A RIGOROUS ALGORITHMIC INVESTIGATION FOR CONSTANT APPROXIMATIONS VIA EXPRESSION SEARCH

SURYANSH S. SHEKHAWAT

Abstract. This paper presents a computational investigation that searches for arithmetic expressions approximating a given target constant using a finite set of base expressions and allowed operators. In our approach, the search is organized by a notion of **DEPTH** (i.e., the number of operations applied) and is limited to a maximum depth (typically 10 or 11) due to computational constraints. We describe the algorithm rigorously, introduce precise definitions and notation, present pseudocode in the algorithmic style, and discuss sample solutions—including approximations for e, φ (the golden ratio), and π with their respective depths and computed absolute errors. We also include an example run showing the number of candidate expressions generated at each depth (with depth 10 evaluating approximately 2.5 million sequences). Finally, we discuss the inherent limitations, including memory (RAM) requirements for deeper searches, and how increased computational power could extend the search depth.

Introduction

The goal of this investigation is to approximate a target constant (for example, $\varphi \approx 1.6180339887$) by constructing arithmetic expressions from a single base constant and a fixed set of allowed operations. In our implementation, we start from a **BASE EXPRESSION** (in our case, $\sqrt{2}$) and combine it using binary operators (such as $+$, **-**, $\times$, $\div$, and $\wedge$) and unary operators (such as **neg**, $\sqrt{\cdot}$, **recip**, and **fact**).

Definition 1 (Depth)**.** *The **DEPTH** of an expression is defined as the number of operator applications required to construct the expression from the base constant. For example, an expression constructed by one operation has depth 2 (the base plus one operator), while recursive combinations lead to higher depths.*

Definition 2 (Allowed Operators)**.** *The allowed operations are split into:*

- ***Binary Operators:*** $\{+, -, \times, \div, \wedge\}$.
- ***Unary Operators:*** $\{neg, \sqrt{\cdot}, recip, fact\}$.

These operators are applied to previously generated expressions.

Definition 3 (Expression Search)**.** *The process of **EXPRESSION SEARCH** recursively combines expressions from lower depths using allowed operators to generate new candidate expressions. At each depth level, the algorithm evaluates the absolute error between the candidate's value and the target constant.*

Date: February 23, 2025.

Algorithm Description

The algorithm performs a parallelized search to generate and evaluate candidate expressions. Below is the pseudocode in the algorithmic style using the `algorithm` and `algorithmic` packages. Key statements are highlighted in **BOLD**.

Algorithm 1 SEARCH FOR EXPRESSION APPROXIMATIONS

INITIALIZE the target constant T, maximum depth $D_{\max}$, time limit T_{limit}, and tolerance τ.
GENERATE BASE EXPRESSIONS: Create a set E_1 containing the base expression(s) (e.g., $\sqrt{2}$).
SET best error $\epsilon_{\text{best}} \leftarrow \infty$, best expression $E_{\text{best}} \leftarrow$ NULL.
for each depth $d = 2$ to $D_{\max}$ **do**
  IF elapsed time $> T_{\text{limit}}$, **THEN** TERMINATE SEARCH.
  INITIALIZE an empty set E_d.
  FOR all valid combinations of expressions from previous depths that sum to $d - 1$:
    FOR each allowed binary operator B:
      COMPUTE $v \leftarrow B(v_1, v_2)$ using expressions from lower depths.
      IF v is a valid number, **THEN** add the new expression to E_d.
    FOR each allowed unary operator U applied to an expression from depth $d - 1$:
      COMPUTE $v \leftarrow U(v_1)$.
      IF v is valid, **THEN** add the new expression to E_d.
  EVALUATE every candidate in E_d: Compute error $\epsilon = |v - T|$.
  IF any candidate has $\epsilon < \epsilon_{\text{best}}$, **THEN** update E_{best}, ϵ_{best}, and RECORD depth d.
  OUTPUT the number of expressions generated at depth d.
  IF $\epsilon_{\text{best}} < \tau$, **THEN** TERMINATE SEARCH AND RETURN E_{best}.
end for
RETURN the best approximation found, its evaluated value, and the associated error.

