Mathematicians' Values and Norms Related to Algorithms

Peter Sassman
Northern Illinois University

Rachel Rupnow
Northern Illinois University

Alastair Fletcher
Northern Illinois University

Algorithms play increasingly important roles in mathematics as they facilitate solving complex problems both in teaching and research contexts. Nevertheless, limited research has examined mathematicians' views of algorithms or the broader goals of the mathematical community that relate to algorithms or work therewith. Based on interviews with nine discrete or computational mathematicians, we highlight three values upheld by a variety of norms related to algorithms. Of note, some of these values and norms appear to be in tension.

Keywords: norms, values, algorithms, mathematical culture

The mathematical community holds many views about what it means to do mathematics. While research has previously examined individual mathematicians' beliefs about what it means to do and teach mathematics (e.g., Rupnow, 2023; Woods & Weber, 2020), recently new attention has been paid to the values and norms of the mathematical community (Dawkins & Weber, 2017; Rupnow & Randazzo, 2023; Vroom et al., 2024). These works have focused on mathematicians' perceptions of how particular tools like proof (Dawkins & Weber, 2017) and definitions (Rupnow & Randazzo, 2023; Vroom et al., 2024) permit particular values to be upheld via norms of practice in the mathematical community. To build on this work, we examine a third tool, algorithms, which arguably is more central to K-14 mathematics instruction, though it remains important for research in discrete mathematics, applied mathematics, and various applications. Moreover, deductive logic plays a key role in both proof and algorithm creation, suggesting the interwoven nature of these practices. We thus address two research questions:

1. Which values and norms are revealed through mathematicians' purposes for creating and using algorithms in research?
2. Which values and norms are revealed through mathematicians' characterization of instruction around algorithms?

Background Literature

Due to the ubiquity of technology of modern life, computing and computational thinking have received increased attention in curricula. Computational thinking originally centered on problem solving in computer science contexts (e.g., Wing, 2006) but has since been expanded to consider other mathematics and science contexts in which computation is relevant. Weintrop et al. (2016) created a taxonomy of computational thinking in mathematics and science practices with four categories: data practices, modeling and simulation practices, computational problem solving practices, and systems thinking practices. Within mathematics, Lockwood et al. (2019) defined computing as a mathematical disciplinary practice, specifically "the practice of using tools to perform mathematical calculations or to develop or implement algorithms in order to accomplish a mathematical goal" (p. 4). Moreover, they highlighted many ways that interviewed mathematicians use computing, including for facilitating experimentation, testing conjectures, approximating, and visualizing. While we note computing and computational thinking are not exclusively focused on algorithms, algorithms are a component of these related constructs.

While many definitions of algorithm exist in the mathematics education literature, Fan and Bockhove (2014) combined many to state: "an algorithm is a fixed set of step-by-step procedures

for solving a (mathematics) problem" (p. 483). Of note, this definition suggests a procedural aspect, but leaves the relevant problem open-ended. Relatedly, algorithms have not always been viewed positively in mathematics education. Kamil and Dominick (1997) characterized algorithms as procedures that obscured conceptual understanding and argued against teaching arithmetic algorithms like borrowing and carrying for subtraction and addition, respectively. In contrast, Rasmussen et al. (2005) focused on the activity around the use of algorithms when characterizing 'algorithmatizing' as a mathematical practice. In particular, they characterized this activity in terms of how students learn to use and understand a generalizable procedure that is useful across varied contexts, and, in positioning this as a mathematical practice, attributed value to this activity. Nevertheless, limited work has attended to how mathematicians characterize algorithms or how they relate algorithms and algorithmatizing to the mathematical community's expectations about the goals of mathematical activity and how mathematics is done.

Theoretical Perspective

The Triadic Network of Justification (Laudan, 1984) is a theoretical framework that characterizes the conceptual connections among axiology, methodology, and factual claims in a community. Axiology relates to ethics and values, methodology relates to commonly accepted norms of practice, and factual claims relate to what is accepted as settled fact in a community. While this framework was originally designed to conceptualize the nature of paradigmatic changes in the scientific community, this framework has been applied in mathematics education to characterize values (axiology), norms (methodology), and theorems (factual claims) of the mathematical community with respect to proof (Dawkins & Weber, 2017) and definitions (Rupnow & Randazzo, 2023; Vroom et al., 2024). In this study, we build on Laudan (1984) in a similar way to conceptualize values and norms related to algorithms.

