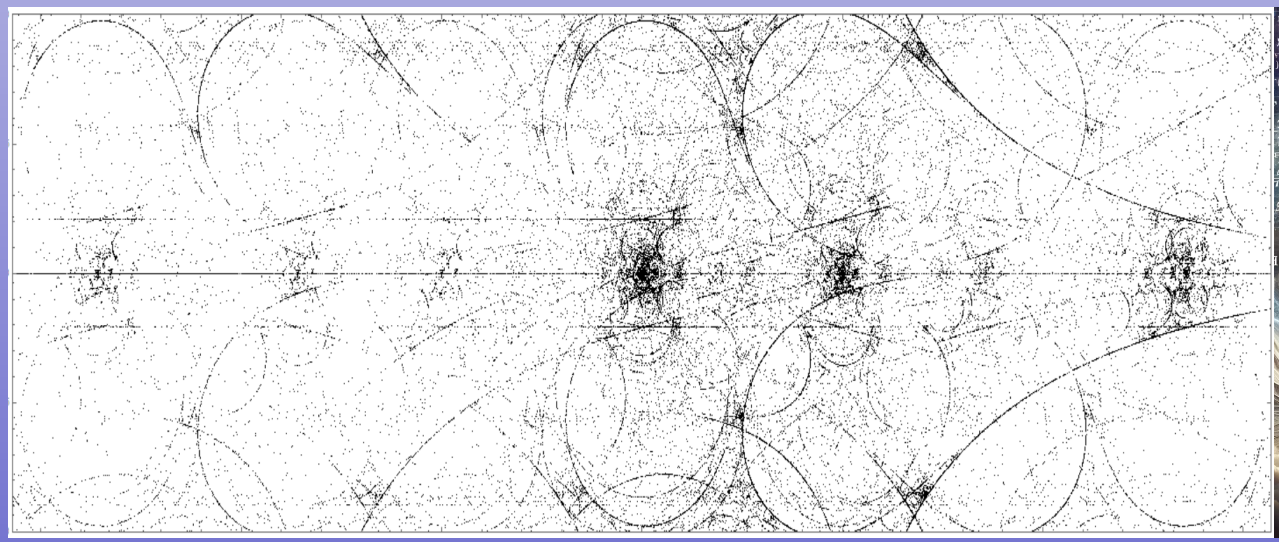
Constant Recognition

Andrzej Odrzywołek

Zakład Teorii Względności i Astrofizyki, Instytut Fizyki UJ

3 kwietnia 2024



How to start discussion on number recognition?

Topic looks clear — how to solve a typical problem

From numerical integration, numerical differential equation solution, experimental data

or insanely complicated analytical calculation etc. you got a number

$$3.645751311064590590501615753639 3$$

While it looks strangely familiar you cannot recall where or when have you seen it.

How to discuss such an elusive topic?

1. as a function approximation

How to start discussion on number recognition?

Topic looks clear — how to solve a typical problem

From numerical integration, numerical differential equation solution, experimental data

or insanely complicated analytical calculation etc. you got a number

$$3.645751311064590590501615753639 3$$

While it looks strangely familiar you cannot recall where or when have you seen it.

How to discuss such an elusive topic?

1. ~~as a function approximation~~ 2009, FAILED

How to start discussion on number recognition?

Topic looks clear — how to solve a typical problem

From numerical integration, numerical differential equation solution, experimental data

or insanely complicated analytical calculation etc. you got a number

$$3.645751311064590590501615753639 3$$

While it looks strangely familiar you cannot recall where or when have you seen it.

How to discuss such an elusive topic?

1. ~~as a function approximation~~ 2009, FAILED
2. as a physical theory, e.g. fine-structure const α

How to start discussion on number recognition?

Topic looks clear — how to solve a typical problem

From numerical integration, numerical differential equation solution, experimental data

or insanely complicated analytical calculation etc. you got a number

$$3.645751311064590590501615753639 3$$

While it looks strangely familiar you cannot recall where or when have you seen it.

How to discuss such an elusive topic?

1. ~~as a function approximation~~ 2009, FAILED
2. ~~as a physical theory, e.g. fine-structure const α~~ 2020, FAILED

How to start discussion on number recognition?

Topic looks clear — how to solve a typical problem

From numerical integration, numerical differential equation solution, experimental data

or insanely complicated analytical calculation etc. you got a number

$$3.645751311064590590501615753639 3$$

While it looks strangely familiar you cannot recall where or when have you seen it.

How to discuss such an elusive topic?

1. ~~as a function approximation~~ 2009, FAILED
2. ~~as a physical theory, e.g. fine-structure const α~~ 2020, FAILED
3. as a mathematical theory of e.g. exp-log numbers etc.

How to start discussion on number recognition?

Topic looks clear — how to solve a typical problem

From numerical integration, numerical differential equation solution, experimental data

or insanely complicated analytical calculation etc. you got a number

$$3.645751311064590590501615753639 3$$

While it looks strangely familiar you cannot recall where or when have you seen it.

How to discuss such an elusive topic?

1. ~~as a function approximation~~ 2009, FAILED
2. ~~as a physical theory, e.g. fine-structure const α~~ 2020, FAILED
3. ~~as a mathematical theory of e.g. exp-log numbers etc.~~ 2016, FAILED

How to start discussion on number recognition?

Topic looks clear — how to solve a typical problem

From numerical integration, numerical differential equation solution, experimental data

or insanely complicated analytical calculation etc. you got a number

$$3.645751311064590590501615753639 3$$

While it looks strangely familiar you cannot recall where or when have you seen it.

How to discuss such an elusive topic?

1. ~~as a function approximation~~ 2009, FAILED
2. ~~as a physical theory, e.g. fine-structure const α~~ 2020, FAILED
3. ~~as a mathematical theory of e.g. exp-log numbers etc.~~ 2016, FAILED
4. as a genetic/evolutionary process (DNA = binary seq. 0111010011100011, amino acids - functional composition)

How to start discussion on number recognition?

Topic looks clear — how to solve a typical problem

From numerical integration, numerical differential equation solution, experimental data

or insanely complicated analytical calculation etc. you got a number

$$3.645751311064590590501615753639 3$$

While it looks strangely familiar you cannot recall where or when have you seen it.

How to discuss such an elusive topic?

