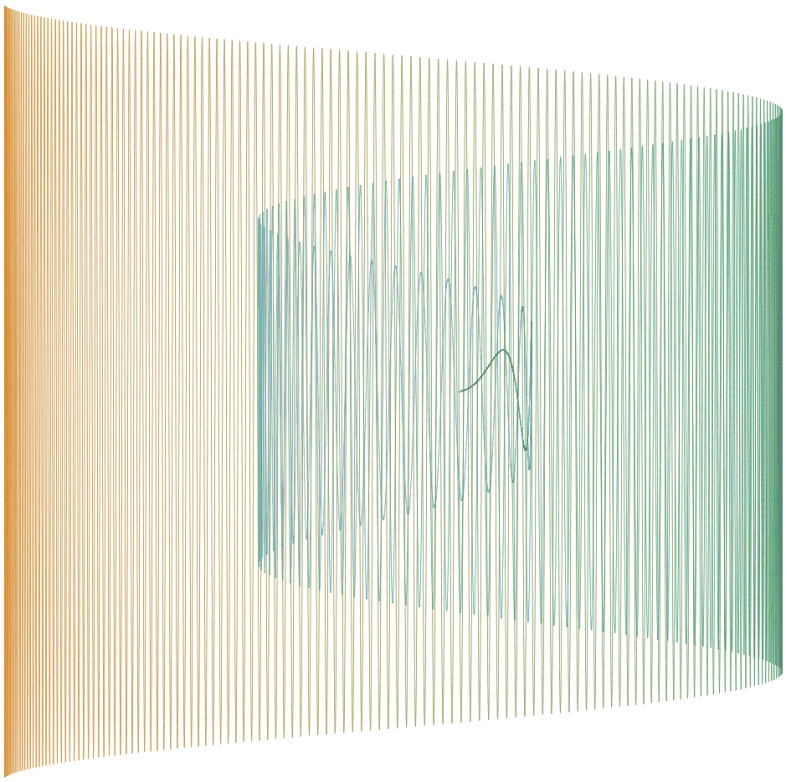
Komputerowe rozwiązywanie problemów matematycznych

Wydział Matematyki i Informatyki UAM · semestr zimowy 2026/27

Twierdzenie Richardsona

Pełny dowód krok po kroku — z definicjami, dodatkami i zadaniami



Krzywa $t \mapsto (t\sin t,\ t\sin t^3)$ dla $0 \le t \le 11$; obraz całej półprostej jest gęsty w płaszczyźnie (Twierdzenie 4.4).

Bartosz Naskręcki

Wersja z 26 października 2026

O tym dokumencie

Wersja do druku rozdziału „Twierdzenie Richardsona” z części „Pogłębienia” książki kursu *Komputerowe rozwiązywanie problemów matematycznych* (WMI UAM, semestr zimowy 2026/27). Wersja internetowa: https://bnaskrecki.faculty.wmi.amu.edu.pl/math-comp/book/richardson/richardson.html

Treść, numeracja definicji, lematów i twierdzeń, wzory oraz wyniki obliczeń są takie same jak w wersji internetowej: dokument jest składany automatycznie z tych samych źródeł, a wyniki i wykresy pochodzą z notatnika wykonanego przy budowie książki. Kod w Pythonie można uruchomić samodzielnie — notatniki `r2_dowod.ipynb` i `r2_przyklad.ipynb` można pobrać ze stron rozdziałów „Pełny dowód krok po kroku” i „Przykład: wyrażenie, którego znaku nie widać”.

Stan na 26 października 2026. © 2026 Bartosz Naskręcki (współtworzone z Claude).

Oznaczenia

twierdzenia, lematy, wnioski i stwierdzenia

definicje, uwagi i przykłady

wskazówki, komentarze o budowie dowodu, tezy

ostrzeżenia: miejsca, w których łatwo o błąd

odsyłacze do literatury

`wynik` wynik obliczenia (tekst z szarą kreską po lewej)

★ … ★★★ trudność problemu: rutynowy … wyzwanie; 💻 — problem wymaga obliczeń

Spis treści

Twierdzenie Richardsona: dlaczego komputer nie rozpozna każdego zera **5**
Twierdzenie . . . 5
Szkic dowodu w sześciu krokach . . . 5
Jak czytać ten rozdział . . . 7

Pojęcia wstępne **8**
1. Algorytmy i rozstrzygalność . . . 8
2. Równania diofantyczne i dziesiąty problem Hilberta . . . 10
3. Wyrażenia, funkcje, języki . . . 11
4. Narzędzia analizy . . . 11
5. Algebra różniczkowa (tylko do części o całkowaniu) . . . 12

Pełny dowód krok po kroku **14**
0. Twierdzenie i plan dowodu . . . 14
1. Wejście arytmetyczne . . . 15
2. Majoranty lokalne . . . 16
3. Kara ważona i twierdzenie o zaokrąglaniu . . . 18
4. Jedna zmienna wystarczy: krzywa gęsta . . . 20
5. Przeniesienie na jedną zmienną . . . 24
6. Problem znaku . . . 26
7. Problem tożsamości (zera) . . . 26
8. Problem całkowania . . . 27
9. Oryginalne, ogólne sformułowanie Richardsona . . . 28
10. Co dokładnie udowodniliśmy . . . 29
11. Wersja współczesna: wystarczą wielomiany i sinus . . . 30

Przykład: wyrażenie, którego znaku nie widać **31**
1. Wyrażenie $A_a(t)$ . . . 31
2. Cztery równania . . . 33
3. Co widzi numeryka . . . 34
4. Pułapka zmiennoprzecinkowa . . . 35
5. Certyfikat dla $a = 3$ . . . 36
6. Certyfikat dla $a = 42$ . . . 39
7. $a = 4$: dowód zamiast numeryki . . . 39
8. $a = 114$: znamy wzór, nie znamy odpowiedzi . . . 39
9. To samo pytanie jako problem zera . . . 41
10. Morał . . . 41
11. Pytania i zadania . . . 42

Dodatki: potęgi dwójki i całka z e^{x^2} **43**
A. Wystarczą potęgi dwójki . . . 43
B. e^{x^2} nie ma elementarnej funkcji pierwotnej — nawet lokalnie . . . 43

Kontekst, konsekwencje i dalsze wyniki **46**
1. Co twierdzenie mówi o systemach algebry komputerowej . . . 46
2. Dlaczego sinus? Granica rozstrzygalności . . . 46

3. Problem stałych i hipoteza Schanuela 47
4. Dalsze wyniki 47
5. Związek z dowodami formalnymi 47
6. Kalendarium 48
7. Literatura z komentarzem 48

Pytania kontrolne i problemy **50**
Pytania kontrolne 50
Problemy 51
Tematy do dalszej eksploracji 52
Odpowiedzi do pytań kontrolnych 52

Twierdzenie Richardsona: dlaczego komputer nie rozpozna każdego zera

> *„Nie jest znana metoda rozstrzygania, czy wyrażenie subelementarne bez zmiennych przedstawia liczbę mniejszą od zera."* — Daniel Richardson (1968)

Na wykładzie 1 (§1.3) padło zdanie: *dla wyrażeń zbudowanych z x, liczb wymiernych, π, $\ln 2$, $\sin$, $\exp$ i modułu nie istnieje algorytm, który rozstrzyga, czy wyrażenie jest tożsamościowo równe zeru.* Ten rozdział jest **pogłębieniem**: zawiera pełny dowód tego twierdzenia (Daniel Richardson, *Journal of Symbolic Logic*, 1968), wszystkie potrzebne pojęcia spoza kursu, kontekst i zadania.

Materiały

- Aplet: jedna zmienna wystarczy — krzywa gęsta, lemat o podnoszeniu, rola wag, sztuczka z modułem.
- „Pojęcia wstępne" — obliczalność, dziesiąty problem Hilberta, języki wyrażeń, narzędzia analizy, algebra różniczkowa.
- „Pełny dowód krok po kroku" — pełny dowód z ilustracjami w Pythonie.
- „Przykład: wyrażenie, którego znaku nie widać" — numeryka mówi „wszędzie dodatnie" dla $x^3 + y^3 + z^3 = a$ z $a = 3, 4, 42, 114$, a prawda bywa różna (dla $a = 114$ nikt jej nie zna).
- „Dodatki" — potęgi dwójki; dlaczego e^{x^2} nie ma elementarnej funkcji pierwotnej.
- „Kontekst, konsekwencje i dalsze wyniki" — konsekwencje dla CAS, granica rozstrzygalności, problem stałych, literatura.
- „Pytania kontrolne i problemy" — 20 pytań kontrolnych i 17 problemów.

Twierdzenie

Niech $\mathcal{A}_1$ oznacza wyrażenia jednej zmiennej x zbudowane z liczb wymiernych, π, $\log 2$, działań $+, -, \cdot$, funkcji exp i sin (z dowolnym zagnieżdżaniem), a $\mathcal{B}_1$ — to samo z modułem $|\cdot|$.

Twierdzenie Richardsona (1968)

1. Nie istnieje algorytm, który dla $A \in \mathcal{A}_1$ rozstrzyga, czy $A(x) < 0$ dla pewnego x.
2. Nie istnieje algorytm, który dla $B \in \mathcal{B}_1$ rozstrzyga, czy $B(x) = 0$ dla wszystkich x.
3. Nie istnieje algorytm, który rozstrzyga, czy dane wyrażenie z $\mathcal{B}_1$ ma funkcję pierwotną wyrażalną w $\mathcal{B}_1$.

Wniosek dla praktyki: **nie istnieje doskonały `simplify`**. Każdy system algebry komputerowej musi czasem odpowiedzieć „nie wiem", pomylić się albo liczyć bez końca.

Szkic dowodu w sześciu krokach

Idea: pokażemy, że gdyby taki algorytm istniał, rozwiązywałby **dziesiąty problem Hilberta** — a ten jest nierozwiązywalny. Każdy krok przekształca jedno pytanie w drugie, równoważne.

Krok 1. Start: nierozstrzygalne równanie w liczbach całkowitych. Istnieje ustalony wielomian $P(y, x_1, \ldots, x_n)$ (u Richardsona z potęgami 2^{x_i}) taki, że żaden algorytm nie rozstrzyga,

czy dana liczba $a \in \mathbb{N}$ należy do zbioru U tych a, dla których równanie $P(a, \mathbf{x}) = 0$ ma rozwiązanie w liczbach naturalnych (Davis–Putnam–Robinson 1961, Matiyasevich 1970).

Krok 2. Z liczb całkowitych do rzeczywistych: sinus wykrywa całkowitość. Ponieważ $\sin(\pi x) = 0 \iff x \in \mathbb{Z}$, rozwiązania całkowite to zera funkcji

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

Ale później będziemy umieli tylko *zbliżać się* do zer, więc potrzebujemy więcej.

Krok 3. Przerwa: małe wartości wymuszają dokładne zero. Richardson mnoży składniki z sinusem przez szybko rosnące **wagi** k_i — wyrażenia ograniczające pochodne P^2 na sąsiedztwie punktu:

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

Jeśli $f(a, \mathbf{x}) \le 1$ dla pewnego $\mathbf{x} \in [0, \infty)^n$, to $\mathbf{x}$ leży tak blisko punktu $\mathbf{m} \in \mathbb{N}^n$, że $P(a, \mathbf{m})^2 < 1$, a ponieważ to liczba całkowita — $P(a, \mathbf{m}) = 0$. Zatem: **albo f ma zero, albo wszędzie $f > 1$.** (Bez wag to nie działa — przykład $P(x) = Nx - 1$.)

Krok 4. Jedna zmienna wystarczy. Funkcje $h(t) = t \sin t$ i $g(t) = t \sin(t^3)$ mają różne „tempa”: na krótkim przedziale przy dużym t funkcja h prawie stoi, a g wykonuje pełne oscylacje o amplitudzie $\approx t$. Dzięki temu krzywa

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

ma obraz **gęsty** w $\mathbb{R}^n$. Niech $F(a, \mathbf{u}) = f(a, u_1^2, \ldots, u_n^2)$ (kwadraty usuwają ograniczenie $x_i \ge 0$). Ponieważ nierówność $F < 1$ jest ostra i zachodzi na całym otoczeniu, a krzywa wchodzi w każde otoczenie, funkcja **jednej** zmiennej $G_a(t) = F\big(a, \Gamma_n(t)\big)$ spełnia: $a \in U \iff G_a(t) < 1$ dla pewnego t.

Krok 5. Problem znaku. Wyrażenie $A_a = G_a - 1$ jest gdzieś ujemne dokładnie wtedy, gdy równanie diofantyczne ma rozwiązanie. Algorytm znaku rozwiązywałby więc problem Hilberta.

Krok 6. Problem zera i całkowania. Moduł zamienia znak na tożsamość: $B_a = |A_a| - A_a$ jest tożsamościowo zerem dokładnie wtedy, gdy $A_a \ge 0$ wszędzie. Dla całkowania mnożymy e^{x^2} (które nie ma elementarnej funkcji pierwotnej) przez wyłącznik S_a, który dla $a \notin U$ jest tożsamościowo zerem, a dla $a \in U$ równa się 1 na pewnym przedziale.

$$\underbrace{\exists\, \mathbf{m} \in \mathbb{N}^n \;\; P(a, \mathbf{m}) = 0}_{\text{Hilbert}} \iff \exists\, \mathbf{x} \in [0, \infty)^n \;\; f(a, \mathbf{x}) \le 1 \iff \exists\, t \;\; G_a(t) < 1$$

$$\iff A_a \text{ gdzieś} < 0 \iff B_a \not\equiv 0.$$

Co jest algorytmem, a co tylko istnieje?

Przekształcenie $a \mapsto G_a \mapsto A_a \mapsto B_a$ to **operacje na drzewach wyrażeń** — algorytm. Punkty, w których $G_a < 1$, istnieją tylko w dowodzie; nikt nie musi ich wyznaczać (a w praktyce leżą astronomicznie daleko — w przykładzie z §5 dowodu przy $t \approx 10^{11}$).

Jak czytać ten rozdział

- Do zrozumienia szkicu wystarczy wykład 1 i podstawy analizy.
- Pełny dowód (rozdział „Pełny dowód krok po kroku”) wymaga pojęć z rozdziału „Pojęcia wstępne”: rozstrzygalność, redukcje, zbiory diofantyczne, twierdzenie o wartości średniej, gęstość.
- Fakty przyjęte bez dowodu: twierdzenia Davisa–Putnama–Robinsona i Matiyasevicha (arytmetyka) oraz twierdzenie Liouville’a (tylko część o całkowaniu). Poza standardowymi twierdzeniami analizy („Pojęcia wstępne”, §4) wszystko inne jest udowodnione.
- Notacja i numeracja lematów w rozdziale „Pełny dowód krok po kroku” odpowiadają oryginałowi: Lemat 2.1 = *Lemma One*, Twierdzenie 3.4 = *Theorem One* §1, Lemat 4.2 = *Lemmas Two* i *Three*, Twierdzenie 4.4 = *Theorem Two*, §6–§8 = *Theorems One–Three* §2.

Rozdział powstał na podstawie oryginalnej pracy Richardsona i opracowań przygotowanych przez prowadzącego; obliczenia w nim są wykonywane przy każdej budowie książki.

Pojęcia wstępne

Dowód twierdzenia Richardsona łączy trzy działy matematyki, które na naszym kursie pojawiają się tylko marginalnie: **teorię obliczalności**, **teorię liczb** (równania diofantyczne) i — w części o całkowaniu — **algebrę różniczkową**. Ten rozdział zbiera wszystkie potrzebne definicje i twierdzenia. Fakty oznaczone jako *zewnętrzne* przyjmujemy bez dowodu (z podaniem źródła); poza standardowymi twierdzeniami analizy z §4 wszystko inne jest w tej książce udowodnione.

Konwencje: $\mathbb{N} = \{0, 1, 2, \dots\}$ (zero należy do $\mathbb{N}$), a dla wektora $\boldsymbol{\eta} = (\eta_1, \dots, \eta_d) \in \mathbb{R}^d$ piszemy $\|\boldsymbol{\eta}\|_\infty = \max_j |\eta_j|$ (norma maksimum).

1. Algorytmy i rozstrzygalność

Definicja 1.1 (algorytm — intuicyjnie)

Algorytm to skończony, jednoznaczny przepis, który krok po kroku przetwarza skończone dane wejściowe (napis nad skończonym alfabetem) i — jeśli się zatrzyma — zwraca skończony wynik. Nie ma ograniczenia na czas ani pamięć, ale każdy krok jest mechaniczny.

Aby mówić o tym, czego algorytm *nie może* zrobić, potrzebujemy precyzyjnego modelu.

Definicja 1.2 (maszyna Turinga)

Maszyna Turinga to piątka $(S, \Sigma, \delta, s_0, F)$: skończony zbiór stanów S ze stanem początkowym $s_0 \in S$ i zbiorem stanów końcowych $F \subseteq S$, skończony alfabet taśmy Σ (z symbolem pustym) oraz funkcja przejścia $\delta\colon S \times \Sigma \to S \times \Sigma \times \{\leftarrow, \rightarrow\}$. Maszyna pracuje na nieskończonej taśmie z głowicą: w każdym kroku odczytuje symbol, zapisuje nowy, przesuwa głowicę i zmienia stan. Obliczenie na danym wejściu może się **zatrzymać** (osiągnąć stan końcowy) albo trwać w nieskończoność.

Teza Churcha–Turinga

Każdy algorytm w sensie intuicyjnym da się zrealizować maszyną Turinga (równoważnie: programem w Pythonie z nieograniczoną pamięcią, rachunkiem lambda, funkcjami rekurencyjnymi…). To nie twierdzenie matematyczne, lecz teza o adekwatności definicji — dotąd żaden „rozsądny” model obliczeń jej nie obalił. Dzięki niej wolno nam w dowodach pisać „istnieje algorytm” zamiast konstruować maszynę Turinga.

Definicja 1.3 (problem decyzyjny, zbiór rozstrzygalny)

Problem decyzyjny to zbiór W skończonych napisów (np. kodów liczb naturalnych albo wyrażeń). Problem W jest **rozstrzygalny** (zbiór W jest *rekurencyjny*), jeśli istnieje algorytm, który dla *każdego* napisu w zatrzymuje się i poprawnie odpowiada, czy $w \in W$. W przeciwnym razie W jest **nierozstrzygalny**.

Definicja 1.4 (zbiór półrozstrzygalny)

Zbiór W jest **półrozstrzygalny** (*obliczalnie przeliczalny*, *rekurencyjnie przeliczalny*), jeśli istnieje algorytm, który zatrzymuje się dokładnie na elementach W (na pozostałych napisach

działa w nieskończoność). Równoważnie: istnieje algorytm wypisujący po kolei wszystkie elementy W.

