A_3(t₀) = -0.999921346614
```

The deviations of the first two coordinates from 1 are of order 10^{-13}, the third is practically exact, and $A_3(t_0) \approx -0.99992$. The number $A_3(t_0)$ computed in 260-digit arithmetic is not yet a proof. The proof is provided by **interval arithmetic**: every operation rounds outward, so the result is an interval that is guaranteed to contain the true value. We write t_0 as a string with 110 significant digits — this is our certificate:

```
t0_napis = mp.nstr(t0, 110)
print("t₀ =", t0_napis)
for dps in (220, 400):
    mp.iv.dps = dps
    I = G(mp.iv.mpf(t0_napis), 3, ctx=mp.iv)
    print(f"{dps} cyfr: G_3(t₀) ∈ {mp.iv.nstr(I, 10)};  A_3(t₀) < 0: {I.b < 1}")
```

```
t₀ = 12566370639513.90533114404791391092572004548277467959535937378558282193200382080
↪  48777993365314271100762109192 21
220 cyfr: G_3(t₀) ∈ [7.865338619e-5, 7.865338619e-5];  A_3(t₀) < 0: True
400 cyfr: G_3(t₀) ∈ [7.865338619e-5, 7.865338619e-5];  A_3(t₀) < 0: True
```

At both precisions the interval is narrow and lies below 1 ($G_3(t_0) \approx 7.87 \cdot 10^{-5}$), so $A_3(t_0) < 0$ is proved. A hundred digits is no exaggeration — here is what happens when we round t_0 more coarsely:

```
mp.iv.dps = 400
for cyfry in (60, 80, 100, 110):
    I = G(mp.iv.mpf(mp.nstr(t0, cyfry)), 3, ctx=mp.iv)
    gora = mp.nstr(mp.mpf(I.b.b), 4)
    print(f"t₀ z {cyfry:>3} cyframi: G_3 ≤ {gora:>11};  A_3 < 0: {I.b < 1}")
```

```
t₀ z  60 cyframi: G_3 ≤  7.852e+497;  A_3 < 0: False
t₀ z  80 cyframi: G_3 ≤  1.841e+339;  A_3 < 0: False
t₀ z 100 cyframi: G_3 ≤     0.08948;  A_3 < 0: True
t₀ z 110 cyframi: G_3 ≤    7.865e-5;  A_3 < 0: True
```

With 60 and 80 digits the rounded point already lies outside the interval on which $A_3 < 0$: there G_3 is of order 10^{497} and 10^{339}, respectively (the resulting intervals are narrow, so $A_3 > 0$ at these points); from 100 digits on, the upper endpoint drops below 1, i.e., $A_3 < 0$ is proved. How wide is the interval around t_0 on which $A_3 < 0$? And what will `float64` see?

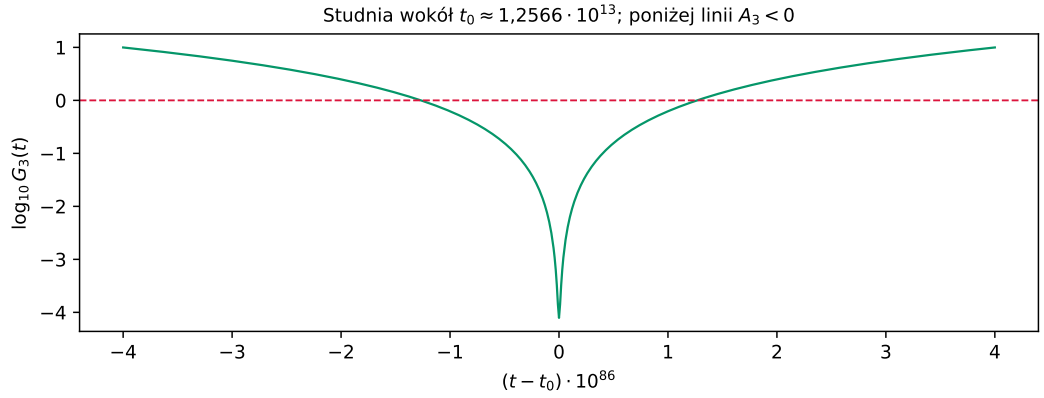
```
def brzeg(kierunek):
    """Odległość od t₀ do miejsca, w którym A_3 wraca do zera."""
    krok = mp.mpf('1e-100')
    while A(t0 + kierunek * krok, 3) < 0:          # oddalamy się, aż A_3 ≥ 0
        krok *= 2
    lo, hi = mp.mpf(0), krok
    for _ in range(200):                           # potem bisekcja
        mid = (lo + hi) / 2
        lo, hi = (mid, hi) if A(t0 + kierunek * mid, 3) < 0 else (lo, mid)
    return lo

szer = brzeg(-1) + brzeg(+1)
print("długość przedziału wokół t₀, na którym A_3 < 0:", mp.nstr(szer, 3))
print("liczba punktów siatki o takim kroku w [0, t₀]:", mp.nstr(t0 / szer, 2))
t64 = float(t0)
print("float64 w t₀:", G64(np.float64(t64), 3), "  (t₀ jako float64:", t64, ")")
print("prawdziwe G_3 w tym punkcie float64:", mp.nstr(G(mp.mpf(t64), 3), 4))
```

```
długość przedziału wokół t₀, na którym A_3 < 0: 2.53e-86
liczba punktów siatki o takim kroku w [0, t₀]: 5.0e+98
float64 w t₀: inf   (t₀ jako float64: 12566370639513.906 )
prawdziwe G_3 w tym punkcie float64: 8.894e+532
```

The interval has length of order 10^{-86}; a grid with such a step would have about $5 \cdot 10^{98}$ points on $[0, t_0]$. `float64` misses this interval and returns infinity: the true value of G_3 at the `float64` point is about $8.9 \cdot 10^{532}$, which exceeds the range of this format. Let us plot $\log_{10} G_3$ in a neighborhood of t_0 on the scale 10^{-86}:

