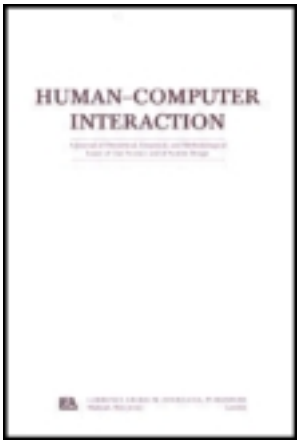
This article was downloaded by: [University of Manitoba Libraries]
On: 21 March 2013, At: 08:44
Publisher: Taylor & Francis
Informa Ltd Registered in England and Wales Registered Number: 1072954 Registered office: Mortimer House, 37-41 Mortimer Street, London W1T 3JH, UK



Human-Computer Interaction
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/hhci20

Challenges and Opportunities for Mathematics Software in Expert Problem Solving

Andrea Bunt [a], Michael Terry [b] & Edward Lank [b]
[a] University of Manitoba, Canada
[b] University of Waterloo, Canada

Accepted author version posted online: 06 Jun 2012.Version of record first published: 11 Mar 2013.

To cite this article: Andrea Bunt , Michael Terry & Edward Lank (2013): Challenges and Opportunities for Mathematics Software in Expert Problem Solving, Human-Computer Interaction, 28:3, 222-264

To link to this article: http://dx.doi.org/10.1080/07370024.2012.697020

PLEASE SCROLL DOWN FOR ARTICLE

Full terms and conditions of use: http://www.tandfonline.com/page/terms-and-conditions

This article may be used for research, teaching, and private study purposes. Any substantial or systematic reproduction, redistribution, reselling, loan, sub-licensing, systematic supply, or distribution in any form to anyone is expressly forbidden.

The publisher does not give any warranty express or implied or make any representation that the contents will be complete or accurate or up to date. The accuracy of any instructions, formulae, and drug doses should be independently verified with primary sources. The publisher shall not be liable for any loss, actions, claims, proceedings, demand, or costs or damages whatsoever or howsoever caused arising directly or indirectly in connection with or arising out of the use of this material.

HUMAN–COMPUTER INTERACTION, 2013, Volume 28, pp. 222–264
Copyright © 2013 Taylor & Francis Group, LLC
ISSN: 0737-0024 print / 1532-7051 online
DOI: 10.1080/07370024.2012.697020

Taylor & Francis
Taylor & Francis Group

Challenges and Opportunities for Mathematics Software in Expert Problem Solving

Andrea Bunt,[1] Michael Terry,[2] and Edward Lank[2]
[1]*University of Manitoba, Canada*
[2]*University of Waterloo, Canada*

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

Computer Algebra Systems and matrix-based mathematics packages provide sophisticated functionality to assist with mathematical problem solving. However, despite their widespread adoption, little work in the human–computer interaction community has examined the extent to which these computational tools support expert problem solving. In this article, we report findings from a qualitative study comparing and contrasting the work practices and software use of practicing researchers in mathematics and engineering who share the goal of developing and defending new mathematical formulations. Our findings indicate that although computational tools are used by both groups to support their work, current mathematics software plays a relatively minor, somewhat untrusted role in the process. Our data suggest that five primary factors limit the applicability of current mathematics software to expert work practices: (a) *a lack of transparency* in how current software derives its computed results; (b) *the lack of clearly defined operational boundaries* indicating whether the system can meaningfully operate on the user's input (whether expressions or data); (c) *the need for free-form two-dimensional input* to support annotations, diagrams, and in-place manipulation of objects of interest; (d) the potential for *transcription problems* when switching between physical and computational

Andrea Bunt is a computer scientist with an interest in cognitive tools, software complexity, and intelligent interactive systems; she is an Assistant Professor in the Department of Computer Science at the University of Manitoba. **Michael Terry** is a computer scientist who uses web-centric data sets to model how people think about and use publicly available interactive systems; he is an Associate Professor in the David R. Cheriton School of Computer Science at the University of Waterloo. **Edward Lank** is a computer scientist with an interest in models of low-level movement in interfaces, pen-based interfaces, and sketch recognition; he is an Associate Professor in the David R. Cheriton School of Computer Science at the University of Waterloo.

CONTENTS

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

media; and (e) *the need for collaboration*, particularly in early stages of problem solving. Each of these issues suggests a concrete direction for future improvement of mathematics software for experts. These findings also have more general implications for the design of computational systems intended to support complex problem solving.

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

1. INTRODUCTION

Computer visionaries such as Vannevar Bush and Douglas Engelbart have long viewed computational tools as a means to augment human intellect, allowing one to solve existing problems faster while opening the door to solving problems previously beyond one's reach (e.g., Bush, 1945; Engelbart, 1968). One of the ways computational systems can achieve these ideals is by providing tools that enable one to work at high levels of abstraction when solving a problem. For example, domain-specific languages such as Logo, SQL, or Verilog allow one to work in languages that are much closer to the problem domain compared to general purpose programming languages. Similarly, photo editing tools like Adobe Photoshop enable an individual to work efficiently with operations that would take minutes or hours to perform in a physical darkroom, with the added benefit of being able to undo actions that do not turn out as hoped. In short, by providing higher level abstractions to problems, computational tools free individuals to focus on solving the problem at hand rather than the tedious, low-level operations that would otherwise be needed to complete the task.