Twierdzenie 1.5 (Post)

Zbiór W jest rozstrzygalny wtedy i tylko wtedy, gdy zarówno W, jak i jego dopełnienie $\overline{W}$ są półrozstrzygalne.

Dowód. Jeśli W jest rozstrzygalny, ten sam algorytm (zmuszony do zapętlenia przy odpowiedzi „nie” albo „tak”) półrozstrzyga W i $\overline{W}$. Odwrotnie: dla danego w uruchamiamy naprzemiennie, krok po kroku, obie procedury półrozstrzygające. Dokładnie jedna z nich się zatrzyma, bo w należy do dokładnie jednego z $W, \overline{W}$; wtedy znamy odpowiedź. ■

Najważniejszy przykład problemu nierozstrzygalnego dotyczy samych programów.

Twierdzenie 1.6 (Turing, 1936: problem stopu)

Zbiór $H = \{(e, w) : \text{program o kodzie } e \text{ zatrzymuje się na wejściu } w\}$ jest półrozstrzygalny, ale nierozstrzygalny.

Dowód. Półrozstrzygalność: symulujemy program e na w i zatrzymujemy się, gdy on się zatrzyma. Przypuśćmy teraz, że jakiś algorytm $\mathsf{STOP}(e, w)$ rozstrzyga H. Skonstruujmy program D, który na wejściu e: jeśli $\mathsf{STOP}(e, e)$ odpowiada „tak”, wpada w nieskończoną pętlę; jeśli „nie” — kończy działanie. Niech d będzie kodem programu D. Czy D zatrzymuje się na wejściu d? Jeśli tak, to $\mathsf{STOP}(d, d) =$ „tak”, więc D się zapętla — sprzeczność. Jeśli nie, to $\mathsf{STOP}(d, d) =$ „nie”, więc D się zatrzymuje — sprzeczność. ■

Z twierdzenia Posta wynika, że **dopełnienie** problemu stopu nie jest nawet półrozstrzygalne.

Nierozstrzygalność przenosi się na inne problemy za pomocą *redukcji*.

Definicja 1.7 (redukcja wielo-jednoznaczna)

Problem V **redukuje się** do problemu W (piszemy $V \leq_m W$), jeśli istnieje funkcja obliczalna r (algorytm zamieniający napisy na napisy) taka, że

$$w \in V \iff r(w) \in W \qquad \text{dla każdego napisu } w.$$

Stwierdzenie 1.8

Jeśli $V \leq_m W$ i V jest nierozstrzygalny, to W jest nierozstrzygalny. Ponadto W jest rozstrzygalny wtedy i tylko wtedy, gdy $\overline{W}$ jest rozstrzygalny, więc redukcja $V \leq_m \overline{W}$ również dowodzi nierozstrzygalności W.

Dowód. Algorytm rozstrzygający W, złożony z r, rozstrzygałby V. Dla dopełnienia wystarczy zamienić odpowiedzi „tak” i „nie”. ■

Ważne: co robi redukcja, a czego nie musi

Funkcja r musi być **obliczalna** — w naszym dowodzie będzie przekształceniem *napisów* (drzew wyrażeń). Natomiast uzasadnienie, że $w \in V \iff r(w) \in W$, może być dowolnie nieefektywne: wolno w nim używać ciągłości, twierdzenia Darboux, istnienia punktów, których nie umiemy wyznaczyć. Algorytm nie musi tych punktów znajdować.

2. Równania diofantyczne i dziesiąty problem Hilberta

Definicja 2.1 (równanie diofantyczne, zbiór diofantyczny)

Równanie diofantyczne to równanie $p(x_1, \dots, x_n) = 0$, gdzie $p \in \mathbb{Z}[x_1, \dots, x_n]$, w którym szukamy rozwiązań całkowitych (u nas: w $\mathbb{N}$). Zbiór $D \subseteq \mathbb{N}$ jest **diofantyczny**, jeśli istnieje wielomian $p(y, x_1, \dots, x_n)$ o współczynnikach całkowitych taki, że

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

Na przykład zbiór liczb złożonych jest diofantyczny: $y \in D \iff \exists a, b \;\; y = (a+2)(b+2)$.

Dziesiąty problem Hilberta (1900) prosił o algorytm, który dla dowolnego równania diofantycznego rozstrzyga, czy ma ono rozwiązanie w liczbach całkowitych. Odpowiedź jest negatywna, a droga do niej prowadziła przez równania z potęgowaniem.

Definicja 2.2 (wykładnicze równanie diofantyczne)

Wykładnicze równanie diofantyczne to równanie $E_1 = E_2$, w którym E_1, E_2 są zbudowane ze zmiennych i stałych naturalnych za pomocą dodawania, mnożenia i potęgowania a^b (także ze zmiennym wykładnikiem). Zbiory definiowane takimi równaniami (jak w Definicji 2.1) nazywamy **wykładniczo diofantycznymi**.

Twierdzenie 2.3 (Davis, Putnam, Robinson, 1961; zewnętrzne)

Każdy zbiór półrozstrzygalny $W \subseteq \mathbb{N}$ jest wykładniczo diofantyczny. W szczególności nie istnieje algorytm rozstrzygający rozwiązalność wykładniczych równań diofantycznych.

Twierdzenie 2.4 (Matiyasevich, 1970; DPRM; zewnętrzne)

Każdy zbiór półrozstrzygalny $W \subseteq \mathbb{N}$ jest diofantyczny. W szczególności dziesiąty problem Hilberta jest nierozwiązywalny.

Matiyasevich pokazał, że relacja $c = a^b$ sama jest diofantyczna. Kluczowe było znalezienie diofantycznej relacji o wykładniczym wzroście — Matiyasevich użył liczb Fibonacciego ($v = F_{2u}$); współczesne wykłady używają zwykle równania Pella. Stąd Twierdzenie 2.4 wynika z 2.3. Wiadomo nawet, że wystarczy dziewięć niewiadomych (Matiyasevich; zob. J. P. Jones, 1982).

Wniosek 2.5 (ustalony nierozstrzygalny wielomian)

Istnieją $n \geq 1$ i wielomian $p(y, x_1, \dots, x_n)$ o współczynnikach całkowitych takie, że zbiór

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

jest nierozstrzygalny.

Dowód. Zbiór $\{e : \text{program } e \text{ zatrzymuje się na wejściu } e\}$ jest półrozstrzygalny (symulacja), ale nierozstrzygalny (dowód Twierdzenia 1.6 pokazuje właśnie to). Twierdzenie 2.4 daje jego diofantyczną definicję. ■

Ważne: wielomian p jest **jeden, ustalony**. Zmienia się tylko parametr a. Richardson (1968), przed wynikiem Matiyasevicha, użył analogicznego wniosku z Twierdzenia 2.3 w postaci, w której potęgowanie występuje tylko jako 2^{x_i} — zob. „Pełny dowód krok po kroku”, §1, i „Dodatki”, dodatek A.

3. Wyrażenia, funkcje, języki

Algorytm nie dostaje „funkcji”, tylko jej **skończony zapis**.

Definicja 3.1 (język wyrażeń, drzewo wyrażenia)

Język wyrażeń zadajemy gramatyką: zbiorem symboli stałych, zmiennych i operacji. Na przykład

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

Każde wyrażenie to skończone **drzewo**: w liściach stałe i zmienne, w węzłach operacje (zob. rozdział „Jak działa system algebry komputerowej?”). Drzewa są skończonymi napisami, więc mogą być danymi wejściowymi algorytmu.

Definicja 3.2 (klasy $\mathcal{A}_d$ i $\mathcal{B}_d$)

$\mathcal{A}_d$ — wyrażenia z powyższej gramatyki w zmiennych $x_1, \dots, x_d$ (*rdzeń analityczny*); każde wyznacza funkcję $\mathbb{R}^d \to \mathbb{R}$ określoną wszędzie. $\mathcal{B}_d$ — to samo, z dodatkową operacją $A \mapsto |A|$. Podstawianie wyrażeń za zmienne jest operacją na drzewach, więc obie klasy są **efektywnie zamknięte na złożenie**.

Różne wyrażenia mogą oznaczać tę samą funkcję: $\sin^2 x + \sin^2(x+\pi/2) - 1$ i 0 to różne drzewa, ale ta sama funkcja (bo $\sin(x+\pi/2) = \cos x$). Pytania, które badamy, dotyczą **funkcji**, a dane wejściowe to **drzewa**. W tej szczelinie mieszka nierozstrzygalność.

Definicja 3.3 (funkcje częściowe i równość z dziedziną)

Funkcja **częściowa** f ma dziedzinę $\operatorname{dom} f \subseteq \mathbb{R}$. Dla sumy, różnicy i iloczynu dziedziną jest przecięcie dziedzin, a $\operatorname{dom}(f \circ g) = \{x \in \operatorname{dom} g : \ g(x) \in \operatorname{dom} f\}$. Piszemy $f \equiv g$, gdy $\operatorname{dom} f = \operatorname{dom} g$ i $f = g$ na tej dziedzinie. W szczególności $f \equiv 0$ oznacza, że f jest określona **wszędzie** i wszędzie równa zeru.

4. Narzędzia analizy

Definicja 4.1 (funkcja analityczna rzeczywista)

Funkcja $f\colon \mathbb{R}^d \to \mathbb{R}$ jest **analityczna**, jeśli w otoczeniu każdego punktu jest sumą zbieżnego szeregu potęgowego. Stałe, współrzędne, exp i sin są analityczne, a sumy, iloczyny i złożenia funkcji analitycznych są analityczne. Zatem **każde wyrażenie z $\mathcal{A}_d$ wyznacza funkcję analityczną**. Wyrażenia z $\mathcal{B}_d$ wyznaczają funkcje ciągłe, ale nie zawsze różniczkowalne (np. $|x|$ w zerze).

Twierdzenie 4.2 (o identyczności)

Jeśli funkcja analityczna na przedziale $I \subseteq \mathbb{R}$ znika na pewnym niepustym przedziale otwartym $J \subseteq I$, to znika na całym I.

Funkcje z modułem nie mają tej własności: $|x| - x$ znika na $[0, \infty)$, ale nie na $\mathbb{R}$. Właśnie dlatego moduł jest potrzebny w części o tożsamości.

Twierdzenie 4.3 (Darboux — o wartości pośredniej)

Jeśli φ jest ciągła na $[\alpha,\beta]$, a c leży między $\varphi(\alpha)$ i $\varphi(\beta)$, to $\varphi(\tau)=c$ dla pewnego $\tau\in[\alpha,\beta]$.

Twierdzenie 4.4 (o wartości średniej; wersja całkowa)

Jeśli $\varphi\colon[\alpha,\beta]\to\mathbb{R}$ jest różniczkowalna, to $|\varphi(\beta)-\varphi(\alpha)|\le(\beta-\alpha)\sup_{[\alpha,\beta]}|\varphi'|$. Dla $Q\colon\mathbb{R}^n\to\mathbb{R}$ klasy C^1 i odcinka $\gamma(t)=\mathbf{x}+t(\mathbf{m}-\mathbf{x})$:

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

Definicja 4.5 (gęstość)

Zbiór $D\subseteq\mathbb{R}^n$ jest **gęsty**, jeśli przecina każdy niepusty zbiór otwarty; równoważnie: dla każdego $\mathbf{a}\in\mathbb{R}^n$ i $\varepsilon>0$ istnieje $\mathbf{d}\in D$ z $\|\mathbf{d}-\mathbf{a}\|_\infty<\varepsilon$.

Fakt 4.6 (ostre nierówności są odporne)

Jeśli $H\colon\mathbb{R}^n\to\mathbb{R}$ jest ciągła i $H(\mathbf{u}_0)<c$, to $H<c$ na pewnym otoczeniu $\mathbf{u}_0$. Dla nierówności nieostrej $H(\mathbf{u}_0)\le c$ analogiczny wniosek nie musi zachodzić.

To drobne spostrzeżenie jest osią całego dowodu: gęsta krzywa nie musi przejść przez żaden konkretny punkt, ale wejdzie w każde otoczenie — wystarczy więc mieć **ostrą** nierówność z zapasem.

Definicja 4.7 (kres dolny a minimum)

$\inf_t G(t)=0$ oznacza, że $G\ge0$ i wartości G są dowolnie bliskie zeru; nie oznacza, że G gdzieś przyjmuje wartość 0 (np. $G(t)=\frac{1}{1+t^2}$ ma na $\mathbb{R}$ kres dolny 0, który nie jest osiągany).

5. Algebra różniczkowa (tylko do części o całkowaniu)

Definicja 5.1 (ciało różniczkowe)

Ciało różniczkowe to ciało K z **derywacją** $D\colon K\to K$: odwzorowaniem addytywnym spełniającym regułę Leibniza $D(uv)=(Du)v+u(Dv)$. **Stałe** to elementy c z $Dc=0$. Przykład: $\mathbb{C}(x)$ (funkcje wymierne) z $D=\frac{d}{dx}$; jego stałe to $\mathbb{C}$.

Definicja 5.2 (rozszerzenie elementarne)

Rozszerzenie $K\subseteq L$ jest **elementarne**, jeśli $L=K(\theta_1,\dots,\theta_s)$, gdzie każde θ_j nad $K(\theta_1,\dots,\theta_{j-1})$ jest (i) algebraiczne, albo (ii) **wykładnicze**: $D\theta_j/\theta_j=Db$ (czyli „$\theta_j=e^b$”), albo (iii) **logarytmiczne**: $D\theta_j=Db/b$ (czyli „$\theta_j=\log b$”), dla pewnego b z poprzedniego ciała. **Funkcja elementarna** to element takiego rozszerzenia $\mathbb{C}(x)$. Funkcje trygonometryczne są elementarne, bo $\sin z=(e^{iz}-e^{-iz})/(2i)$.

Twierdzenie 5.3 (Liouville; postać Rosenlichta; zewnętrzne)

Niech K będzie ciałem różniczkowym charakterystyki 0 o ciele stałych $\mathbb{C}$ i niech $a\in K$ ma funkcję pierwotną w pewnym elementarnym rozszerzeniu K o tych samych stałych. Wtedy

istnieją $v \in K$, niezerowe $u_1, \dots, u_s \in K$ i stałe $c_1, \dots, c_s \in \mathbb{C}$ takie, że

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

Innymi słowy: jeśli całka jest elementarna, to jest postaci „funkcja z K plus kombinacja logarytmów funkcji z K”. Z tego twierdzenia wyprowadzimy w rozdziale „Dodatki”, że e^{x^2} nie ma elementarnej funkcji pierwotnej na żadnym przedziale.

Gdzie szukać dowodów faktów zewnętrznych

- Twierdzenie 1.6 i redukcje: M. Sipser, *Introduction to the Theory of Computation*, rozdz. 3–5.
- Twierdzenia 2.3–2.4: M. Davis, *Hilbert’s Tenth Problem is Unsolvable*, Amer. Math. Monthly 80 (1973) — przystępny, kompletny wykład (nagrody Chauvenet i Steele); Yu. Matiyasevich, *Hilbert’s Tenth Problem*, MIT Press 1993.
- Twierdzenie 5.3: M. Rosenlicht, Pacific J. Math. 24 (1968); B. Conrad, *Impossibility theorems for elementary integration* (notatki, tw. 4.1).

Pełny dowód krok po kroku

Ten rozdział zawiera kompletny dowód twierdzenia Richardsona w postaci z oryginalnej pracy (D. Richardson, *Some undecidable problems involving elementary functions of a real variable*, J. Symbolic Logic **33** (1968), 514–520), z uzupełnionymi szczegółami. Poza standardowymi twierdzeniami analizy jedynymi faktami przyjętymi bez dowodu są twierdzenie Davisa–Putnama–Robinsona (i, w §11, Matiyasevicha) oraz — tylko w części o całkowaniu — twierdzenie Liouville'a; zob. rozdział „Pojęcia wstępne". Każdy krok konstrukcji zilustrowaliśmy obliczeniem w Pythonie.

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

0. Twierdzenie i plan dowodu

Przypomnijmy (Definicja 3.2 w rozdziale „Pojęcia wstępne"): $\mathcal{A}_1$ to wyrażenia jednej zmiennej zbudowane z liczb wymiernych, π, $\log 2$, zmiennej x, działań $+, -, \cdot$ oraz funkcji exp i sin (z dowolnym zagnieżdżaniem); $\mathcal{B}_1$ to to samo z modułem $|\cdot|$.

> **Twierdzenie R (Richardson, 1968 — postać konkretna)**
>
> 1. **(A — problem znaku.)** Nie istnieje algorytm, który dla dowolnego wyrażenia $A \in \mathcal{A}_1$ rozstrzyga, czy $A(x) < 0$ dla pewnego $x \in \mathbb{R}$.
> 2. **(B — problem zera/tożsamości.)** Nie istnieje algorytm, który dla dowolnego wyrażenia $B \in \mathcal{B}_1$ rozstrzyga, czy $B(x) = 0$ dla wszystkich $x \in \mathbb{R}$.
> 3. **(C — problem całkowania.)** Nie istnieje algorytm, który dla dowolnego $J \in \mathcal{B}_1$ rozstrzyga, czy istnieje $H \in \mathcal{B}_1$ różniczkowalne na $\mathbb{R}$ z $H' = J$.

W §9 podajemy oryginalne, ogólniejsze sformułowanie Richardsona (dowolne efektywnie zadane klasy funkcji, także częściowych). Wszystkie trzy części wynikają z jednej konstrukcji: dla ustalonego nierozstrzygalnego zbioru $U \subseteq \mathbb{N}$ budujemy algorytmicznie, dla każdego $a \in \mathbb{N}$, wyrażenie $G_a \in \mathcal{A}_1$ o **dychotomii z przerwą**

$$\begin{aligned} a \in U: &\quad G_a \geq 0 \text{ wszędzie i } \inf_t G_a(t) = 0, \\ a \notin U: &\quad G_a(t) > 1 \text{ dla każdego } t \in \mathbb{R}. \end{aligned}$$

Krok	Konstrukcja	Cel
§1	wielomian P z potęgami 2^{x_i}	kodujemy $a \in U$ jako istnienie **całkowitego** zera
§2	majoranty lokalne k_i	kontrolujemy zmianę P^2 przy zaokrąglaniu, w każdym miejscu przestrzeni
§3	kara ważona f, kwadraty współrzędnych F	„$f \leq 1$" wymusza całkowite zero — **przerwa** między 0 a 1

Krok	Konstrukcja	Cel
§4	krzywa $\Gamma_n(t)$ z $t\sin t$ i $t\sin t^3$	jedna zmienna, obraz gęsty w $\mathbb{R}^n$
§5	$G_a(t) = F(a, \Gamma_n(t))$	dychotomia dla jednej zmiennej
§6–8	$G_a - 1$, moduł, odcięcie	problemy znaku, tożsamości, całkowania

Cały dowód to łańcuch równoważności, prawdziwy dla każdego $a \in \mathbb{N}$:

$$\begin{aligned} a \in U &\iff \exists\, \mathbf{m} \in \mathbb{N}^n \;\; P(a, \mathbf{m}) = 0 && \text{(§1)} \\ &\iff \exists\, \mathbf{u} \in \mathbb{R}^n \;\; F(a, \mathbf{u}) = 0 && \text{(§3)} \\ &\iff \exists\, t \in \mathbb{R} \;\; G_a(t) < 1 && \text{(§4–5)} \\ &\iff A_a = G_a - 1 \text{ jest gdzieś ujemne} && \text{(§6)} \\ &\iff B_a = |A_a| - A_a \not\equiv 0 && \text{(§7)} \\ &\iff J_a = e^{x^2} S_a \text{ nie ma funkcji pierwotnej w } \mathcal{B}_1 && \text{(§8).} \end{aligned}$$

1. Wejście arytmetyczne

Twierdzenie 1.1 (Davis–Putnam–Robinson w postaci użytej przez Richardsona; zewnętrzne)

Istnieją $n \geq 2$ i wielomian $p(Y, X_1, \dots, X_n, Z_1, \dots, Z_n)$ o współczynnikach całkowitych takie, że dla

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

zbiór

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

jest nierozstrzygalny.