1. ~~as a function approximation~~ 2009, FAILED
2. ~~as a physical theory, e.g. fine-structure const α~~ 2020, FAILED
3. ~~as a mathematical theory of e.g. exp-log numbers etc.~~ 2016, FAILED
4. ~~as a genetic/evolutionary process (DNA = binary seq. 01110100111000 11, amino acids - functional composition)~~ FAILED

How to start discussion on number recognition?

Topic looks clear — how to solve a typical problem

From numerical integration, numerical differential equation solution, experimental data

or insanely complicated analytical calculation etc. you got a number

$$3.645751311064590590501615753639 3$$

While it looks strangely familiar you cannot recall where or when have you seen it.

How to discuss such an elusive topic?

1. ~~as a function approximation~~ 2009, FAILED
2. ~~as a physical theory, e.g. fine-structure const α~~ 2020, FAILED
3. ~~as a mathematical theory of e.g. exp-log numbers etc.~~ 2016, FAILED
4. ~~as a genetic/evolutionary process (DNA = binary seq. 011101001110001 1, amino acids - functional composition)~~ FAILED
5. as a data compression

How to start discussion on number recognition?

Topic looks clear — how to solve a typical problem

From numerical integration, numerical differential equation solution, experimental data

or insanely complicated analytical calculation etc. you got a number

$$3.645751311064590590501615753639 3$$

While it looks strangely familiar you cannot recall where or when have you seen it.

How to discuss such an elusive topic?

1. ~~as a function approximation~~ 2009, FAILED
2. ~~as a physical theory, e.g. fine-structure const α~~ 2020, FAILED
3. ~~as a mathematical theory of e.g. exp-log numbers etc.~~ 2016, FAILED
4. ~~as a genetic/evolutionary process (DNA = binary seq. 011101001110001 1, amino acids - functional composition)~~ FAILED
5. as a data compression
6. as an artificial intelligence (pattern recognition, database search, **brute force search**)

How to start discussion on number recognition?

Topic looks clear — how to solve a typical problem

From numerical integration, numerical differential equation solution, experimental data

or insanely complicated analytical calculation etc. you got a number

$$3.645751311064590590501615753639 3$$

While it looks strangely familiar you cannot recall where or when have you seen it.

How to discuss such an elusive topic?

1. ~~as a function approximation~~ 2009, FAILED
2. ~~as a physical theory, e.g. fine-structure const α~~ 2020, FAILED
3. ~~as a mathematical theory of e.g. exp-log numbers etc.~~ 2016, FAILED
4. ~~as a genetic/evolutionary process (DNA = binary seq. 011101001110001 1, amino acids - functional composition)~~ FAILED
5. as a data compression
6. as an artificial intelligence (pattern recognition, database search, **brute force search**)
7. as a random search (statistical process, *MonteCarlo method*)

How to start discussion on number recognition?

Topic looks clear — how to solve a typical problem

From numerical integration, numerical differential equation solution, experimental data

or insanely complicated analytical calculation etc. you got a number

$$3.645751311064590590501615753639 3$$

While it looks strangely familiar you cannot recall where or when have you seen it.

How to discuss such an elusive topic?

1. ~~as a function approximation~~ 2009, FAILED
2. ~~as a physical theory, e.g. fine-structure const α~~ 2020, FAILED
3. ~~as a mathematical theory of e.g. exp-log numbers etc.~~ 2016, FAILED
4. ~~as a genetic/evolutionary process (DNA = binary seq. 011101001110001 1, amino acids - functional composition)~~ FAILED
5. as a data compression
6. as an artificial intelligence (pattern recognition, database search, **brute force search**)
7. as a random search (statistical process, *MonteCarlo method*)
8. **Last but not least:** *a practical tool which might be occasionally required/used*

Gedankenexperiment: is problem solvable at all?

The experiment shows stated problem is uniquely solvable under assumptions:

1. unknown number can be (in principle) obtained/computed to arbitrary accuracy, i.e., on demand we could obtain more significant digits
2. computational device/system used to compute it is known in advance
3. length of calculation steps is finite (in practice small, up to several terms)

Experiment proceeds as follows:

- give someone scientific calculator
- let him press secretly several buttons
- print out numerical result

Question: can you reverse above process, i.e., given numerical output recover original sequence?
Answer: YES, one can run brute force search, pressing all possible sequence of buttons, increasing sequence length until you get "identical" result.

Practical implementation(s)

Practice shows inverse-calc approach (or equivalent, like expression tree or pre-calculated database search) is THE ONLY working one:

1. RIES by Robert Munafo mrob.com { ANSI C }
2. ISC/ISC2 by Simon Plouffe et. al (lost tech) [https://www.plouffe.fr/] { Maple}
3. Ask Constants by David R. Stoutemyer [https://math.hawaii.edu/~dale/AskConstants/AskConstants.html] { Mathematica }
4. Constant Recognition by A.O. [https://th.if.uj.edu.pl/~odrzywolek/WASM/index_parallel.html] { Mma, C }

Any other software/method is heuristic wandering in the dark, e.g:

- Maple identify()
- Wolfram Alpha
- SymPy nsimplify() { Python }

Noteworthy, any Symbolic Regression (SR) general software is able to recognize constants as special case of zero-variable constant functions, e.g. PySR by Miles Cranmer { Julia, Python }.

Be useful

Original implementation of the software at `https://github.com/VA00/SymbolicRegressionPackage` *was either slow (Mathematica version) or very obscure (C code written by other Mathematica code) and required both Mma and Linux/gcc basic skills to run, not to mention plenty of options of unknown to laymans importance. Output was RPN code encoded as base-n numbers, and yet other Mma code was required to decipher it.*

Be useful

Original implementation of the software at `https://github.com/VA00/SymbolicRegressionPackage` *was either slow (Mathematica version) or very obscure (C code written by other Mathematica code) and required both Mma and Linux/gcc basic skills to run, not to mention plenty of options of unknown to laymans importance. Output was RPN code encoded as base-n numbers, and yet other Mma code was required to decipher it.*

Above was not useful ... unfortunately also for referees

Potential user need web application/website, where input number can be pasted, and candidate expression(s) in convenient form (like Mathematica, Python etc.) returned.