```
xs = [mp.mpf(i) / 100 for i in range(-400, 401)]
ys = [float(mp.log10(G(t0 + v * mp.mpf('1e-86'), 3))) for v in xs]
fig, ax = plt.subplots(figsize=(8, 3.2))
ax.plot([float(v) for v in xs], ys, color=kolory[3], lw=1.2)
ax.axhline(0, color='crimson', lw=1, ls='--')
ax.set_xlabel(r'$(t - t_0)\cdot 10^{86}$')
ax.set_ylabel(r'$\log_{10} G_3(t)$')
ax.set_title(r'Studnia wokół $t_0\approx1{,}2566\cdot10^{13}$; poniżej linii $A_3<0$',
             fontsize=10)
fig.tight_layout()
plt.show()
```



Negative values exist, but on an interval of length of order 10^{-86} around 10^{13} — no grid or random sampling will find them, and `float64` cannot even represent t_0 with the required accuracy. We do not claim that t_0 is the smallest such point; it is the point indicated by the construction in the proof.

6. A certificate for $a = 42$

The same construction, only around the 2019 solution. The weights at this point have about 171 digits, so the coordinates of $\Gamma_3(t)$ must hit the target to within about 10^{-344}:

```
m42 = rozwiazania[42]
mp.mp.dps = 60
M = sum((mp.mpf(v)**2 + 2)**3 for v in m42) + 42**2 + 2
wagi42 = [38 * (mp.mpf(v)**2 + 2)**2 * M for v in m42]
eps = 1 / (16 * mp.pi * max(wagi42)**2 * mp.sqrt(6))
print("wagi:", [mp.nstr(k, 4) for k in wagi42])
print("szacowana potrzebna dokładność:", mp.nstr(eps, 3))

mp.mp.dps = 3300
start = time.time()
t42 = podnies3(m42, mp.mpf('1e-345'), mp.mpf('1e-345'))
print(f"czas konstrukcji: {time.time() - start:.0f} s;   t₀ ≈ {mp.nstr(t42, 15)}")
t42_napis = mp.nstr(t42, 2800)
for dps in (3300, 4200):
    mp.iv.dps = dps
    I = G(mp.iv.mpf(t42_napis), 42, ctx=mp.iv)
    gora = mp.nstr(mp.mpf(I.b.b), 6)
    print(f"{dps} cyfr: G_42(t₀) ≤ {gora};  A_42(t₀) < 0: {I.b < 1}")
for cyfry in (2750, 2770):
    I = G(mp.iv.mpf(mp.nstr(t42, cyfry)), 42, ctx=mp.iv)
    print(f"t₀ z {cyfry} cyframi: A_42 < 0: {I.b < 1}")
przed = len(t42_napis.split('.')[0])
print(f"certyfikat: {przed} cyfr przed przecinkiem; początek: {t42_napis[:30]}…")
```

```
wagi: ['8.694e+170', '8.649e+170', '5.211e+167']
szacowana potrzebna dokładność: 1.07e-344
czas konstrukcji: 52 s;   t₀ ≈ 1.25663706394919e+346
3300 cyfr: G_42(t₀) ≤ 5.2935e-5;  A_42(t₀) < 0: True
4200 cyfr: G_42(t₀) ≤ 5.2935e-5;  A_42(t₀) < 0: True
t₀ z 2750 cyframi: A_42 < 0: False
t₀ z 2770 cyframi: A_42 < 0: True
certyfikat: 347 cyfr przed przecinkiem; początek: 125663706394919141951352900551…
```

The certificate for $a = 42$ is a number of order 10^{346} that has to be given to about 2770 digits. The computer finds it in a minute — but only because we know a solution of the equation. Before September 2019 nobody could point to such a t, nor did anyone know whether it existed.

7. $a = 4$: a proof instead of numerics

A sum of three cubes never leaves the residue 4 modulo 9 (the cell in §2), so the equation for $a = 4$ has no solutions, and by Proposition P.1: $A_4(t) > 0$ **for every** $t \in \mathbb{R}$**.** No sampling will prove this — after all, A_3 looked exactly the same. The question is settled by a three-line number-theoretic argument that numerics knows nothing about.

8. $a = 114$: we know the formula, not the answer

We have $114 \equiv 6 \pmod 9$, so the residue modulo 9 rules nothing out, and nobody has found a solution. So today **nobody knows whether** $A_{114}(t) \geq 0$ **for all** t. Heath-Brown's conjecture (1992) predicts that every integer not congruent to 4 or 5 modulo 9 is a sum of three cubes (and

in infinitely many ways). If it is right, A_{114} is negative somewhere — at a place that today we cannot even estimate.

Do we know the function A_{114}?

Yes, completely. It is the formula from §1 for $a = 114$:

$$A_{114}(t) = 256\Big[(u_1^3 + u_2^3 + u_3^3 - 114)^2 + \sum_{i=1}^{3} \sin^2(\pi u_i)\, k_i^4\Big] - 1,$$

$$k_i = 38\,(u_i^2 + 2)^2 M, \qquad M = \sum_{j=1}^{3} (u_j^2 + 2)^3 + 12998,$$

where $s = t \sin t^3$, $u_1 = t \sin t$, $u_2 = s \sin s$, $u_3 = s \sin s^3$. No solution of the equation appears in it — the machine from the proof uses only the polynomial P_{114}. We can compute the value $A_{114}(t)$ at any point with guaranteed accuracy (interval arithmetic, as in §5). The only thing we do not know is the answer to the question whether $A_{114}(t) < 0$ for some t.

A solution of the equation is not a component of the expression but a **certificate** of its negativity. If $A_{114}(t) < 0$, then $f_{114}(\Gamma_3(t)) < 1$, and then — as in the proof of Proposition P.1 — rounding the coordinates of $\Gamma_3(t)$ to the nearest integers gives a solution of the equation $x_1^3 + x_2^3 + x_3^3 = 114$: the point on the curve is itself a witness (Remark 5.4 of the proof). Searching for t with $A_{114}(t) < 0$ is therefore searching for a solution. In §5–§6 we went the other way: from a known solution to t_0.

Does this prove Richardson's theorem?

No — and no single example will. Undecidability is a property of a **family** of questions: there is no algorithm that answers correctly for every expression. A single expression has a definite answer, so an algorithm that outputs it does exist: "print yes" or "print no" — although we may not know which. The family A_a may not be enough either: if Heath-Brown's conjecture were true, the answer would depend only on $a \bmod 9$, and would therefore be decidable. Undecidability requires the polynomial from the DPRM theorem (§1 and §11 of the proof).

A single case can exhibit practical difficulty ($a = 3$, $a = 42$), the limit of present-day knowledge ($a = 114$), or unprovability. The last is produced by the same machine (§3–§6 and §11 of the proof) applied to another, enormous polynomial Q; denote the resulting expression by A_Q.

- By the DPRM theorem there is a Q for which the equation $Q = 0$ has a solution in natural numbers if and only if ZFC set theory is inconsistent. If ZFC is consistent, then the statement "$A_Q(t) \geq 0$ for all t" is true but cannot be proved in ZFC: the chain of equivalences from the proof can be carried out in ZFC, so such a proof would yield a proof of the consistency of ZFC within ZFC itself, contrary to Gödel's second incompleteness theorem.
- Davis, Matiyasevich and Robinson (1976) gave a Q for which the equation $Q = 0$ has no solutions if and only if the Riemann hypothesis holds. Then "$A_Q \geq 0$ everywhere" is equivalent to the Riemann hypothesis.

The family A_a shows something else: even when the general answer may be simple, **individual cases** can be extremely hard — and numerics cannot tell the easy ones from the hard ones.

9. The same question as a zero problem

With the absolute value (§7 of the proof): $B_a = |A_a| - A_a$ is identically zero if and only if $A_a \geq 0$ everywhere. Hence $B_4 \equiv 0$, $B_3 \not\equiv 0$, $B_{42} \not\equiv 0$, and about B_{114} nobody knows. What will a computer algebra system say?