Wynika to z Twierdzenia 2.3 w rozdziale „Pojęcia wstępne”: potęgowanie o dowolnej podstawie $c = a^b$ da się wyrazić przez potęgi dwójki (dowód w rozdziale „Dodatki”, dodatek A), a układ równań $E_1 = \cdots = E_r = 0$ jest równoważny jednemu równaniu $\sum E_j^2 = 0$. Zmienne „nieme” pozwalają założyć $n \geq 2$. Wielomian p jest **ustalony raz na zawsze**; zmienia się tylko a.

Z wielomianu P korzystamy wyłącznie przez dwie własności:

- **(P1)** $P \in \mathcal{A}_{n+1}$, bo $2^x = \exp(x \log 2)$ — stąd w języku stała $\log 2$;
- **(P2)** $P(a, \mathbf{m}) \in \mathbb{Z}$ dla $a \in \mathbb{N}$, $\mathbf{m} \in \mathbb{N}^n$; w szczególności

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

Własność (P2) to jedyne miejsce, w którym działa *dyskretność* liczb całkowitych: małe przybliżone zero staje się dokładnym zerem.

Uwaga 1.2 (o stałej π)

W definicji wyrażeń „subelementarnych” Richardson (s. 515) nie wymienia π, choć zaraz używa $\sin(\pi x_i)$, a π występuje w założeniach twierdzenia głównego. Włączamy π od początku. Dzięki temu $\cos u = \sin(u + \pi/2)$ i pochodne pozostają w języku (Lemat 2.3).

Uwaga 1.3 (skąd $\log 2$ i $\exp$)

Funkcja exp i stała $\log 2$ są potrzebne **tylko** do zapisania 2^{x_i}, czyli tylko dlatego, że w 1968 r. znano nierozstrzygalność równań *wykładniczo* diofantycznych. Po twierdzeniu Matiyasevicha

(1970) można wziąć zwykły wielomian i wtedy cała konstrukcja obywa się bez exp i $\log 2$ — zob. §11.

2. Majoranty lokalne

Będziemy zaokrąglać rzeczywiste punkty do całkowitych i musimy kontrolować, jak przy tym zmienia się P^2. Punkty mogą leżeć dowolnie daleko, więc ograniczenie pochodnej na jednym zbiorze zwartym nie wystarczy: potrzebujemy **wzoru** — wyrażenia, które w każdym punkcie ogranicza funkcję na całym sąsiedztwie tego punktu.

Lemat 2.1 (majoranty lokalne; Richardson, *Lemma One*)

Dla każdego wyrażenia $A \in \mathcal{A}_d$ można algorytmicznie skonstruować $M_A \in \mathcal{A}_d$ takie, że dla każdego $\mathbf{z} \in \mathbb{R}^d$

$$M_A(\mathbf{z}) > 1 \qquad \text{oraz} \qquad |A(\mathbf{z}+\boldsymbol{\eta})| < M_A(\mathbf{z}) \quad \text{dla każdego } \boldsymbol{\eta} \text{ z } \|\boldsymbol{\eta}\|_\infty \le 1.$$

Dowód. Definiujemy M_A rekurencyjnie po drzewie wyrażenia A:

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

Każda prawa strona należy do $\mathcal{A}_d$, a rekurencja kończy się, bo drzewo jest skończone. Sprawdzamy obie własności przez indukcję po budowie wyrażenia; ustalmy $\mathbf{z}$ i $\boldsymbol{\eta}$ z $\|\boldsymbol{\eta}\|_\infty \le 1$.

- *Stała c.* Wartość nie zależy od punktu, a $c^2 + 2 - |c| = \left(|c| - \tfrac{1}{2}\right)^2 + \tfrac{7}{4} > 0$, więc $|c| < c^2 + 2$; oczywiście $c^2 + 2 \ge 2 > 1$.
- *Zmienna x_j.* $|z_j + \eta_j| \le |z_j| + 1 < z_j^2 + 2$, bo $z_j^2 + 2 - (|z_j| + 1) = \left(|z_j| - \tfrac{1}{2}\right)^2 + \tfrac{3}{4} > 0$.
- *Suma i różnica.* Z nierówności trójkąta $|A \pm B| \le |A| + |B| < M_A + M_B$ (wartości w $\mathbf{z}+\boldsymbol{\eta}$, majoranty w $\mathbf{z}$); suma majorant przekracza 2.
- *Iloczyn.* $|AB| = |A|\,|B| < M_A M_B$, bo obie majoranty są dodatnie; iloczyn liczb > 1 jest > 1.
- *Wykładnicza.* $e^A \le e^{|A|} < e^{M_A}$ z monotoniczności, a $e^{M_A} > e > 1$.
- *Sinus.* $|\sin A| \le 1 < 2$. ∎

Konstrukcja nigdy nie pyta o znak ani o wartość żadnego wyrażenia. Dla stałych z gramatyki $(q,\ \pi,\ \log 2)$ znak jest znany, więc działałoby też $|c| + 2$, jak w oryginale. Bierzemy $c^2 + 2$, bo ta reguła działa bez zmian także dla dowolnego podwyrażenia bez zmiennych traktowanego jako stała (tak robi funkcja `majoranta` poniżej), np. $\pi - \frac{22}{7}$, którego znaku algorytm nie musi umieć rozstrzygać. Oto implementacja na drzewach SymPy:

```
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
```

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

Lemat 2.1 obiecuje nierówność $|A(\mathbf{z}+\boldsymbol{\eta})| < M_A(\mathbf{z})$ dla każdego przesunięcia z $\|\boldsymbol{\eta}\|_\infty \le 1$. Sprawdźmy ją w 20 000 losowych parach $(\mathbf{z}, \boldsymbol{\eta})$ i policzmy naruszenia:

```
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
```

Żadnego naruszenia — jak być powinno: lemat jest udowodniony, a test go tylko ilustruje. Wynik $\exp((u^2+2)(v^2+2)) + 6$ (a nie $+2$) bierze się stąd, że SymPy zapisuje $-\sin u$ jako iloczyn $(-1)\cdot\sin u$, a reguła iloczynu daje $3\cdot 2$. To wciąż poprawna majoranta. Ogólniej:

Uwaga 2.2 (majoranta zależy od zapisu)

M_A jest funkcją **drzewa** A, a nie funkcji, którą A oznacza: drzewa $x-x$ i 0 dostają majoranty $2(x^2+2)$ i 2 (SymPy uprościłby $x-x$ do 0 już przy budowie drzewa). Nie przeszkadza to w niczym — redukcja potrzebuje *jakiegoś* udowodnionego ograniczenia danego wzorem, nie najlepszego.

Lemat 2.3 (efektywne różniczkowanie)

Dla $A \in \mathcal{A}_d$ wyrażenie na $\partial A/\partial x_j$ w języku $\mathcal{A}_d$ można skonstruować algorytmicznie.

Dowód. Rekurencyjnie: $\partial_j c = 0$, $\partial_j x_i \in \{0,1\}$, reguły sumy i iloczynu, $\partial_j e^A = e^A \partial_j A$ oraz $\partial_j \sin A = \sin\!\big(A + \tfrac{\pi}{2}\big)\,\partial_j A$. ■

Definicja 2.4 (wagi)

Niech $Q(y,\mathbf{x}) = P(y,\mathbf{x})^2$. Dla $i = 1,\dots,n$ kładziemy $k_i = M_{\partial Q/\partial x_i}$ (Lemat 2.1 zastosowany do wyrażenia z Lematu 2.3, w zmiennych $(y, x_1, \dots, x_n)$, z zerowym zaburzeniem y). Wtedy $k_i \in \mathcal{A}_{n+1}$, $k_i > 1$ oraz

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

Zwróćmy uwagę: pochodna jest liczona w punkcie $\mathbf{z}$, a waga w **środku** $\mathbf{x}$. Jedna waga kontroluje pochodną na całej kostce wokół $\mathbf{x}$.

3. Kara ważona i twierdzenie o zaokrąglaniu

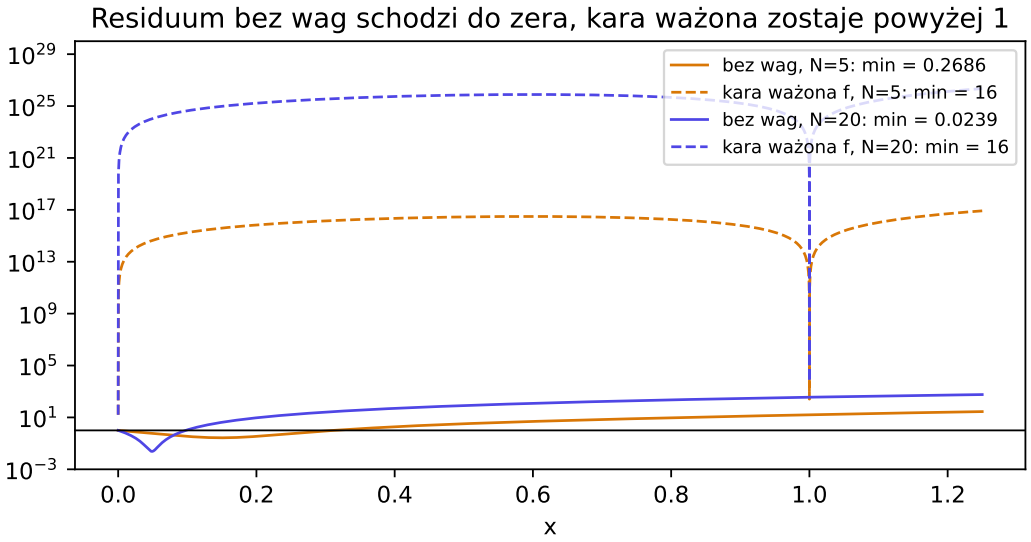
Na zbiorze $[0,\infty)^n$ zera wyrażenia $P(y,\mathbf{x})^2+\sum_i \sin^2(\pi x_i)$ to dokładnie rozwiązania całkowite (bo $\sin(\pi x)=0 \iff x\in\mathbb{Z}$). Później zastąpimy jednak zmienne punktami gęstej krzywej, która **dowolnie blisko podchodzi** do zera, ale nie musi w nie trafić. Potrzebujemy więc twierdzenia o **małych wartościach**, a nie tylko o zerach. Naiwne residuum tego nie daje:

Przykład 3.1 (bez wag nie ma przerwy)

Dla $N\geq 2$ niech $P_N(x)=Nx-1$: brak pierwiastka w $\mathbb{N}$. A jednak $P_N(1/N)^2+\sin^2(\pi/N)=\sin^2(\pi/N)\to 0$ przy $N\to\infty$. Żaden ustalony próg nie oddziela „tak” od „nie” w całej rodzinie: wielomian staje się coraz czulszy na zaburzenia.

Porównajmy na wykresie residuum bez wag z karą ważoną f (Definicja 3.2 poniżej, tu $n=1$) dla $N=5$ i $N=20$:

```
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()
```



Bez wag minimum residuum spada z 0,27 ($N=5$) do 0,024 ($N=20$), a kara ważona ma w obu przypadkach minimum 16 (w $x=0$, gdzie $P_N(0)^2=1$), wysoko nad progiem 1.

Definicja 3.2 (kara ważona; Richardson, s. 516)

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

To wyrażenie z $\mathcal{A}_{n+1}$, wszędzie nieujemne. Składnik P^2 mierzy błąd równania, składniki z sinusem — odległość od kraty $\mathbb{Z}^n$; wagi żądają tym większej bliskości kraty, im większe są pochodne P^2 w danym miejscu.

Lemat 3.3 (sinus mierzy odległość od liczby całkowitej)

Niech $x \ge 0$, $m \in \mathbb{N}$ — najbliższa x liczba naturalna (przy remisie dowolna), $d = |x-m|$. Wtedy $0 \le d \le \tfrac12$, $|\sin(\pi x)| = \sin(\pi d)$ oraz

$$2d \le |\sin(\pi x)| \qquad \text{i w konsekwencji} \qquad d \le \big(\sin^2(\pi x)\big)^{1/4}.$$

Dowód. Okresowość i symetria sinusa dają $|\sin(\pi x)| = |\sin(\pi(x-m))| = \sin(\pi d)$, bo $0 \le \pi d \le \pi/2$. Na $[0,\tfrac12]$ funkcja $t \mapsto \sin(\pi t)$ jest wklęsła (druga pochodna $-\pi^2\sin(\pi t) \le 0$), więc leży nad cięciwą łączącą $(0,0)$ i $(\tfrac12,1)$: $\sin(\pi d) \ge 2d$. Ponieważ $d \le \tfrac12 \le 2$, mamy $2d \ge d^2$; zatem $d^2 \le \sin(\pi d) = |\sin(\pi x)|$ i po spierwiastkowaniu $d \le |\sin \pi x|^{1/2} = (\sin^2 \pi x)^{1/4}$. ■

(Richardson dowodzi $t^2 \le \sin(\pi t)$ na $[0,\tfrac12]$ badaniem funkcji $\sin(\pi t) - t^2$; argument z wklęsłością jest krótszy.)

Twierdzenie 3.4 (o zaokrąglaniu; Richardson, Theorem One)

Dla każdego $a \in \mathbb{N}$ następujące warunki są równoważne:

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

W szczególności, jeśli $a \notin U$, to $f(a,\mathbf{x}) > 1$ w każdym punkcie $\mathbf{x} \ge 0$.

Dowód. (1)⇒(2): w całkowitym zerze $\mathbf{m}$ znikają P i wszystkie $\sin(\pi m_i)$. (2)⇒(3) jest oczywiste. Pozostaje (3)⇒(1). Niech $\mathbf{x} \ge 0$ i $f(a,\mathbf{x}) \le 1$.

Krok 1 (każdy składnik jest mały). Wszystkie składniki w nawiasie są nieujemne, więc

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

Pierwiastek czwartego stopnia (wagi są dodatnie) daje $\big(\sin^2(\pi x_i)\big)^{1/4} k_i(a,\mathbf{x}) \le \frac{1}{n+1}$.

Krok 2 (zaokrąglamy). Niech m_i będzie najbliższą x_i liczbą naturalną i $d_i = |m_i - x_i| \le \tfrac12$. Z Lematu 3.3

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

Krok 3 (szacujemy wzdłuż całego odcinka). Niech $\gamma(t) = \mathbf{x} + t(\mathbf{m}-\mathbf{x})$, $t \in [0,1]$. Każdy punkt odcinka leży w odległości $\le \tfrac12$ od $\mathbf{x}$ w normie maksimum, więc stosuje się (2.1). Z Twierdzenia 4.4 w rozdziale „Pojęcia wstępne”:

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

Zatem, z kroku 1 i (3.1),

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

gdzie ostatnia nierówność jest ostra, bo $(n+1)^{-4} < (n+1)^{-1}$.

Krok 4 (całkowitość). $a \in \mathbb{N}$ i $\mathbf{m} \in \mathbb{N}^n$, więc z (P2) $Q(a,\mathbf{m}) = P(a,\mathbf{m})^2$ jest nieujemną liczbą całkowitą mniejszą od 1, czyli 0. Stąd $P(a,\mathbf{m}) = 0$. ■

Podział pracy w dowodzie

Nieujemne współrzędne zaokrąglają się do liczb **naturalnych** (to ważne, bo 2^x dla ujemnych całkowitych nie jest całkowite). Wagi kontrolują cały odcinek, nie tylko końce. Czynnik $(n+1)^4$ zostawia zapas poniżej 1. Całkowitość pojawia się dopiero w ostatnim kroku, gdzie przybliżone zero staje się dokładnym.

Uwaga 3.5 (wariant z kwadratami)

Te same wagi działają w prostszej karze $f_{\text{alt}} = (n+1)^2[P^2 + \sum_i \sin^2(\pi x_i)k_i^2]$: z $f_{\text{alt}} \le 1$ i silniejszej nierówności $2d_i \le |\sin \pi x_i|$ dostajemy $d_i k_i \le \frac{1}{2(n+1)}$, więc $Q(a,\mathbf{m}) \le (n+1)^{-2} + \frac{n}{2(n+1)} < 1$. W dalszym ciągu trzymamy się czwartych potęg Richardsona.

Uwaga 3.6 (przerwa jest nawet jednostajna)

Jeśli $f(a,\mathbf{x}) \le s$, ten sam rachunek daje $Q(a,\mathbf{m}) \le \varphi(s) = \frac{s}{(n+1)^4} + \frac{n\,s^{1/4}}{n+1}$. Ponieważ $\varphi(1) < 1$, a φ jest ciągła, istnieje $s_0 > 1$ (zależne tylko od n) z $\varphi(s_0) < 1$. Wtedy dla $a \notin U$ mamy $f(a,\cdot) > s_0$, czyli $\inf f \ge s_0 > 1$. Ta wzmocniona wersja nie jest dalej potrzebna.

Ostatni krok tej części usuwa ograniczenie $\mathbf{x} \ge 0$: kwadraty przebiegają cały zbiór $[0,\infty)^n$.

Wniosek 3.7 (przerwa na całym $\mathbb{R}^n$; Richardson, Corollary)

Niech $F(y,u_1,\dots,u_n) = f(y,u_1^2,\dots,u_n^2)$. Wtedy $F \in \mathcal{A}_{n+1}$, $F \ge 0$ i dla $a \in \mathbb{N}$:

- jeśli $a \in U$, to $F(a,\mathbf{u}) = 0$ dla pewnego $\mathbf{u} \in \mathbb{R}^n$;
- jeśli $a \notin U$, to $F(a,\mathbf{u}) > 1$ dla **każdego** $\mathbf{u} \in \mathbb{R}^n$.