Options:

- package to download/install (unlikely to succeed, due to occasional random use)
- client-server WWW application with easy to use frontend
- WASM

Be useful

Original implementation of the software at `https://github.com/VA00/SymbolicRegressionPackage` *was either slow (Mathematica version) or very obscure (C code written by other Mathematica code) and required both Mma and Linux/gcc basic skills to run, not to mention plenty of options of unknown to laymans importance. Output was RPN code encoded as base-n numbers, and yet other Mma code was required to decipher it.*

Above was not useful ... unfortunately also for referees

Potential user need web application/website, where input number can be pasted, and candidate expression(s) in convenient form (like Mathematica, Python etc.) returned.

Options:

- package to download/install (unlikely to succeed, due to occasional random use)
- client-server WWW application with easy to use frontend
- WASM

Emergence of LLMs with coding capabilities

- *30 Nov 2022* Chat GPT 3.5
- 2023: Chat GPT4, Code Interpreter, Copilot, Gemini, Claude, Mixtral ...
- good at JavaScript & Web development including client-server apps and frontends (training corpus include millions lines of code, manuals, and troubleshooting)
- now "everyone" can create web applications, with "minimal" effort

Problems with client-server apps

Indeed, in 1-2 week I was able to create such an app and deploy it to Azure Cloud. Noteworthy, Java Script parts were entirely written by AI. But ...

1. there are no free servers anymore
2. existing cloud solutions (Microsoft Azure, Google, Amazon Cloud) require to pay via CreditCard, only 3 months trials
3. no option for compute-bound problems
4. paranoid security do not allow us to use own computers
5. all of the above applies to database-based constant recognizers, what explain why they are all dead as of now
6. modern internet developement assume web app must earn for itself to survive (Ads, subscriptions, donations, sponsors)

However, there is still light in the tunnel: **WASM** option!

WASM Constant Recognition app

WASM (Web Assembly) is increasingly more popular system, allowing to run native plain C code within internet browser (Opera, Firefox, Chrome, Edge, ...).

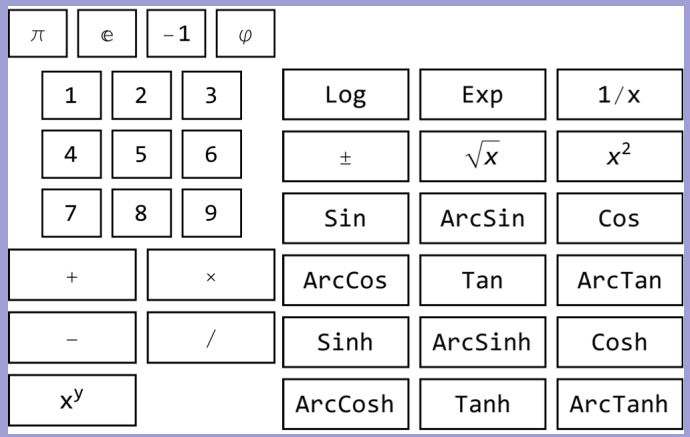
- run untouched C code
- special compiler Emscripten (emcc) create virtual machine LLVM code
- code is executed withing competely isolated browser environment
- runs at speed equal to, say $1/3$ of unoptimized bare metal version ...
- ...what is still thousands times faster than Python/JS/Mma code
- C code is mixed with JavaScript, and AI was used to handle „glue code”
- **last but not least**, it is end-user who provide and pay for nearly all resources (hardware, maintenence, staff, electricity) for search to run
- surprisingly, WASM app can be used in very long search (tested by B.D. for 2 weeks) and use massively parallel computations (tested on 80-thread Brave browser run from our 56-core 2xXeonGold machine)

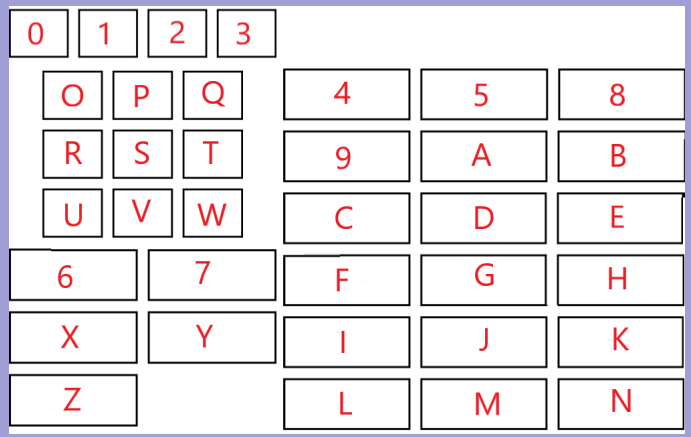
Bottom line: WASM web app is nearly as powerful as native C code, and instantly available for anyone who have at least smartphone with internet browser. For people like me this is breakthrough, bringing my C codes and experience **back from the dead**.

What do you need?

To create "simple" search app one must:

1. decide and fix RPN calculator used (std. 36-button sci calc)
2. use real or complex numbers?
3. fix search order (subsequent base-36 numbers `itoa()`)
4. check for valid RPN codes (stack machine)
5. define precision & stop criteria for search (rel. err $<=$ 16 DBL_EPSILON)





Every possible formula is encoded as base-36 integer, e.g. $\pi^{e} - e^{\pi} \equiv$ 01Z10ZX.

Let's show some examples

- definite & indefinite integration (poor man integrator)
- formula simplification
- π day formulas
- physical constants (Koide formula, $\frac{1}{\alpha} = 137.035999$)
- memorization of large integers
- smartphone & tablet benchmarking

Optimization & bottlenecks

Major bottlenecks & unsolved problems

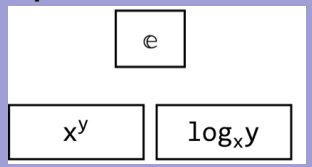
- `itoa()` in base-36 numbers takes up to 95% of search time (ideas: use base-32 or base-64 numbers, sequential `itoa()`)
- checking RPN code syntax (solution: generate ONLY valid codes using, e.g., formal grammar)
- compute poor (solution: use GPU and become GPU-poor)

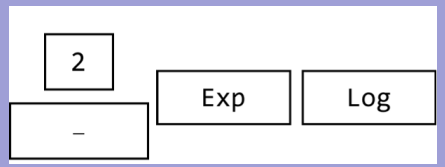
NOTE: optimization efficiency heavily hardware/compiler dependent, especially comparing `emcc` and `gcc/icx` performance.

High-level optimization

Do search efficiency depend on calc used?