In this article, we consider a class of software that has long held the promise of enabling people to work with higher level abstractions, namely, mathematics software. This class of software includes Computer Algebra Systems (CAS), which enable users to rapidly solve, simplify, and otherwise manipulate a wide range of symbolic expressions (e.g., Macsyma, Maxima, Maple, and Mathematica); and tools dedicated to matrix manipulations (e.g., Matlab and Octave). Although there exists a wide range of other mathematics software (e.g., statistics software, software for scientific computation, etc.), in this article, we focus on CAS and software for matrix manipulations.

Although mathematics tools are widely used (e.g., Maple sold close to 800,000 new licenses when a new version was released in 2005; Maplesoft, 2005), there has been little work to understand how they augment mathematical problem-solving practices by *professional users*. Instead, prior work in this space has focused primarily on CAS use in educational settings, where students use the tools as part of the mathematics curriculum (e.g., Artigue, 2002; Heid, 1988; Leinback, Pountney, & Etchells, 2002; Pierce, Herbert & Giri, 2004). Although this prior work provides insight into how the tools support learning, it is unclear whether previous findings are applicable to computational tool use by highly knowledgeable mathematicians solving new problems, a context in which this type of software has the potential to substantially impact cognitively demanding tasks.

This article reports the results of research examining the practices of working professionals and researchers for whom mathematics is an integral, if not central, part of their work. We performed semi-structured interviews of 20 researchers working in a university research setting to understand our participants' daily practices, the full range of tools they employ, and the artifacts they rely on and produce as part of their work. The interviews were conducted in the participants' places of work, where we took photographs of both their work environments and their artifacts to help us better understand their overall goals, workflow, and use of tools and media.

Although our primary motivation was to understand how mathematics software fits into their practices, our study sought to form a holistic view of our participants' work practices to uncover why these software systems are, or are not, used to solve real-world problems.

Our study participants can be classified into two general categories—*theoreticians* and *engineers*. In an earlier publication, we reported how theoreticians perceive and use mathematics software in their day-to-day work (Bunt, Terry, & Lank, 2009). In this article, we elaborate on these findings and complement them with data from a second group of participants, engineers.

The theoreticians who participated in our study work in the area of theoretical mathematics with the goal of producing new mathematical knowledge. In contrast, the engineers employ mathematics to build and validate mathematical models and controllers of physical entities and phenomena.

Despite the different uses of mathematics between the two groups, our study reveals a number of clear, shared themes in the perceptions and uses of computational mathematical tools across the groups. As an example, we found that both groups, as a matter of principle, generally distrust the results of mathematics software, and thus feel the need to validate the system's output. Such recurrent themes provide some generalization to our earlier findings arising from the study of the theoreticians alone. At the same time, the differences discovered between the two groups' use of mathematical software enriches our understanding of the various dimensions one should consider in the design and evaluation of this general class of software. For example, in contrast to the theoreticians, the engineers in our study use mathematics software to build complex simulations. Features that support debugging and verification of these simulations would thus be of great value to the engineers, while of comparatively little value to the theoreticians.

To summarize our results, we found that mathematical tools are used to perform the following tasks:

- To find solutions or simplifications to long, "tedious" expressions. As an example, a user may input a long, complex symbolic expression and ask the system to simplify it. Both groups take advantage of these features, though theoreticians typically have greater need for this type of capability.
- To search through large solution spaces for examples or counterexamples of potential solutions to a problem. A small number of participants representing both groups occasionally engage in this activity.
- To perform rapid, successive manipulations of expressions to understand their nature or to detect patterns. A small number of users across both groups indicated using computational tools for this purpose.
- To create simulations of models of the phenomenon under study (engineers).
- To make sense of the large data sets that arise from model validation (engineers).

These tasks range from those that are quite straightforward to perform with the mathematics software (e.g., simplification), to those that are quite involved, such as

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

data analysis and the creation of simulations and models. However, despite these uses of mathematics software, our findings reveal that our participants choose to do a large amount of their work by hand using physical media. Their reluctance for using mathematics software, and their corresponding preferences for using physical media, arise primarily from the following issues in current offerings:

1. *The Lack of Transparency*: Given the goal of discovering, producing, and proving new mathematical formulations, there is a concomitant need to be able to communicate and defend these formulations. However, the computational tools used by our participants lack sufficient *transparency* in communicating many of their internal processes. Specifically, given input and a command (whether the input is purely symbolic in nature, or a combination of mathematical models and data), the mathematical software produces an output, but often without accompanying information explaining how that result was derived. Without a clear explanation of how the system's output was derived—an explanation of the intermediate steps—mathematicians are not always able to argue or demonstrate the correctness of their solution, a stringent requirement of their work.
2. *The Lack of Clearly Defined Operational Boundaries*: Current mathematical software is able to perform operations on many different types of mathematical expressions and data, but our participants indicated that they cannot always predict whether the system can properly operate on their input, making the output of the software sometimes suspect. In essence, our participants cannot produce robust mental models of the system's capabilities and specific limitations.
3. *The Lack of Free-Form Two-Dimensional (2D) Representations*: Although tools such as CAS could be used in various phases of the mathematical problem-solving process, we found that the software's rigid input/output format can also deter its use, especially in earlier problem-solving phases where mathematicians tend to move fluidly between informal notes, sketches, formulae, and mathematical arguments.
4. *Transcription Problems*: Moving between physical media and computational media requires transcribing expressions into a format the computer can understand. A number of participants commented that this process can be error prone, with errors difficult to diagnose. Eliminating physical media is currently not an option because of the unique advantages it confers during early phases of problem solving.
5. *Limited Support for Collaboration*: Mathematics is often a highly collaborative activity with individuals working around a shared whiteboard or piece of paper. Collaboration using physical media is very natural, whereas collaboration that takes place around a shared computer console requires coordination of physical input devices. These requirements reduce the feasibility of using computational tools for collocated, collaborative work.

