Condition Numbers & Probability for Explaining Algorithms

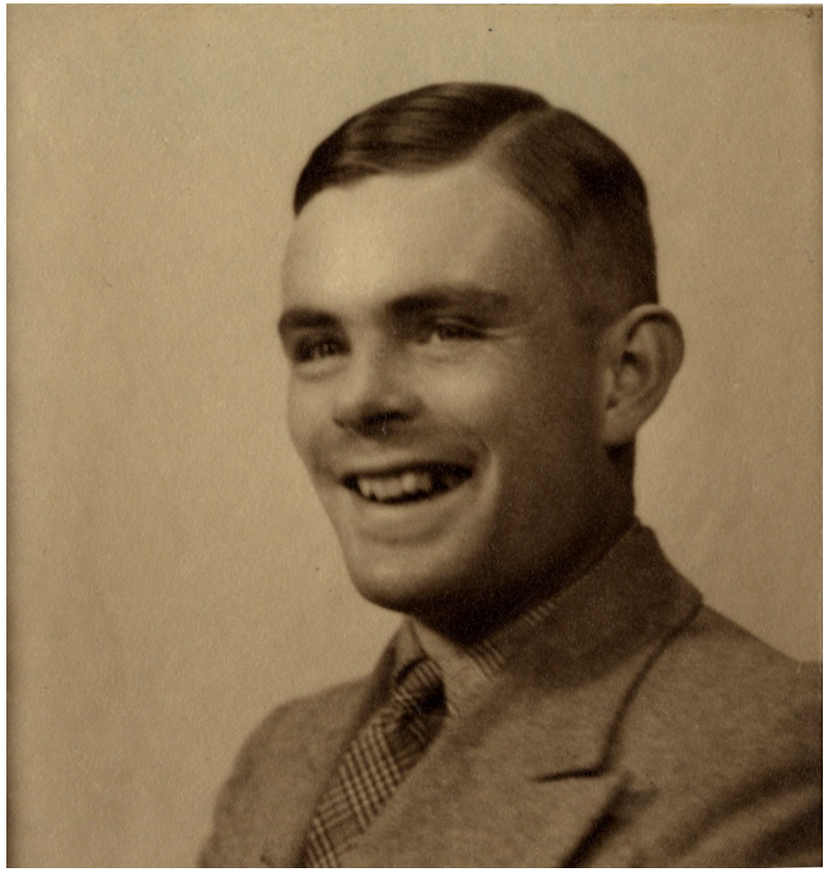
Josué Tonelli-Cueto

UTSA
The University of Texas at San Antonio™

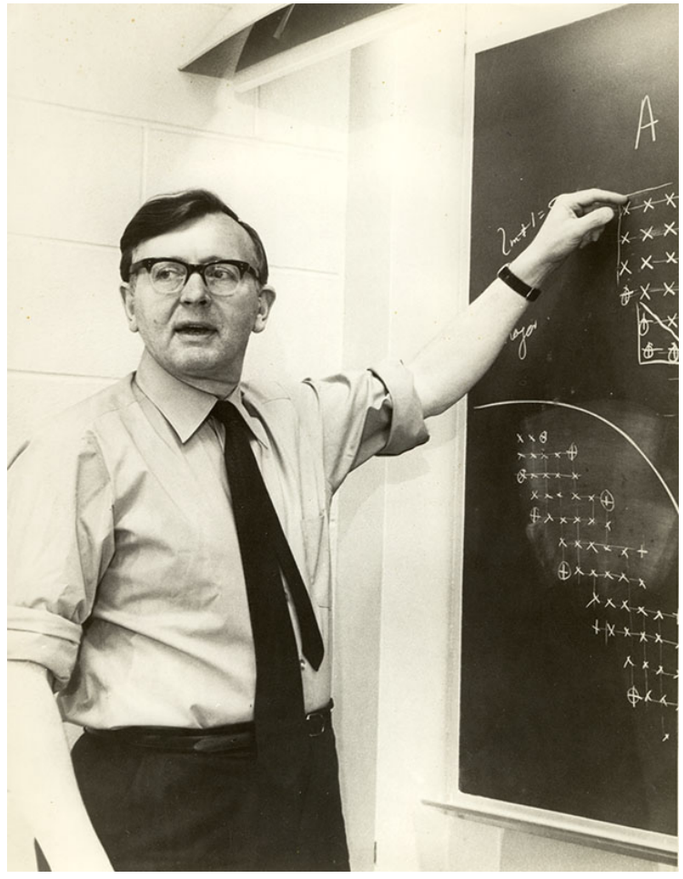
CodEx Seminar

A Foundational Myth

Turing vs. Wilkinson



Source: King's College [ATM/K/7/11]