1. 36-button calculator is culmination of human experience: only truly useful functions survived
2. hints from evolutionary biology: genetic code do not look like frozen incident
3. less calc buttons (e.g. 3-button system $x^y, \log_x y, e$) – more chance to solve all optimization issues!





Above two examples are fully operational calcs, capable of generating ANY elementary function!

3-criteria for STOP

What if the target number is known only with certain accuracy?

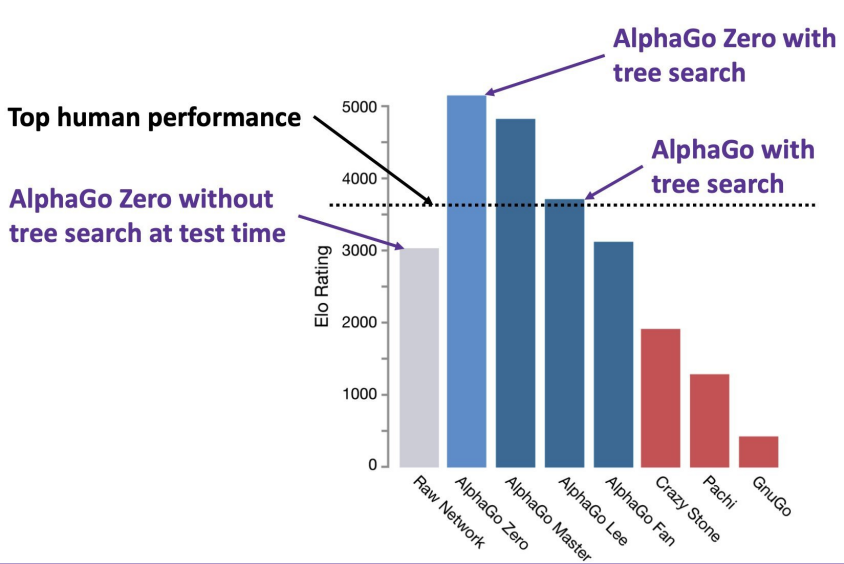
For infinite precision target STOP criterion for search is trivial: numbers must match (either literally or numerically).
For finite precision, like e.g., $\mathrm{z} = 3.14$ we can:

- compute compression ratio of original decimal string length to base-10 representation of best code; if compression ratio $\mathrm{r} > 1$, then we got hit, if $\mathrm{r} \ll 1$ further search is pointless
- construct quasi-convergent subsequence of progressively better approximation; next one is statistically expected to be e-fold improvement over previous in terms of relative precision — premature drop of error is "smoking gun" of true formula discovery
- one can replace sequential search with random sample from (presumably known) statistical distribution of target numbers (e.g. exp-log) – and estimate probability

See `https://arxiv.org/abs/2002.12690` for details.

Remark 1

Brute-force seach is highly underrated form of artificial intelligence.



"Person" (including person-like AI systems) appear much more "intelligent" when using brute-force to solve certain sub-problems. Especially if it is done in secret, and others do not know about such abilities. Imagine Microsoft running ConstantRecognition using 350, 000 GPU's recently ordered?

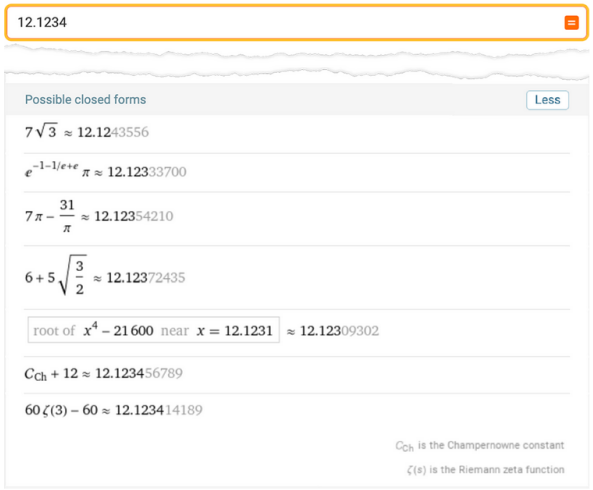
Remark 2

From Stephen Wolfram blog:

```
In[•]:= FindFormula [EntityValue ["Planet", {"OrbitPeriod", "SemimajorAxis"}], p]

Out[•]= (0.251894 + 0.0135544 QuantityMagnitude [p, Days]^0.7) au
```

There's an even more minimal example of this kind of thing in recognizing numbers. Type a number into Wolfram|Alpha and it'll try to tell you what "possible closed forms" for the number might be:



Constant recognition problem, in contrast to general Symbolic Regression of functions of many variables, (incl. Neural Networks), could be solved exactly. We can, in principle, enumerate all possible formulas down to double precision, resulting in several petabytes of data, and check how to optimally identify/compress them.

In contrast, for genuine function of say 15 variables, we barely can generate all permutations of input variables. Forget about any operations or functions applied to them.

AI: is this really different from random search?

Remark 3

Brute-force search is very limited, especially for those who are in GPU-poor class. However, there are plenty of methods *deceivingly* extending our capabilities:

- random search (Monte Carlo) methods
- genetic/evolutionary algorithms (mutations,
- directed search (gradient, tree, neighborhood)
- database search (find things others already did)

All of the above converge much faster, but skip a lot of potential simpler solutions.

In fact, proper constant identification algorithm is ***the least efficient*** *in terms of approximation!*