Collectively, these findings signal clear issues in current mathematics software that limit their potential for effectively and seamlessly supporting work with a strong

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

mathematical component. These findings also suggest paths forward for improving the design of mathematics software for professional use, a topic that we expand on later in this article.

The remainder of this article is structured as follows. We contextualize this research by first considering previous work in the area of mathematical tools. We then describe our study design and detail its findings. We contrast these findings with those from educational research, derive a set of implications for the design of mathematical software for professionals, and discuss our study's limitations. We conclude with directions for future research.

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

2. RELATED WORK

A wide range of software, both commercial and experimental, has been developed to support mathematical work. As mentioned in the Introduction, for the purposes of this article, we are most concerned with two classes of such software: CAS and matrix-based mathematics software. CAS support symbolic computation involving mathematical expressions. Examples of popular CAS offerings include Maple, Mathematica, and Maxima. The second class of software focuses on matrix-based mathematics and includes packages such as Matlab and its open source equivalent, Octave. Other classes of mathematical software, such as S, R, SPPS, or JMP (which are more geared toward data processing and statistics), are beyond the scope of this work.

Past human–computer interaction research has examined these two classes of software from two different perspectives: through laboratory studies assessing the general feature sets offered by this type of software (including expression entry capabilities), and via studies examining the impact of CAS in educational contexts.

2.1. Assessing General Feature Sets of Mathematical Software

In a laboratory evaluation involving high-school students as participants, Oviatt and colleagues studied the impact of different media on mathematical problem-solving performance (Oviatt, Arther, & Cohen, 2006; Oviatt & Cohen, 2010). The experiment examined students' ability to solve math word problems with four types of interfaces: pen and paper, an Anoto pen, a pen-based Tablet PC, and a graphical equation editor. The authors found that problem-solving performance was better with pen and paper or the Anoto pen compared to the other two conditions and that the performance differences were particularly dramatic for students with low prior math knowledge. Drawing on Cognitive Load Theory (vanMerrienboer & Sweller, 2005), the authors attribute their results to the familiarity students have with entering and manipulating expressions with physical media, leading to comparatively higher cognitive load when using digital media for these tasks. The results of this study also suggest that physical media might better support utilizing and moving between the different representational forms required for this type of problem solving (e.g., prose, diagrams, and formulae).

Given mathematical expressions' two-dimensional nature and use of special notation (both of which push the limits of traditional keyboard and mouse input), other studies have focused more specifically on the problem of entering mathematical expressions. For example, Anthony, Yang, and Koedinger (2005) compared the efficiency with which users could input expressions using four different input techniques: pen-based entry, speech, pen plus speech, and keyboard and mouse using Microsoft's equation editor. The authors found that expression entry with a keyboard and mouse was significantly slower and more error prone than the other three conditions, particularly for equations with a large number of mathematical symbols (as opposed to equations consisting primarily of characters found on the keyboard). On a post-session questionnaire, participants also rated pen-based entry higher than the other conditions for its suitability to entering mathematical expressions.

Expression entry has also been examined in the context of *pen-math* systems, systems that use a Tablet PC as an interface to mathematics software. The goal of these systems is to create a more natural input interface. For example, LaViola (2007) and Labahn et al. (2008) both assessed a user's ability to enter expressions, correct system recognition errors, and solve a number of small problems with pen-math systems. The evaluations have shown that expression recognition and correction can be challenging in such systems but that users are able to complete their tasks effectively once their expressions have been recognized. Thus, although studies by Oviatt and colleagues (Oviatt et al., 2006; Oviatt & Cohen, 2010) and Anthony et al. (2005) suggested advantages to pen-based input, current recognition engines for pen-math systems likely need to be improved and refined before such systems can offer compelling environments for mathematical problem solving.

2.2. CAS Use in Educational Settings

Moving outside of the laboratory setting, there is a relatively large body of work within the education research community investigating how computational mathematics tools, particularly CAS, integrate with high-school and undergraduate education (e.g., Artigue, 2002; Heid, 1988; Leinback et al., 2002; Pierce et al., 2004; Pierce & Stacey, 2001; Robinson & Burns, 2009; Ruthven, 2002). This work has articulated three main advantages of using computational tools in the classroom. The first is students' ability to learn higher level concepts through active experimentation with different expressions (e.g., Leinback et al., 2002; Pierce & Stacey, 2001). It is important to note that this type of experimentation is too time-consuming to do by hand, making it an activity only possible with computational tools. The second potential benefit seen for using CAS in the classroom is that some of the low-level work can be delegated to the computer (such as low-level calculations), enabling students to focus on higher level problem-solving processes and decision making (Leinback et al., 2002; Ruthven, 2002). Students appear to benefit from both of the previous two advantages without a noticeable impact on their calculation skills, despite having less classroom time devoted to this latter activity (Heid, 1988). The third benefit that has been observed is that computational tools allow students to

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

work on interesting problems, particularly ones from the "real world," where these problems would otherwise be infeasible to do by hand given the complexity of the calculations (Artigue, 2002; Robinson & Burns, 2009).