```
class Limit(Exception):
    pass

def z_limitem(f, sekundy):
    """Wywołuje f(), ale przerywa po zadanej liczbie sekund."""
    def stop(*_):
        raise Limit
    stary = signal.signal(signal.SIGALRM, stop)
    signal.alarm(sekundy)
    try:
        return f()
    except Limit:
        return f"(brak odpowiedzi po {sekundy} s)"
    finally:
        signal.alarm(0)
        signal.signal(signal.SIGALRM, stary)

for a in (3, 4):
    Aa = A_symb(a)
    Ba = sp.Abs(Aa) - Aa
    nieujemne, zero_B = Aa.is_nonnegative, Ba.is_zero
    print(f"a = {a}:  A.is_nonnegative = {nieujemne},  B.is_zero = {zero_B}")
    print(f"        B.equals(0) = {z_limitem(lambda: Ba.equals(0), 5)}")
```

```
a = 3:  A.is_nonnegative = None,  B.is_zero = None
        B.equals(0) = (brak odpowiedzi po 5 s)
a = 4:  A.is_nonnegative = None,  B.is_zero = None
        B.equals(0) = (brak odpowiedzi po 5 s)
```

SymPy honestly answers "I don't know" (`None`) or does not answer within 5 seconds — in the same way for both values of a, even though the answers are opposite.

10. The moral

- **Numerics sees only the points at which it computes.** The sign of A_a depends on the arithmetic of integers, which no finite sampling can see: the negative values live on intervals of length 10^{-86} (for $a = 3$) around 10^{13} — or farther out.
- **Negativity can be certified by a finite computation; nonnegativity, in general, cannot.** A certificate (t_0 + interval arithmetic) proves that $A_3 < 0$ and $A_{42} < 0$ somewhere; this is exactly the semi-decidability of Proposition 10.1. The positivity of A_4 requires a proof (modulo 9), and for A_{114} there is no proof.
- **Even an "easy yes" can be hard in practice:** the trivial $1^3 + 1^3 + 1^3 = 3$ yields a witness with about 100 digits, the solution for 42 one with about 2770. Larger solutions — even worse.
- Richardson's theorem says that no algorithm can cope with all such questions. This example shows that the difficulty is not merely theoretical.

11. Questions and exercises

1. Why do both certificates begin with the digits 12566…? *(Hint: what does the construction in Lemma 4.2 start from?)*
2. Which coordinate of $\Gamma_3(t)$ changes fastest near t_0? Estimate its derivative and explain where the interval length of order 10^{-86} comes from.
3. Why does t_0 have to be known to about 100 digits, although $t_0 \approx 10^{13}$? Compare with the result in §5.
4. $30 = 2220422932^3 + (-2218888517)^3 + (-283059965)^3$. Estimate (without computing the certificate) the order of magnitude of t_0 and the number of digits needed. *(Hint: the weights grow like $|\mathbf{m}|^{10}$.)*
5. 💻 Compute the certificate for $a = 29 = 3^3 + 1^3 + 1^3$ and compare it with $a = 3$.
6. 💻 Build the analogous expressions for the Pell equations $x^2 - 61y^2 = -1$ (smallest positive solution: $x = 29718$, $y = 3805$) and $x^2 - 61y^2 = 2$ (no solutions: 2 is not a quadratic residue modulo 61). Compare the numerics and estimate t_0 for the first one.
7. What would happen if the weights k_i were replaced by ones? *(Cf. Example 3.1.)*

References. A. R. Booker, *Cracking the problem with 33*, Research in Number Theory (2019), arXiv:1903.04284 · A. R. Booker, A. V. Sutherland, *On a question of Mordell*, Proc. Natl. Acad. Sci. USA 118 (2021), no. 11, doi:10.1073/pnas.2022377118, arXiv:2007.01209 (the solutions for 42 and new ones for 3) · Sums of three cubes (Wikipedia; as of September 12, 2026, the open cases below 1000 are 114, 390, 627, 633, 732, 921, and 975) · Epoch AI: Sum of Three Cubes.

Appendices: powers of two and the integral of e^{x^2}

Two supplements to the complete proof. Appendix A explains why in the arithmetic input (Theorem 1.1 in "The complete proof, step by step") powers with base 2 suffice — hence the constant $\log 2$ in Richardson's language. Appendix B proves that e^{x^2} is a **local obstruction** to integration in the language with absolute value (Definition 8.1 in "The complete proof, step by step"), which completes the proof of part (C).

A. Powers of two suffice

The Davis–Putnam–Robinson theorem (Theorem 2.3 in "Preliminaries") allows exponentiation a^b with an arbitrary base. We show that the relation $c = a^b$ can be expressed by equations in which exponentiation occurs only as $2^{(\cdot)}$.

Lemma A.1

For positive integers a, b, c let $k = ab + a + 1$ and $M = 2^k - a$. Then

$$c = a^b \iff \exists\, q, s \in \mathbb{N}: \quad 2^{kb} = c + q\,(2^k - a) \quad \text{and} \quad c + s + 1 = 2^k - a.$$

Proof. First, the estimates. Since $a \le 2^a$, we have $a^b \le 2^{ab}$. Moreover,

$$M - 2^{ab} = 2^{ab+a+1} - a - 2^{ab} = 2^{ab}(2^{a+1} - 1) - a > 0,$$

so $0 < a^b \le 2^{ab} < M$. Since $2^k \equiv a \pmod M$, we get $2^{kb} = (2^k)^b \equiv a^b \pmod M$.

"⇐": the second equation says that $0 < c < M$ (since $s \ge 0$ and $c \ge 1$), and the first that $c \equiv 2^{kb} \equiv a^b \pmod M$. Two numbers in the interval $(0, M)$ that are congruent modulo M are equal, so $c = a^b$. "⇒": if $c = a^b$, then $2^{kb} - c$ is a multiple of M (nonnegative, since $2^{kb} \ge 2^{ab} \ge a^b$), which gives $q \in \mathbb{N}$, and $s = M - c - 1 \ge 0$. ■

To make the exponents **variables**, we introduce new unknowns k and ℓ with the equations $k - ab - a - 1 = 0$ and $\ell - kb = 0$; then only 2^k and 2^ℓ occur. Any exponential expression can be broken down, by naming intermediate results as new variables, into a finite system of equations of this type, and the system $E_1 = \cdots = E_r = 0$ is replaced by the single equation $\sum_j E_j^2 = 0$. Positive variables are written as $x + 1$ with $x \in \mathbb{N}$, using $2^{x+1} = 2 \cdot 2^x$. The cases $b = 0$ (then $c = 1$) and $a = 0 < b$ (then $c = 0$) are added as disjuncts, since $E = 0 \vee E' = 0 \iff EE' = 0$. We obtain a polynomial with integer coefficients in the variables x_i and the powers 2^{x_i} — exactly the form of Theorem 1.1 in "The complete proof, step by step".

B. e^{x^2} has no elementary antiderivative — not even locally