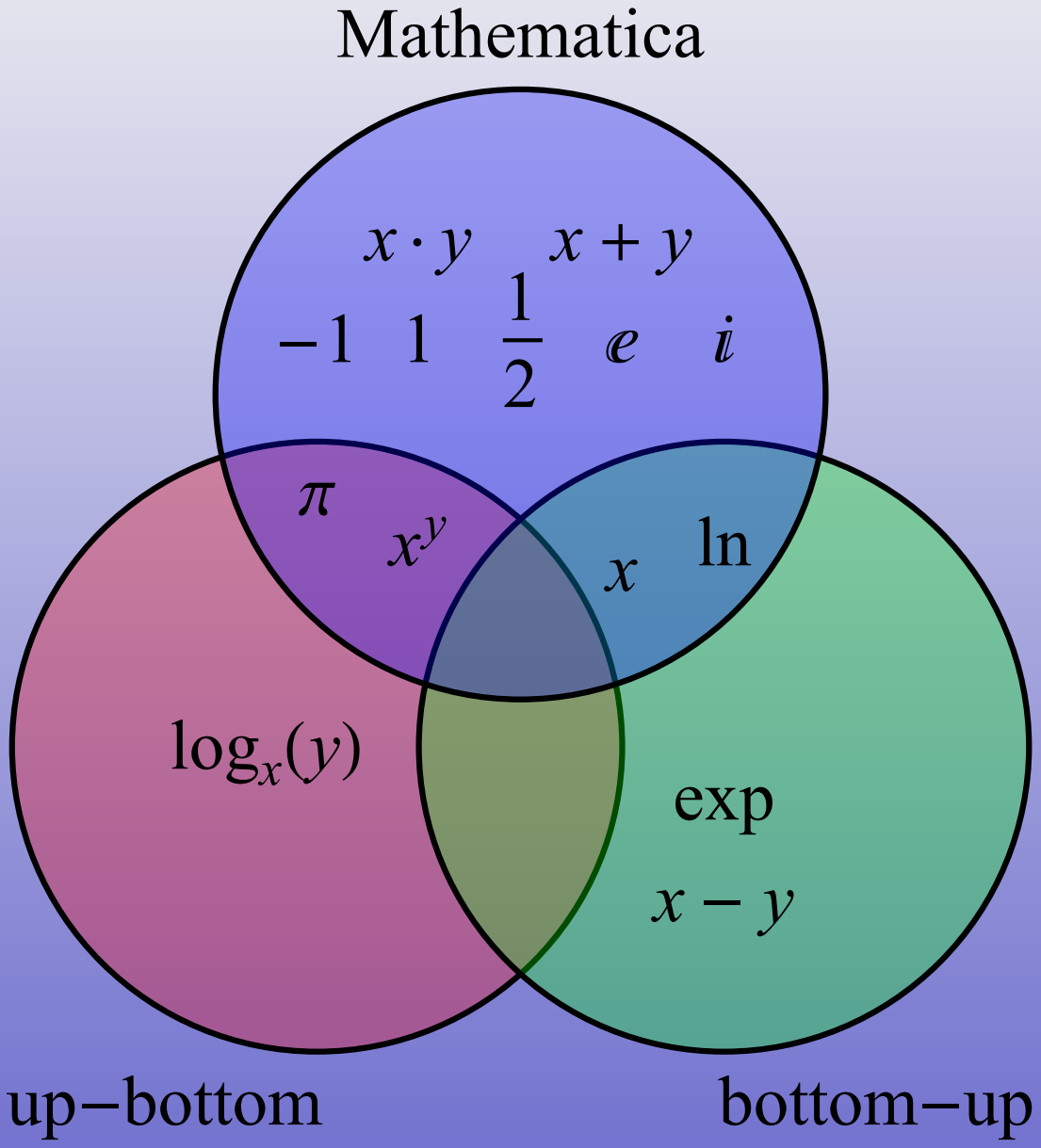
Conclusions

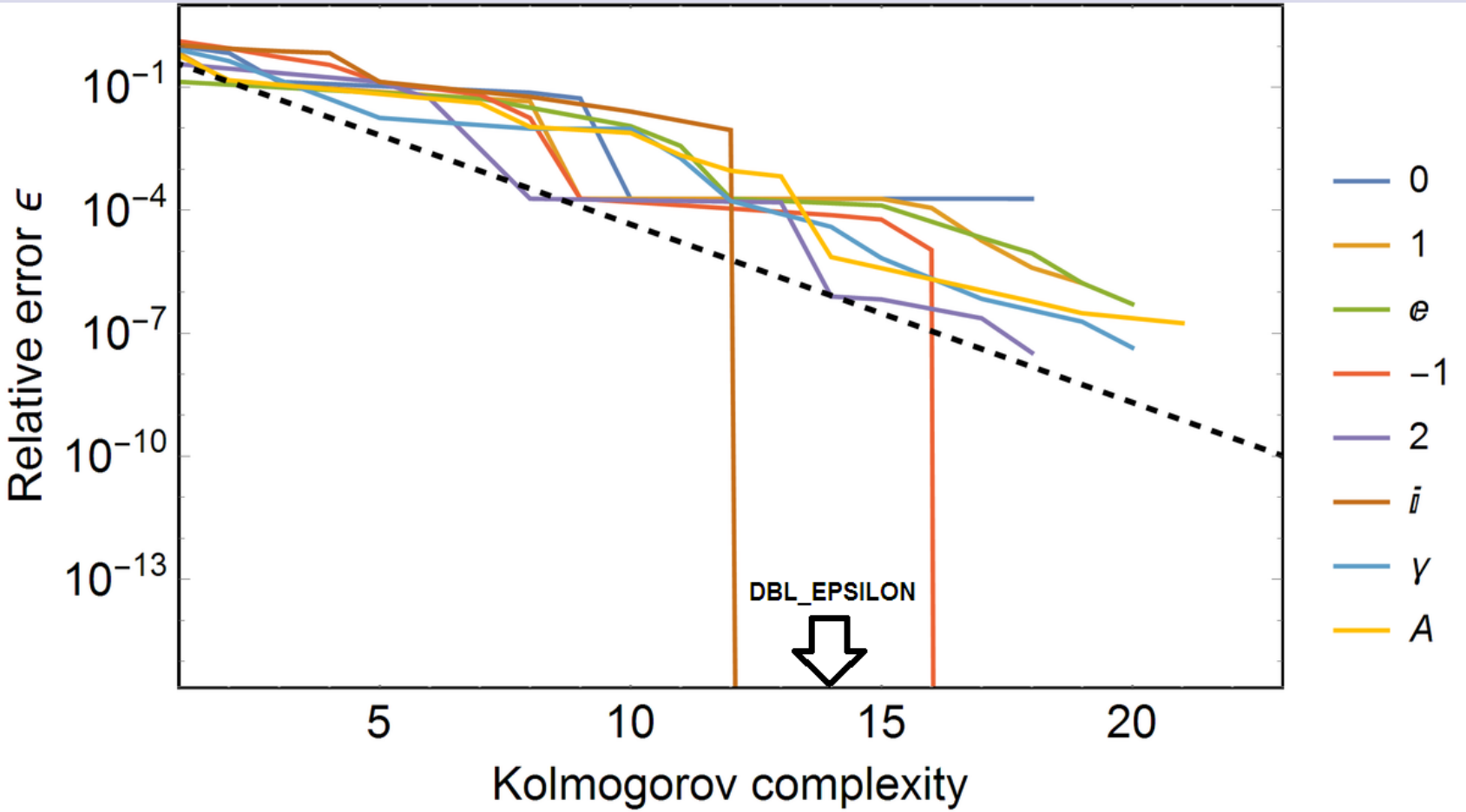
1. Thanks to **WASM** & AI technology "legacy" C code for Constant Recognition available to a wide audience *via* simple internet browser
2. Unsolved problems:
   - RPN code validator
   - sequential/parallelized expression enumeration (Karol Urbański: use NLTK/formal grammar?)
   - optimal button (instruction) set
   - low-level optimization (LLVM assembly, vectorization, GPU?)