Although a number of advantages of mathematics software have been noted, research in this space has also enumerated the challenges associated with using this type of software in the classroom. First, students often have difficulty coping with the vast array of commands available in current offerings (Artigue, 2002; Pierce et al., 2004). Second, similar to findings from the laboratory experiments described in the previous section, the syntax required to input expressions into such systems can impose additional cognitive load not present with paper-based work (Pierce et al., 2004; Robinson & Burns, 2009). Third, some students have difficulty translating CAS output into representations that they understand, particularly when the output differs significantly from the input expressions. Artigue (2002) argued that this problem is less likely to occur when students are manipulating expressions using physical media, where they are able to see the intermediary steps. Finally, some students feel that they learn more when solving problems by hand, or that "real mathematics" is done by hand, not by computers (Pierce & Stacey, 2001). Some researchers believe that the aforementioned challenges can be addressed through a combination of careful lesson and exercise planning (Leinback et al., 2002) and by having the teacher emphasize and motivate the use of computers for mathematics (Artigue, 2002; Pierce et al., 2004). As we show later, some of these very issues and concerns also arise in professional use (most notably, the need to see the intermediary steps required to achieve the output, and the sense that problem solving is better done by hand than with software), indicating that these issues are not limited to educational contexts.

These initial studies, both in the laboratory and in educational settings, provide important insights into the potential benefits and limitations of current CAS software and other computational tools for mathematics. However, these studies characterize only *short-term* use of such software, typically in fairly well-defined, well-directed ways. For example, in the classroom, students are working on structured, well-defined problems, typically with a known, correct answer. Furthermore, the tools are used to help students understand and apply fairly standard mathematical concepts. In contrast, there has been very limited study of how these mathematical tools are utilized by professional mathematicians who are motivated to develop new mathematical representations. In this latter context, individuals are attempting to solve unstructured and ill-defined problems, for which the "correct" solution is often unknown. Thus, although some researchers have argued for the value that computational tools can bring to mathematics research (e.g., Borwein, Bailey, & Girgensohn, 2004), their true impact and utility are less known.

To the best of our knowledge, there has been only one other study examining computational mathematics tool usage within mathematics research. Quilan (2007) conducted a large-scale web survey (containing primarily of closed questions) designed to understand which computational tools are used in mathematics research (including typesetting software), how frequently, and for what purposes. Through his study, Quilan found that the majority of participants reported at least monthly use of

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

mathematics software, with Maple, Matlab, and Mathematica being the most commonly used packages. Unfortunately, the survey was not able to clearly distinguish why the mathematics software was used. Thus, although it is known that computational mathematics tools are used within professional work practices, there is limited knowledge of *how* and for *what reasons*. A deeper and more contextualized understanding of how these tools are adopted and applied in professional environments would be valuable, both to guide future design efforts and to identify open research problems. We turn now to our work intended to address these open research questions.

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

3. STUDY OVERVIEW

In this research, our primary goal is to understand how mathematics software is utilized by working mathematicians. We also wish to gain insight into how this software could be better designed to integrate with desired practices. We are thus interested in answering the following research questions:

- What are the goals of the mathematicians? What are they seeking to accomplish? What is the "product" or output of their work?
- What characterizes the mathematicians' workflow? That is, how do they accomplish their work?
- Which tools are used in mathematical problem solving (e.g., paper, whiteboard, mathematics software), at which points in the work process, and for what reasons?
- What types of tasks are best supported by the different tools and why?
- What preferences do they have with respect to tools and media?

To answer these questions, we interviewed two groups of mathematicians who work in a university setting. We first interviewed a group of theoreticians, the results of which were reported in (Bunt et al., 2009). We then sought to extend this work by interviewing a group of applied researchers (engineers) to complement the data from the theoreticians. Although the study took place in two phases, we used a common method and analytical process for both groups of participants.

3.1. Method

We used semi-structured interviews in participants' places of work (where possible) to examine the study's research questions. Interviews lasted approximately 30 to 45 min each. During the interviews, we asked participants to educate us about their research practices and how they perform their work. To ground the interviews and assist with recall, we asked participants to walk us through specific instances of recent research work. As participants described specific instances of their recent work, we asked them to discuss which tools they used to complete the various pieces of their

work and their reasons for doing so. At the end of the interviews, we also asked about general preferences with respect to different tools/media.

Interviews took place at locations identified by participants as their primary workspace (either offices or labs), enabling us to photograph their work environments. Conducting the interviews in their workplace also allowed us to view, discuss, and document samples of relevant work materials. One interview was held in the first author's office because of the participant's concern about disturbing others in his shared workspace. This participant brought his current working materials with him to the interview (his laptop and paper work). All photography was at the request of the interviewer (as opposed to being directed by the participant).