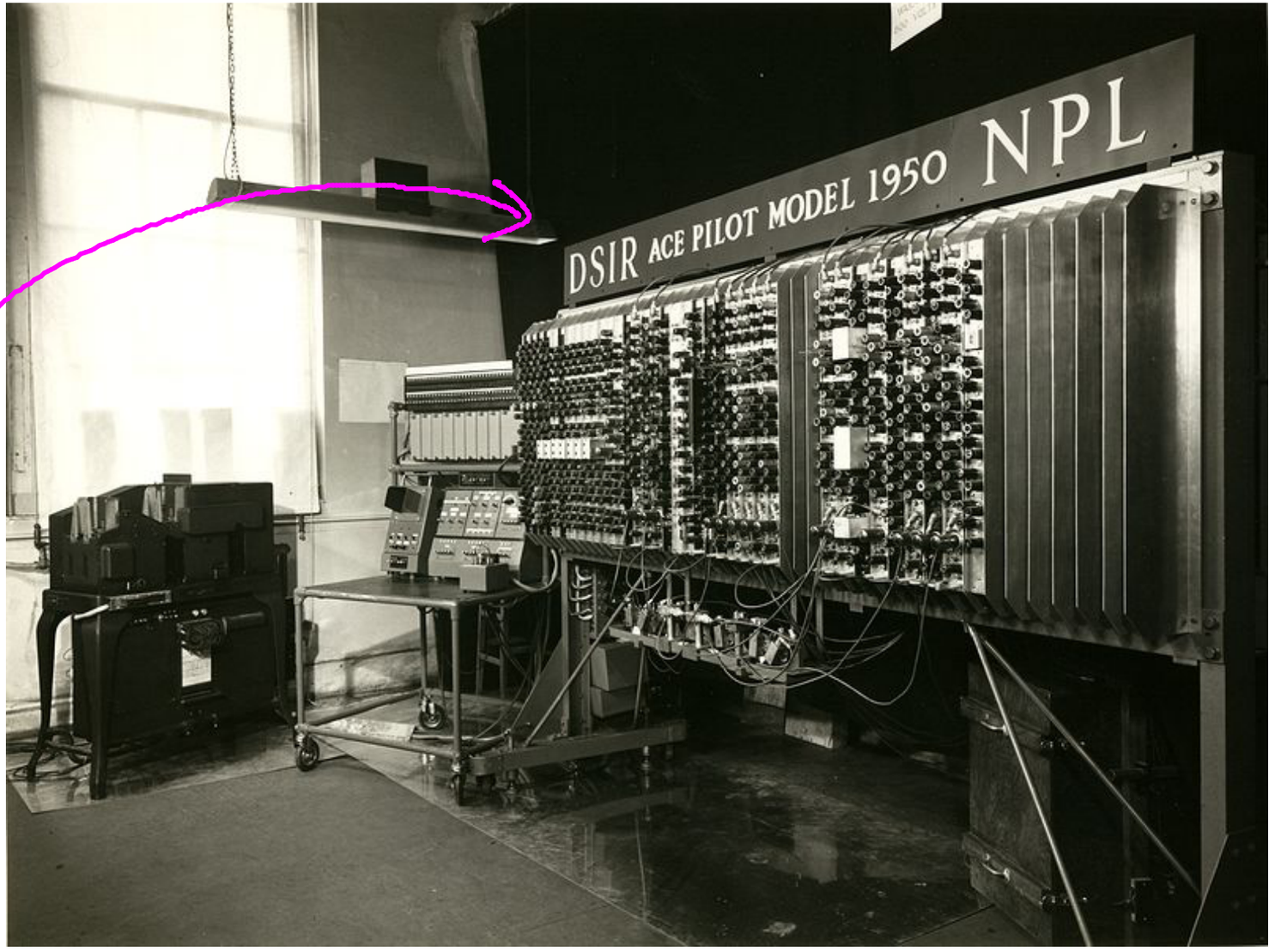


Source: U. of Manchester

Source: 1970 Turing Lecture

We are in 1946...
at the NPL in Manchester

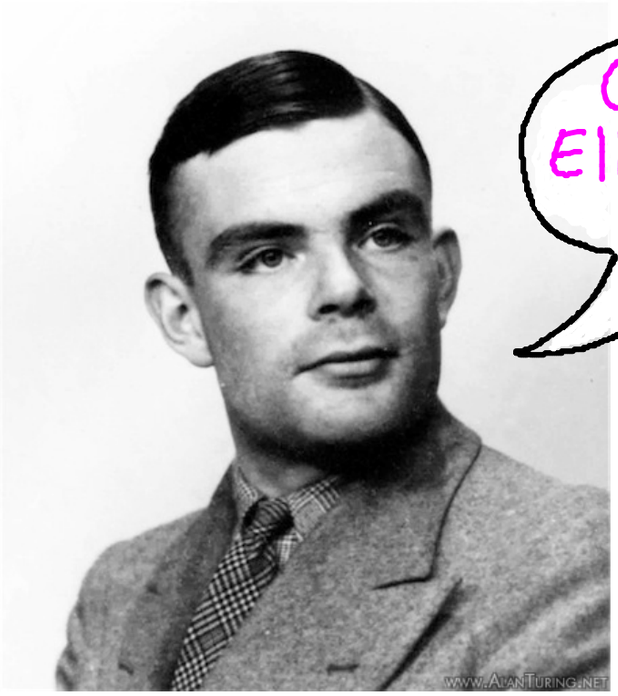
The computer 4 years after!



Source: U. of Manchester

However, it happened that some time after my arrival, a system of 18 equations arrived in Mathematics Division and after talking around it for some time we finally decided to abandon theorizing and to solve it. A system of 18 is surprisingly formidable, even when one has had previous experience with 12, and we accordingly decided on a joint effort.