   still quench search speed 20x-100x below theoretical benchmark set by RIES
3. scattered community in the field still lacking common name *(constant recognition, inverse calculator, number recognition/identification, formula finder, numerical simplification, expression guess, expression tree enumeration, symbolic regression, broken calculator, inverse RPN, Plouffe inverter, wild constant hunt, AI Feynmann, pattern recognition, genetic search, brute-force/extreme optimization, sentence generation)*
4. Unfinished work: JavaScript parallelization, implementation of 3 STOP criteria [compression, entropy, quasi-convergence]
5. Any feedback from potential users appreciated (Thanks Bogdan Damski!)
   - **single answer** *VS* candidates list
   - more options *VS* **less options**
   - **real numbers** *VS* complex numbers
   - faster search *VS* **exhaustive search**
   - **use compute** *VS* use disk/memory
6. Future: use complex tetration?

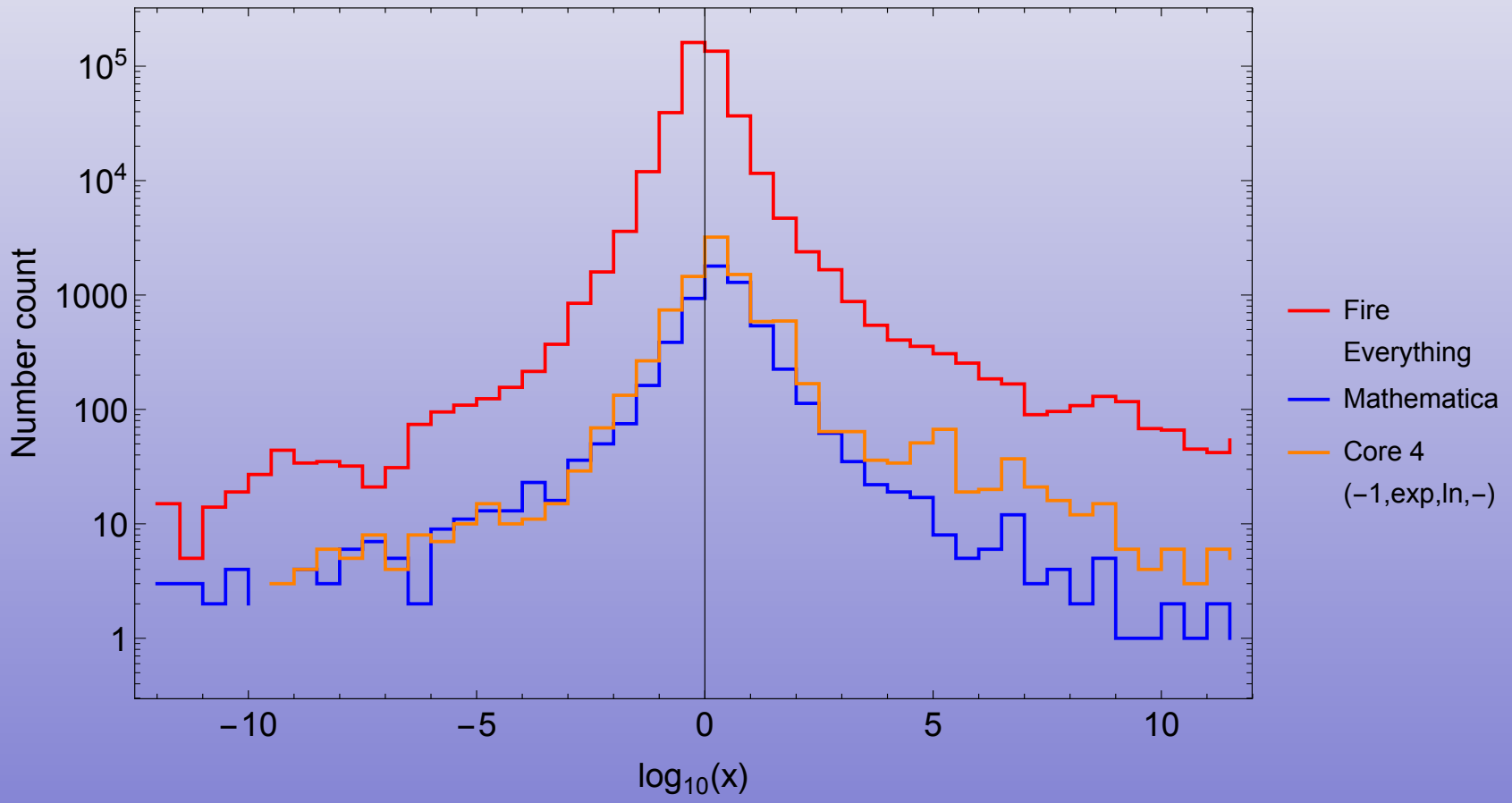
Extra Slides

ChatGPT4-V: what is 3.421175121605644176146814 ?