The concepts needed (differential field, elementary extension, Liouville's theorem) are in "Preliminaries", §5. We work with **germs** of meromorphic functions at a real point (complex values are allowed): a germ is a function defined on some neighborhood, where functions that agree after shrinking the neighborhood are identified. A germ with zero derivative is a complex constant, so fields of such germs containing $\mathbb{C}$ have field of constants exactly $\mathbb{C}$ — the hypothesis of Liouville's theorem is satisfied.

B.1. Transcendence of e^{x^2}

Lemma B.1

The function $t = e^{x^2}$ is transcendental over $\mathbb{C}(x)$.

Proof. If not, then after clearing denominators we would have $\sum_{j=0}^{m} p_j(x)e^{jx^2} = 0$ with $p_j \in \mathbb{C}[x]$, $p_m \neq 0$, $m \geq 1$, on some neighborhood, and by the identity theorem on the whole real line. Dividing by $p_m(x)e^{mx^2}$ for large $x > 0$, we get $1 + \sum_{j<m} \frac{p_j(x)}{p_m(x)} e^{-(m-j)x^2} = 0$. Each term of the sum tends to zero (a rational function grows at most like a power of x, while $e^{-(m-j)x^2}$ decays faster than any reciprocal of a power), which gives the contradiction $1 = 0$. ■

Hence $K = \mathbb{C}(x, t)$ is the field of rational functions in the variable t with coefficients in $\mathbb{C}(x)$, with the derivation

$$D = \frac{\partial}{\partial x} + 2xt\,\frac{\partial}{\partial t}, \qquad Dx = 1, \quad Dt = 2xt.$$

B.2. A rational equation forced by Liouville's theorem

Proposition B.2

If e^{x^2} has an elementary antiderivative on some nonempty open interval, then there is a rational function $r \in \mathbb{C}(x)$ satisfying

$$r'(x) + 2x\,r(x) = 1. \tag{B.1}$$

Conversely, such an r would give the antiderivative $r(x)e^{x^2}$.

Proof. An elementary tower containing the antiderivative can be built over K (we adjoin its generators after t; the types of the extensions are preserved), with the same field of constants $\mathbb{C}$. Liouville's theorem (Theorem 5.3 in "Preliminaries") for $a = t$ gives

$$t = Dv + \sum_{j=1}^{s} c_j \frac{Du_j}{u_j}, \qquad v \in \mathbb{C}(x,t),\ u_j \in \mathbb{C}(x,t)^{\times}. \tag{B.2}$$

We compare formal Laurent expansions at the point $t = \infty$, i.e., series $\sum_{k \leq m} a_k(x)t^k$ with $a_k \in \mathbb{C}(x)$ (every rational function of t has a unique such expansion). The derivation acts term by term:

$$D(a(x)t^k) = (a'(x) + 2kx\,a(x))t^k,$$

so differentiating a series with only negative powers produces no nonnegative powers.

Logarithmic derivatives produce no positive powers of t. We write every nonzero $u \in \mathbb{C}(x,t)$ as $u = a(x)t^m(1 + w)$, $a \neq 0$, $m \in \mathbb{Z}$, with w a series in negative powers. Then

$$\frac{Du}{u} = \frac{a'}{a} + 2mx + \frac{Dw}{1+w},$$

and the last term contains only negative powers of t.

The coefficient of t^1. Let $r(x)$ be the coefficient of t^1 in the expansion of v (zero if there is none). The coefficient of t^1 in Dv is $r' + 2xr$; the logarithmic derivatives contribute 0; on the left-hand side of (B.2) it is 1. This gives (B.1). Conversely, $\frac{d}{dx}(re^{x^2}) = (r' + 2xr)e^{x^2}$. ■

Lemma B.3

Equation (B.1) has no solution $r \in \mathbb{C}(x)$.

Proof. Suppose that r has a pole of order $m \geq 1$ at $\alpha \in \mathbb{C}$, with leading term $c(x-\alpha)^{-m}$, $c \neq 0$. Then r' has leading term $-mc(x-\alpha)^{-m-1}$, while $2xr$ has a pole of order at most m, so it cannot cancel it: $r' + 2xr$ would have a pole, whereas the right-hand side of (B.1) is 1. Hence r is a polynomial. The zero polynomial does not satisfy (B.1); if r has degree $d \geq 0$ and leading coefficient $c \neq 0$, then $r' + 2xr$ has degree $d+1 \geq 1$ (the term $2cx^{d+1}$), so it is not the constant 1. ■

B.3. The absolute value does not help

Lemma B.4 (local removal of absolute values)

Let f be given by an expression in $\mathcal{B}_1$. For every nonempty open interval I there is a nonempty open interval $J \subseteq I$ on which f equals some expression in $\mathcal{A}_1$ (without absolute value); in particular, f is elementary and analytic on J.

Proof. Induction on the expression tree, for an arbitrary input interval. Constants and x need no change. For a binary operation, we apply the induction hypothesis to one argument on I, then to the other on the resulting subinterval; on the smaller interval both replacements hold simultaneously. For exp and sin, we apply it to the argument. For $|B|$: on some $J_0 \subseteq I$ we have $B = B_0$ with $B_0 \in \mathcal{A}_1$, analytic. If $B_0 \equiv 0$ on J_0, we replace $|B|$ by 0. Otherwise we pick a point with $B_0 \neq 0$ and (by continuity) a neighborhood $J \subseteq J_0$ on which the sign is constant; we replace $|B|$ by B_0 or $-B_0$. ■

This proof is **existential**: it gives no algorithm for determining which subexpressions vanish identically — and it need not, since the lemma serves only to show that e^{x^2} is an obstruction; the reduction algorithm does not use it.

Corollary B.5 (e^{x^2} is a local obstruction)

No function given by an expression in $\mathcal{B}_1$ has derivative equal to e^{x^2} on any nonempty open interval. Hence $\Delta(x) = e^{x^2}$ satisfies the hypothesis of Theorem 8.3 in "The complete proof, step by step".

Proof. If $H' = e^{x^2}$ on I, then by Lemma B.4, on some $J \subseteq I$ we have $H = H_0$ with $H_0 \in \mathcal{A}_1$, so $H_0' = e^{x^2}$ on J, and the germ of H_0 is elementary ($\sin z = (e^{iz} - e^{-iz})/(2i)$). This contradicts Proposition B.2 and Lemma B.3. ■

The same method in a language with square roots

In Richardson's example the language contains the real square root (then $|x| = \sqrt{x^2}$). Lemma B.4 carries over: if an expression is defined on an interval, then the radicand is nonnegative there and either vanishes identically, or on a smaller interval is positive and the square root is an analytic branch, algebraic over the preceding field.

An antiderivative in the ordinary sense does of course exist: $\int_0^x e^{s^2}\,ds$ is smooth on the whole real line (it is $\frac{\sqrt{\pi}}{2}\operatorname{erfi} x$). The obstruction concerns the **form** of the antiderivative, not its existence. SymPy behaves in accordance with this theory:

```
import sympy as sp
x = sp.symbols('x')
```

```
sp.integrate(sp.exp(x**2), x)        # sqrt(pi)*erfi(x)/2 — funkcja specjalna, nie
↪  elementarna
```

Context, consequences and further results

1. What the theorem says about computer algebra systems

Corollary 7.2 in "The complete proof, step by step" says that for expressions with sin, exp, π, $\log 2$ and the absolute value there **is no** canonical-form algorithm. Every function like `simplify` is therefore doomed to one of two fates: either it sometimes fails to recognize zero, or it sometimes gets it wrong. CAS designers choose the former — and use several strategies worth knowing:

1. **Subclasses with a canonical form.** Polynomials and rational functions have a canonical form (Lecture 1, §1.3). For exponential-logarithmic expressions, *structure theorems* (Risch) reduce zero recognition for functions to zero recognition for **constants**.
2. **Randomized tests.** For polynomial and algebraic identities, testing at random points modulo a large prime gives a probabilistic guarantee (the Schwartz–Zippel lemma, Lecture 1, §1.4).
3. **Certified numerics.** Interval arithmetic can **prove** that an expression is *not* zero (the interval does not contain zero) — this is how Proposition 10.1 in "The complete proof, step by step" works. But it will never prove that an expression *is* zero: an interval containing zero does not rule out a very small nonzero value, and a computation at finitely many points says nothing about the others.
4. **Heuristics and the answer "I don't know".** SymPy's `equals` method returns `None` when it cannot decide; `simplify` may return an unsimplified expression.

> **The lesson**
>
> "Nonzero" can be **established by computation**. "Equal to zero" requires a **proof**. Richardson's theorem says that in a sufficiently rich language there is no algorithm that always finds such a proof.

2. Why the sine? The boundary of decidability

The heart of the proof is the equivalence $\sin(\pi x) = 0 \iff x \in \mathbb{Z}$: the sine makes it possible to **define the integers inside the real numbers**, and then arithmetic gets in, together with its undecidability (Hilbert's tenth problem). Without periodicity the situation is completely different:

Language (one or several variables)	Is "$\exists x\ A(x) < 0$" decidable?
polynomials with rational coefficients	**yes** — Tarski (1948/1951): the theory of real closed fields admits quantifier elimination; a practical algorithm: cylindrical algebraic decomposition (Collins, 1975)
polynomials and exp	**yes, if Schanuel's conjecture is true** — Macintyre and Wilkie (1996)
polynomials, sin, π	**no** — Richardson's construction (1968) with Matiyasevich's polynomial (1970); Caviness (1970)
integers, x, $\sin x^n$, $\sin(x \sin x^n)$	**no** — Laczkovich (2003), without π

Wilkie (1996) proved that the structure $(\mathbb{R}, +, \cdot, <, \exp)$ is *o-minimal*: every subset of the line definable in it is a finite union of intervals and points. In such a structure $\mathbb{Z}$ cannot be defined — and that is exactly why there is hope for decidability. The sine restricted to a compact interval is likewise "tame" (van den Dries, 1986). Trouble arises only with the sine on the **whole** line, with its infinitely many zeros.

3. The constant problem and Schanuel's conjecture

Richardson noted in 1968 that it is not known whether the sign of expressions **without variables** can be decided. The question of whether a given elementary constant (e.g., one built from rational numbers, π, e, exp, log) equals zero remains open.

> **Schanuel's conjecture**
>
> If the complex numbers $z_1, \dots, z_n$ are linearly independent over $\mathbb{Q}$, then the transcendence degree of the field $\mathbb{Q}(z_1, \dots, z_n, e^{z_1}, \dots, e^{z_n})$ over $\mathbb{Q}$ is at least n.

The conjecture implies, among other things, the algebraic independence of e and π — today we do not even know whether $e + \pi$ is irrational. Richardson (1997) gave an algorithm for zero recognition of exponential-logarithmic constants that **always answers correctly when it halts** and **always halts if Schanuel's conjecture is true**. This is a good example of how open problems in number theory translate into questions about algorithms in CAS — e.g., the Risch algorithm for integration in finite terms needs precisely a zero test for constants.

4. Further results

- **Caviness (1970)** observed that once Hilbert's tenth problem had been solved, e^x and $\log 2$ could be removed from the theorem (see §11 in "The complete proof, step by step").
- **Wang (1974)** proved that the **existence of a zero** of elementary functions is undecidable. (A version of this result can be derived from our construction and the intermediate value theorem — this is Problem 9 in "Review questions and problems".)
- **Laczkovich (2003)** removed π and restricted the use of composition: in the ring generated by the integers and the functions x, $\sin x^n$, $\sin(x \sin x^n)$ $(n = 1, 2, \dots)$ it is undecidable whether a function takes a positive value and whether it has a root.
- **Richardson (1997)** — an algorithm for constants, conditional on Schanuel's conjecture (§3).

5. Connection with formal proofs

