# Genetic Improvement of Data for Maths Functions[*]

William B. Langdon
W.Langdon@cs.ucl.ac.uk
Department of Computer Science, UCL
London, UK

Oliver Krauss
oliver.krauss@fh-hagenberg.at
University of Applied Sciences Upper Austria
Hagenberg, Austria

## ABSTRACT

Genetic Improvement (GI) can be used to give better quality software and to create new functionality.

We show that GI can evolve the PowerPC open source GNU C runtime library square root function into cube root, binary logarithm log2 and reciprocal square root. The GI cbrt is competitive in run-time performance and our inverted square root $x^{-\frac{1}{2}}$ is far more accurate than the approximation used in the Quake video game. We use CMA-ES to adapt constants in a Newton-Raphson table, originally from glibc's sqrt, for other double precision mathematics functions. Such automatically customised math libraries might be used for mobile or low resource, IoT, mote, smart dust, bespoke cyber-physical systems.

Evolutionary Computing (EC) can be used to not only adapt source code but also data, such as numerical constants, and could enable a new way to conduct software data maintenance. This is an exciting opportunity for the GECCO and optimisation communities.

## CCS CONCEPTS

• **Software and its engineering → Search-based software engineering**.

## KEYWORDS

evolutionary computing, software engineering, search based software engineering, SBSE, computer-aided software engineering, CASE, autorepair, software maintenance of empirical constants, data transplantation, glibc, vector normalisation, Newton's method

**ACM Reference Format:**
William B. Langdon and Oliver Krauss. 2021. Genetic Improvement of Data for Maths Functions. In *2021 Genetic and Evolutionary Computation Conference Companion (GECCO '21 Companion), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3449726.3462730

## 1 INTRODUCTION

We hope to alert the optimisation community to a new opportunity to apply their existing tools and techniques to an exciting and important potential application: optimising and maintaining existing software.

[*]We summarise our ACM TELO article [8]. See https://dlnext.acm.org/journal/telo.

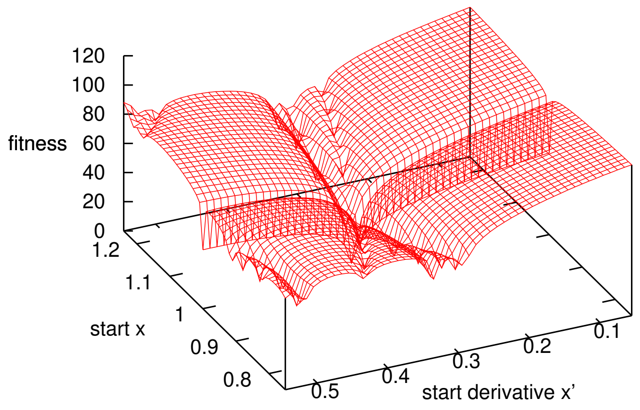

**Figure 1: First fitness landscape for GI cube root (goal is to minimise). The global optima is at 1.0,0.333,0. This fitness guides CMA-ES to find the first pair of initial values of $x$ and the derivative of $x$ for the Newton-Raphson solution of $\sqrt[3]{\ }$.**

**Table 1: GI accuracy [8, 9] and total time time for CMA-ES**

| Start | Evolved | | accuracy | seconds |
|---|---|---|---|---|
| sqrt → | cbrt() | $\sqrt[3]{x}$ | double precision, i.e. $\leq 6.7\ 10^{-16}$ | 270 |
| sqrt → | log2() | $\log_2 x$ | double precision, i.e. $\leq 2.2\ 10^{-16}$ | 6 |
| sqrt → | invsqrt() | $x^{-1/2}$ | double precision, i.e. $\leq 2.2\ 10^{-16}$ | 6 |
| sqrt → | reciprocol | $x^{-1}$ | double precision, i.e. $\leq 2.2\ 10^{-16}$ | 6 |

"Genetic Improvement of Data for Maths Functions" [8] is the first journal publication in which we optimise numbers in source code to generate new functionality. These mathematical functions use a table containing start values for the Newton-Raphson method. Newton's method is often used to reduce runtime of complex arithmetic operations. We evolve values in an existing table to give new functions. This method is not just for mathematical functions: we have also used genetic improvement (GI) to improve the answers given by a program by tuning 50 000 parameters buried in its source code [6]. We used CMA-ES [3] on 1024 floating point numbers within glibc's implementation of square root for the PowerPC and manually adapted the function and derivative to give a C implementation of cube root. Not only does the new $\sqrt[3]{\ }$ code give double precision accuracy but, subsequently we were able to show, it is competitive with standard implementations for C++ and indeed it is faster than Java [4]. The journal article also extends this to other double precision mathematical functions (see Table 1).