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

Dowód. Odwzorowanie $\mathbf{u} \mapsto (u_1^2,\dots,u_n^2)$ przekształca $\mathbb{R}^n$ **na** $[0,\infty)^n$ (świadkiem dla $\mathbf{m}$ jest $u_i = \sqrt{m_i}$). Teza wynika z Twierdzenia 3.4. Pierwiastki pojawiają się tylko w dowodzie; wyrażenie F zawiera same kwadraty. ■

4. Jedna zmienna wystarczy: krzywa gęsta

Mamy już nierozstrzygalną nierówność w n zmiennych. Richardson zastępuje wszystkie zmienne funkcjami **jednej** zmiennej, używając dwóch oscylacji o różnych skalach:

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

Idea: blisko dużego v, na przedziale długości rzędu δ/v funkcja h zmienia się o co najwyżej δ (bo $|h'(t)| \le 1 + t$), a faza t^3 zmienia się o około $3v^2 \cdot \delta/v = 3\delta v$ — dowolnie dużo. Zatem h prawie stoi, a g (o amplitudzie $\approx v$) przebiega cały przedział $[-v,v]$.

Lemat 4.1 (dowolnie odległe rozwiązania $h(v)=a$)

Dla każdego $a\in\mathbb{R}$ i $L>0$ istnieje $v>L$ z $h(v)=a$.

Dowód. Wybierzmy $k\in\mathbb{N}$ tak duże, że $p=2\pi k+\frac{\pi}{2}>\max\{L,|a|\}$, i niech $q=2\pi k+\frac{3\pi}{2}$. Wtedy $h(p)=p>a$ i $h(q)=-q<a$, więc z twierdzenia Darboux $h(v)=a$ dla pewnego $v\in(p,q)$; oczywiście $v>L$. ∎

Lemat 4.2 (podnoszenie: jedna współrzędna przybliżona, druga dokładna; Richardson, *Lemmas Two* i *Three*)

Dla dowolnych $a,b\in\mathbb{R}$, $\delta>0$ i $M\geq 0$ istnieje $w>M$ takie, że

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

Dowód. Wybierzmy $R>\max\{M,|b|,1,4\pi/\delta\}$. Z Lematu 4.1 bierzemy $v>\sqrt{R^2+\delta/2}$ z $h(v)=a$ i kładziemy $u=\sqrt{v^2-\delta/2}$. Wtedy $v>u>R$ oraz $v^2-u^2=\delta/2$.

Przedział $[u,v]$ jest krótki dla h. Ponieważ $u>1$,

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

Faza t^3 zmienia się na nim o więcej niż okres. Korzystając z $v^2=u^2+\delta/2$:

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

g przechodzi przez b. Przedział otwarty (u^3,v^3) ma długość $>2\pi$, więc zawiera liczbę $\theta_+\equiv\frac{\pi}{2}$ i liczbę $\theta_-\equiv\frac{3\pi}{2}\pmod{2\pi}$ (obie progresje mają krok 2π). Ich pierwiastki sześcienne $t_\pm=\theta_\pm^{1/3}$ leżą w (u,v) i

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

Z twierdzenia Darboux istnieje w między t_+ i t_- (więc $w\in(u,v)$, $w>M$) z $g(w)=b$.

h zostaje blisko a. Dla $s\in[u,v]$: $|h'(s)|=|\sin s+s\cos s|\leq 1+s\leq 1+v$. Z twierdzenia o wartości średniej i (4.1)

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

Ten dowód jest konstruktywny i daje się wykonać numerycznie. Ponieważ faza t^3 jest olbrzymia, liczymy w arytmetyce wysokiej precyzji (`mpmath`, 80 cyfr):

```
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):
```

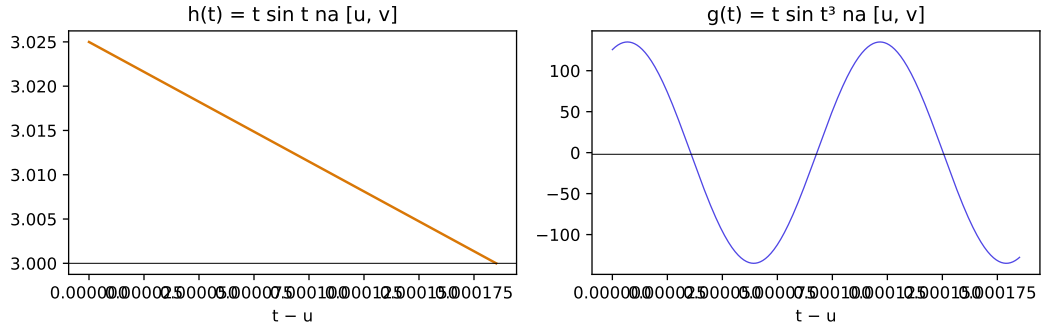
```
    """Lemat 4.2: zwraca w > M z |h(w) - a| < delta i g(w) = b — dokładnie według
    ↪ dowodu."""
    a, b, delta = mp.mpf(a), mp.mpf(b), mp.mpf(delta)
    R = max(mp.mpf(M), abs(b), mp.mpf(1), 4 * mp.pi / delta) * (1 + mp.mpf('1e-9'))
    dolna = mp.sqrt(R**2 + delta / 2)                          # 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
```

Otrzymaliśmy $w \approx 135{,}07$: $|h(w) - 3| \approx 0{,}02 < \delta$, a $g(w)$ różni się od -2 o około $4 \cdot 10^{-73}$, czyli na granicy precyzji obliczeń. Na przedziale $[u, v]$, długości około $1{,}85 \cdot 10^{-4}$, funkcja h prawie stoi, a g wykonuje ponad półtorej oscylacji o amplitudzie około 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()
```



Teraz składamy krzywą w $\mathbb{R}^n$. Piszemy $g^{[j]}$ dla j-krotnego **złożenia** g (nie potęgi), $g^{[0]}(t) = t$.

Definicja 4.3 (krzywa Richardsona)

Dla $n \geq 2$:

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

Pierwsze $n-1$ współrzędnych zaczyna się od h, ostatnia to iteracja samego g; np. $\Gamma_2 = (h, g)$, $\Gamma_4 = (h(t), h(g(t)), h(g(g(t))), g(g(g(t))))$. Zachodzi rekurencja

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

Twierdzenie 4.4 (gęstość z dokładną ostatnią współrzędną; Richardson, Theorem Two)

Dla $n \geq 2$, $\mathbf{a} \in \mathbb{R}^n$, $\delta > 0$ i $M \geq 0$ istnieje $w > M$ takie, że

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

W szczególności zbiór $\Gamma_n((M, \infty))$ jest gęsty w $\mathbb{R}^n$.

Dowód. Indukcja względem n. Dla $n = 2$ to Lemat 4.2. Załóżmy tezę dla n i weźmy cel $(a_1, \ldots, a_{n+1})$. Z założenia indukcyjnego dla „ogona” $(a_2, \ldots, a_{n+1})$ (z $M = 0$) dostajemy $w^* > 0$, dla którego pierwsze $n-1$ współrzędnych $\Gamma_n(w^*)$ przybliżają $a_2, \ldots, a_n$ z dokładnością δ, a ostatnia równa się a_{n+1}. Z Lematu 4.2 dla celów a_1 i w^* dostajemy $w > M$ z $|h(w) - a_1| < \delta$ oraz $g(w) = w^*$. Z (4.3) $\Gamma_{n+1}(w) = (h(w), \Gamma_n(w^*))$ — to dokładnie żądane przybliżenie. Gęstość: każdy niepusty zbiór otwarty zawiera kostkę $\{\mathbf{u} : \|\mathbf{u} - \mathbf{a}\|_\infty < \delta\}$. ■

Uwaga 4.5 (dlaczego ostatnia współrzędna jest dokładna)

Krok indukcyjny żąda $g(w) = w^*$ **dokładnie**. Gdyby było tylko $g(w) \approx w^*$, błąd zostałby wzmocniony przez bardzo stromo zmieniające się iteracje g; zapanować nad nim można tylko osobnym argumentem z ciągłości Γ_n w punkcie w^*, z astronomicznie małą tolerancją. Dokładne „podniesienie” ostatniej współrzędnej sprawia, że cały ogon jest odtworzony bez błędu.

Sprawdźmy to numerycznie w $\mathbb{R}^3$: trafiamy w $(1{,}5;\ -4;\ 7)$ z $\delta = 0{,}05$.

```
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
```

Dwie pierwsze współrzędne różnią się od celu o 0,008 i 0,010, czyli o mniej niż $\delta = 0{,}05$, a trzecia równa się 7 z dokładnością do precyzji obliczeń, jak przewiduje Twierdzenie 4.4.

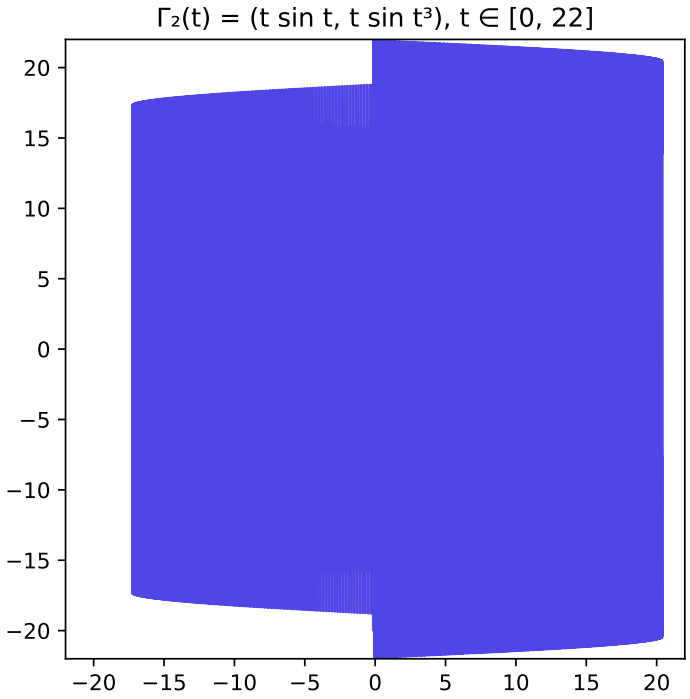
Uwaga 4.6 (gęstość, a nie surjekcja)

Obraz Γ_n jest gęsty, ale ma **miarę zero** w $\mathbb{R}^n$ dla $n \geq 2$: na zwartym przedziale parametru długości ℓ krzywa jest lipschitzowska ze stałą L, więc jej obraz da się pokryć N kostkami o boku $2L\ell/N$, o łącznej objętości $(2L\ell)^n N^{1-n} \to 0$; cała prosta parametru jest przeliczalną sumą takich przedziałów. Dlatego dokładne trafianie we wszystkie współrzędne jest złym

celem — i nie jest potrzebne: wystarczy wejść w otoczenie punktu (Fakt 4.6 w rozdziale „Pojęcia wstępne”).

Narysujmy początkowy fragment krzywej Γ_2, dla $t \in [0, 22]$:

```
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()
```



Na rysunku krzywa zlewa się w jednolitą plamę: dla $t \le 22$ faza t^3 wykonuje około 1700 obrotów, więc kolejne przebiegi drugiej współrzędnej leżą bardzo gęsto. To efekt grubości linii, bo sam obraz krzywej ma miarę zero (Uwaga 4.6).

5. Przeniesienie na jedną zmienną

Lemat 5.1 (co zachowuje podstawienie gęstej krzywej)

Niech $H\colon \mathbb{R}^n \to \mathbb{R}$ będzie ciągła, $c \in \mathbb{R}$, $M \ge 0$. Wtedy

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

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

Dowód. „⇐”: weźmy $\mathbf{u} = \Gamma_n(t)$. „⇒”: jeśli $H(\mathbf{u}_0) < c$, to z ciągłości $H < c$ na kostce $\|\mathbf{u}-\mathbf{u}_0\|_\infty < \eta$, a z Twierdzenia 4.4 krzywa $\Gamma_n((M,\infty))$ w nią wchodzi. Równość kresów: krzywa leży w $\mathbb{R}^n$, więc inf po krzywej $\ge$ inf po $\mathbb{R}^n$; a dla każdego $c > \inf_{\mathbb{R}^n} H$ pierwsza część daje punkt krzywej z $H < c$. ■

Dla nierówności **nieostrej** analogiczna równoważność dla dowolnego zbioru gęstego jest fałszywa: $D = \mathbb{R} \setminus \{0\}$ jest gęsty w $\mathbb{R}$, $H(z) = z^2$ ma na D i na $\mathbb{R}$ ten sam kres dolny 0, a jednak $H(z) \le 0$ ma rozwiązanie tylko w $\mathbb{R}$. Dlatego najpierw zbudowaliśmy przerwę (§3).

Definicja 5.2

$G(y,t) = F(y, \Gamma_n(t))$ (podstawienie n wyrażeń za n zmiennych) oraz $G_a(t) = G(a,t)$ dla $a \in \mathbb{N}$. Zatem $G \in \mathcal{A}_2$, $G_a \in \mathcal{A}_1$, a G_a otrzymujemy z a algorytmicznie (podstawiając liczbę a za y).

Twierdzenie 5.3 (dychotomia dla jednej zmiennej; Richardson, Corollary to Theorem Two)

Każda funkcja G_a jest nieujemna. Ponadto:

1. jeśli $a \in U$, to dla każdego $\varepsilon > 0$ i $M \ge 0$ istnieje $t > M$ z $G_a(t) < \varepsilon$;
2. jeśli $a \notin U$, to $G_a(t) > 1$ dla każdego $t \in \mathbb{R}$.

W szczególności $a \in U \iff \exists t\ G_a(t) < 1 \iff \exists t\ G_a(t) \le 1 \iff \inf_t G_a(t) = 0$.

Dowód. Nieujemność: $F \ge 0$. Jeśli $a \in U$, Wniosek 3.7 daje $\mathbf{u}_0$ z $F(a, \mathbf{u}_0) = 0$; Lemat 5.1 dla $H = F(a, \cdot)$ i $c = \varepsilon$ daje tezę. Jeśli $a \notin U$, to $F(a, \mathbf{u}) > 1$ w **każdym** punkcie, w szczególności w punktach $\Gamma_n(t)$. ■

Uwaga 5.4 (dlaczego nie twierdzimy, że G_a ma zero)

Zero funkcji $F(a, \cdot)$ daje dowolnie małe wartości G_a (gęstość + ciągłość), ale nie zero G_a — krzywa może ominąć ten punkt. Informację arytmetyczną niesie ostry próg 1, chroniony przez twierdzenie o zaokrąglaniu. Kolejność konstrukcji ma znaczenie: najpierw przerwa (§3), potem krzywa (§4). W kierunku „$G_a(t) \le 1 \Rightarrow a \in U$” nie używamy gęstości, tylko przerwy: punkt $\Gamma_n(t)$ sam jest świadkiem $F \le 1$.

Cały łańcuch na zabawkowym przykładzie

Prawdziwy wielomian z Twierdzenia 1.1 jest ogromny. Możemy jednak prześledzić **tę samą konstrukcję** na „równaniu diofantycznym” w dwóch zmiennych: $x_1^2 + x_2^2 = 5$ (ma rozwiązanie $(1,2)$) oraz $x_1^2 + x_2^2 = 3$ (nie ma). Budujemy wagi, karę f, F i $G(t) = F(\Gamma_2(t))$, a potem — dla przypadku „tak” — szukamy punktu z $G(t) < 1$, celując krzywą w zero F, czyli w $\mathbf{u} = (1, \sqrt{2})$ (bo f znika w $\mathbf{x} = (1,2) = (u_1^2, u_2^2)$).