Proof assistants have decision procedures for **decidable** fragments of mathematics: `ring` (identities in commutative rings — a canonical form), `linarith` and `omega` (linear arithmetic), `norm_num` (numerical calculations), `decide` (finite checks). Richardson's theorem explains why there is not — and cannot be — a complete tactic "prove every identity involving sin, exp and absolute value": such proofs have to be found (by a human, a heuristic, an AI model), and the kernel merely **checks** them. Peano Lab is similar: Peano arithmetic is undecidable (Church, 1936), so the `auto` tactic is necessarily incomplete.

6. Timeline

Year	Event
1900	Hilbert poses the tenth problem: an algorithm for Diophantine equations
1936	Turing and Church: undecidability of the halting problem and of arithmetic
1948/51	Tarski: decidability of the theory of real closed fields
1961	Davis, Putnam, Robinson: semi-decidable sets are exponential Diophantine
1968	Richardson: undecidable problems for elementary functions
1970	Matiyasevich: semi-decidable sets are Diophantine; Caviness: removal of e^x, $\log 2$
1974	Wang: undecidability of the existence of zeros
1996	Wilkie: o-minimality of $\mathbb{R}_{\exp}$; Macintyre–Wilkie: decidability assuming Schanuel's conjecture
1997	Richardson: "How to recognise zero"
2003	Laczkovich: removal of π

7. Annotated bibliography

Primary source

- D. Richardson, *Some undecidable problems involving elementary functions of a real variable*, Journal of Symbolic Logic 33(4) (1968), 514–520, doi:10.2307/2271358. Seven pages; worth reading after this chapter — all the steps will be familiar.

Arithmetic and computability

- M. Davis, H. Putnam, J. Robinson, *The decision problem for exponential Diophantine equations*, Annals of Mathematics 74(3) (1961), 425–436, doi:10.2307/1970289.
- Yu. V. Matiyasevich, *Enumerable sets are Diophantine*, Soviet Mathematics Doklady 11 (1970), 354–358 (original: Doklady Akademii Nauk SSSR 191 (1970), 279–282).
- M. Davis, *Hilbert's Tenth Problem is Unsolvable*, American Mathematical Monthly 80(3) (1973), 233–269 — **the best starting point**: a complete, self-contained proof (Chauvenet and Steele Prizes).
- Yu. V. Matiyasevich, *Hilbert's Tenth Problem*, MIT Press 1993 — a monograph.
- J. P. Jones, *Universal Diophantine equation*, Journal of Symbolic Logic 47(3) (1982), 549–571.
- M. Sipser, *Introduction to the Theory of Computation*, Cengage — the chapters on decidability and reductions.
- B. Poonen, *Undecidable problems: a sampler*, in: J. Kennedy (ed.), *Interpreting Gödel*, Cambridge University Press 2014, 211–241; arXiv:1204.0299 — a survey of undecidability across different areas of mathematics, including Richardson's theorem.

Extensions of Richardson's theorem

- B. F. Caviness, *On canonical forms and simplification*, Journal of the ACM 17(2) (1970), 385–396, doi:10.1145/321574.321591.
- P. S. Wang, *The undecidability of the existence of zeros of real elementary functions*, Journal of the ACM 21(4) (1974), 586–589, doi:10.1145/321850.321856.
- M. Laczkovich, *The removal of π from some undecidable problems involving elementary functions*, Proceedings of the AMS 131(7) (2003), 2235–2240, doi:10.1090/S0002-9939-02-06753-9.
- D. Richardson, *How to recognise zero*, Journal of Symbolic Computation 24 (1997), 627–645.

Decidability and o-minimality

- A. Tarski, *A Decision Method for Elementary Algebra and Geometry*, RAND 1948; 2nd ed.: University of California Press 1951.
- G. E. Collins, *Quantifier elimination for real closed fields by cylindrical algebraic decomposition*, Lecture Notes in Computer Science 33 (1975), 134–183.
- A. J. Wilkie, *Model completeness results for expansions of the ordered field of real numbers by restricted Pfaffian functions and the exponential function*, Journal of the AMS 9(4) (1996), 1051–1094.
- A. Macintyre, A. J. Wilkie, *On the decidability of the real exponential field*, in: P. Odifreddi (ed.), *Kreiseliana*, A K Peters 1996, 441–467.

Integration in finite terms

- M. Rosenlicht, *Liouville's theorem on functions with elementary integrals*, Pacific Journal of Mathematics 24(1) (1968), 153–161, doi:10.2140/pjm.1968.24.153.
- B. Conrad, *Impossibility theorems for elementary integration*, notes (available on the author's website) — an accessible proof of Liouville's theorem and of the criterion for $f e^{g}$.
- M. Bronstein, *Symbolic Integration I: Transcendental Functions*, 2nd ed., Springer 2005 — the Risch algorithm in practice.

Review questions and problems

The review questions test your understanding of the successive steps of the proof — the answers are collected at the end of the chapter. The problems are marked by difficulty (⭐ — routine, ⭐⭐ — require an idea, ⭐⭐⭐ — challenge) and with 💻 if they require computation. The numbering of lemmas and theorems refers to "The complete proof, step by step".

Review questions

1. What is the input of the identity problem: a function or an expression? Why does this distinction matter?

2. Does Richardson's theorem imply that a computer cannot verify the identity $\sin^2 x + \cos^2 x = 1$?

3. Why does the language need a periodic function? What happens if we allow only polynomials?

4. What are the weights k_i^4 in the penalty f for? What would happen without them?

5. Why, in Theorem 3.4, do we round to the nearest **natural** number rather than the nearest integer? Where does this matter?

6. Where exactly in the proof do we use the fact that P takes integer values?

7. Why does the rounding theorem concern the inequality $f \leq 1$ and not just the equation $f = 0$?

8. Does the curve Γ_n pass through every point of $\mathbb{R}^n$? Is that needed?

9. Why in Lemma 4.2 do we require $g(w) = b$ **exactly**, not just approximately?

10. For $a \in U$, must the function G_a have a zero?