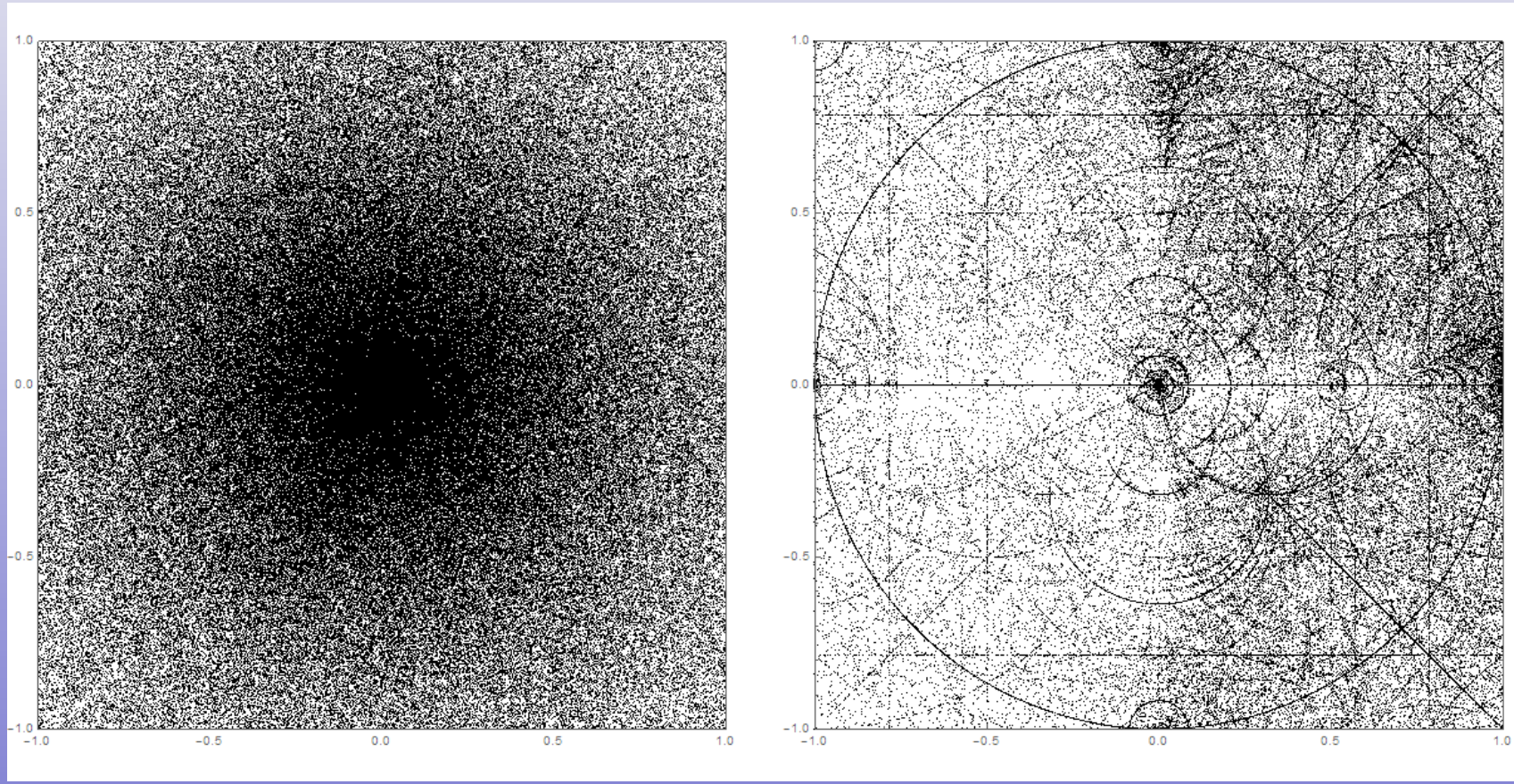

Mathematica
$x \cdot y$
$x + y$
-1
1
$\frac{1}{2}$
e
i
π
x^y
x
ln
$\log_x(y)$
exp
$x - y$
up−bottom
bottom−up


Relative error ϵ
10^−1
10^−4
10^−7
10^−10
10^−13
DBL_EPSILON
5
10
15
20
Kolmogorov complexity
0
1
e
−1
2
i
γ
A