We collected data by audio-recording the interviews and taking digital photographs, with three exceptions. The audio-recording device failed during one interview. Immediately following this interview, the interviewer created detailed notes and later had the participant review the notes for accuracy. Two participants declined to have photographs taken of their working materials for reasons of privacy.

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

3.2. Participants

Participants were recruited in a university setting. However, and of importance, our focus was *not* on how these individuals teach or learn mathematics in this setting but on how they employ mathematics in their research.

Twenty mathematics researchers were interviewed. We divide the participants into two categories—*theoreticians* and *engineers*. Theoreticians are mathematicians who have the broad research goal of deriving new mathematical knowledge. The engineers, on the other hand, build mathematical models of phenomena, and thus are more applied in their use of mathematics. We note that not all participants in the engineers category are in engineering departments, though their work clearly falls in the area of the applied use of mathematics. Nine theoreticians (eight male, one female) and 11 engineers (nine male, two females) participated in the study. Participants ranged in age from mid-20s to early 60s. Other demographic data, such as prior computer training, were not collected.

Although classified into these two general categories, our participants represent a wide range of specialties within these categories. All participants have advanced mathematics knowledge, with all but three participants at the PhD level or higher, although seven of the participants were graduate students. The title and research area of each participant is listed in Figure 1. We use the prefix "T" to refer to theoreticians and "E" for the engineers. We omit gender information in Figure 1 to guard participant anonymity.

3.3. Data Analysis

As described in Section 3, our study was conducted in two phases. For each participant group, data were analyzed by the three authors employing methods from Contextual Inquiry (Beyer & Holtzblatt, 1998).

FIGURE 1. Participant backgrounds

Group	Participant	Title	Research Area
Theoreticians	T1	Postdoc	Theoretical Computer Science
	T2	Ph.D. student	Quantum Computing
	T3	Faculty	Applied Math
	T4	M.Sc. student	Pure Math
	T5	Postdoc	Symbolic Computation
	T6	Postdoc	Mechanical Engineering & Applied Math
	T7	Ph.D. student	Pure Math
	T8	Faculty	Applied Math
	T9	Faculty	Theoretical Computer Science
Engineers	E10	Faculty	Electrical and Computer Engineering
	E11	Faculty	Electrical and Computer Engineering
	E12	Faculty	Systems Design Engineering
	E13	Faculty	Physics
	E14	Faculty	Mechanical Engineering
	E15	Postdoc	Electrical and Computer Engineering
	E16	Faculty	System's Design Engineering
	E17	Ph.D. student	Physics and Astronomy
	E18	M.Sc. student	Electrical and Computer Engineering
	E19	M.Sc. student	Computer Science (Graphics)
	E20	Ph.D. student	Electrical and Computer Engineering

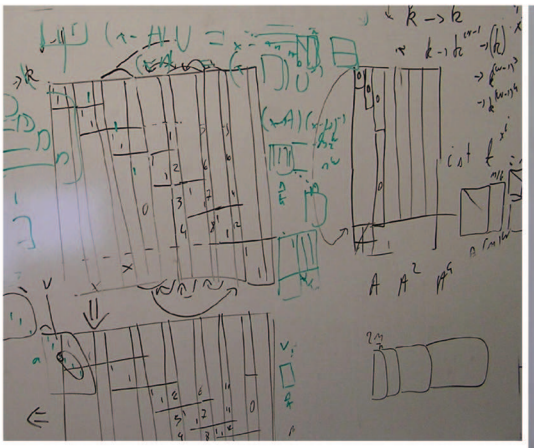
Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

The main findings from our analysis of the theoretician's data were initially reported in Bunt et al. (2009). Data from this first phase were analyzed by creating two separate affinity diagrams, one examining participants' responses and one clustering and categorizing their work artifacts. These affinity diagrams revealed common themes in work practices and goals, as well as common conventions within the artifacts themselves. The affinity diagram for the work artifacts was particularly helpful in identifying commonalities in structure, annotations, and so on, that might not have been explicitly discussed by participants during the interviews.

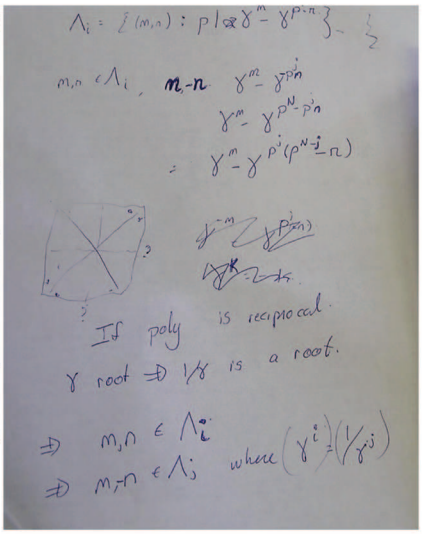
In analyzing the theoreticians' data, we also discovered a strong temporal theme within the work artifacts. In particular, we found that the samples collected served to document the progression of mathematical solutions from early problem-solving stages to final solution forms. Thus, in addition to the affinity diagrams, we developed a timeline composed of pictures of the artifacts. The timeline incorporated samples from all theoreticians, which had the benefit of providing multiple example artifacts from similar points in the problem-solving process. From this timeline, we were able to identify general trends in how their solutions evolve over time, the characteristics of their work at various stages, and how participants move between different tools and media as the work progresses. Figure 2 shows a subset of this timeline.