11. Where in the proof is the absolute value needed, and why can't it simply be omitted?

12. Is the problem "is $A(x) < 0$ for some x?" semi-decidable? And the problem "is $A \equiv 0$?"

13. How does the identity problem differ from the constant problem?

14. In the reduction, what must be computable, and what merely has to exist?

15. Why in Lemma 2.1 is the majorant of a constant c equal to $c^2 + 2$ rather than $|c| + 2$, as in Richardson's paper?

16. What role does $\log 2$ play in the original theorem? Is it necessary?

17. Why does the threshold $\frac{1}{2}$, rather than 1, appear in the integration part?

18. Does Richardson's theorem contradict the existence of the Risch algorithm (integration in finite terms)?

19. A numerical program in `float64` returned $G_a(t) \approx 0.5$. Do we now know that $a \in U$?

20. Why do we define the curve Γ_n for $n \geq 2$?

Problems

1. ★ Prove the inequalities from Lemma 2.1: $|c| < c^2 + 2$ for $c \in \mathbb{R}$ and $|z| + 1 < z^2 + 2$. What are the smallest values of the differences?

2. ★ 💻 Build by hand the majorant from Lemma 2.1 for $A(x, y) = x \sin y - e^x$, and then check the result with the function `majoranta` from "The complete proof, step by step" and with a randomized test of the inequality $|A(\mathbf{z} + \boldsymbol{\eta})| < M_A(\mathbf{z})$.

3. ★ Extend Lemma 2.1 to the language with absolute value: propose a rule for $M_{|A|}$ and prove it. Why does Lemma 2.3 (differentiation) not extend, and why does the proof require the polynomial P to be analytic?

4. ★★ Prove in full the variant with squares (Remark 3.5): $f_{\text{alt}} = (n+1)^2[P^2 + \sum_i \sin^2(\pi x_i) k_i^2]$ also yields a gap.

5. ★★ 💻 For $n = 2$, find the number $s_0 > 1$ from Remark 3.6 (the largest you can) and prove that $\varphi(s_0) < 1$. What is the meaning of s_0 in the case $a \notin U$?

6. ★★ Can $g(t) = t \sin t^3$ be replaced by $t \sin t^2$ in the Richardson curve? Analyze the proof of Lemma 4.2: compare the change of the phase with the admissible change of h on the same interval. (Hint: for the phase t^k, the ratio of the derivatives of the phase and of h behaves like $k\, t^{k-1}/t$.)

7. ★★ Fill in the details of Remark 4.6: prove that the image of Γ_2 has Lebesgue measure zero in $\mathbb{R}^2$ and yet is dense.

8. ★★ Show that the problem "is $A \geq 0$ everywhere?" for $A \in \mathcal{A}_1$ is not semi-decidable. (Hint: Proposition 10.1 and Post's theorem.)

9. ★★ **Existence of a zero.** Prove that there is no algorithm that, given $A \in \mathcal{A}_1$, decides whether A has a zero. Hint: show that for every a there is a point at which $G_a > 1$ — estimate $F(a, \mathbf{u})$ at points with $u_1^2 = \frac{1}{2}$ and use density — and then apply the intermediate value theorem to $G_a - 1$. (Cf. Wang, 1974.)

10. ★★ Prove that the relation "$A \not\equiv B$" for $A, B \in \mathcal{B}_1$ is semi-decidable, while the relation "$A \equiv B$" is not.

11. ★★ Decidability for polynomials in one variable without Tarski's theorem: show that $p \in \mathbb{Q}[x]$ takes a negative value if and only if it has a negative leading coefficient, odd degree, or a real root of odd multiplicity. Describe an algorithm (Sturm sequences) and implement it in SymPy (`sp.sturm`, `sp.sqf_list`). 💻

12. ★★ 💻 Check Lemma A.1 from "Appendices: powers of two and the integral of e^{x^2}" numerically for all $1 \leq a, b \leq 6$: compute k, M, q, s and confirm the equivalence.

13. ★★ 💻 Reproduce the construction from Lemma 4.2 for the target $(a, b) = (0, 100)$ and $\delta = 10^{-3}$. Estimate how many digits of precision are needed for the computation of $g(w)$ to be reliable, and check this by changing `mp.mp.dps`.

14. ★★★ 💻 For the toy equation $x_1^2 + x_2^2 = 5$ from §5, find a point t at which $G(t) < \frac{1}{2}$ (such a point is needed in the integration part). How should δ be chosen? How large is t, and what precision is needed?

15. ★★★ Prove Lemma B.4 from "Appendices: powers of two and the integral of e^{x^2}" for the language with the square root instead of the absolute value (the details of the remark after Corollary B.5).

16. ★★★ Show that the function $\dfrac{\sin x}{x}$ (extended by the value 1 at zero) has no elementary antiderivative. Hint: the criterion for fe^g from Conrad's notes (consider e^{ix}/x) and an analogue of Lemma B.3.

17. ★★★ Read Laczkovich's paper (2003) and explain what replaces the condition $\sin(\pi x) = 0$, given that the language has no π.

Topics for further exploration

- **Dense curves versus space-filling curves.** Compare the Richardson curve (dense, of measure zero, endless) with the Hilbert curve (a continuous **surjection** of a segment onto a square; see the course *Obliczenia matematyczne wspierane komputerowo* (in Polish), the chapter on the Hilbert curve). What does each of them give in dimension-reduction proofs?
- **Hunting for zeros that a CAS cannot see.** Collect identities that SymPy's `simplify` cannot handle (nested radicals, branches of the logarithm, Ramanujan's identities) and classify the reasons for the failures.
- **The constant problem and Schanuel's conjecture.** Present the algorithm from Richardson's paper (1997) and explain exactly where the conjecture is needed.
- **Hilbert's tenth problem.** Based on Davis's article (1973), explain why the relation $c = a^b$ is Diophantine (the Pell equation).
- **Formalization in Lean.** Formalize Lemma 3.3 ($2d \le \sin(\pi d)$ on $[0, \frac{1}{2}]$) or Lemma 4.1 in Lean 4 + Mathlib. Which facts about the sine are already in Mathlib?