```
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
```

Dla równania rozwiązywalnego punkt z $G(t) < 1$ istnieje, ale nasza konstrukcja znajduje go dopiero przy $t \approx 1{,}26 \cdot 10^{11}$ i trzeba go liczyć z ponad stu cyframi precyzji (faza $t^3 \approx 2 \cdot 10^{33}$). Dla równania nierozwiązywalnego nawet „najlepsza próba” daje $G \approx 81 > 1$ — zgodnie z twierdzeniem, które gwarantuje $G > 1$ **wszędzie**. Redukcja jest efektywna jako przekształcenie wyrażeń, ale świadkowie mogą być astronomicznie daleko: to kolejny powód, dla którego numeryczne „sprawdzanie” nie zastąpi rozstrzygania.

Zobacz też

Większy przykład: rodzina wyrażeń z równania $x^3 + y^3 + z^3 = a$ — w tym przypadek, w którym świadek ma około 2770 cyfr, i przypadek, w którym nikt nie zna odpowiedzi — w rozdziale „Przykład: wyrażenie, którego znaku nie widać”.

6. Problem znaku

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

Nie istnieje algorytm, który dla $A \in \mathcal{A}_1$ rozstrzyga, czy $\exists t\ A(t) < 0$.

Dowód. Dla $a \in \mathbb{N}$ budujemy algorytmicznie $A_a(t) = G_a(t) - 1$. Z Twierdzenia 5.3 $a \in U \iff \exists t\ A_a(t) < 0$. Algorytm dla problemu znaku rozstrzygałby U, wbrew Twierdzeniu 1.1. ■

Trudne egzemplarze nie zawierają modułu: każde A_a jest analityczne i określone wszędzie. Z Twierdzenia 5.3 redukcja działa także, gdy świadka szukamy tylko wśród $t > 0$ lub $t > M$. Zamiana A na $-A$ daje nierozstrzygalność pytania „czy A jest gdzieś dodatnie”, a zamiana odpowiedzi — pytania „czy $A \geq 0$ wszędzie”.

7. Problem tożsamości (zera)

Moduł zamienia pytanie o **znak** na pytanie o **tożsamość**:

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

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

Nie istnieje algorytm, który dla $B \in \mathcal{B}_1$ rozstrzyga, czy $B \equiv 0$.

Dowód. Niech $B_a(t) = |G_a(t)-1| - (G_a(t)-1) = |A_a(t)| - A_a(t)$. Jeśli $a \notin U$, to $G_a > 1$ wszędzie, więc $B_a \equiv 0$. Jeśli $a \in U$, to $G_a(t_0) < 1$ w pewnym t_0, więc $B_a(t_0) = 2(1 - G_a(t_0)) > 0$. Zatem $B_a \equiv 0 \iff a \notin U$; algorytm dla tożsamości (z zamienioną odpowiedzią) rozstrzygałby U. ■

Wniosek 7.2 (równość i postaci kanoniczne)

Równość funkcji wyznaczonych przez dwa wyrażenia z $\mathcal{B}_1$ jest nierozstrzygalna. W szczególności **nie istnieje** algorytm przypisujący każdemu wyrażeniu z $\mathcal{B}_1$ skończoną postać kanoniczną (taką, że postaci są równe wtedy i tylko wtedy, gdy funkcje są równe).

Dowód. $B \equiv C \iff B - C \equiv 0$. Algorytm postaci kanonicznej rozstrzygałby równość porównaniem dwóch napisów. ■

To jest precyzyjna treść wniosku z wykładu 1, że każda funkcja `simplify` musi dla niektórych wyrażeń zawieść: nie rozpoznać zera, pomylić się albo nie skończyć obliczeń (zob. „Jak działa system algebry komputerowej?").

Uwaga 7.3 (rola modułu)

Moduł jest mostem od znaku do tożsamości. Dla czysto analitycznego języka $\mathcal{A}_1$ ten dowód **nie** daje nierozstrzygalności tożsamości: funkcja analityczna znikająca na przedziale znika wszędzie (Twierdzenie 4.2 w rozdziale „Pojęcia wstępne"), więc „płaskowyż zer" typu $|s| - s$ jest niemożliwy. W odróżnieniu od funkcji analitycznych funkcje z modułem mają dzielniki zera: $(|x| + x)(|x| - x) \equiv 0$, choć żaden czynnik nie jest tożsamościowo zerem.

8. Problem całkowania

Pytamy, czy dla danego $J \in \mathcal{B}_1$ istnieje **wyrażenie** $H \in \mathcal{B}_1$ różniczkowalne na $\mathbb{R}$ z $H' = J$ — pytanie o *reprezentowalną* funkcję pierwotną, nie o istnienie funkcji pierwotnej w ogóle (ta, dla ciągłego J, zawsze istnieje: $\int_0^x J$).

Definicja 8.1 (lokalna przeszkoda dla całkowania)

Funkcja Δ określona na $\mathbb{R}$ i reprezentowana w języku $\mathcal{E}$ jest **lokalną przeszkodą**, jeśli nie istnieją $H \in \mathcal{E}$ i niepusty przedział otwarty I, na którym H jest różniczkowalna i $H' = \Delta$.

W rozdziale „Dodatki" (dodatek B) dowodzimy, że $\Delta(x) = e^{x^2}$ jest lokalną przeszkodą dla $\mathcal{B}_1$ (twierdzenie Liouville'a plus lokalne usuwanie modułów).

Potrzebujemy wyrażenia, które w przypadku „nie" jest tożsamościowo zerem, a w przypadku „tak" równa się **dokładnie** 1 **na pewnym przedziale**. Richardson buduje je z obciętego odejmowania:

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

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

$S_a \in \mathcal{B}_1$ jest ciągła i ma wartości w $[0, 1]$.

Lemat 8.2 (wyłącznik)

Jeśli $a \notin U$, to $S_a \equiv 0$. Jeśli $a \in U$, to $S_a = 1$ na pewnym niepustym przedziale otwartym.

Dowód. W pierwszym przypadku $G_a > 1$ wszędzie. W drugim Twierdzenie 5.3 daje t_0 z $G_a(t_0) < \frac{1}{2}$ (tu potrzebujemy pełnej siły „$\inf G_a = 0$”, a nie tylko „$G_a < 1$ gdzieś”), a z ciągłości $G_a < \frac{1}{2}$ na otoczeniu t_0. ■

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

Niech język $\mathcal{E}$ funkcji określonych wszędzie efektywnie zawiera $\mathcal{B}_1$, będzie efektywnie zamknięty na mnożenie i zawiera lokalną przeszkodę Δ. Wtedy nie istnieje algorytm rozstrzygający, czy dane $J \in \mathcal{E}$ ma funkcję pierwotną w $\mathcal{E}$. W szczególności dotyczy to $\mathcal{B}_1$ z $\Delta = e^{x^2}$.

Dowód. Niech $J_a = \Delta \cdot S_a$. Jeśli $a \notin U$, to $J_a \equiv 0$ i funkcja stała 0 jest jej funkcją pierwotną w $\mathcal{E}$. Jeśli $a \in U$, to na pewnym przedziale I mamy $S_a = 1$, czyli $J_a = \Delta$; funkcja pierwotna $H \in \mathcal{E}$ dla J_a spełniałaby $H' = \Delta$ na I, wbrew definicji przeszkody. Zatem J_a ma funkcję pierwotną w $\mathcal{E}$ wtedy i tylko wtedy, gdy $a \notin U$. ■

Uwaga 8.4 (problem jest globalny)

Dla $a \in U$ funkcja J_a może znikać na innym przedziale — tam funkcja pierwotna 0 jest elementarna. Przeszkoda leży na przedziale, gdzie $S_a = 1$. Pytamy o **jedno** wyrażenie dobre na całej prostej.

9. Oryginalne, ogólne sformułowanie Richardsona

Richardson pracuje z dowolnym zbiorem wyrażeń E reprezentujących funkcje **częściowe** i ze zbiorem funkcji E^*. Zakłada, że E^* zawiera identyczność i stałe wymierne, a dla $A, B \in E$ można **algorytmicznie** zbudować wyrażenia dla $A \pm B$, $A \cdot B$, $A(B(x))$. Równość $f \equiv g$ rozumie z dziedziną (Definicja 3.3 w rozdziale „Pojęcia wstępne”). Warunki:

- **(R1)** E^* zawiera stałe π, $\log 2$ i funkcje e^x, $\sin x$;
- **(R2)** istnieje $\mu \in E^*$ z $\mu(x) = |x|$ dla $x \neq 0$ (wartość $\mu(0)$ dowolna lub nieokreślona);
- **(R3)** istnieje określona wszędzie $\Delta \in E^*$ taka, że dla żadnego $f \in E^*$ i żadnego przedziału I nie zachodzi $f' \equiv \Delta$ na I.

Twierdzenie 9.1 (Richardson, 1968)

Przy powyższych założeniach efektywności:

1. (R1) ⇒ nierozstrzygalne jest pytanie, czy wyrażenie przyjmuje wartość ujemną w pewnym punkcie dziedziny;
2. (R1)+(R2) ⇒ nierozstrzygalny jest problem tożsamości $A \equiv 0$;
3. (R1)+(R2)+(R3) ⇒ nierozstrzygalny jest problem całkowania $\exists f \in E^*\ f' \equiv A$.

Dowód. Wyrażenia G_a z §5 mają reprezentantów w E: ustalamy raz na zawsze wyrażenia dla skończenie wielu stałych i funkcji bazowych konstrukcji (x, e^x, $\sin x$) i stosujemy efektywne konstruktory wzdłuż drzewa; liczbę a budujemy z 1 przez dodawanie. To daje (1) przez $G_a - 1$.

(2): niech $\widetilde{B}_a = \mu(G_a - 1) - (G_a - 1)$. Jeśli $a \notin U$, argument μ nigdy nie jest zerem, więc $\widetilde{B}_a$ jest określone wszędzie i równe 0. Jeśli $a \in U$, w punkcie t_0 z $G_a(t_0) < 1$ argument jest niezerowy i $\widetilde{B}_a(t_0) = 2(1 - G_a(t_0)) > 0$; zatem $\widetilde{B}_a \not\equiv 0$ niezależnie od zachowania w innych punktach.

(3): w odcięciu zastępujemy moduł przez μ: $q_a = 2 - 2G_a$, $T_a = \frac{\mu(q_a)+q_a}{2}$, $\tilde{S}_a = 1 - \frac{\mu(1-T_a)+(1-T_a)}{2}$. Jeśli $a \notin U$: $q_a < 0$, więc $T_a = 0$ wszędzie, a zewnętrzne μ dostaje 1, więc $\tilde{S}_a$ jest zerem określonym wszędzie. Jeśli $a \in U$: na przedziale z $G_a < \frac{1}{2}$ mamy $q_a > 1$, $T_a = q_a$, $1 - T_a < 0$, wszystkie argumenty μ są niezerowe i $\tilde{S}_a = 1$. Dalej jak w Twierdzeniu 8.3, z $\tilde{J}_a = \Delta\, \tilde{S}_a$ (określoność Δ wszędzie gwarantuje, że w przypadku „nie” $\tilde{J}_a$ jest zerem określonym wszędzie). ■

Na progach $G_a = 1$ i $G_a = \frac{1}{2}$ wyrażenia z μ mogą być nieokreślone — dowód ich unika: w przypadku „nie” $G_a > 1$ wszędzie, a w przypadku „tak” używamy przedziału z $G_a < \frac{1}{2}$.

10. Co dokładnie udowodniliśmy

Audyt efektywności. Algorytm redukcji wykonuje wyłącznie operacje na wyrażeniach: bierze ustalone P, tworzy $Q = P^2$, różniczkuje, buduje wagi z reguł Lematu 2.1, składa f, podstawia kwadraty, potem współrzędne krzywej, wreszcie liczbę a za y. Nigdy nie szuka pierwiastków, punktów krzywej, minimów ani znaków stałych. Te obiekty występują tylko w **dowodzie**, że zbudowane wyrażenie ma żądaną własność.

> **Stwierdzenie 10.1 (ujemność jest półrozstrzygalna)**
>
> Istnieje algorytm, który dla $A \in \mathcal{B}_1$ zatrzymuje się wtedy i tylko wtedy, gdy $A(t) < 0$ dla pewnego t.

Dowód. Wartości stałych, exp, sin i $|\cdot|$ w punktach wymiernych da się przybliżać z gwarantowanym błędem, a skończone złożenie zachowuje tę własność. Wyliczamy pary $(q, k) \in \mathbb{Q} \times \mathbb{N}$ i dla każdej liczymy wymierne $r_{q,k}$ z $|r_{q,k} - A(q)| \le 2^{-k}$; zatrzymujemy się, gdy $r_{q,k} + 2^{-k} < 0$. Test nigdy się nie myli. Jeśli $A(t_0) < 0$, to z ciągłości $A < 0$ w otoczeniu t_0, więc w pewnym wymiernym q; dla dużego k test się powiedzie. ■

Ta sama metoda półrozstrzyga **różność od zera**. Z twierdzenia Posta (Twierdzenie 1.5 w rozdziale „Pojęcia wstępne”) i Twierdzenia 7.1: **tożsamość z zerem nie jest nawet półrozstrzygalna** — inaczej, łącząc obie procedury, rozstrzygalibyśmy ją. Oto działająca procedura półrozstrzygająca z arytmetyką przedziałową:

```
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
```

W pierwszym przykładzie procedura znalazła certyfikat w $q = 1$: cały przedział wartości leży poniżej zera. W drugim wynik `None` oznacza tylko wyczerpanie limitu kroków, a nie odpowiedź „nie”: bez limitu procedura szukałaby certyfikatu w nieskończoność. Twierdzenie Richardsona mówi, że żadnej takiej procedury nie da się uzupełnić procedurą przeciwną, która zatrzymuje się dokładnie wtedy, gdy $A \geq 0$ wszędzie (por. problem 8 w rozdziale „Pytania kontrolne i problemy”).

Trzy różne pytania o zero.

Pytanie	Postać logiczna	Co wiemy
tożsamość z zerem	$\forall t\ A(t) = 0$	nierozstrzygalne w $\mathcal{B}_1$ (Twierdzenie 7.1)
istnienie zera	$\exists t\ A(t) = 0$	nierozstrzygalne (Wang 1974; Laczkovich 2003) — nie wynika z samej gęstości krzywej
czy stała jest zerem	$c = 0$, bez zmiennej	**problem otwarty**; rozstrzygalny przy hipotezie Schanuela dla stałych wykładniczo-logarytmicznych (Richardson 1997)

Richardson pisał w 1968 r. (s. 515): „Nie jest znana metoda rozstrzygania, czy wyrażenie subelementarne bez zmiennych przedstawia liczbę mniejszą od zera. Czy taka metoda istnieje, jest problemem otwartym.” Więcej w rozdziale „Kontekst, konsekwencje i dalsze wyniki”.

11. Wersja współczesna: wystarczą wielomiany i sinus

Twierdzenie Matiyasevicha (Twierdzenie 2.4 w rozdziale „Pojęcia wstępne”) pozwala wziąć jako P zwykły wielomian $p(y, \mathbf{x}) \in \mathbb{Z}[y, \mathbf{x}]$ z nierozstrzygalnym zbiorem U. Wtedy pochodne $Q = p^2$ są wielomianami i wagi można wziąć wielomianowe: jeśli $\partial Q/\partial x_i = \sum_{\alpha} c_{i,\alpha}\, y^{\alpha_0} x_1^{\alpha_1} \cdots x_n^{\alpha_n}$, to

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

spełnia (2.1) (współczynniki są znanymi liczbami całkowitymi, więc $|c_{i,\alpha}|$ liczymy od razu — to nie jest operacja modułu w języku). Kara, kwadraty i krzywa h, g używają tylko działań arytmetycznych i sinusa.

> **Wniosek 11.1 (Caviness 1970 po Matiyasevichu)**
>
> Niech $\mathcal{S}_1$ będzie językiem generowanym przez liczby wymierne, π, zmienną x, działania $+, -, \cdot$ i funkcję sin. Problem znaku jest nierozstrzygalny dla $\mathcal{S}_1$; po dodaniu modułu nierozstrzygalny jest także problem tożsamości.

Dowód. Powtarzamy §3–§5 z wielomianowym P i powyższymi wagami, a potem §6–§7 bez zmian. ■

Laczkovich (2003) usunął także π: w pierścieniu generowanym przez liczby całkowite oraz funkcje x, $\sin x^n$, $\sin(x \sin x^n)$ $(n = 1, 2, \ldots)$ nierozstrzygalne jest, czy funkcja przyjmuje wartość dodatnią i czy ma pierwiastek. Zob. rozdział „Kontekst, konsekwencje i dalsze wyniki”.

Przykład: wyrażenie, którego znaku nie widać

Twierdzenie Richardsona mówi, że **nie istnieje algorytm** rozstrzygający, czy wyrażenie jest gdzieś ujemne. Jak to wygląda na konkretnym przykładzie? Weźmiemy maszynę z dowodu („Pełny dowód krok po kroku", §2–§5) i włożymy do niej równanie

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

Dla każdej liczby a maszyna produkuje jawne wyrażenie $A_a(t)$ jednej zmiennej, zbudowane wyłącznie z liczb całkowitych, π, t, działań $+, -, \cdot$ i sinusa. Zapytamy: **czy $A_a(t) \geq 0$ dla wszystkich t?** — dla czterech wartości: $a = 3,\ 4,\ 42,\ 114$.

Numeryka da tę samą odpowiedź we wszystkich czterech przypadkach. Prawda bywa jednak różna:

- $a = 4$: tak, $A_4 > 0$ wszędzie — dowód zajmuje trzy linijki teorii liczb;
- $a = 3$: nie — ale miejsce, w którym $A_3 < 0$, leży w okolicy $t \approx 10^{13}$, a t trzeba podać z dokładnością do około stu cyfr;
- $a = 42$: nie — wiadomo to dopiero od 2019 roku, a odpowiednie t ma około 2770 cyfr znaczących;
- $a = 114$: **nikt nie wie**.

```
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. Wyrażenie $A_a(t)$

Postępujemy dokładnie według dowodu, z jedną zmianą: niewiadome są **całkowite** (dowolnego znaku), a nie naturalne.

- **Równanie:** $P_a(\mathbf{x}) = x_1^3 + x_2^3 + x_3^3 - a$, $n = 3$ niewiadome.
- **Wagi** (Definicja 2.4): $k_i = M_{\partial Q/\partial x_i}$ dla $Q = P_a^2$. Ponieważ $\partial Q/\partial x_i = 6x_i^2 P_a$, Lemat 2.1 daje
$$k_i(\mathbf{x}) = 38\,(x_i^2 + 2)^2\, M_a(\mathbf{x}), \qquad M_a(\mathbf{x}) = \sum_{j=1}^{3} (x_j^2 + 2)^3 + a^2 + 2.$$
- **Kara** (Definicja 3.2): $f_a(\mathbf{x}) = 4^4\big[P_a(\mathbf{x})^2 + \sum_{i=1}^{3} \sin^2(\pi x_i)\, k_i(\mathbf{x})^4\big]$.
- **Krzywa** (Definicja 4.3): $\Gamma_3(t) = (h(t),\, h(g(t)),\, g(g(t)))$, gdzie $h(t) = t\sin t$, $g(t) = t\sin t^3$.
- **Wynik** (Definicja 5.2): $G_a(t) = f_a(\Gamma_3(t))$ i $A_a(t) = G_a(t) - 1$.

Po rozpisaniu: dla $s = t\sin t^3$ oraz $u_1 = t\sin t$, $u_2 = s\sin s$, $u_3 = s\sin s^3$

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

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

Stwierdzenie P.1 (Twierdzenie 5.3 dla niewiadomych całkowitych)

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

1. Jeśli $a = m_1^3+m_2^3+m_3^3$ dla pewnych $m_i \in \mathbb{Z}$, to dla każdego $\varepsilon > 0$ i $M \geq 0$ istnieje $t > M$ z $A_a(t) < -1+\varepsilon$; w szczególności $A_a(t) < 0$ dla pewnego t.
2. Jeśli takich m_i nie ma, to $A_a(t) > 0$ dla **każdego** $t \in \mathbb{R}$.