Results and Discussion

The algorithm was run with a maximum depth of 10. Several candidate expressions approximating the target constants were discovered, and the number of candidate expressions generated at each depth was recorded. In total, approximately 2.5 million expressions were evaluated at depth 10. Below are selected candidate approximations along with their computed absolute errors (to 15-decimal precision):

$$e \sim \frac{\sqrt{\pi^{\pi} + \pi^{\pi}}}{\pi} \quad (\textbf{Depth } d = 10;\ \epsilon \approx 5.10915427902570 \times 10^{-5})$$

$$\phi \sim \sqrt{2}^{\sqrt{\sqrt{2}}^{\sqrt{\sqrt{2}} + \frac{1}{\sqrt{2}}}} \quad (\textbf{Depth } d = 10;\ \epsilon \approx 3.06691394710893 \times 10^{-4})$$

$$\pi \sim \left(\sqrt{\sqrt{\phi}\,(\phi + \phi)}\right)^{\phi} \quad (\textbf{Depth } d = 9;\ \epsilon \approx 2.44848846482600 \times 10^{-6})$$

$$e \sim \frac{1}{\sqrt{2}^{\sqrt{2}^{\frac{1}{\sqrt{2}}}} - \sqrt{\sqrt{2}}} \quad (\textbf{Depth } d = 10;\ \epsilon \approx 2.18525236984390 \times 10^{-5})$$

$$\phi \sim \frac{\pi}{\pi^{-\pi^{-1}}} - \frac{1}{\sqrt{\pi}} \quad (\textbf{Depth not recorded})$$

Limitations and Computational Considerations

The current investigation is limited by the maximum depth ($d \leq 10$ or 11) that can be feasibly searched. Mathematically, if $N(d)$ denotes the number of candidate expressions at depth d, then the recursive construction implies a recurrence of the form

$$N(d) \approx \sum_{i=1}^{d-1} N(i)\, N(d-i),$$

which exhibits super-exponential growth. Approximating

$$N(d) \sim C \cdot n^{d},$$

for some constants $C > 0$ and $n > 1$, even a small increase in d leads to a dramatic increase in the search space.

For instance, at depth 10 the search generated roughly 2.5 million expressions. Assuming each candidate expression requires about 200 bytes of memory, the depth 10 search consumes roughly 8 GB of RAM. Increasing the depth to $d = 11$ may approximately double the memory usage to around 16 GB, while a search at $d = 12$ could require over 32 GB of RAM. Thus, even with a doubling of computational resources, only an incremental increase in d may be feasible. Alternative strategies, such as more aggressive pruning or heuristic-guided search, might be necessary to extend the search depth without incurring prohibitive hardware demands.

Example Run for ϕ

Below is an example run of the code, presented in aligned LaTeX equations for clarity:

Depth 1: $\sqrt{2} = 1.414213562373$ (error 0.203820426377)

Depth 2: 3 expressions generated.

Depth 3: $(\sqrt{2})^{\sqrt{2}} = 1.632526919438$ (error 0.014492930688)

Depth 3: 13 expressions generated.

Depth 4: 67 expressions generated.

Depth 5: $\sqrt{\sqrt{2} + \sqrt{\sqrt{2}}} = 1.613511908037$ (error 0.004522080713)

Depth 5: 358 expressions generated.

Depth 6: 2036 expressions generated.

Depth 7: $\sqrt{\sqrt{2} + \frac{\sqrt{2}}{\sqrt{\sqrt{2}}}} = 1.613511908037$ (error 0.004522080713)

Depth 7: 11864 expressions generated.

Depth 8: $\sqrt{\frac{\sqrt{2}}{\sqrt{2}} + (\sqrt{2})^{\sqrt{2}}} = 1.622506369614$ (error 0.004472380864)

Depth 8: $\frac{1}{\sqrt{\sqrt{\sqrt{2}} - \frac{1}{\sqrt{2}}}} = 1.615003273073$ (error 0.003030715676)

Depth 8: 71140 expressions generated.