Glibc is the GNU C runtime library. It is a key component of the Free Software Movement's support for the widely used C programming language and core to GNU and GNU/Linux systems. Note glibc does not support cube root.

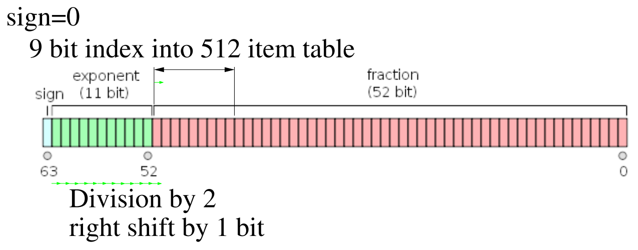

**Figure 2: Glibc sqrt divides double precision float exponent by two with right shift and extracts nine bit index (0..511).**

Almost all software contains numbers, e.g. the glibc source code contains more than a million integers. These tend to be set when the code is written. Many are indeed fixed, but some of them represent tunable parameters whose values are often chosen by the initial software developer before first contact with the users, and never subsequently updated. The Evolutionary Computation (EC) community has the opportunity to develop new tools for automatic update of source code parameters. Uses include: 1) first initial tuning, and also retuning in response to 2) increased user load, 3) changing user expectations or user behaviour and 4) to support new hardware. (E.g. increased/decreased RAM, GPU, multi-core, mobile computing, internet-of-things (IOT) and smart dust or mote computing.)

## 2 SQUARE ROOT ⇒ CUBE ROOT VIA DATA

If there is no hardware support for square root, glibc uses the Newton-Raphson method to iteratively solve $x^2 - a = 0$ (i.e. to find $x = a^{\frac{1}{2}}$) [13]. To get double precision accuracy quickly, Newton-Raphson needs to start with a good approximation of $\sqrt{\ }$. To achieve this, glibc divides the double precision number range into 512 bins (Figure 2) and stores good start values for Newton-Raphson in each.

To convert the open source glibc C code to support cube root, we need a little manual code manipulation, e.g. to remove the trap for negative values of $x$ and to divide the exponent (11 green bits in Figure 2) by 3 rather than 2. However our article shows it is easy to use evolutionary computation to mutate the "fixed" sqrt numbers into numbers for the new cube root function. The evolved $\sqrt[3]{\ }$ function has been tested many thousands of times to show it does indeed give the true double precision cube root.

The CMA-ES fitness function (Figure 1) uses the values in the table to try and calculate the cube root. It cubes the calculated value. This is compared to the original number and the absolute difference becomes the fitness value guiding CMA-ES. (See the journal paper [8] for details.)

## 3 CONCLUSIONS

Genetic Improvement (GI) [15] has been widely used to automatically fix bugs [12], speed up programs [7], reduce energy consumption [2] and transplant code [14] [1]. In our ACM TELO paper [8], we show GI can be applied, not just to code, but also to data. Giving functionally improved software primarily by evolving floating point numbers in its source code.

The method gives in all cases correct double precision answers. It was able to rapidly evolve hundreds of C source code constants. (See Table 1.)

In a world addicted to software, software maintenance is a huge cost. Most research concentrates on maintaining source code and, as yet, there is little research on maintaining numbers within programs. Therefore both automated data maintenance and data transplantation could be vital new areas for optimisation research.

## Acknowledgements

We are grateful for the assistance of Justyna Petke and Roy Longbottom and thank Simon Tatham for PuTTY. Funded by EPSRC GGGP and InfoTestSS grants EP/M025853/1 EP/P005888/1.

*Code.* Our code is available via https://github.com/oliver-krauss/Replication_GI_Division_Free_Division or http://www.cs.ucl.ac.uk/staff/W.Langdon/ftp/gp-code/gi_cbrt.tar.gz

## REFERENCES

[1] Earl T. Barr et al. 2015. Automated Software Transplantation. In *International Symposium on Software Testing and Analysis, ISSTA 2015*, Tao Xie and Michal Young (Eds.). ACM, Baltimore, Maryland, USA, 257–269. http://dx.doi.org/10.1145/2771783.2771796 ACM SIGSOFT Distinguished Paper Award.