Methods

Mathematicians at R1 institutions in the United States who were listed on their departmental website as researchers of applied/computational mathematics or combinatorics (or a similar description) were emailed to find participants. The second author conducted 60-minute remote, screen-recorded interviews with nine such mathematicians. Five participants' research aligned more with applied/computational mathematics and four aligned more with combinatorics. All names are gender-neutral pseudonyms. Interview questions focused on participants' views of algorithms and their use for different populations in both research and teaching contexts.

Using MAXQDA 2020 software (VERBI Software, 2019), all three researchers initially open-coded two transcripts to create norm codes and value categories for algorithms in alignment with Laudan (1984). After discussing our open codes, we created a codebook of norms. We then re-coded at the paragraph level (i.e., came to agreement on the norms present in each paragraph of the transcript) and came to consensus on the first two interviews. Finally, two researchers individually coded each of the remaining seven interviews and then discussed and came to consensus on the norms present in each paragraph. This process aligns with codebook thematic analysis (Braun et al., 2019) in that we entered analysis with some idea of the types of codes we would use and desired to reach consensus on the norms in each transcript but did not enter analysis with specific norms (codes) or values (themes) in mind.

Results

After analyzing the data through open-coding, we obtained a variety of norms (codes, italicized) which we gathered into three values (themes): algorithms are useful for solving

problems, correctness of algorithms needs to be verified, and algorithms need to be clear to people and communicable to others. These values with supporting norms are shown in Table 1.

Table 1. Algorithm Values, Norms, and Frequencies.

Values	Norms	Frequency (n = 9)
Algorithms are useful for solving problems	Algorithms are a tool	9
	Problems/contexts drive the use/creation of algorithms	9
	Algorithms can be used to scaffold thinking/learning	6
	Students need support in writing/implementing algorithms	5
	Coding knowledge helpful for using/proving properties of algorithms	5
	Utility of implementation of algorithms is more important than proving correctness	4
	Using/creating algorithms drives research on convergence/stability properties	3
Correctness of algorithms needs to be verified	Deductive logic is important for algorithms	9
	Proving correctness of algorithms is important	8
	Students need support in proving aspects of algorithms	4
	Students need to develop intellectual need for proving aspects of algorithms	2
Algorithms need to be clear to people and communicable to others	Algorithms need to be understood (enough, depending on context)	9
	Algorithms should be presented/tested with examples	8
	Assumptions/clarity of an algorithm (to humans) is important	7
	People should know the limitations of algorithms	6
	Algorithms should be presented in pseudocode	3

Algorithm Values

Algorithms are useful for solving problems. This value looks at the utility of algorithms for accomplishing a variety of tasks. The first norm, *Algorithms are a tool* encompasses a variety of ways that algorithms are useful for accomplishing some other purpose, such as for solving a numerical problem, generating data or objects, or helping prove a theorem. We observed references to this norm by all nine participants. For example, Ashton described how algorithms could be used to generate useful objects: "[Algorithms] have several purposes. One is to construct mathematical objects or examples." Similarly, Emery stated that algorithms may be used to create an object that was known to exist but whose form was unknown: "For example, if a mathematical result proves the existence of some object or structure in a given context, then an algorithm may be devised to produce it." Foster, who works in applied mathematics, described one way algorithms can be used for proving: "A lot of the ways you prove existence of solutions is to use an algorithm. So the granddaddy one is the Contraction Mapping Principle, but there are lots of them." In particular, the Contraction Mapping Principle, also known as the Banach fixed point theorem, guarantees the existence and uniqueness of fixed points in a metric space and has varied applications such as finding solutions to differential equations and root finding.

Connected to the idea that algorithms can be used to solve problems, is the norm *Problems/contexts drive the use/creation of algorithms*, which was also coded for all participants. This norm highlights how the need to solve a problem in a specific context may promote the creation or use of an algorithm. For example, Gideon stated:

> Monte Carlo is anything that involves randomness, which includes applications in mathematics…And so the early pioneers in the Monte Carlo method were the math physicists who worked on the atomic bomb. And so the Monte Carlo simulations were going to give an indication of whether the damage was going to be astronomical or under control. And so that was one of the first users of the Monte Carlo sim[ulation]. But since then, the subject has exploded with millions of other applications.