Dowód. Dowód Twierdzenia 3.4 przenosi się słowo w słowo: w kroku 2 zaokrąglamy x_i do najbliższej liczby całkowitej (dowolnego znaku), a w kroku 4 liczba $P_a(\mathbf{m})$ jest całkowita, bo P_a ma współczynniki całkowite. Dostajemy $f_a(\mathbf{x}) > 1$ dla wszystkich $\mathbf{x} \in \mathbb{R}^3$, gdy rozwiązań nie ma — kwadraty z Wniosku 3.7 nie są więc potrzebne. Gdy $P_a(\mathbf{m}) = 0$, mamy $f_a(\mathbf{m}) = 0$, a gęstość krzywej (Twierdzenie 4.4) i ciągłość f_a dają punkty $t > M$ z $G_a(t) < \varepsilon$ — jak w dowodzie Twierdzenia 5.3. ■

Sprawdźmy, że wagi to rzeczywiście majoranty z Lematu 2.1 (ta sama funkcja `majoranta` co w §2 dowodu):

```
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)
```

Wagi zgadzają się z Lematem 2.1 tożsamościowo (porównanie po rozwinięciu jest symboliczne, a nie numeryczne). Zbudujmy teraz całe wyrażenie $A_a(t)$ w SymPy i sprawdźmy, że rzeczywiście zawiera tylko liczby całkowite, π, t, działania i sinus:

```
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
```

Jedyną funkcją w drzewie jest sinus, a funkcja `count_ops` naliczyła w nim 223 działania — wyrażenie jest duże, ale całkowicie jawne. Do obliczeń używamy tej samej formuły zapisanej bezpośrednio — raz dla zwykłych liczb (`ctx=mp`), raz dla przedziałów (`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
```

Obie wersje zgadzają się w trzech punktach kontrolnych z błędem względnym rzędu 10^{-50}, czyli na granicy ustawionej precyzji (50 cyfr).

2. Cztery równania

a	rozwiązanie w liczbach całkowitych	od kiedy wiadomo
3	$1^3+1^3+1^3$ (także $4^3+4^3+(-5)^3$)	od zawsze
4	**nie ma** (reszty modulo 9)	elementarne
42	$(-80538738812075974)^3 + 80435758145817515^3 + 12602123297335631^3$	A. Booker i A. Sutherland, 2019
114	**nie wiadomo**	najmniejszy otwarty przypadek

Rozwiązanie dla 42 znaleziono po 1,3 mln godzin obliczeń w rozproszonej sieci Charity Engine; wszystkie liczby poniżej 114 (poza tymi, które wyklucza reszta modulo 9) mają już znane przedstawienia, a 114 jest najmniejszym otwartym przypadkiem (stan z września 2026 r.; zob. literaturę na końcu). Sprawdźmy, co się da sprawdzić:

```
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)})
```

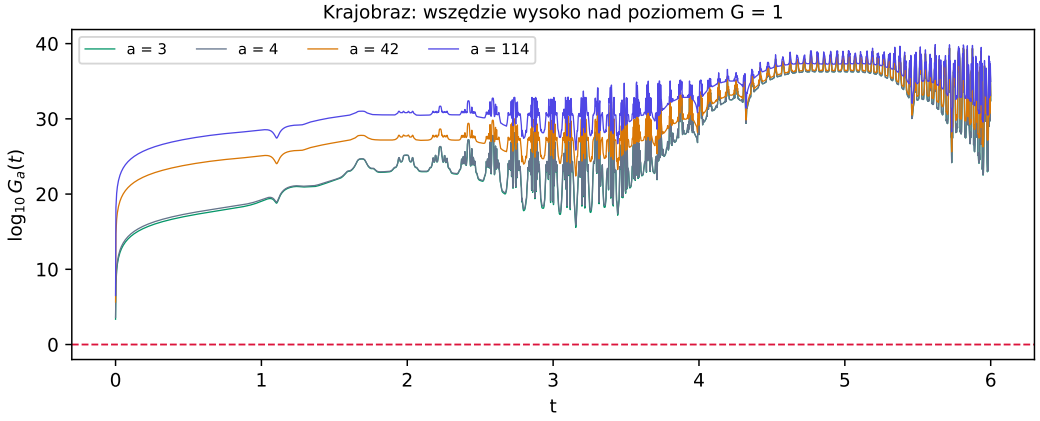
```
a = 3: suma sześcianów (1, 1, 1) = 3
a = 42: suma sześcianów (-80538738812075974, 80435758145817515, 12602123297335631) =
 ↪  42
reszty modulo 9 sum trzech sześcianów: [0, 1, 2, 3, 6, 7, 8]
a mod 9: {3: 3, 4: 4, 42: 6, 114: 6}
```

Sześciany dają modulo 9 tylko reszty $0, \pm 1$, więc suma trzech sześcianów nie może dawać reszty 4 ani 5. Dla $a = 4$ rozwiązań nie ma; dla 42 i 114 reszta modulo 9 niczego nie wyklucza.

3. Co widzi numeryka

Narysujmy $\log_{10} G_a(t)$ na przedziale $[0, 6]$; przerywana linia to poziom $G_a = 1$, czyli $A_a = 0$:

```
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()
```



Na całym przedziale $[0,6]$ wszystkie cztery wykresy leżą wysoko nad poziomem $G_a = 1$; najniżej, w $t = 0$, jest $G_3(0) = 2304$, czyli $\log_{10} G_3 \approx 3{,}4$. Sprawdźmy jeszcze gęstą siatkę na $[0,20]$ i 3000 losowych punktów z $[0,10^6]$, licząc z precyzją 90 cyfr, żeby $\sin t^3$ i $\sin s^3$ były policzone poprawnie:

```
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
```

Najmniejsza wartość na siatce to $G_a(0) = 256a^2$: krzywa przechodzi przez punkt kratowy $(0,0,0)$, ale tam $P_a = -a \neq 0$. W punktach losowych najmniejsze wartości są dla wszystkich a takie same aż do wielu cyfr — dla dużych t stała a ginie wobec u_i^3.

Numeryka mówi to samo dla wszystkich czterech równań: A_a jest dodatnie, i to ogromnie. Gdybyśmy na tej podstawie zgadywali, uznalibyśmy, że żadne z czterech równań nie ma rozwiązań — i pomylilibyśmy się co najmniej dwa razy (dla $a = 3$ i $a = 42$).

4. Pułapka zmiennoprzecinkowa

Ta sama formuła w zwykłej arytmetyce `float64`:

```
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
```

Dla małych t wyniki się zgadzają. Dla $t = 10^5$ liczba $t^3 = 10^{15}$ jest w `float64` jeszcze dokładna ($10^{15} < 2^{53} \approx 9 \cdot 10^{15}$) i $\sin t^3$ wychodzi poprawnie. Dokładne nie jest już jednak $s = t \sin t^3$: błąd rzędu 10^{-12} daje w $s^3 \approx 6 \cdot 10^{14}$ błąd około 0,05, a $\sin s^3$ zależy od s^3 modulo 2π. Dalej `float64` liczy sinusy „innych” liczb niż trzeba.

5. Certyfikat dla $a = 3$

Ponieważ $1^3 + 1^3 + 1^3 = 3$, Stwierdzenie P.1 gwarantuje ujemne wartości A_3. Gdzie? Dowód jest konstruktywny: Twierdzenie 4.4 dla $n = 3$ to dwukrotne użycie Lematu 4.2 — najpierw s z $\Gamma_2(s) \approx (1, 1)$, potem w z $h(w) \approx 1$ i $g(w) = s$; wtedy $\Gamma_3(w) = (h(w), \Gamma_2(g(w))) \approx (1, 1, 1)$.

Jak blisko trzeba trafić? W punkcie $(1, 1, 1)$ wagi są równe $k_i = 31464$, więc $G_3 \approx 256 \sum_i \pi^2 \varepsilon_i^2 k_i^4$ dla odchyleń ε_i — trzeba $|\varepsilon_i| \lesssim 10^{-11}$. Bierzemy $\delta = 10^{-12}$; dokładnie sprawdzimy wynik na końcu.

```
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
```

Odchylenia dwóch pierwszych współrzędnych od 1 są rzędu 10^{-13}, trzecia jest praktycznie dokładna, a $A_3(t_0) \approx -0{,}99992$. Liczba $A_3(t_0)$ policzona w arytmetyce 260-cyfrowej to jeszcze nie dowód. Dowodem jest **arytmetyka przedziałowa**: każde działanie zaokrągla na zewnątrz, więc wynik to przedział, który na pewno zawiera prawdziwą wartość. Zapisujemy t_0 jako napis o 110 cyfrach znaczących — to jest nasz certyfikat:

```
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
```

W obu precyzjach przedział jest wąski i leży poniżej 1 ($G_3(t_0) \approx 7{,}87 \cdot 10^{-5}$), więc $A_3(t_0) < 0$ jest udowodnione. Sto cyfr to nie przesada — oto, co się dzieje, gdy t_0 zaokrąglimy mocniej:

```
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
```

Przy 60 i 80 cyfrach zaokrąglony punkt leży już poza przedziałem, na którym $A_3 < 0$: G_3 jest tam rzędu 10^{497} i 10^{339} (przedziały wyniku są wąskie, więc w tych punktach $A_3 > 0$); od 100 cyfr górny kraniec spada poniżej 1, czyli $A_3 < 0$ jest udowodnione. Jak szeroki jest przedział wokół t_0, na którym $A_3 < 0$? I co zobaczy `float64`?

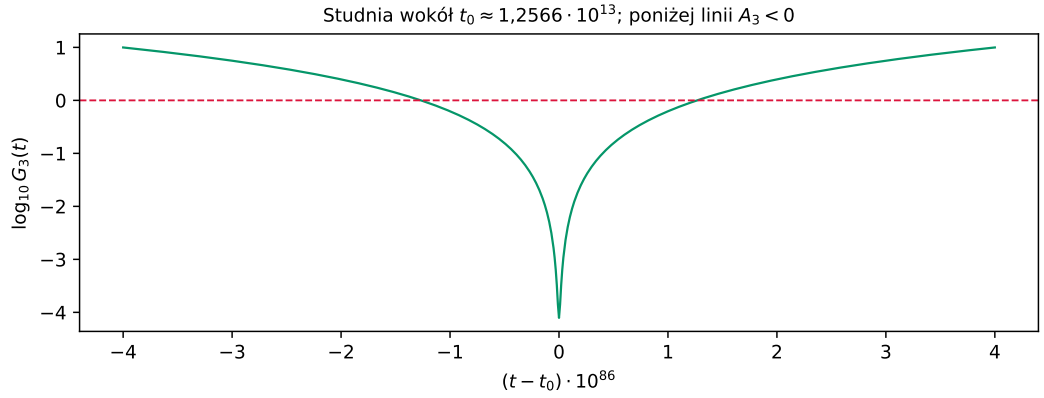
```
def brzeg(kierunek):
    """Odległość od t₀ do miejsca, w którym A_3 wraca do zera."""
    krok = mp.mpf('1e-100')
    while A(t0 + kierunek * krok, 3) < 0:          # oddalamy się, aż A_3 ≥ 0
        krok *= 2
    lo, hi = mp.mpf(0), krok
    for _ in range(200):                           # potem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

Przedział ma długość rzędu 10^{-86}; siatka o takim kroku miałaby na $[0, t_0]$ około $5 \cdot 10^{98}$ punktów. `float64` nie trafia w ten przedział i zwraca nieskończoność: prawdziwa wartość G_3 w punkcie `float64` wynosi około $8{,}9 \cdot 10^{532}$, czyli przekracza zakres tego formatu. Narysujmy $\log_{10} G_3$ w otoczeniu t_0 w skali 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()
```



Ujemne wartości istnieją, ale na przedziale długości rzędu 10^{-86} w okolicy 10^{13} — żadna siatka ani losowanie ich nie znajdzie, a `float64` nie potrafi nawet zapisać t_0 z potrzebną dokładnością. Nie twierdzimy, że t_0 jest najmniejszym takim miejscem; to miejsce wskazane przez konstrukcję z dowodu.

6. Certyfikat dla $a = 42$

Ta sama konstrukcja, tylko wokół rozwiązania z 2019 roku. Wagi w tym punkcie mają około 171 cyfr, więc współrzędne $\Gamma_3(t)$ muszą trafić z dokładnością około 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: 51 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…
```

Certyfikat dla $a = 42$ to liczba rzędu 10^{346}, którą trzeba podać z dokładnością do około 2770 cyfr. Komputer liczy ją w minutę — ale tylko dlatego, że znamy rozwiązanie równania. Przed wrześniem 2019 roku nikt nie potrafił wskazać takiego t ani nie wiedział, czy ono istnieje.

7. $a = 4$: dowód zamiast numeryki

Modulo 9 suma trzech sześcianów nie daje reszty 4 (komórka w §2), więc równanie dla $a = 4$ nie ma rozwiązań i ze Stwierdzenia P.1: $A_4(t) > 0$ **dla każdego** $t \in \mathbb{R}$**.** Żadne próbkowanie tego nie udowodni — przecież A_3 wyglądało tak samo. Rozstrzyga trzylinijkowy argument z teorii liczb, którego numeryka nie zna.

8. $a = 114$: znamy wzór, nie znamy odpowiedzi

$114 \equiv 6 \pmod 9$, więc reszta modulo 9 niczego nie wyklucza, a rozwiązania nikt nie znalazł. Zatem dziś **nikt nie wie, czy** $A_{114}(t) \geq 0$ **dla wszystkich** t. Hipoteza Heath-Browna (1992) przewiduje, że każda liczba całkowita, która nie przystaje do 4 ani do 5 modulo 9, jest sumą trzech

sześcianów (i to na nieskończenie wiele sposobów). Jeśli ma rację, A_{114} jest gdzieś ujemne — w miejscu, którego nie potrafimy dziś nawet oszacować.

Czy znamy funkcję A_{114}?

Tak, w całości. To wzór z §1 dla $a = 114$:

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

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

gdzie $s = t \sin t^3$, $u_1 = t \sin t$, $u_2 = s \sin s$, $u_3 = s \sin s^3$. Nie występuje w nim żadne rozwiązanie równania — maszyna z dowodu korzysta tylko z wielomianu P_{114}. Wartość $A_{114}(t)$ w każdym punkcie policzymy z gwarantowaną dokładnością (arytmetyka przedziałowa, jak w §5). Nie znamy tylko odpowiedzi na pytanie, czy $A_{114}(t) < 0$ dla pewnego t.

Rozwiązanie równania nie jest składnikiem wyrażenia, lecz **certyfikatem** jego ujemności. Jeśli $A_{114}(t) < 0$, to $f_{114}(\Gamma_3(t)) < 1$, a wtedy — jak w dowodzie Stwierdzenia P.1 — zaokrąglenie współrzędnych $\Gamma_3(t)$ do najbliższych liczb całkowitych daje rozwiązanie równania $x_1^3 + x_2^3 + x_3^3 = 114$: punkt krzywej sam jest świadkiem (Uwaga 5.4 dowodu). Szukanie t z $A_{114}(t) < 0$ jest więc szukaniem rozwiązania. W §5–§6 szliśmy w drugą stronę: od znanego rozwiązania do t_0.

Czy to dowodzi twierdzenia Richardsona?

Nie — i żaden pojedynczy przykład tego nie zrobi. Nierozstrzygalność jest własnością **rodziny** pytań: nie istnieje algorytm, który odpowiada poprawnie dla każdego wyrażenia. Pojedyncze wyrażenie ma ustaloną odpowiedź, więc algorytm, który ją wypisuje, istnieje: „wypisz tak” albo „wypisz nie” — choć możemy nie wiedzieć który. Rodzina A_a też może nie wystarczyć: gdyby hipoteza Heath-Browna była prawdziwa, odpowiedź zależałaby tylko od a mod 9, byłaby więc rozstrzygalna. Do nierozstrzygalności potrzebny jest wielomian z twierdzenia DPRM (§1 i §11 dowodu).

Pojedynczy przypadek może pokazać trudność praktyczną ($a = 3$, $a = 42$), granicę dzisiejszej wiedzy ($a = 114$) albo niedowodliwość. To ostatnie daje ta sama maszyna (§3–§6 i §11 dowodu) zastosowana do innego, olbrzymiego wielomianu Q; otrzymane wyrażenie oznaczmy A_Q.

- Z twierdzenia DPRM istnieje Q, dla którego równanie $Q = 0$ ma rozwiązanie w liczbach naturalnych wtedy i tylko wtedy, gdy teoria mnogości ZFC jest sprzeczna. Jeśli ZFC jest niesprzeczna, to zdanie „$A_Q(t) \geq 0$ dla wszystkich t” jest prawdziwe, ale nie da się go udowodnić w ZFC: łańcuch równoważności z dowodu da się przeprowadzić w ZFC, więc taki dowód dawałby dowód niesprzeczności ZFC w samej ZFC, wbrew drugiemu twierdzeniu Gödla.
- Davis, Matiyasevich i Robinson (1976) podali Q, dla którego równanie $Q = 0$ nie ma rozwiązań wtedy i tylko wtedy, gdy zachodzi hipoteza Riemanna. Wtedy „$A_Q \geq 0$ wszędzie” jest równoważne hipotezie Riemanna.

Rodzina A_a pokazuje co innego: nawet gdy ogólna odpowiedź może być prosta, **pojedyncze przypadki** bywają skrajnie trudne — a numeryka nie odróżnia łatwych od trudnych.