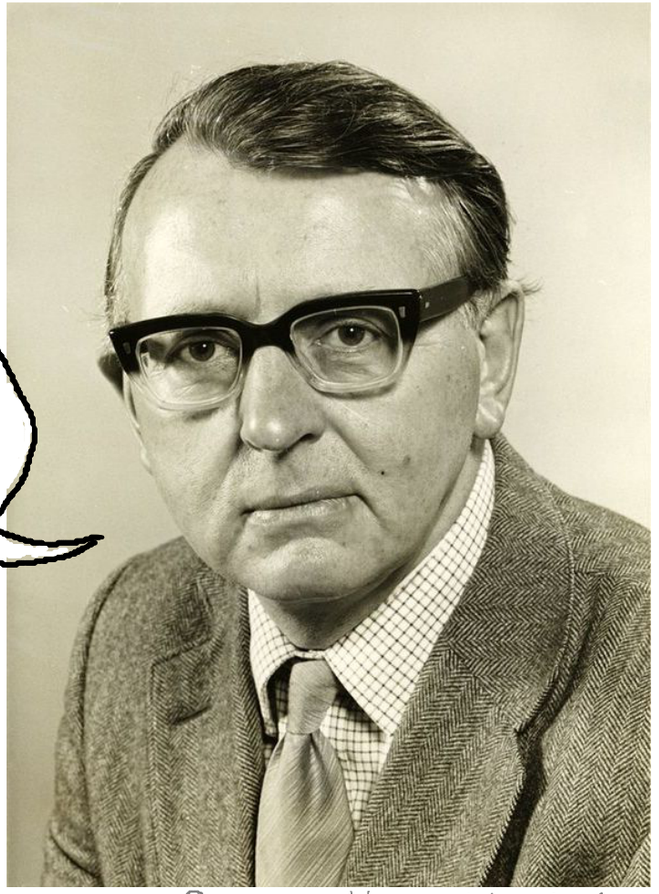
Wilkinson, 1970 Turing Lecture



Source: Beryl Turing & King's College

Gaussian Elimination will not work!

It will work! Let's do it with complete pivoting.



Source: U. of Manchester

And it succeeded!

I suppose this must be regarded as a defeat for Turing since he, at that time, was a keener adherent than any of the rest of us to the pessimistic school.

ROUNDING-OFF ERRORS IN MATRIX PROCESSES

By A. M. TURING

(*National Physical Laboratory, Teddington, Middlesex*)

[Received 4 November 1947]

The second round undoubtedly went to Turing!

Wilkinson, 1970 Turing Lecture

Why

do some algorithms perform better than predicted?

Not an isolated phenomenon:

the Simplex Method

Linear Programming

$\max\ c^T x$

$\text{s.t.}\ Ax \leq b$

Problem

Danzig (1947)

Simplex Method

↑ Very efficient in practice,

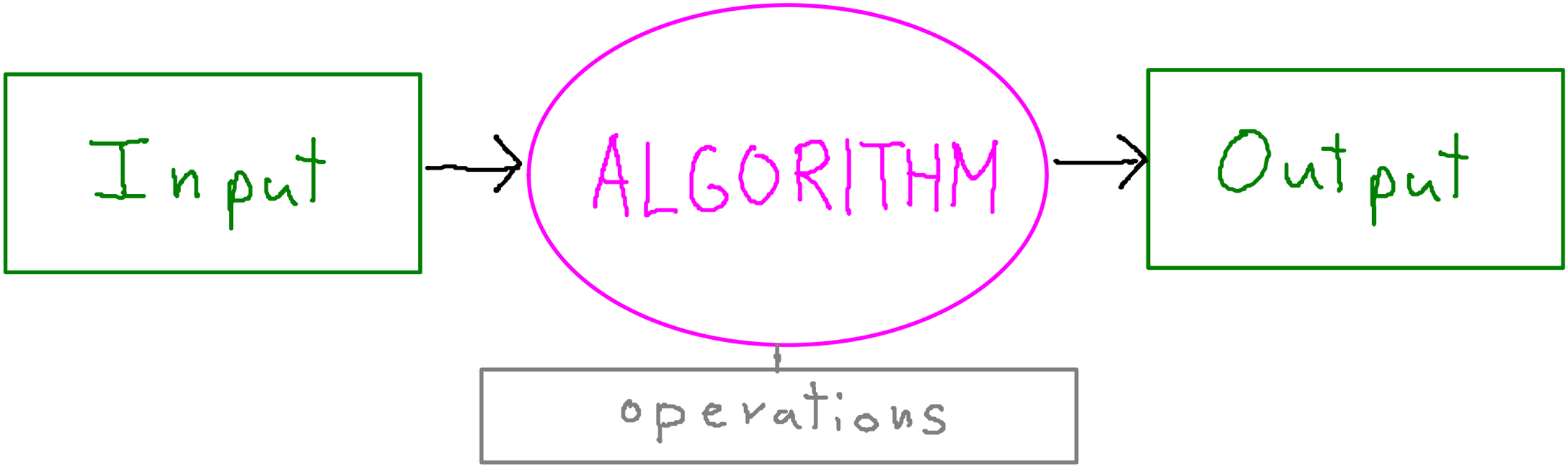
but... why?