Notice the need to examine random processes in a nuclear reaction led to the creation of new a new type of algorithm, which is now applicable in many other settings. While all but one participant described the idea that *proving correctness of algorithms is important* (discussed further below), four of the participants also stated that this proof may not be the most important aspect of some algorithms. The norm *Utility of implementation of algorithms is more important than proving correctness* specifies that the usefulness of an algorithm may be more important than definitive proof that it works as expected. For example, Foster said:

> Sometimes those are actually very hard problems to show that a certain algorithm, which seems to produce an answer, actually produces the answer you want it to produce. And it's not always the case it does.…I mean, there are algorithms in numerical analysis, for example, that I know of, where nobody knows that it produces the answer you want. It appears to, but there's no theorem that says, "Okay, you follow this algorithm, and you will get closer and closer to the answer you want."

Of note, even mathematicians will use algorithms without rigorous proofs of convergence or of other properties that would assure correctness of results.

The norm *Using/creating algorithms drives research on convergence/stability properties* spotlights that using or creating algorithms may lead to new research on proving properties of these algorithms. Three participants had statements aligned with this norm, such as Becker:

> One of our recent papers where we had an algorithm that was basically designed to describe how to find a stability region that's very complicated to get… Other authors had given condition[s] for determining the region, but they couldn't actually do the computations to find it because it involved a good bit of ergodic theory.…So we actually give the algorithm in the paper, but the proof of the result that the algorithm actually does that was the real meat of the paper and showing that we got the region that we claim we got, that this thing was actually doable.

Five participants discussed students needing support with algorithms as described with the norm *Students need support in writing/implementing algorithms*. For example, Harley said:

> In some ways, algorithms can be more concrete for students to understand in theory. But in practice, as I say, students will struggle with algorithms. Even though it's something that you can actually compute through steps, it's hard for them to grasp the whole thing and it gets very tedious if you actually want to work out the mechanics.

The final two norms look at how algorithms can also be helpful for learning, and types of knowledge that may be useful in working with algorithms. The norm *Algorithms can be used to scaffold thinking/learning* was coded in six respondents' interviews and describes how algorithms are helpful for scaffolding the learning process. For example, Devin stated:

> What the algorithm does, the algorithm is going to allow you to compartmentalize what is

> happening, what is it going to, how is it getting there, and where are you going. It helps you with creating that pattern. It helps you to create that sequence of understanding so that what you want to happen, there is a logical process to it, right?

Here, Devin described how algorithms can scaffold an approach to a problem by breaking it into smaller parts that can be more easily understood and connected to each other.

Finally, the norm *Coding knowledge is helpful for using/proving properties of algorithms* was part of four participants' responses. For example, Devin said:

> This is when our plan meets the real world….So this is when we use our software, right? So we translate our algorithm into a mathematical software, right? So I may use MATLAB or I may use Mathematica or something like that. And I may find these numerical [simulations] that approximate the clinical information.

Here Devin emphasized the importance of being able to translate a mathematical algorithm into computer code so it could be solved in a large-scale, real world application.

Correctness of algorithms needs to be verified. This value focuses on the importance of deductive logic and aspects of proof to algorithms. The first norm, *Deductive logic is important for algorithms*, was part of every participant's response. For this norm, participants described how algorithms can intersect with deductive logic. Ashton provided the following example:

> I want to construct a big data set....So without thinking about what's going to happen at each individual step, I'm going to think like, "First, I need a function that does this, and then the output of that function is going to go into another function that does this. And I need to range over this set of inputs or whatever," and you put it together kind of in this modular way.

Here, Ashton described building an algorithm, step by step, following a deductive process.

While some norms of the prior value especially centered utility, participants also said that it is important to prove correctness of algorithms, coded with the norm *Proving correctness of algorithms is important*, which was mentioned by 8 participants. Ira described this in the following quote: "A new algorithm has got to be analyzed for stability and accuracy before it's at all useful. I think that's just one of the essential steps in developing an algorithm.

Similar to the norm about needing to support students in writing algorithms, the norm *Students need support in proving aspects of algorithms* centers the proving aspects. Four participants were coded with this norm, such as Emery, who said:

> When students are in a mathematics course in graph theory, what they're having difficulty with is the proofs, generally. And running through an example subject to an algorithm is not so much where they find the difficulty but rather something that they use to help them with difficulties in other aspects of proof.

Notice that Emery viewed proving as generally more difficult for students than understanding algorithms, at least in their context of graph theory.