After collecting data from the engineers in the second phase of the study, we used a similar analysis procedure. Specifically, we created two affinity diagrams (one for interview statements and another for the artifacts) and a timeline. When analyzing the engineers' data, we looked for evidence of the themes identified in our first round

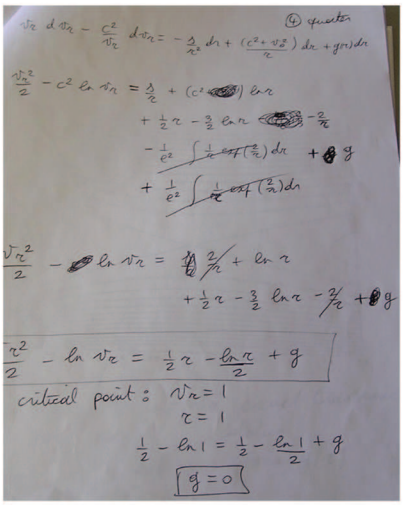
FIGURE 2. Example images illustrating a subset of the timeline present in the theoreticians' work. (a) is an example of early work on one participant's whiteboard and (b) is another participant's work on paper. In the early stages, expressions and diagrams are rough, with little attention paid to alignment or formal prose. As the work progresses (c) shows that there is increased structure, with greater attention paid to alignment. In (c), however, items are still being actively manipulated as the derivation unfolds. Finally, in (d), we see an example where the narrative has reached a more formal state—the writing is neat, the structure is clean, and rhetorical conventions are used in a more rigorous fashion. (Color figure available online.)



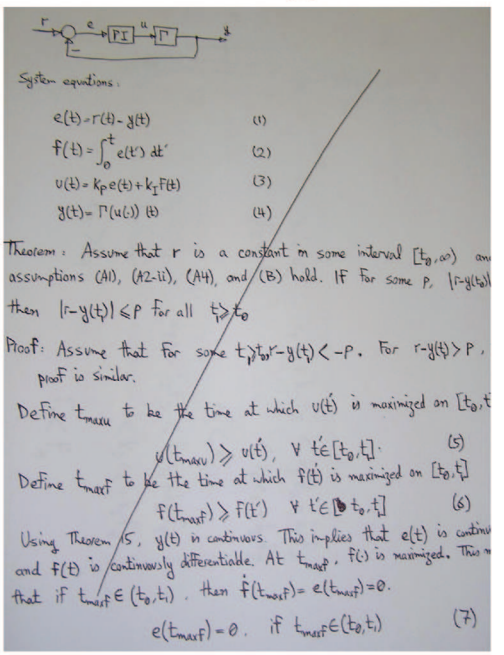
(a)



(b)



(c)



(d)

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

of analysis and for any new emergent themes. Data from the theoreticians were subsequently reanalyzed in light of any new themes that emerged from this second data set.

To ensure proper interpretation of the data, the authors held interpretation sessions amongst themselves to discuss the findings derived from the data analysis. We also conducted a small number of follow-up interviews with participants to verify our interpretations.

4. STUDY FINDINGS

In this section, we examine the goals and practices of our participants, as well as the artifacts and tools employed by them. This description of their goals, workflows, tools, and artifacts establishes a context within which we can consider the current and potential roles of computational tools in mathematical problem solving. In the next section, we use this context to enumerate limitations of current computational tools for our participants. We note that when discussing our findings, our intention is to describe the practices and perceptions of this particular group of professional mathematicians. We discuss the potential generalizability of our findings to other professional mathematicians in Section 6.3.

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

4.1. Goal/Product of Work

As would be expected, there are some clear differences between our two groups of participants with respect to goals and workflow. In fact, given the diverse research areas of our participants, there is no single definitive representation of what our participants are trying to accomplish and how they go about accomplishing this work. However, the similarities observed across participants reveal issues likely to be of concern for many expert users of computer-based tools for mathematics.

Our data reveal that the primary goal of our participants' work is to develop *new mathematical formulations*, as opposed to applying existing mathematical techniques to well-known, well-defined problems (such as performing a statistical test of significance). In addition to developing these new mathematical formulations, both theoreticians and engineers must demonstrate and/or argue for the new formulations' correctness. This specific goal of constructing new mathematical knowledge has important consequences for the role of computation in our participants' work process, as we show.

Although both the theoreticians and the engineers share the same broad goal, the specific output of their work differs. For the theoretical mathematicians, the output of their work tends to be a formalized *mathematical narrative*. The purpose of the narrative is to describe the *transformation* of mathematical entities from an initial form to another, more desirable form and to prove the correctness of this transformation. The narrative itself is a mixture of prose, mathematical expressions, and graphs and

diagrams of the mathematical phenomena under study (e.g., Figure 2D). It is a highly structured document following established conventions in rhetorical style and the visual presentation of mathematical material.

For the engineers, the output of their work tends to be a mathematically formulated model or controller (where a controller is a set of laws defining how a modeled entity should behave to achieve a particular goal), and a demonstration of correctness. We expand on each of these specific outcomes next.