- **Decidable subclasses.** Implement a sign test for multivariate polynomials on small examples (cylindrical algebraic decomposition in Sage/Mathematica) and compare it with our semi-deciding interval procedure.

Answers to the review questions

1. An expression — a finite tree that an algorithm can read. A function is an infinite object. Different expressions may denote the same function (e.g., $\sin^2 x + \cos^2 x$ and 1), and the question concerns the function. Undecidability lives precisely in the passage from notation to meaning.

2. No. The theorem says that there is no **single** algorithm that is correct for **all** expressions in the class. Specific identities (and huge subclasses) are decided by a computer without difficulty; SymPy simplifies $\sin^2 x + \cos^2 x$ to 1.

3. The equation $\sin(\pi x) = 0$ defines the set $\mathbb{Z}$, and through it arithmetic and Hilbert's tenth problem enter the real numbers. For polynomials alone, the sign problem (and even arbitrary sentences with quantifiers) is decidable — Tarski's theorem.

4. They control the change of P^2 when a point is rounded to the nearest lattice point, in places where the derivatives of P^2 are large. Without weights, a small residual does not force an integer zero — Example 3.1: $P_N(x) = Nx - 1$ has no root in $\mathbb{N}$, while the unweighted residual attains $\sin^2(\pi/N) \to 0$.

5. The polynomial P contains powers 2^{x_i}, which are integers only for $x_i \in \mathbb{N}$. For nonnegative x_i the nearest integer is a natural number, so property (P2) applies. That is why we first prove the theorem on the set $[0, \infty)^n$ and then substitute squares.

6. Only in the last step of Theorem 3.4: $0 \le P(a, \mathbf{m})^2 < 1$ and $P(a, \mathbf{m}) \in \mathbb{Z}$ give $P(a, \mathbf{m}) = 0$. All the rest of the proof is analysis.

7. Because later we substitute a dense curve, which yields arbitrarily **small** values, but not necessarily zeros. A gap is needed: either there is a zero, or $f > 1$ everywhere.

8. No — its image has measure zero (Remark 4.6). It is not needed: it suffices that the curve enters every neighborhood, because the strict inequality $F < c$ holds on a whole neighborhood (Lemma 5.1).

9. So that the induction in Theorem 4.4 works without accumulating errors: the successive coordinates are iterates of the very steep function g, so even a small error in $g(w)$ would be enormously amplified. An approximation would also suffice (by the continuity of Γ_n), but with an astronomically small tolerance. Exact equality reproduces the entire "tail" of the curve without error.

10. Not necessarily: we only know that $\inf G_a = 0$ (arbitrarily small values). The curve may miss the zero of the function $F(a, \cdot)$. The threshold 1 is enough for the proof.

11. In part (B): $B_a = |A_a| - A_a$ is identically zero exactly when $A_a \geq 0$. An analytic function that vanishes on an interval vanishes everywhere, so in the language $\mathcal{A}_1$ we cannot build such a "plateau of zeros" — this proof does not give undecidability of the identity problem without absolute value.

12. The first — yes (Proposition 10.1: we search rational points with certified approximations). The second — no: its complement ("$A \not\equiv 0$") is semi-decidable, so if it were semi-decidable itself, it would be decidable (Post's theorem), contrary to Theorem 7.1.

13. Identity: an expression with a variable, the question $\forall t\ A(t) = 0$ — undecidable. Constants: an expression without variables, the question $c = 0$ — status unknown (decidable for exponential-logarithmic constants if Schanuel's conjecture is true).

14. The transformation $a \mapsto G_a$ (and further A_a, B_a, J_a) must be computable — these are operations on expression trees. The points $\mathbf{m}$, $\mathbf{u}_0$, t_0, roots and minima exist only in the correctness proof; the algorithm never computes them.

15. For the constants of the grammar $(q, \pi, \log 2)$, $|c|+2$ would also work, since their sign is known. The rule $c^2 + 2$, however, is purely syntactic and also works for any variable-free subexpression treated as a constant, e.g., $\pi - \frac{22}{7}$, whose sign the algorithm need not be able to determine.

16. It is used to write $2^x = \exp(x \log 2)$ in the polynomial from the Davis–Putnam–Robinson theorem. After Matiyasevich's theorem an ordinary polynomial suffices, so exp and $\log 2$ are unnecessary (§11).

17. We need an interval on which the switch S_a is **exactly** equal to 1, and this requires $G_a \leq \frac{1}{2}$. For this, "$G_a < 1$ somewhere" is not enough — we need the full information $\inf G_a = 0$ from Theorem 5.3.

18. No. The Risch algorithm decides whether the integral of a function from suitable differential fields is elementary, **assuming** that zero can be recognized in those fields. Richardson shows that for the language with sin (on the whole line) and absolute value there is no such test; for constants the problem is open.

19. Only if the computation is **certified** (e.g., by interval arithmetic with an upper bound < 1). Then yes, since $G_a(t) < 1 \Rightarrow a \in U$. An ordinary `float` can be wrong, especially since the witnesses lie at huge t (see the example in §5 of "The complete proof, step by step").

20. For $n = 1$ the formula would give $\Gamma_1(t) = t$, and the image (M, ∞) is not dense in $\mathbb{R}$. The number of variables can always be increased with "dummy" variables, so the assumption $n \geq 2$ costs nothing.