Although the previous norm noted that students need support in proving, two participants expanded on this idea in the norm *Students need to develop intellectual need for proving aspects of algorithms*. Becker described:

> …when [computer science students] started having to prove the correctness of the algorithms, they didn't like it at all which I was like, "This is a senior-level math class. Mathematicians prove things, you've got to—because I knew I had CS majors, I was like, You're going to have to prove things in here just to help you understand."

Here, Becker described computer science students as not being as interested in proving properties of algorithms, and thus a need to communicate the importance of proof to algorithms.

This speaks to the potential tension between communities in the role of proof in algorithms.

Algorithms need to be clear to people and communicable to others. The final value focuses on the clear presentation and communication of algorithms. The first norm supporting this value is *Algorithms need to be understood (enough, depending on context)*, which was part of every participant's interview. Gideon illustrates this norm in the following:

> In a calculus course, the best you can hope for is illustrate an algorithm with some pictures such as gradient descent or tangent slopes or splines. And so the picture is often the link between the theory on one side and the program on the other side.

Here, Gideon described students as unlikely to fully understand an algorithm, but with the aid of illustrations, able to understand it enough for the course. In research, sometimes mathematicians could also limit which parts of an algorithm they understand, as Devin described:

> Anytime I am into a system, I try to break it because when you figure out a problem, then you can figure out why is this a problem. And I could create more algorithms from understanding that problem. And that's how you get an algorithm from an algorithm because there [is] embeddedness in things, right? And so why was that embedded? And that's how you unfold the different models. What I'm saying is someone else may create a model which you have no idea why they created that model.

Here Devin described the importance of understanding why aspects of an algorithm work while acknowledging that not all details of the design process need to be known. In contrast, Cooper set a high standard for creating new algorithms in mathematical research:

> A colleague of mine [is] designing spectral methods that give better results for certain stiff differential equations. That's where the ball game's totally different because that's where you really have to analyze how does my code [do] what it does, why does it do it? How is it superior to what the status quo is? And is it only superior for these equations, or is it really something that could potentially be the next upgrade for MATLAB or something like that?

Cooper noted that when generating a new algorithm, it is important to establish whether and how the new algorithm is better, as well as determine some sense of the contexts in which it is better.

All but one participant had responses coded with the norm *Algorithms should be presented/tested with examples*. Ira explained this as follows:

> Oh, you can never build a reliable code without testing it on a problem where you know the answer. This is something I drum into my students. We need analytical solutions as benchmarks for the code that we're developing in order to be able to solve systems where we don't know what the answer is.

Here, Ira described the need for testing an algorithm with examples and known answers, so it may be used in cases where the answer is not known. Others highlighted the utility of examples for understanding an algorithm, such as Ashton:

> If something's just described verbally in a paragraph, that's always a real pain. And lots of mathematicians who don't do so much computation will describe things just like in this big block of a paragraph or two that's just impossible to read. So the best thing is if you put it in pseudocode, and then if the authors or if wherever you're getting the algorithm from hasn't provided an example, then you start with the simplest example you could possibly come up with and then just slowly ratchet up the level of complexity until … what should happen as you do that is that each step of the algorithm becomes clear why it's needed. So sometimes you'll have certain loops in an algorithm or certain branches that don't kick in unless you encounter a certain situation …And so for a while,

> you'll be like, "I don't know why this part's here," and you just need to find an example where it triggers it. And a good writer will give you those examples.

Here Ashton noted that good writers assist readers by providing examples that illustrate why all parts of the algorithm are useful. This segment also was coded with the norm *Algorithms should be presented in pseudocode*, which was coded in three participants' interviews, because it notes that pseudocode is easier to translate to a working algorithm in the computer language of one's choice (or to understand for oneself) than a large block of text.

Another norm, the *Assumptions/clarity of an algorithm (to humans) is important*, was observed in seven participants' interviews. Becker for example, said:

> So you have to know the constraints upon the problem itself for which this will actually give you the solution because it's generally not always going to give you all possible solutions.…So it's the same thing with algorithms. You have to understand the conditions upon which classes of problems this algorithm will actually solve.

Here Becker described the importance of looking at the assumptions of an algorithm, to make sure your system satisfies the given conditions for which the algorithm works.

The final norm is *People should know the limitations of algorithms*, which was coded in six participants' interviews. Ashton illustrates this norm with the following:

> So sometimes people will just be like, oh, okay, great, we can solve this with this algorithm. And they don't think about the fact that the algorithm doesn't see the quality of the data. And oftentimes, the algorithms that are used in science and public policy and stuff really are biased by not just the data, but by the weights that are chosen, like, you set these parameters.