Spielman & Teng (2001)

Justify Simplex Method using smoothed analysis

Complexity of Algorithms

Complexity of (Traditional) Algorithms

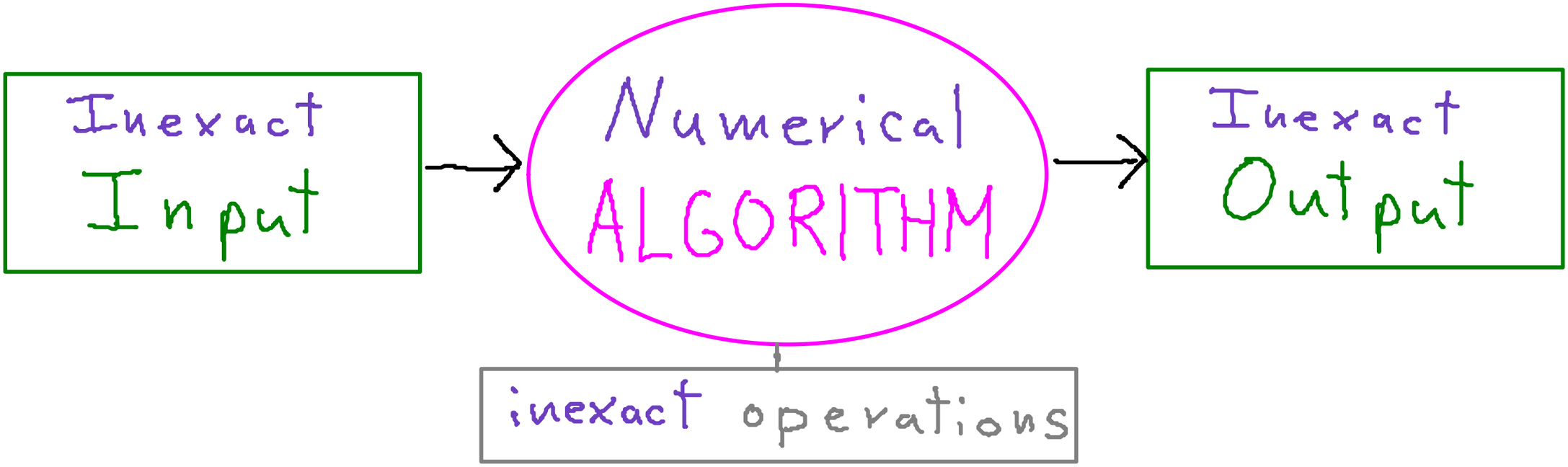


Worst-case form of complexity estimate:

$$\text{run-time}(\text{ALGORITHM}, \text{Input}) \leq g(\text{size}(\text{Input}))$$

⚠ sometimes size has several parameters
(e.g. #variables, degree...)

Complexity of Numerical Algorithms



⚠ usual form of complexity fails!

ALL INPUTS OF THE SAME SIZE ARE EQUAL,
BUT SOME INPUTS ARE MORE EQUAL
THAN OTHERS

Condition Numbers

(Turing) (Goldstine, von Neumann)

cond(Input):
measure of numerical sensitivity of Input

cond big ⇒ small variations of Input → big variations of Output

cond small ⇒ 'big' variations of Input → small variations of Output

cond is a property of the computational problem, not of the algorithm!

Turing Condition Number

$$A \in \mathbb{C}^{n \times n}$$

$$\text{cond}(A) := \|A\| \, \|A^{-1}\|$$

Linear System: $Ax = b$

$$\text{rel-error}(x) \lesssim \text{cond}(A) \max\{\text{rel-error}(A), \text{rel-error}(b)\}$$

Condition-based Complexity

(Turing) (Goldstine, von Neumann)

$$\text{run-time}(\text{ALGORITHM}, \text{Input}) \leq g(\text{size}(\text{Input}), \text{cond}(\text{Input}))$$

Can we have
a complexity estimate
of a numerical algorithm
only depending on size?

Randomize your Input

(Goldstine & von Neumann) (Smale) (Demmel)

Random Input → Probabilistic Complexity

How do we randomize the Input?

Choice depends on the context!

Probabilistic Complexity

(Goldstine & von Neumann) (Smale) (Demmel)

$$\mathbb{P}_{input}\left[\text{run-time}(\text{ALGORITHM}, input) \geq t\right] \leq \delta(s,t)$$

where $\text{size}(input) \leq s$

...and if we are lucky

$$\mathbb{E}_{input}\left[\text{run-time}(\text{ALGORITHM}, input)\right] \leq \delta(s)$$

Smoothed Complexity

(Spielman & Teng)

$$\sup_{\substack{\text{Input} \\ \text{size(Input)}=s}} \mathbb{P}_{\text{noise}}\left[\text{run-time}(\text{ALGORITHM}, \text{Input} + \sigma\,\text{noise}) \geq t\right] \leq f(s,t,\sigma)$$

... and if we are lucky

$$\sup_{\substack{\text{Input} \\ \text{size(Input)}=s}} \mathbb{E}_{\text{noise}}\left[\text{run-time}(\text{ALGORITHM}, \text{Input} + \sigma\,\text{noise})\right] \leq f(s,\sigma)$$

Why Smoothed is better?