Mathematical Narratives

The theoreticians' mathematical narrative serves dual purposes: It communicates the mathematical phenomena to others, but just as important, it argues for the correctness of the work. Thus, although the end result may be the derivation of a new mathematical formula, it is the description of the derivation and the argumentation for its correctness that form the primary contributions of the work.[1] Consequently, the mathematical narrative simultaneously *contains* and *constitutes* the results of the mathematician's work.

As an example, T3 examines properties of perfect numbers—numbers whose divisors sum to equal the original number. His work involves finding properties and bounds on these numbers, as he describes with the following quote:

> Every single one of them that is known is even, but they have no proof why there's no odd ones. But they do come up with these bizarrely insane large upper bounds. Like: "If there is one then it has to be bigger than 10^{300}." "If there is one it has to have more than 75 factors." . . . these bizarrely large upper bounds. . . . You can't just use a brute-force algorithm and search all odd numbers up to 10^{300}. . . So you have to use some somewhat sophisticated methods, which involve a lot of symbolic computation. [T3]

Although T3's problem can be plainly stated, developing a mathematically rigorous, tight, and compete description of the phenomenon under study is elusive. Progress on this problem moves forward piecemeal, with individual mathematicians making small contributions bit by bit. Accordingly, as each new component of this problem's larger solution is discovered, it must be rigorously argued and defended in a mathematical narrative.

Engineers' Models

The engineers, on the other hand, have the goal of developing a mathematical model that will (typically) have some real-world application. The engineers in our study modeled a wide range of phenomena. For example, in the following quote, E14 describes work on modeling flames. Other types of phenomena being modeled by

[1]Note that although we use "formula" here as an example, the role of the narrative is the same for other types of mathematical work, such as proving a mathematical relationship or concept.

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

our participants included air-traffic control patterns and circuits for use in quantum computers.

> Quite a lot [of the work involves math] because I work in modeling. So I do numerical simulations of flames. So of course there is [the] physical or even chemical engineering aspect, but there is even more applied math aspects to it because we use numerical methods. . . . We obviously use computers. We have . . . equations for species and we look at . . . how to model some of the terms in these equations. [E14]

As with the theoreticians, it is critical that the mathematical formulations derived are accurate, as this participant underscores:

> We do physically accurate simulations of materials. So, it's not so much about how it renders on the screen, it's the actual mathematics, are they correct or not. [E19]

The method by which the engineers demonstrate the correctness of their work differs somewhat from the theoreticians. For most of the engineers we studied, demonstrating the correctness of their work consists of graphically depicting the results of a simulation and arguing that it matches either expectations or experimentally gathered data (which may be collected and published by other researchers). Thus, for these participants, there is less of an emphasis on mathematical proofs or arguments about how the model was derived, and more emphasis on demonstrating that the output of the model matches expectations.

Summarizing Goals of Both Groups

Although the specific goals of the theoreticians and engineers differ, both groups are attempting to derive and defend new mathematical formulations. In the case of the theoreticians, the goal is to demonstrate that, based on a logical argument (a proof), new insight into the structure of mathematics has been created. In the case of the engineers, the goal is to demonstrate that the mathematics applied to the problem at hand accurately models the behavior of the phenomenon, or maintains the desired behavior of a system over time. Out of necessity, both groups must employ sophisticated mathematical concepts and transformations to achieve their respective goals. It is this manipulation of mathematics for novel ends that interests us in this work, as prior work has examined computational support for mathematics almost solely in educational contexts.

4.2. Workflow

Across both groups, we observed a common pattern of problem solving, even though the precise details and objects of concern necessarily differ between participants. The overall process is akin to design practices, such as those described by

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

Schon (1983) in the *Reflective Practitioner*: It is an iterative process in which the solution gradually evolves to become more and more refined.

T6, a theoretician, provides a cursory summarization of this process:

> Okay, this is how I work. First of all, I think about the problem. I draw some meaningless figures like this [artifact] and then I translate what I see to some equations. Then I write my equations down [in a way] that is readable by someone else, like this [artifact]. . . . And then I type it and then I submit it. [T6]

Although somewhat dismissive of the utility of some of his practices (specifically, the "meaningless figures" which nonetheless play an important role in understanding the problem space), T6's rough description of his workflow is indicative of other participants' work practices. For example, E11 describes a similar step-by-step process for problem solving:

> [E11] And so basically you'll start to, well more or less brainstorming, seeing what choices of these control inputs can influence . . . you know, first of all we try to formulate the objective mathematically.
>
> *[Interviewer] Would that [paper artifact] be an example?*
>
> [E11] Well this is a bit further along. This is where, at this point, what I was trying to do is come up with a simulation. So basically the idea is we first develop the theory on paper, once that we're convinced that it's correct, we try and do a simulation. Although of course, it is not so Step 1, Step 2. It's like, as we're developing the theory . . . we're not sure . . . that we're on the right track. You say "OK this seems to be correct. I have this conjecture that seems to be correct. Let's verify first by simulation." If it works in simulation, then we'll try to formally prove it mathematically.

In the preceding description, we see a gradual development of a solution that moves from initial, rough representations (brainstorming to a theory sketched out on paper) to gradually more refined representations (simulation, followed by a formal mathematical proof). This overall process was echoed by our other participants, as well.