[2] Bobby R. Bruce et al. 2016. Deep Parameter Optimisation for Face Detection Using the Viola-Jones Algorithm in OpenCV. In *Proceedings of the 8th International Symposium on Search Based Software Engineering, SSBSE 2016 (LNCS, Vol. 9962)*, Federica Sarro and Kalyanmoy Deb (Eds.). Springer, Raleigh, North Carolina, USA, 238–243. http://dx.doi.org/10.1007/978-3-319-47106-8_18

[3] Nikolaus Hansen and Andreas Ostermeier. 2001. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation* 9, 2 (Summer 2001), 159–195. http://dx.doi.org/10.1162/106365601750190398

[4] Oliver Krauss and W. B. Langdon. 2020. Automatically Evolving Lookup Tables for Function Approximation. In *EuroGP 2020: Proceedings of the 23rd European Conference on Genetic Programming (LNCS, Vol. 12101)*, Ting Hu et al. (Eds.). Springer Verlag, Seville, Spain, 84–100. http://dx.doi.org/10.1007/978-3-030-44094-7_6

[5] W. B. Langdon. 2019. Genetic Improvement of Data gives double precision invsqrt. In *7th edition of GI @ GECCO 2019*, Brad Alexander et al. (Eds.). ACM, Prague, Czech Republic, 1709–1714. http://dx.doi.org/10.1145/3319619.3326800

[6] William B. Langdon et al. 2018. Evolving better RNAfold structure prediction. In *EuroGP 2018: Proceedings of the 21st European Conference on Genetic Programming (LNCS, Vol. 10781)*, Mauro Castelli et al. (Eds.). Springer Verlag, Parma, Italy, 220–236. http://dx.doi.org/10.1007/978-3-319-77553-1_14

[7] William B. Langdon and Mark Harman. 2015. Optimising Existing Software with Genetic Programming. *IEEE Transactions on Evolutionary Computation* 19, 1 (Feb. 2015), 118–135. http://dx.doi.org/10.1109/TEVC.2013.2281544

[8] William B. Langdon and Oliver Krauss. [n.d.]. Genetic Improvement of Data for Maths Functions. *ACM Transactions on Evolutionary Learning and Optimization* ([n. d.]). http://www.cs.ucl.ac.uk/staff/W.Langdon/ftp/papers/Langdon_TELO.pdf to appear.

[9] William B. Langdon and Oliver Krauss. 2020. Evolving sqrt into 1/x via Software Data Maintenance. In *9th edition of GI @ GECCO 2020*, Brad Alexander et al. (Eds.). ACM, Internet, 1928–1936. http://dx.doi.org/10.1145/3377929.3398110

[10] William B. Langdon and Justyna Petke. 2018. Evolving Better Software Parameters. In *SSBSE 2018 Hot off the Press Track (LNCS, Vol. 11036)*, Thelma Elita Colanzi and Phil McMinn (Eds.). Springer, Montpellier, France, 363–369. http://dx.doi.org/10.1007/978-3-319-99241-9_22

[11] W. B. Langdon and Justyna Petke. 2019. Genetic Improvement of Data gives Binary Logarithm from sqrt. In *GECCO '19: Proceedings of the Genetic and Evolutionary Computation Conference Companion*, Richard Allmendinger et al. (Eds.). ACM, Prague, Czech Republic, 413–414. http://dx.doi.org/10.1145/3319619.3321954

[12] Claire Le Goues et al. 2019. Automated Program Repair. *Commun. ACM* 62, 12 (Dec. 2019), 56–65. http://dx.doi.org/10.1145/3318162

[13] P. W. Markstein. 1990. Computation of elementary functions on the IBM RISC System/6000 processor. *IBM Journal of Research and Development* 34, 1 (Jan 1990), 111–119. http://dx.doi.org/10.1147/rd.341.0111

[14] Justyna Petke et al. 2014. Using Genetic Improvement and Code Transplants to Specialise a C++ Program to a Problem Class. In *17th European Conference on Genetic Programming (LNCS, Vol. 8599)*, Miguel Nicolau et al. (Eds.). Springer, Granada, Spain, 137–149. http://dx.doi.org/10.1007/978-3-662-44303-3_12

[15] Justyna Petke et al. 2018. Genetic Improvement of Software: a Comprehensive Survey. *IEEE Transactions on Evolutionary Computation* 22, 3 (June 2018), 415–432. http://dx.doi.org/doi:10.1109/TEVC.2017.2693219