Worst-case form of complexity estimate

$$\text{run-time}(\text{ALGORITHM}, \text{Input}) \leq f(\text{size}(\text{Input}))$$

$\uparrow \ \sigma \to 0$

Smoothed form of complexity estimates

$$\sup_{\substack{\text{Input} \\ \text{size}(\text{Input})=s}} \mathbb{P}_{\text{noise}}\left[\text{run-time}(\text{ALGORITHM}, \text{Input} + \sigma\,\text{noise}) \geq t\right] \leq f(s,t,\sigma)$$

$\downarrow \ \sigma \to \infty$

Probabilistic form of complexity estimates

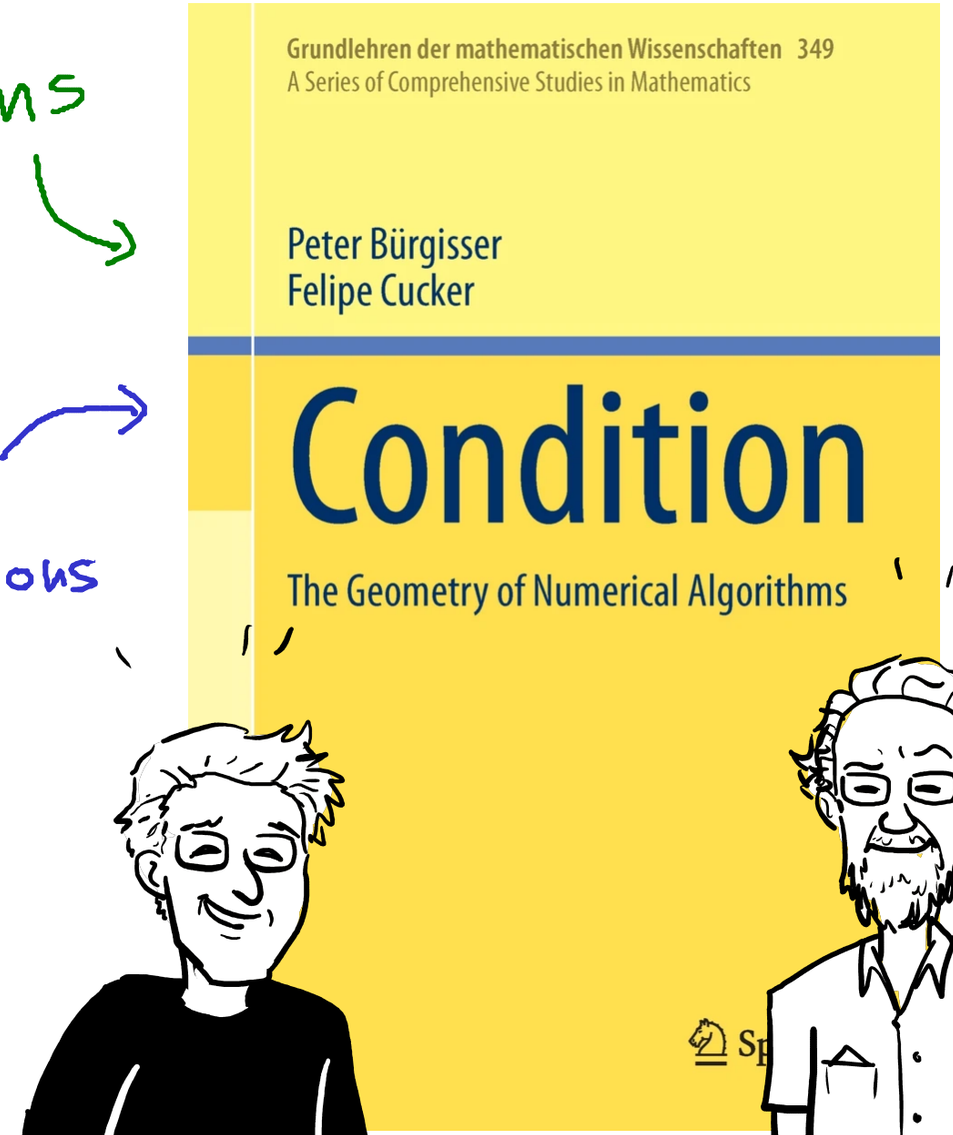
$$\mathbb{P}_{\text{input}}\left[\text{run-time}(\text{ALGORITHM}, \text{input}) \geq t\right] \leq f(s,t)$$

Where to find all the details?

Linear Systems

Systems of Polynomial Equations

Linear Programming (Interior Point Method)



Peter Bürgisser

Drawings by Jorge Cham

Felipe Cucker

A Case Study

of the Framework in Action:

the DESCARTES Solver

For finding real roots

of real univariate polynomials

Joint work of

Elias Tsigaridas

Josué Tonelli-Cuesto

Alperen A. Ergür

Photo while working on this project

Real Root Isolation I: The Problem

INPUT:

$f \in \mathbb{Z}[X]$

We can also handle continuous inputs!

OUTPUT:

Intervals $J_1, \ldots, J_k$ s.t.

0) $J_i = (a_i, b_i)$ with $a_i, b_i \in \mathbb{Q}$

1) $Z(f) \cap \mathbb{R} \subseteq \bigcup_{i=1}^{k} J_i$

2) $\forall i, \#\, Z(f) \cap J_i = 1$

INPUT SIZE PARAMETERS:

d : degree of f

τ : bit-size of coefficients of f

MEASURE OF RUN-TIME

Bit complexity

Real Root Isolation II: The State of the Art

Sturm Solver	$\tilde{\mathcal{O}}_B(d^4\tau^2)$
Descartes Solver	$\tilde{\mathcal{O}}_B(d^4\tau^2)$
ANewDsc (Sagraloff & Mehlhorn; 2016)	$\tilde{\mathcal{O}}_B(d^3+d^2\tau)$
Pan's Algorithm (Pan; 2002)	$\tilde{\mathcal{O}}_B(d^2\tau)$

Q: Can we beat the champion?