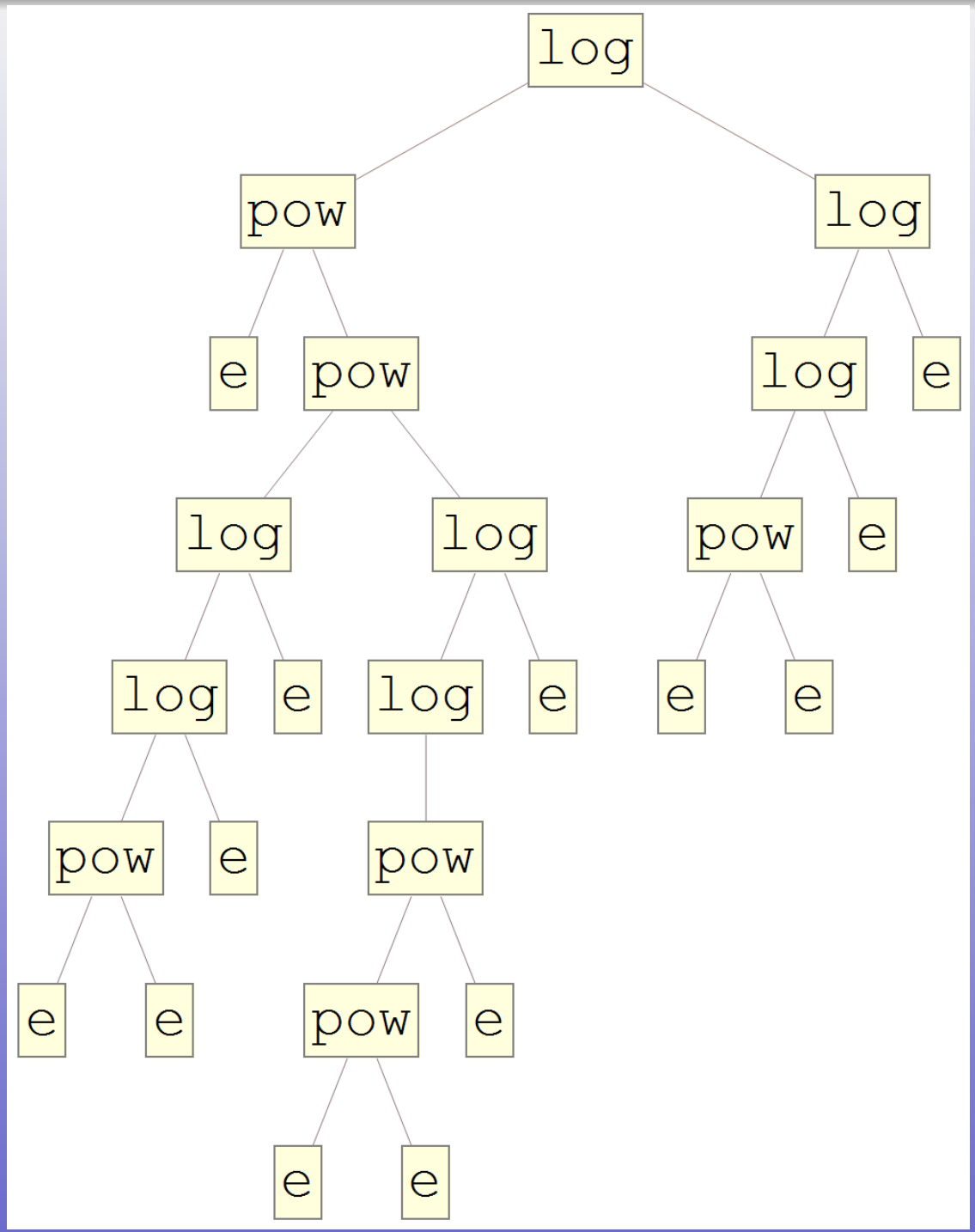
Histogram

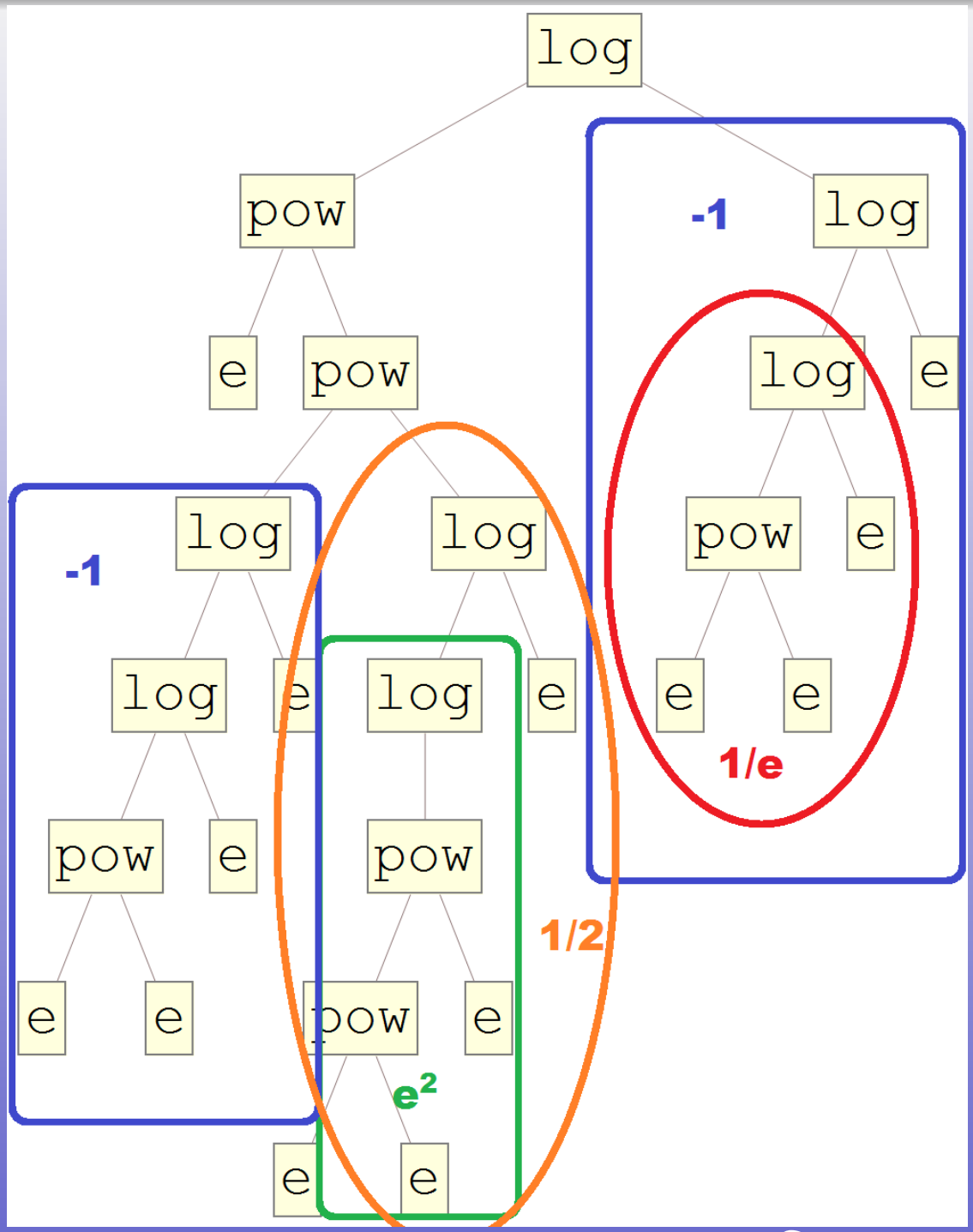

Number count
10^5
10^4
1000
100
10
1
−10
−5
0
5
10
$\log_{10}(x)$
Fire
Everything
Mathematica
Core 4
(−1,exp,ln,−)

Base CALC1 numbers on complex plane



Characteristic	**DNA/RNA**	**Computers**
Target	Survival	Truth
Digits	C G T A	0 1
Numeral system	base-4	binary
Word length	3	8
Word capacity	$4^3 = 64$	$2^8 = 256$
Encoded entities	20 amino acids	36 calculator buttons
Full seq. example	Gly-Arg-Gly-Asp-Ser	NINE, EULER, SINH, TAN, PLUS
Short seq.	GRGDS	$\mathrm{w1ig6}_{36}$
De-serialization	protein folding	RPN seq. evaluation
Results	Biological function	Mathematical function
Example result	Cell adhesion	$9 + \mathrm{tg}\,\sinh \mathrm{e}$
Value	Morphogenesis etc.	12.1234764057


log
pow
log
e
pow
log
e
log
log
pow
e
log
e
log
e
e
e
pow
e
pow
e
e
pow
e
e
e


log
pow
e
pow
-1
log
log
e
log
log
-1
pow
e
log
e
log
e
e
e
1/e
pow
e
pow
1/2
e
e
pow
e
e^2
e
e