9. To samo pytanie jako problem zera

Z modułem (§7 dowodu): $B_a = |A_a| - A_a$ jest tożsamościowo równe zeru wtedy i tylko wtedy, gdy $A_a \geq 0$ wszędzie. Zatem $B_4 \equiv 0$, $B_3 \not\equiv 0$, $B_{42} \not\equiv 0$, a o B_{114} nikt nie wie. Co powie system algebry komputerowej?

```
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 uczciwie odpowiada „nie wiem” (`None`) albo nie odpowiada w ciągu 5 sekund — dla obu wartości a jednakowo, choć odpowiedzi są przeciwne.

10. Morał

- **Numeryka widzi tylko punkty, w których liczy.** Znak A_a zależy od arytmetyki liczb całkowitych, której żadne skończone próbkowanie nie widzi: ujemne wartości mieszkają na przedziałach długości 10^{-86} (dla $a = 3$) w okolicy 10^{13} — albo dalej.
- **Ujemność da się zaświadczyć skończonym obliczeniem, nieujemności — w ogólności nie.** Certyfikat (t_0 + arytmetyka przedziałowa) dowodzi, że $A_3 < 0$ i $A_{42} < 0$ gdzieś; to dokładnie półrozstrzygalność ze Stwierdzenia 10.1. Dodatniość A_4 wymaga dowodu (modulo 9), a dla A_{114} dowodu nie ma.
- **Nawet „łatwe tak” bywa praktycznie trudne:** banalne $1^3 + 1^3 + 1^3 = 3$ daje świadka z około 100 cyframi, rozwiązanie dla 42 — z około 2770. Większe rozwiązania — jeszcze gorzej.
- Twierdzenie Richardsona mówi, że żaden algorytm nie poradzi sobie ze wszystkimi takimi pytaniami. Ten przykład pokazuje, że trudność nie jest tylko teoretyczna.

11. Pytania i zadania

1. Dlaczego oba certyfikaty zaczynają się od cyfr 12566…? *(Wskazówka: od czego zaczyna się konstrukcja w Lemacie 4.2?)*
2. Która współrzędna $\Gamma_3(t)$ zmienia się najszybciej w pobliżu t_0? Oszacuj jej pochodną i wyjaśnij, skąd długość przedziału rzędu 10^{-86}.
3. Dlaczego t_0 trzeba znać z dokładnością do około 100 cyfr, choć $t_0 \approx 10^{13}$? Porównaj z wynikiem z §5.
4. $30 = 2220422932^3 + (-2218888517)^3 + (-283059965)^3$. Oszacuj (bez liczenia certyfikatu) rząd wielkości t_0 i liczbę potrzebnych cyfr. *(Wskazówka: wagi rosną jak $|\mathbf{m}|^{10}$.)*
5. 💻 Policz certyfikat dla $a = 29 = 3^3 + 1^3 + 1^3$ i porównaj z $a = 3$.
6. 💻 Zbuduj analogiczne wyrażenia dla równań Pella $x^2 - 61y^2 = -1$ (najmniejsze rozwiązanie dodatnie: $x = 29718$, $y = 3805$) i $x^2 - 61y^2 = 2$ (brak rozwiązań: 2 nie jest resztą kwadratową modulo 61). Porównaj numerykę i oszacuj t_0 dla pierwszego.
7. Co by się stało, gdyby wagi k_i zastąpić jedynkami? *(Por. Przykład 3.1.)*

Literatura. 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), nr 11, doi:10.1073/pnas.2022377118, arXiv:2007.01209 (rozwiązania dla 42 i nowe dla 3) · Sums of three cubes (Wikipedia; stan z 12.09.2026: otwarte przypadki poniżej 1000 to 114, 390, 627, 633, 732, 921, 975) · Epoch AI: Sum of Three Cubes.

Dodatki: potęgi dwójki i całka z e^{x^2}

Dwa uzupełnienia do pełnego dowodu. Dodatek A uzasadnia, dlaczego w wejściu arytmetycznym (Twierdzenie 1.1 w rozdziale „Pełny dowód krok po kroku") wystarczą potęgi o podstawie 2 — stąd w języku Richardsona stała $\log 2$. Dodatek B dowodzi, że e^{x^2} jest **lokalną przeszkodą** dla całkowania w języku z modułem (Definicja 8.1 w rozdziale „Pełny dowód krok po kroku"), co kończy dowód części (C).

A. Wystarczą potęgi dwójki

Twierdzenie Davisa–Putnama–Robinsona (Twierdzenie 2.3 w rozdziale „Pojęcia wstępne") dopuszcza potęgowanie a^b o dowolnej podstawie. Pokażemy, że relację $c = a^b$ można wyrazić równaniami, w których potęgowanie występuje tylko jako $2^{(\cdot)}$.

Lemat A.1

Dla dodatnich liczb całkowitych a, b, c niech $k = ab + a + 1$ i $M = 2^k - a$. Wtedy

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

Dowód. Najpierw oszacowania. Ponieważ $a \le 2^a$, mamy $a^b \le 2^{ab}$. Ponadto

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

więc $0 < a^b \le 2^{ab} < M$. Ponieważ $2^k \equiv a \pmod M$, dostajemy $2^{kb} = (2^k)^b \equiv a^b \pmod M$.

„⇐": drugie równanie mówi, że $0 < c < M$ (bo $s \ge 0$ i $c \ge 1$), a pierwsze, że $c \equiv 2^{kb} \equiv a^b \pmod M$. Dwie liczby z przedziału $(0, M)$ przystające modulo M są równe, więc $c = a^b$. „⇒": jeśli $c = a^b$, to $2^{kb} - c$ jest wielokrotnością M (nieujemną, bo $2^{kb} \ge 2^{ab} \ge a^b$), co daje $q \in \mathbb{N}$, a $s = M - c - 1 \ge 0$. ■

Aby wykładniki były **zmiennymi**, wprowadzamy nowe niewiadome k i ℓ z równaniami $k - ab - a - 1 = 0$ oraz $\ell - kb = 0$; wtedy występują tylko 2^k i 2^ℓ. Dowolne wyrażenie wykładnicze rozkładamy, nazywając wyniki pośrednie nowymi zmiennymi, na skończony układ równań tego typu, a układ $E_1 = \cdots = E_r = 0$ zastępujemy jednym równaniem $\sum_j E_j^2 = 0$. Zmienne dodatnie zapisujemy jako $x + 1$ z $x \in \mathbb{N}$, korzystając z $2^{x+1} = 2 \cdot 2^x$. Przypadki $b = 0$ (wtedy $c = 1$) i $a = 0 < b$ (wtedy $c = 0$) dołączamy alternatywą, bo $E = 0 \vee E' = 0 \iff EE' = 0$. Otrzymujemy wielomian o współczynnikach całkowitych w zmiennych x_i i potęgach 2^{x_i} — dokładnie postać z Twierdzenia 1.1 w rozdziale „Pełny dowód krok po kroku".

B. e^{x^2} nie ma elementarnej funkcji pierwotnej — nawet lokalnie

Potrzebne pojęcia (ciało różniczkowe, rozszerzenie elementarne, twierdzenie Liouville'a) są w rozdziale „Pojęcia wstępne", §5. Pracujemy z **zarodkami** funkcji meromorficznych w otoczeniu punktu rzeczywistego (dopuszczamy wartości zespolone): zarodek to funkcja określona na pewnym otoczeniu, z utożsamieniem funkcji równych po zmniejszeniu otoczenia. Zarodek o zerowej pochodnej jest stałą zespoloną, więc ciała takich zarodków zawierające $\mathbb{C}$ mają ciało stałych dokładnie $\mathbb{C}$ — założenie twierdzenia Liouville'a jest spełnione.

B.1. Transcendencja e^{x^2}

Lemat B.1

Funkcja $t = e^{x^2}$ jest przestępna nad $\mathbb{C}(x)$.

Dowód. Gdyby nie, po usunięciu mianowników mielibyśmy $\sum_{j=0}^{m} p_j(x)e^{jx^2} = 0$ z $p_j \in \mathbb{C}[x]$, $p_m \neq 0$, $m \geq 1$, na pewnym otoczeniu, a z twierdzenia o identyczności — na całej prostej. Dzieląc przez $p_m(x)e^{mx^2}$ dla dużych $x > 0$ dostajemy $1 + \sum_{j<m} \frac{p_j(x)}{p_m(x)} e^{-(m-j)x^2} = 0$. Każdy składnik sumy dąży do zera (funkcja wymierna rośnie co najwyżej jak potęga x, a $e^{-(m-j)x^2}$ maleje szybciej niż każda odwrotność potęgi), co daje sprzeczność $1 = 0$. ■

Zatem $K = \mathbb{C}(x, t)$ to ciało funkcji wymiernych zmiennej t o współczynnikach z $\mathbb{C}(x)$, z derywacją

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

B.2. Równanie wymierne wymuszone przez Liouville'a

Stwierdzenie B.2

Jeśli e^{x^2} ma elementarną funkcję pierwotną na jakimś niepustym przedziale otwartym, to istnieje funkcja wymierna $r \in \mathbb{C}(x)$ spełniająca

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

Odwrotnie, taka r dawałaby funkcję pierwotną $r(x)e^{x^2}$.

Dowód. Wieżę elementarną zawierającą funkcję pierwotną możemy zbudować nad K (dołączamy jej generatory po t; typy rozszerzeń się zachowują), z tym samym ciałem stałych $\mathbb{C}$. Twierdzenie Liouville'a (Twierdzenie 5.3 w rozdziale „Pojęcia wstępne") dla $a = t$ daje

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

Porównujemy formalne rozwinięcia Laurenta w punkcie $t = \infty$, tj. szeregi $\sum_{k \leq m} a_k(x)t^k$ z $a_k \in \mathbb{C}(x)$ (każda funkcja wymierna zmiennej t ma jednoznaczne takie rozwinięcie). Derywacja działa wyraz po wyrazie:

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

więc różniczkowanie szeregu o samych ujemnych potęgach nie tworzy potęg nieujemnych.

Pochodne logarytmiczne nie dają dodatnich potęg t. Każde niezerowe $u \in \mathbb{C}(x,t)$ zapisujemy jako $u = a(x)t^m(1+w)$, $a \neq 0$, $m \in \mathbb{Z}$, w — szereg o ujemnych potęgach. Wtedy

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

a ostatni składnik zawiera tylko ujemne potęgi t.

Współczynnik przy t^1. Niech $r(x)$ będzie współczynnikiem przy t^1 w rozwinięciu v (zero, jeśli go brak). Współczynnik przy t^1 w Dv to $r' + 2xr$; pochodne logarytmiczne wnoszą 0; po lewej stronie (B.2) stoi 1. Stąd (B.1). Odwrotnie, $\frac{d}{dx}(re^{x^2}) = (r' + 2xr)e^{x^2}$. ■

Lemat B.3

Równanie (B.1) nie ma rozwiązania $r \in \mathbb{C}(x)$.

Dowód. Przypuśćmy, że r ma biegun rzędu $m \geq 1$ w $\alpha \in \mathbb{C}$, z wyrazem wiodącym $c(x-\alpha)^{-m}$, $c \neq 0$. Wtedy r' ma wyraz wiodący $-mc(x-\alpha)^{-m-1}$, a $2xr$ ma biegun rzędu co najwyżej m, więc nie może go znieść: $r' + 2xr$ miałoby biegun, a prawa strona (B.1) to 1. Zatem r jest wielomianem. Wielomian zerowy nie spełnia (B.1); jeśli r ma stopień $d \geq 0$ i współczynnik wiodący $c \neq 0$, to $r' + 2xr$ ma stopień $d+1 \geq 1$ (wyraz $2cx^{d+1}$), więc nie jest stałą 1. ■

B.3. Moduł nie pomaga

Lemat B.4 (lokalne usuwanie modułów)

Niech f będzie dana wyrażeniem z $\mathcal{B}_1$. Dla każdego niepustego przedziału otwartego I istnieje niepusty przedział otwarty $J \subseteq I$, na którym f jest równa pewnemu wyrażeniu z $\mathcal{A}_1$ (bez modułu); w szczególności f jest na J elementarna i analityczna.

Dowód. Indukcja po drzewie wyrażenia, dla dowolnego przedziału wejściowego. Stałe i x nie wymagają zmian. Dla działania dwuargumentowego stosujemy założenie indukcyjne do jednego argumentu na I, potem do drugiego na otrzymanym podprzedziale; na mniejszym przedziale oba zastąpienia działają jednocześnie. Dla exp i sin — do argumentu. Dla $|B|$: na pewnym $J_0 \subseteq I$ mamy $B = B_0$ z $B_0 \in \mathcal{A}_1$, analitycznym. Jeśli $B_0 \equiv 0$ na J_0, zastępujemy $|B|$ przez 0. W przeciwnym razie wybieramy punkt z $B_0 \neq 0$ i (z ciągłości) otoczenie $J \subseteq J_0$ o stałym znaku; zastępujemy $|B|$ przez B_0 lub $-B_0$. ■

Ten dowód jest **egzystencjalny**: nie daje algorytmu ustalającego, które podwyrażenia znikają tożsamościowo — i nie musi, bo lemat służy tylko do wykazania, że e^{x^2} jest przeszkodą; algorytm redukcji z niego nie korzysta.

Wniosek B.5 (e^{x^2} jest lokalną przeszkodą)

Żadna funkcja dana wyrażeniem z $\mathcal{B}_1$ nie ma pochodnej równej e^{x^2} na żadnym niepustym przedziale otwartym. Zatem $\Delta(x) = e^{x^2}$ spełnia założenie Twierdzenia 8.3 w rozdziale „Pełny dowód krok po kroku”.

Dowód. Jeśli $H' = e^{x^2}$ na I, to z Lematu B.4 na pewnym $J \subseteq I$ mamy $H = H_0$ z $H_0 \in \mathcal{A}_1$, więc $H_0' = e^{x^2}$ na J, a zarodek H_0 jest elementarny ($\sin z = (e^{iz} - e^{-iz})/(2i)$). To przeczy Stwierdzeniu B.2 i Lematowi B.3. ■

Ta sama metoda w języku z pierwiastkiem

W przykładzie Richardsona język zawiera rzeczywisty pierwiastek kwadratowy (wtedy $|x| = \sqrt{x^2}$). Lemat B.4 przenosi się: jeśli wyrażenie jest określone na przedziale, to wyrażenie podpierwiastkowe jest tam nieujemne i albo znika tożsamościowo, albo na mniejszym przedziale jest dodatnie i pierwiastek jest gałęzią analityczną, algebraiczną nad poprzednim ciałem.

Funkcja pierwotna w zwykłym sensie oczywiście istnieje: $\int_0^x e^{s^2}\,ds$ jest gładka na całej prostej (to $\frac{\sqrt{\pi}}{2}\operatorname{erfi} x$). Przeszkoda dotyczy **postaci** funkcji pierwotnej, a nie jej istnienia. SymPy zachowuje się zgodnie z tą teorią:

```
import sympy as sp
x = sp.symbols('x')
sp.integrate(sp.exp(x**2), x)        # sqrt(pi)*erfi(x)/2 — funkcja specjalna, nie
↪ elementarna
```

Kontekst, konsekwencje i dalsze wyniki

1. Co twierdzenie mówi o systemach algebry komputerowej

Wniosek 7.2 z rozdziału „Pełny dowód krok po kroku” mówi, że dla wyrażeń z sin, exp, π, $\log 2$ i modułem **nie istnieje** algorytm postaci kanonicznej. Każda funkcja w rodzaju `simplify` jest więc skazana na jedno z dwojga: albo czasem nie rozpozna zera, albo czasem się pomyli. Projektanci CAS wybierają pierwsze — i stosują kilka strategii, które warto znać:

1. **Podklasy z postacią kanoniczną.** Wielomiany i funkcje wymierne mają postać kanoniczną (wykład 1, §1.3). Dla wyrażeń wykładniczo-logarytmicznych *twierdzenia o strukturze* (Risch) sprowadzają rozpoznawanie zera funkcji do rozpoznawania zera **stałych**.
2. **Testy losowe.** Dla tożsamości wielomianowych i algebraicznych test w losowych punktach modulo duża liczba pierwsza daje gwarancję probabilistyczną (lemat Schwartza–Zippela, wykład 1, §1.4).
3. **Numeryka z certyfikatem.** Arytmetyka przedziałowa potrafi **udowodnić**, że wyrażenie *nie jest* zerem (przedział nie zawiera zera) — tak działa Stwierdzenie 10.1 w rozdziale „Pełny dowód krok po kroku”. Nigdy jednak nie udowodni, że *jest* zerem: przedział zawierający zero nie wyklucza bardzo małej wartości niezerowej, a obliczenie w skończenie wielu punktach nic nie mówi o pozostałych.
4. **Heurystyki i odpowiedź „nie wiem”.** SymPy w metodzie `equals` zwraca `None`, gdy nie potrafi rozstrzygnąć; `simplify` może zwrócić wyrażenie nieuproszczone.

> **Lekcja**
>
> „Różne od zera” można **wykazać obliczeniem**. „Równe zeru” wymaga **dowodu**. Twierdzenie Richardsona mówi, że w dostatecznie bogatym języku nie ma algorytmu, który zawsze taki dowód znajdzie.

2. Dlaczego sinus? Granica rozstrzygalności

Sercem dowodu jest równoważność $\sin(\pi x) = 0 \iff x \in \mathbb{Z}$: sinus pozwala **zdefiniować liczby całkowite wewnątrz liczb rzeczywistych**, a wtedy do środka wchodzi arytmetyka z jej nierozstrzygalnością (dziesiąty problem Hilberta). Bez okresowości sytuacja jest zupełnie inna:

Język (jedna lub wiele zmiennych)	Czy „$\exists x\ A(x) < 0$” jest rozstrzygalne?
wielomiany o współczynnikach wymiernych	**tak** — Tarski (1948/1951): teoria ciał rzeczywiście domkniętych ma eliminację kwantyfikatorów; praktyczny algorytm: rozkład cylindryczny (Collins, 1975)
wielomiany i exp	**tak, jeśli prawdziwa jest hipoteza Schanuela** — Macintyre i Wilkie (1996)
wielomiany, sin, π	**nie** — konstrukcja Richardsona (1968) z wielomianem Matiyasevicha (1970); Caviness (1970)
liczby całkowite, x, $\sin x^n$, $\sin(x \sin x^n)$	**nie** — Laczkovich (2003), bez π

Wilkie (1996) udowodnił, że struktura $(\mathbb{R}, +, \cdot, <, \exp)$ jest *o-minimalna*: każdy definiowalny w niej podzbiór prostej jest skończoną sumą przedziałów i punktów. W takiej strukturze nie da się zdefiniować $\mathbb{Z}$ — i właśnie dlatego jest nadzieja na rozstrzygalność. Podobnie „oswojony" jest sinus obcięty do odcinka (van den Dries, 1986). Kłopot sprawia dopiero sinus na **całej** prostej, z nieskończenie wieloma zerami.

3. Problem stałych i hipoteza Schanuela

Richardson zanotował w 1968 r., że nie wiadomo, czy da się rozstrzygać znak wyrażeń **bez zmiennych**. Pytanie, czy dana stała elementarna (np. zbudowana z liczb wymiernych, π, e, exp, log) jest równa zeru, pozostaje otwarte.

> **Hipoteza Schanuela**
>
> Jeśli liczby zespolone $z_1, \dots, z_n$ są liniowo niezależne nad $\mathbb{Q}$, to stopień przestępności ciała $\mathbb{Q}(z_1, \dots, z_n, e^{z_1}, \dots, e^{z_n})$ nad $\mathbb{Q}$ wynosi co najmniej n.

Hipoteza implikuje m.in. algebraiczną niezależność e i π — dziś nie wiemy nawet, czy $e + \pi$ jest liczbą niewymierną. Richardson (1997) podał algorytm rozpoznawania zera stałych wykładniczo-logarytmicznych, który **zawsze odpowiada poprawnie, gdy się zatrzyma**, i **zatrzymuje się zawsze, jeśli hipoteza Schanuela jest prawdziwa**. To dobry przykład, jak otwarte problemy teorii liczb przekładają się na pytania o algorytmy w CAS — np. algorytm Rischa całkowania w skończonej postaci potrzebuje właśnie testu zera dla stałych.

4. Dalsze wyniki

- **Caviness (1970)** zauważył, że po rozwiązaniu dziesiątego problemu Hilberta z twierdzenia można usunąć e^x i $\log 2$ (zob. §11 w rozdziale „Pełny dowód krok po kroku").
- **Wang (1974)** wykazał nierozstrzygalność **istnienia miejsca zerowego** funkcji elementarnych. (Wersję tego wyniku można wyprowadzić z naszej konstrukcji i twierdzenia Darboux — to problem 9 w rozdziale „Pytania kontrolne i problemy".)
- **Laczkovich (2003)** usunął π i ograniczył użycie złożenia: w pierścieniu generowanym przez liczby całkowite i funkcje x, $\sin x^n$, $\sin(x \sin x^n)$ $(n = 1, 2, \dots)$ nierozstrzygalne jest, czy funkcja przyjmuje wartość dodatnią i czy ma pierwiastek.
- **Richardson (1997)** — algorytm dla stałych, warunkowy względem hipotezy Schanuela (§3).

5. Związek z dowodami formalnymi

Asystenci dowodu mają procedury decyzyjne dla **rozstrzygalnych** fragmentów matematyki: `ring` (tożsamości w pierścieniach przemiennych — postać kanoniczna), `linarith` i `omega` (arytmetyka liniowa), `norm_num` (rachunki na liczbach), `decide` (skończone sprawdzenia). Twierdzenie Richardsona wyjaśnia, dlaczego nie ma — i nie może być — kompletnej taktyki „udowodnij każdą tożsamość z sin, exp i modułem": takie dowody trzeba znaleźć (człowiek, heurystyka, model AI), a jądro jedynie je **sprawdza**. W Peano Lab jest podobnie: arytmetyka Peana jest nierozstrzygalna (Church, 1936), więc taktyka `auto` z konieczności jest niepełna.

6. Kalendarium

Rok	Wydarzenie
1900	Hilbert stawia dziesiąty problem: algorytm dla równań diofantycznych
1936	Turing i Church: nierozstrzygalność problemu stopu i arytmetyki
1948/51	Tarski: rozstrzygalność teorii ciał rzeczywiście domkniętych
1961	Davis, Putnam, Robinson: zbiory półrozstrzygalne są wykładniczo diofantyczne
1968	Richardson: nierozstrzygalne problemy dla funkcji elementarnych
1970	Matiyasevich: zbiory półrozstrzygalne są diofantyczne; Caviness: usunięcie e^x, $\log 2$
1974	Wang: nierozstrzygalność istnienia zer
1996	Wilkie: o-minimalność $\mathbb{R}_{\exp}$; Macintyre–Wilkie: rozstrzygalność przy hipotezie Schanuela
1997	Richardson: „How to recognise zero”
2003	Laczkovich: usunięcie π

7. Literatura z komentarzem

Źródło pierwotne

- 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. Siedem stron; warto przeczytać po tym rozdziale — wszystkie kroki będą znajome.

Arytmetyka i obliczalność

- 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 (oryginał: Doklady Akademii Nauk SSSR 191 (1970), 279–282).
- M. Davis, *Hilbert's Tenth Problem is Unsolvable*, American Mathematical Monthly 80(3) (1973), 233–269 — **najlepszy punkt startu**: kompletny, samowystarczalny dowód (nagrody Chauvenet i Steele).
- Yu. V. Matiyasevich, *Hilbert's Tenth Problem*, MIT Press 1993 — monografia.
- J. P. Jones, *Universal Diophantine equation*, Journal of Symbolic Logic 47(3) (1982), 549–571.
- M. Sipser, *Introduction to the Theory of Computation*, Cengage — rozdziały o rozstrzygalności i redukcjach.
- B. Poonen, *Undecidable problems: a sampler*, w: J. Kennedy (red.), *Interpreting Gödel*, Cambridge University Press 2014, 211–241; arXiv:1204.0299 — przegląd nierozstrzygalności w różnych działach matematyki, w tym twierdzenia Richardsona.

Rozszerzenia twierdzenia Richardsona

- 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.

Rozstrzygalność i o-minimalność

- A. Tarski, *A Decision Method for Elementary Algebra and Geometry*, RAND 1948; wyd. 2: 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*, w: P. Odifreddi (red.), *Kreiseliana*, A K Peters 1996, 441–467.

Całkowanie w skończonej postaci

- 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*, notatki (dostępne na stronie autora) — przystępny dowód twierdzenia Liouville'a i kryterium dla $f e^{g}$.
- M. Bronstein, *Symbolic Integration I: Transcendental Functions*, wyd. 2, Springer 2005 — algorytm Rischa w praktyce.

Pytania kontrolne i problemy

Pytania kontrolne sprawdzają zrozumienie kolejnych kroków dowodu — odpowiedzi zebrano na końcu rozdziału. Problemy są oznaczone trudnością (⭐ — rutynowe, ⭐⭐ — wymagają pomysłu, ⭐⭐⭐ — wyzwanie) oraz 💻, jeśli wymagają obliczeń. Numeracja lematów i twierdzeń odnosi się do „Pełny dowód krok po kroku".

Pytania kontrolne

1. Co jest daną wejściową w problemie tożsamości: funkcja czy wyrażenie? Dlaczego to rozróżnienie jest istotne?

2. Czy z twierdzenia Richardsona wynika, że komputer nie potrafi sprawdzić tożsamości $\sin^2 x + \cos^2 x = 1$?

3. Dlaczego w języku potrzebna jest funkcja okresowa? Co się dzieje, gdy dopuszczamy tylko wielomiany?

4. Do czego służą wagi k_i^4 w karze f? Co by się stało bez nich?

5. Dlaczego w Twierdzeniu 3.4 zaokrąglamy do najbliższej liczby **naturalnej**, a nie całkowitej? Gdzie to ma znaczenie?

6. W którym dokładnie miejscu dowodu korzystamy z tego, że P przyjmuje wartości całkowite?

7. Dlaczego twierdzenie o zaokrąglaniu dotyczy nierówności $f \leq 1$, a nie tylko równania $f = 0$?

8. Czy krzywa Γ_n przechodzi przez każdy punkt $\mathbb{R}^n$? Czy to potrzebne?

9. Dlaczego w Lemacie 4.2 żądamy $g(w) = b$ **dokładnie**, a nie tylko w przybliżeniu?

10. Czy dla $a \in U$ funkcja G_a musi mieć miejsce zerowe?

11. Gdzie w dowodzie potrzebny jest moduł i dlaczego nie da się go po prostu pominąć?

12. Czy problem „czy $A(x) < 0$ dla pewnego x" jest półrozstrzygalny? A problem „czy $A \equiv 0$"?

13. Czym różni się problem tożsamości od problemu stałych?

14. Co w redukcji musi być obliczalne, a co wystarczy, że istnieje?

15. Dlaczego w Lemacie 2.1 majorantą stałej c jest $c^2 + 2$, a nie $|c| + 2$, jak u Richardsona?

16. Jaką rolę pełni $\log 2$ w oryginalnym twierdzeniu? Czy jest konieczna?

17. Dlaczego w części o całkowaniu pojawia się próg $\frac{1}{2}$, a nie 1?

18. Czy twierdzenie Richardsona przeczy istnieniu algorytmu Rischa (całkowanie w skończonej postaci)?

19. Program numeryczny w `float64` zwrócił $G_a(t) \approx 0{,}5$. Czy wiemy już, że $a \in U$?

20. Dlaczego krzywą Γ_n definiujemy dla $n \geq 2$?

Problemy

1. ★ Udowodnij nierówności z Lematu 2.1: $|c| < c^2 + 2$ dla $c \in \mathbb{R}$ oraz $|z| + 1 < z^2 + 2$. Jakie są najmniejsze wartości różnic?

2. ★ 💻 Zbuduj ręcznie majorantę z Lematu 2.1 dla $A(x, y) = x \sin y - e^x$, a potem sprawdź wynik funkcją `majoranta` z rozdziału „Pełny dowód krok po kroku” i losowym testem nierówności $|A(\mathbf{z} + \boldsymbol{\eta})| < M_A(\mathbf{z})$.

3. ★ Rozszerz Lemat 2.1 na język z modułem: zaproponuj regułę dla $M_{|A|}$ i udowodnij ją. Dlaczego Lemat 2.3 (różniczkowanie) się nie rozszerza i dlaczego dowód wymaga, by wielomian P był analityczny?

4. ★★ Udowodnij w pełni wariant z kwadratami (Uwaga 3.5): $f_{\text{alt}} = (n{+}1)^2[P^2 + \sum_i \sin^2(\pi x_i) k_i^2]$ też daje przerwę.

5. ★★ 💻 Dla $n = 2$ wyznacz liczbę $s_0 > 1$ z Uwagi 3.6 (największą, jaką potrafisz) i udowodnij, że $\varphi(s_0) < 1$. Jaki jest sens s_0 dla przypadku $a \notin U$?

6. ★★ Czy w krzywej Richardsona można zastąpić $g(t) = t \sin t^3$ przez $t \sin t^2$? Przeanalizuj dowód Lematu 4.2: porównaj zmianę fazy z dopuszczalną zmianą h na tym samym przedziale. (Wskazówka: dla fazy t^k stosunek pochodnych fazy i h zachowuje się jak $k\,t^{k-1}/t$.)

7. ★★ Uzupełnij szczegóły Uwagi 4.6: udowodnij, że obraz Γ_2 ma miarę Lebesgue’a zero w $\mathbb{R}^2$, a mimo to jest gęsty.

8. ★★ Pokaż, że problem „czy $A \geq 0$ wszędzie” dla $A \in \mathcal{A}_1$ nie jest półrozstrzygalny. (Wskazówka: Stwierdzenie 10.1 i twierdzenie Posta.)

9. ★★ **Istnienie miejsca zerowego.** Udowodnij, że nie istnieje algorytm, który dla $A \in \mathcal{A}_1$ rozstrzyga, czy A ma miejsce zerowe. Wskazówka: pokaż, że $G_a > 1$ w pewnym punkcie, i to dla każdego a — oszacuj $F(a, \mathbf{u})$ w punktach z $u_1^2 = \frac{1}{2}$ i użyj gęstości — a potem zastosuj twierdzenie Darboux do $G_a - 1$. (Por. Wang, 1974.)

10. ★★ Udowodnij, że relacja „$A \not\equiv B$” dla $A, B \in \mathcal{B}_1$ jest półrozstrzygalna, a relacja „$A \equiv B$” nie jest.

11. ★★ Rozstrzygalność dla wielomianów jednej zmiennej bez twierdzenia Tarskiego: pokaż, że $p \in \mathbb{Q}[x]$ przyjmuje wartość ujemną wtedy i tylko wtedy, gdy ma ujemny współczynnik wiodący, nieparzysty stopień albo pierwiastek rzeczywisty nieparzystej krotności. Opisz algorytm (ciągi Sturma) i zaimplementuj go w SymPy (`sp.sturm`, `sp.sqf_list`). 💻

12. ★★ 💻 Sprawdź Lemat A.1 z rozdziału „Dodatki” numerycznie dla wszystkich $1 \leq a, b \leq 6$: wyznacz k, M, q, s i potwierdź równoważność.

13. ★★ 💻 Odtwórz konstrukcję z Lematu 4.2 dla celu $(a, b) = (0, 100)$ i $\delta = 10^{-3}$. Oszacuj, ile cyfr precyzji potrzeba, aby obliczenie $g(w)$ było wiarygodne, i sprawdź to, zmieniając `mp.mp.dps`.

14. ★★★ 💻 Dla zabawkowego równania $x_1^2 + x_2^2 = 5$ z §5 znajdź punkt t, w którym $G(t) < \frac{1}{2}$ (taki punkt jest potrzebny w części o całkowaniu). Jak dobrać δ? Jak duże jest t i jaka precyzja jest potrzebna?

15. ★★★ Udowodnij Lemat B.4 z rozdziału „Dodatki” dla języka z pierwiastkiem kwadratowym zamiast modułu (szczegóły z uwagi po Wniosku B.5).

16. ★★★ Pokaż, że funkcja $\dfrac{\sin x}{x}$ (uzupełniona wartością 1 w zerze) nie ma elementarnej funkcji pierwotnej. Wskazówka: kryterium dla $f e^g$ z notatek Conrada (rozważ e^{ix}/x) i analog Lematu B.3.

17. ★★★ Przeczytaj pracę Laczkovicha (2003) i wyjaśnij, czym zastąpiono warunek $\sin(\pi x) = 0$, skoro w języku nie ma π.

Tematy do dalszej eksploracji

- **Krzywe gęste a krzywe wypełniające.** Porównaj krzywą Richardsona (gęsta, miary zero, bez końca) z krzywą Hilberta (ciągła **surjekcja** odcinka na kwadrat; zob. kurs *Obliczenia matematyczne wspierane komputerowo*, rozdział o krzywej Hilberta). Co każda z nich daje w dowodach redukcji wymiaru?
- **Polowanie na zera, których CAS nie widzi.** Zbierz tożsamości, z którymi `simplify` w SymPy sobie nie radzi (zagnieżdżone pierwiastki, gałęzie logarytmu, tożsamości Ramanujana), i sklasyfikuj przyczyny porażek.
- **Problem stałych i hipoteza Schanuela.** Zreferuj algorytm z pracy Richardsona (1997) i wyjaśnij, gdzie dokładnie potrzebna jest hipoteza.
- **Dziesiąty problem Hilberta.** Na podstawie artykułu Davisa (1973) przedstaw, dlaczego relacja $c = a^b$ jest diofantyczna (równanie Pella).
- **Formalizacja w Lean.** Sformalizuj w Lean 4 + Mathlib Lemat 3.3 ($2d \le \sin(\pi d)$ na $[0, \frac{1}{2}]$) albo Lemat 4.1. Które fakty o sinusie są już w Mathlib?
- **Rozstrzygalne podklasy.** Zaimplementuj test znaku dla wielomianów wielu zmiennych na małych przykładach (rozkład cylindryczny w Sage/Mathematica) i porównaj z naszą półrozstrzygającą procedurą przedziałową.

Odpowiedzi do pytań kontrolnych

1. Wyrażenie — skończone drzewo, które algorytm może przeczytać. Funkcja to obiekt nieskończony. Różne wyrażenia mogą oznaczać tę samą funkcję (np. $\sin^2 x + \cos^2 x$ i 1), a pytanie dotyczy funkcji. Nierozstrzygalność mieszka właśnie w przejściu od zapisu do znaczenia.

2. Nie. Twierdzenie mówi, że nie istnieje **jeden** algorytm poprawny dla **wszystkich** wyrażeń z klasy. Konkretne tożsamości (i ogromne podklasy) komputer rozstrzyga bez trudu; SymPy upraszcza $\sin^2 x + \cos^2 x$ do 1.

3. Równanie $\sin(\pi x) = 0$ definiuje zbiór $\mathbb{Z}$, a przez niego do liczb rzeczywistych wchodzi arytmetyka i dziesiąty problem Hilberta. Dla samych wielomianów problem znaku (a nawet dowolne zdania z kwantyfikatorami) jest rozstrzygalny — twierdzenie Tarskiego.

4. Kontrolują zmianę P^2 przy zaokrągleniu punktu do najbliższego punktu kratowego tam, gdzie pochodne P^2 są duże. Bez wag małe residuum nie wymusza całkowitego zera — Przykład 3.1: $P_N(x) = Nx - 1$ nie ma pierwiastka w $\mathbb{N}$, a residuum bez wag osiąga $\sin^2(\pi/N) \to 0$.

5. Wielomian P zawiera potęgi 2^{x_i}, które są całkowite tylko dla $x_i \in \mathbb{N}$. Dla nieujemnych x_i najbliższa liczba całkowita jest naturalna, więc własność (P2) działa. Dlatego najpierw dowodzimy twierdzenia na zbiorze $[0, \infty)^n$, a potem podstawiamy kwadraty.

6. Tylko w ostatnim kroku Twierdzenia 3.4: $0 \le P(a, \mathbf{m})^2 < 1$ i $P(a, \mathbf{m}) \in \mathbb{Z}$ dają $P(a, \mathbf{m}) = 0$. Cała reszta dowodu to analiza.

7. Bo później podstawiamy krzywą gęstą, która daje dowolnie **małe** wartości, ale niekoniecznie zera. Potrzebna jest przerwa: albo istnieje zero, albo wszędzie $f > 1$.

8. Nie — jej obraz ma miarę zero (Uwaga 4.6). Nie jest to potrzebne: wystarczy, że wchodzi w każde otoczenie, bo ostra nierówność $F < c$ zachodzi na całym otoczeniu (Lemat 5.1).

9. Żeby indukcja w Twierdzeniu 4.4 działała bez kumulacji błędów: kolejne współrzędne to iteracje bardzo stromej funkcji g, więc nawet mały błąd w $g(w)$ zostałby ogromnie wzmocniony. Przybliżenie też by wystarczyło (z ciągłości Γ_n), ale z astronomicznie małą tolerancją. Dokładna równość odtwarza cały „ogon" krzywej bez błędu.

10. Nie musi: wiemy tylko, że $\inf G_a = 0$ (wartości dowolnie małe). Krzywa może ominąć zero funkcji $F(a, \cdot)$. Do dowodu wystarcza próg 1.

11. W części (B): $B_a = |A_a| - A_a$ jest tożsamościowo zerem dokładnie wtedy, gdy $A_a \geq 0$. Funkcja analityczna, która znika na przedziale, znika wszędzie, więc w języku $\mathcal{A}_1$ nie zbudujemy takiego „płaskowyżu zer" — ten dowód nie daje nierozstrzygalności tożsamości bez modułu.

12. Pierwszy — tak (Stwierdzenie 10.1: przeszukujemy punkty wymierne z certyfikowanymi przybliżeniami). Drugi — nie: jego dopełnienie („$A \not\equiv 0$") jest półrozstrzygalne, więc gdyby sam był półrozstrzygalny, byłby rozstrzygalny (twierdzenie Posta), wbrew Twierdzeniu 7.1.

13. Tożsamość: wyrażenie ze zmienną, pytanie $\forall t\ A(t) = 0$ — nierozstrzygalne. Stałe: wyrażenie bez zmiennych, pytanie $c = 0$ — status nieznany (rozstrzygalne dla stałych wykładniczo-logarytmicznych, jeśli prawdziwa jest hipoteza Schanuela).

14. Obliczalne musi być przekształcenie $a \mapsto G_a$ (i dalej A_a, B_a, J_a) — operacje na drzewach wyrażeń. Punkty $\mathbf{m}$, $\mathbf{u}_0$, t_0, pierwiastki, minima istnieją tylko w dowodzie poprawności; algorytm nigdy ich nie wyznacza.

15. Dla stałych z gramatyki $(q, \pi, \log 2)$ $|c| + 2$ też by działało, bo ich znak jest znany. Reguła $c^2 + 2$ jest jednak czysto składniowa i działa także dla dowolnego podwyrażenia bez zmiennych traktowanego jako stała, np. $\pi - \frac{22}{7}$, którego znaku algorytm nie musi umieć ustalić.

16. Służy do zapisu $2^x = \exp(x \log 2)$ w wielomianie z twierdzenia Davisa–Putnama–Robinsona. Po twierdzeniu Matiyasevicha wystarczy zwykły wielomian, więc exp i $\log 2$ są zbędne (§11).

17. Potrzebujemy przedziału, na którym wyłącznik S_a jest **dokładnie** równy 1, a to wymaga $G_a \leq \frac{1}{2}$. Do tego nie wystarczy „$G_a < 1$ gdzieś" — potrzebna jest pełna informacja $\inf G_a = 0$ z Twierdzenia 5.3.

18. Nie. Algorytm Rischa rozstrzyga elementarność całki dla funkcji z odpowiednich ciał różniczkowych, **zakładając** możliwość rozpoznawania zera w tych ciałach. Richardson pokazuje, że dla języka z sin (na całej prostej) i modułem takiego testu nie ma; dla stałych problem jest otwarty.

19. Tylko jeśli obliczenie jest **certyfikowane** (np. arytmetyką przedziałową z górnym oszacowaniem < 1). Wtedy tak, bo $G_a(t) < 1 \Rightarrow a \in U$. Zwykły `float` może się mylić, zwłaszcza że świadkowie leżą przy ogromnych t (zob. przykład w §5 rozdziału „Pełny dowód krok po kroku").

20. Dla $n = 1$ wzór dałby $\Gamma_1(t) = t$, a obraz (M, ∞) nie jest gęsty w $\mathbb{R}$. Liczbę zmiennych można zawsze zwiększyć zmiennymi „niemymi", więc założenie $n \geq 2$ nic nie kosztuje.