Real Root Isolation III:

What do we wish?

$$\tilde{\mathcal{O}}_B(d\tau)$$

We wish to find real roots
almost as fast as we read the polynomial!

Real Root Isolation IV:

Are we being pessimistic?

DESCARTES SOLVER
seems to behave faster in practice!

Why?

SPOILER:

DESCARTES

is almost-optimal on average!

What do we mean?

Real Root Isolation V:

Beyond pessimism

$$\mathbb{E}_{g}\{\text{cost}(\text{SOLVER}, g)^{\ell} \mid \text{bit-size}(g) \leq \tau, \deg(g) \leq d\}$$

What's a 'good' random model for g?

↑

Many choices of randomness 😱

Beyond pessimism I: Uniform Random Bit Polynomials & A Simple Main Theorem

$$F = \sum_{k=0}^{d} F_k X^k$$

s.t. $F_k \sim \mathcal{U}([-2^\tau, 2^\tau] \cap \mathbb{Z})$ independent

SIMPLE MAIN THM

$$\mathbb{E}\, \text{cost}(\text{DESCARTES}, F) = \tilde{\mathcal{O}}_B(d^2 + d\tau)$$

On average, DESCARTES is almost optimal!

Beyond pessimism II: Random Bit Polynomials

$$f = \sum_{k=0}^{d} f_k X^k \in \mathbb{Z}[X]$$

s.t. f_k independent

bit-size of f:

$$\tau(f) := \min\{\tau \mid \forall k, \mathbb{P}(|f_k| \leq 2^\tau) = 1\}$$

weight of f:

No middle indexes!

$$w(f) := \max\{\mathbb{P}(f_k = c) \mid c \in \mathbb{R}, k \in \{0, d\}\}$$

uniformity of f:

$$u(f) := \ln\left(w(f)\left(1 + 2^{\tau(f)+1}\right)\right)$$

Beyond pessimism IV: Examples of Random Bit Polynomials I

- Support control $\{0, d\} \subseteq A$

$$f = \sum_{k \in A} f_k X^k \quad \text{with} \quad f_k \sim \mathcal{U}([-2^\tau, 2^\tau] \cap \mathbb{Z})$$

... then $u(f) = 0$

- Sign control $\sigma \in \{-1, +1\}^{\{0, \dots, d\}}$

$$f = \sum_{k=1}^{d} f_k X^k \quad \text{with} \quad f_k \sim \mathcal{U}(\sigma_k([1, 2^\tau] \cap \mathbb{N}))$$

... then $u(f) \leqslant \ln 3$

Beyond pessimism V:
Examples of Random Bit Polynomials II

- Exact bitsize

$$F = \sum_{k=1}^{d} f_k X^k \quad \text{with} \quad f_k \sim \mathcal{U}(\{n \in \mathbb{Z} \mid \lfloor \log n \rfloor = r\})$$

... then $u(f) \leqslant \ln 3$

+ their combinations

Our random model is flexible!

Beyond pessimism V: Smoothed case included!

$F = \sum_{k=1}^{d} f_k X^k$ random bit polynomial

$g = \sum_{k=1}^{d} g_k X^k$ fix polynomial of entries of size τ

$\sigma \in \mathbb{Z} \setminus \{0\}$

Then:

$F_\sigma = g + \sigma f$ random bit polynomial

& $u(F_\sigma) \leq 1 + u(f) + \max\{\tau - \tau(f), \tau(\sigma)\}$

Beyond pessimism III:

Main Theorem

Main Thm

$$\mathbb{E}\,\text{cost}(\text{Descartes}, f) = \tilde{O}_B\left(d^2 + d\tau\right)\left(1 + u(f)\right)^4$$

Note: f uniform $\Rightarrow u(f) = 0$

Claim: For many cases, $u(f) = O(1)$

If $\tau = \Omega(d)$, almost like reading!

On average, Descartes is almost optimal!

Summing Up:

Descartes

is almost-optimal on average!

Descartes Solver I: Rule of Signs

$V(f) := \#$ sign variations of $f_0, f_1, \ldots$

THM (Descartes' rule of signs)