Ashton noted that the quality of the outputs of an algorithm depends on the quality of the inputs as well as the weights given to various criteria, all of which can fundamentally impact the validity of results.

Discussion

This study examined nine mathematicians' norms and values related to algorithms in the contexts of research and teaching. We observed three values: algorithms are useful for solving problems, the correctness of algorithms needs to be verified, and algorithms need to be clear to people and communicable to others. Throughout the interviews, the utility of algorithms for a variety of purposes was very clear. Many uses were highlighted including using algorithms to prove or test conjectures, create examples, or find specific solutions. Participants also expected algorithms to produce correct outcomes, both based on their deductive structure and additional work focused on providing proof that algorithms have desirable properties. Clarity and communication were also viewed as important aspects of algorithms, though participants did vary in how much understanding or clarity is deemed sufficient, especially depending on context.

However, unlike prior research examining mathematicians' values and norms related to proofs and definitions (Dawkins & Weber, 2017; Rupnow & Randazzo, 2023), we saw some disagreement on the importance of some values. While some participants centered the utility of algorithms as most important, with some suggesting proof was desirable but not necessary (especially for solving real-life problems), others believed proof of accuracy or particular properties was paramount. This may in part stem from mathematicians viewing "algorithms" as encompassing different things, based on their mathematical training, common uses for algorithms in their work, and the expectations of the stakeholders they serve. Nevertheless, more research is needed to better understand mathematicians' values related to computing or algorithmatizing and how they interrelate with their view of other mathematical practices.

References

Braun, V., Clarke, V., Hayfield, N., & Terry, G. (2019). Thematic analysis. In Liamputtong P. (Ed.) *Handbook of Research Methods in Health Social Sciences*. Springer: Singapore. https://doi.org/10.1007/978-981-10-5251-4_103

Dawkins, P.C. & Weber, K. (2017). Values and norms of proof for mathematicians and students. *Educational Studies in Mathematics, 95*(2), 123–142.

Fan, L., & Bockhove, C. (2014). Rethinking the role of algorithms in school mathematics: A conceptual model with focus on cognitive development. *ZDM: Mathematics Education, 46*, 481–492. https://doi.org/10.1007/s11858-014-0590-2

Kamil, C., & Dominick, A. (1997). To teach or not to teach algorithms. *Journal of Mathematical Behavior, 16*(1), 51–61. https://doi.org/10.1016/S0732-3123(97)90007-9

Laudan, L. (1984). *Science and Values: The Aims of Science and their Role in Scientific Debate*. Los Angeles: University of California Press.

Lockwood, E., DeJarnette, A. F., & Thomas, M. (2019). Computing as a mathematical disciplinary practice. *Journal of Mathematical Behavior, 54*, Article 100688. https://doi.org/10.1016/j.jmathb.2019.01.004

Rasmussen, C., Zandieh, M., King, K., & Teppo, A. (2005). Advancing mathematical activity: A Practice-Oriented View of Advanced Mathematical Thinking. *Mathematical Thinking and Learning, 7*(1), 51–73. https://doi.org/10.1207/s15327833mtl0701_4

Rupnow, R. (2023). Mathematicians' beliefs, instruction, and students' beliefs: How related are they? *International Journal of Mathematical Education in Science and Technology*, *51*(10), 2147–2175. https://doi.org/10.1080/0020739X.2021.1998684

Rupnow, R., & Randazzo, B. (2023). Norms of mathematical definitions: Imposing constraints, permitting choice, or both? *Educational Studies in Mathematics, 114*(2), 297–314. https://doi.org/10.1007/s10649-023-10227-y

VERBI Software (2019). MAXQDA 2020 [computer software]. Berlin, Germany: VERBI Software. Available from maxqda.com.

Vroom, K., Alzaga Elizondo, T., Barbosa, J.S., & Strand, S., II. (2024). Teaching practices that support revising definition drafts to adhere to mathematical norms. *Educational Studies in Mathematics*, *117*, 285–302. https://doi.org/10.1007/s10649-024-10331-7

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, *25*, 127–147. https://doi.org/10.1007/s10956-015-9581-5

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.

Woods, C., & Weber, K. (2020). The relationship between mathematicians' pedagogical goals, orientations, and common teaching practices in advanced mathematics. *The Journal of Mathematical Behavior, 59,* Article 100792.