Depth 9: $\sqrt{2} \times (\sqrt{2})^{\frac{1}{\sqrt{2}+\sqrt{\sqrt{2}}}} = 1.615583163058$ (error 0.002450825692)

Depth 9: $\sqrt{2} + \left((\sqrt{2} - \sqrt{\sqrt{\sqrt{2}}})^{\sqrt{2}}\right) = 1.617097760584$ (error 0.000936228166)

Depth 9: $\sqrt{\sqrt{2} + \sqrt{\sqrt{\sqrt{2} + \frac{1}{\sqrt{2}}}}} = 1.618968422610$ (error 0.000934433860)

Depth 9: 434136 expressions generated.

Depth 10: $\sqrt{2} \times \frac{\sqrt{2}}{(\sqrt{2})^{\left((\sqrt{2})^{-\sqrt{2}}\right)}} = 1.617455854164$ (error 0.000578134586)

Depth 10: $(\sqrt{2})^{\left((\sqrt{\sqrt{2}})^{\sqrt{\sqrt{2}}+\frac{1}{\sqrt{2}}}\right)} = 1.618340680145$ (error 0.000306691395)

Depth 10: $\sqrt{2} + \frac{1}{\sqrt{\left((\sqrt{2} \times (\sqrt{2} + \sqrt{2}))\right)!}} = 1.618337707605$ (error 0.000303718855)

Depth 10: $\sqrt{\sqrt{2}} + \frac{1}{\sqrt{2} + \frac{1}{\sqrt{\sqrt{\sqrt{2}}}}} = 1.618167499313$ (error 0.000133510563)

Depth 10: $\left((\sqrt{2} \times \sqrt{2})\right)^{\sqrt{\sqrt{2}-\frac{1}{\sqrt{2}}}} = 1.618137779249$ (error 0.000103790499)

Depth 10: 2.5 million expressions generated.

Conclusion

We have presented a rigorous algorithmic investigation for approximating constants by generating arithmetic expressions from a base constant using allowed operations. Our method—organized by a notion of **DEPTH**—produces candidate expressions which are then evaluated against a target constant. Sample solutions for e, ϕ, and π have been discussed with absolute errors computed to 15-decimal precision. The example run demonstrates that, for instance, at depth 10 the algorithm evaluates approximately 2.5 million candidate expressions. We have also detailed the dramatic increase in memory requirements for deeper searches (with depth 10 consuming roughly 8 GB of RAM, and projections indicating that depth 11 and 12 may require around 16 GB and 32 GB, respectively). Future work may focus on improved search strategies and heuristics to enable deeper exploration of the expression space without incurring prohibitive hardware demands.

References

(1) Borwein, P., Borwein, J., & Plouffe, S. *The Inverse Symbolic Calculator.* Canadian Centre for Experimental and Constructive Mathematics, 1995.[**?**]

(2) Raayoni, G., et al. *The Ramanujan Machine: Automatically Generated Conjectures on Fundamental Constants. Nature*, vol. 590, 2021, pp. 67–73.[**?**]

(3) Chudnovsky, D. & Chudnovsky, G. *Approximations and the Computation of Pi. Computers in Mathematics*, 1990.[**?**]

(4) Beckmann, P. *A History of* π. St. Martin's Press, 1971.[**?**]

(5) Wikipedia. *Transcendental number.* Retrieved from `https://en.wikipedia.org/wiki/Transcendental_number`.[**?**]

(6) Wikipedia. *Golden ratio.* Retrieved from `https://en.wikipedia.org/wiki/Golden_ratio`.[**?**]

(7) Bailey, D., Borwein, J., & Plouffe, S. *The BBP Algorithm for Pi. American Mathematical Monthly*, 2007.[**?**]

(8) ProofWiki. $e^{\pi\sqrt{163}}$ *almost an integer – historical note.* Retrieved from `https://proofwiki.org/wiki/Almost_Integer`.[**?**]

(9) Conway, J. H. & Guy, R. K. *The Book of Numbers.* Springer, 1996.[**?**]

Email address: `shekhawatsuryansh@gmail.com`