$$\#Z(f, \mathbb{R}_>) \leq V(f)$$

Moreover,

$$V(f) \leq 1 \Rightarrow \text{Equality}$$

Portrait by Frans Hal
Source: WikiMedia Commons

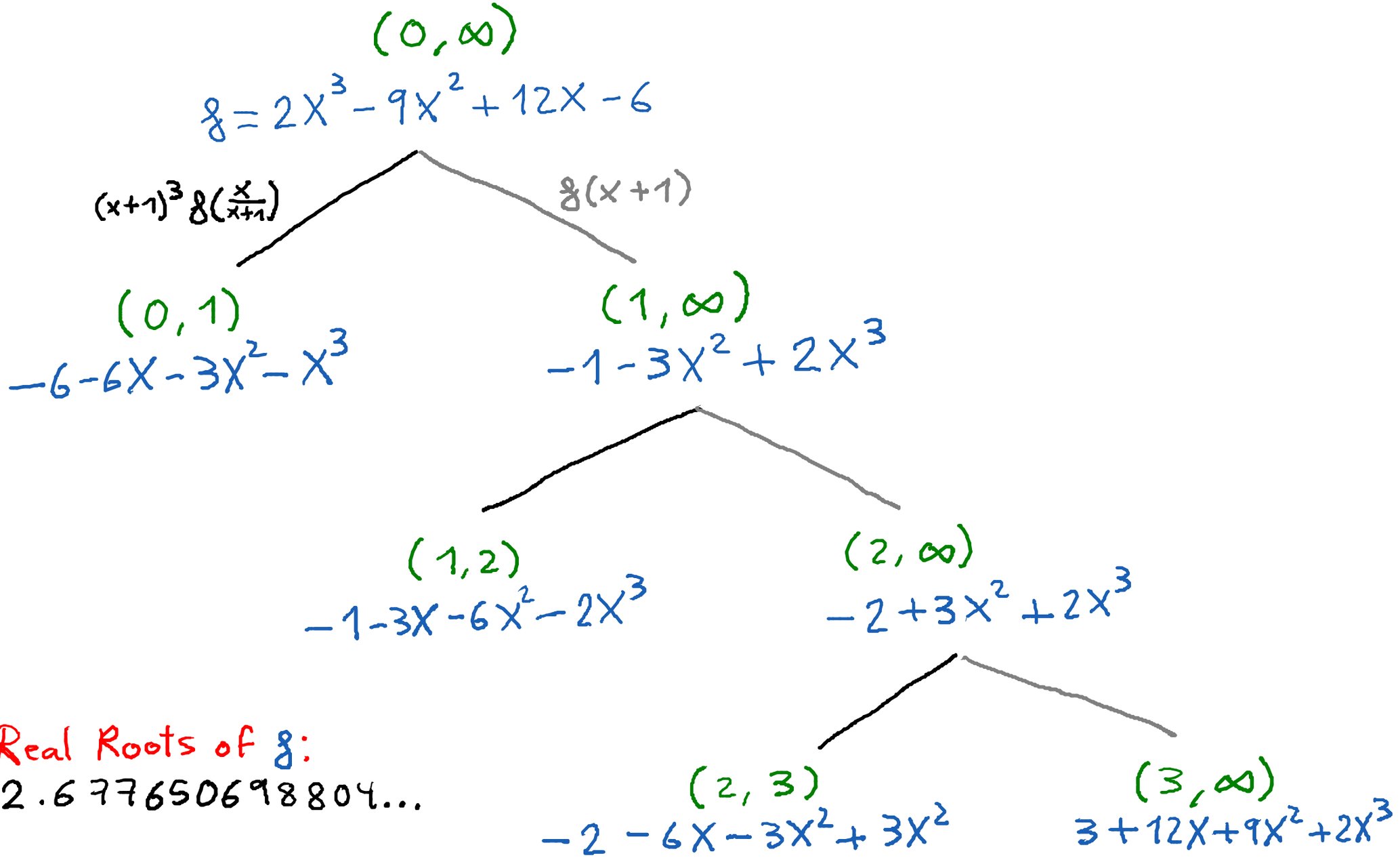
COR

$$\#Z(f,(a,b)) \leq V(f,(a,b)) := V\left((X+1)^d f\left(\frac{bX+a}{X+1}\right)\right)$$

$(0,\infty) \to (a,b)$ bijection

Descartes Solver II:

Rule of Signs in Action



Real Roots of g:

2.677650698804...

Descartes Solver III:

The Descartes' Oracle

1) Overcounting: $\#Z(f,J) \leq V(f,J)$

2) Exactness I: $V(f,J) \leq 1 \Rightarrow$ Equality

3) Exactness II:

$\#Z(f, D(m(J), cw(J))) \leq k \Rightarrow V(f,J) \leq k$

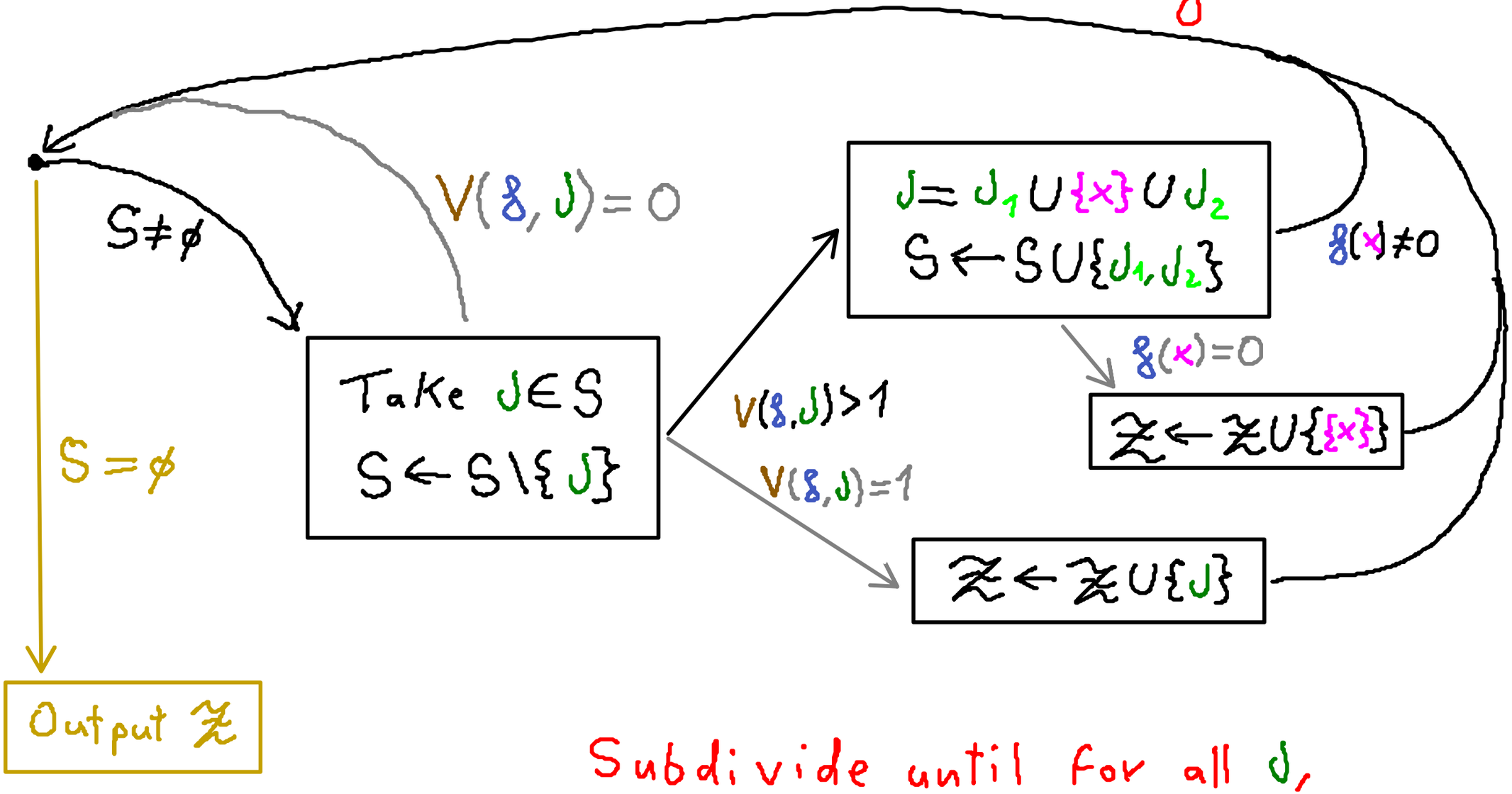
Obreshkoff's Thm: Descartes sees the complex roots around!

4) Subadditivity:

$\dot{\bigcup} J_i \subseteq J \Rightarrow \sum V(f,J_i) \leq V(f,J)$

Descartes Solver IV

The Algorithm



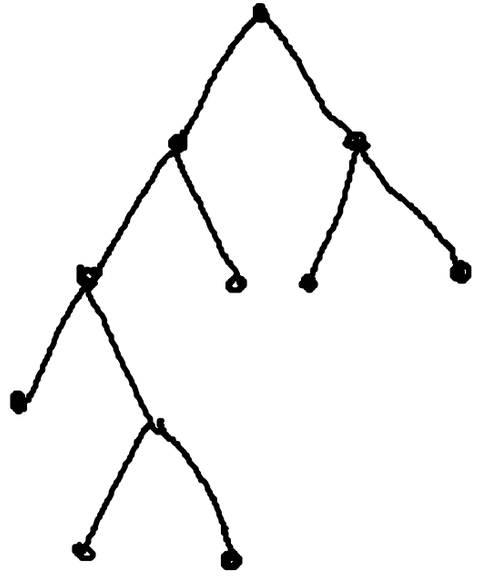
Subdivide until for all J,

$V(f, J) \leq 1$!

Descartes Solver V:

Descartes' tree

$\Upsilon(f, I)$



size of $\Upsilon(f, I)$

run-time of DESCARTES(f, I)

We only need to control the size of subdiv. tree!

The Ingredients of the Analysis I: Condition Numbers

$$C(f) := \max_{x \in [-1,1]} \frac{\sum_{k=0}^{d} |f_k|}{\max\{|f(x)|, |f'(x)|/d\}}$$

$C(f) = \infty \Leftrightarrow f$ has a singular root in $[-1,1]$

Upper bounds on $C(f)$

→ Lower bounds for root separation of f

→ Upper bounds for depth of DESCARTES' tree

The Ingredients of the Analysis II: Bounds For Number of Complex Roots

Upper bounds for

We only care about nearby roots!

complex roots of f around $[-1, 1]$

↓

Upper bounds

For width of DESCARTES' tree

The Ingredients of the Analysis III: Probabilistic Toolbox

Ball's smoothing:

$x \in \mathbb{Z}^N$ discrete random variable

$y \in \mathbb{R}^N$ s.t. $y_i \sim \mathcal{U}((-1/2, 1/2))$ i.i.d.

Then: $x + y$ continuous random var.

We can use our old cont. toolbox!

⚠ I am omitting a lot of technical details.

TAKE HOME MESSAGE:

to EXPLAIN

the SUCCESS of some ALGORITHMS,

we need

CONDITION NUMBERS & PROBABILITY

to avoid PESSIMISTIC ESTIMATES

Eskerrik Asko

zure arretagatik!

Transl.: Thank you for your attention!