If we more carefully analyze the work artifacts and descriptions of participants' problem-solving process, we can break down the workflow into the following problem-solving phases:

- *Ideation*: A brainstorming phase where solution ideas are generated (theoreticians and engineers).
- *Evolution*: Ideas are explored and examined in detail by solving, deriving, and constructing mathematical proofs (theoreticians), or by developing mathematical models (engineers). It is important to note that both groups continually *validate*

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

the correctness of their evolving solution using means appropriate to the problem space.
- *Formalization*: The results of the previous two phases are refined such that the work becomes a more complete mathematical narrative (theoreticians) or functional model (engineers).
- *Dissemination*: The work is prepared so it can be presented to others, either via publication or a more formal presentation to a supervisor (theoreticians and engineers).

Although these phases are represented above as discrete entities, in practice the boundaries between them are nebulous, with much overlap. For example, as noted by E11 earlier, if a simulation doesn't work out, then E11 returns to "Step 1, Step 2," that is, ideation and evolution, to develop a new simulation. Despite the fluidity between phases, they are useful in considering how mathematical software tools can best fit into the overall problem-solving process. We describe how these phases are realized by each of the groups next.

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

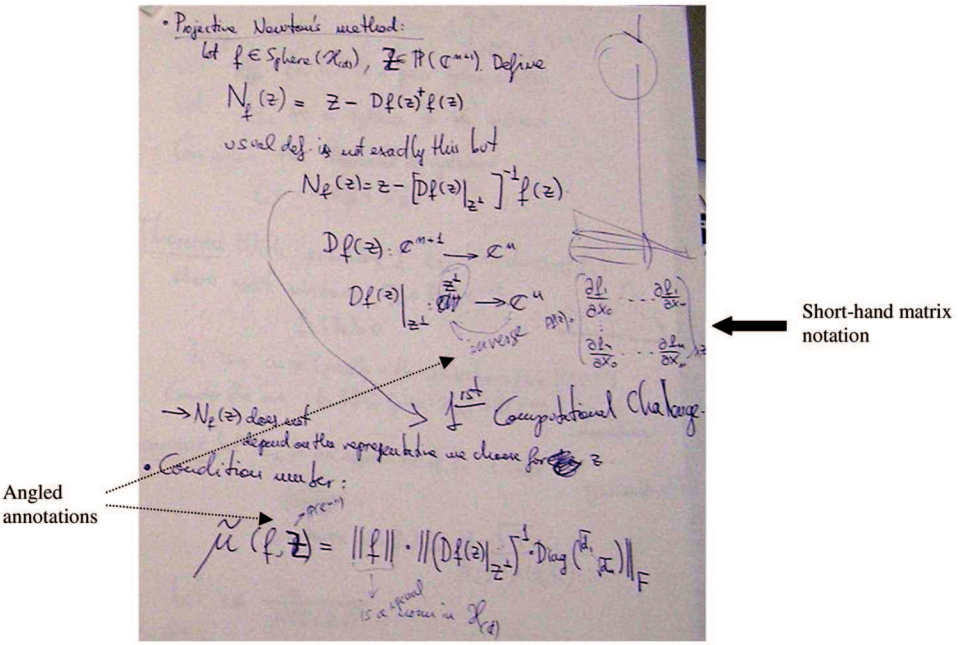
Theoreticians: Developing the Mathematical Narrative

In the early stages of the theoreticians' work, a given problem is represented using rough sketches; basic diagrams; and the informal use of text, mathematical formulae, and rudimentary mathematical argumentation. Figure 2A shows one mathematician's whiteboard at an early stage, whereas Figure 2B shows early work on a piece of paper. Although one can observe some of the elements and conventions eventually used in a final narrative (e.g., the use of whitespace, indentation, and labels to visually structure the document), these initial representations are intended for the mathematician himself, rather than a third party. These practices are indicative of the ideation phase and the early evolution of the final solution.

Work appears to progress in theoretical mathematics through continual manipulation of the mathematical entities on the physical media. Figure 2C and Figure 3 show snapshots of this early work, embellished with notes, content crossed out, and the manipulation of expressions into other forms.

As the theoretician becomes more and more confident in his solution, the workflow enters the formalization phase and the narrative begins to form. In this stage, the representational forms become more structured and detailed, and the arguments become more explicit and refined. This gradual move to more formal representations not only helps prepare the document for eventual presentation to a third party but also serves as a problem-solving tool itself. More specifically, as one moves to more formal representations and argumentation, the problem is subjected to increasing levels of mathematical rigor, which can uncover flaws not obvious with earlier, rougher, higher level representational forms. For example, Figure 2D shows a solution at an advanced stage where the presentation has achieved a fairly high degree of formalization. However, the "slash" through the page indicates that the solution

FIGURE 3. Another example of early work using pen and paper. Of note are the use of annotations at an angle to distinguish them from the primary narrative and short-hand notation for matrix definition (on middle, right-hand side of the paper). (Color figure available online.)



was found to be incorrect at this late stage.[2] After reaching a final, formalized form that the theoretician believes to be valid, the work can move to the dissemination phase, which includes preparing a submission for a conference or journal.

One thing that should be emphasized about the theoreticians' workflow is the primary use of physical media *throughout* the problem-solving phases.

Engineers: Developing and Validating the Models

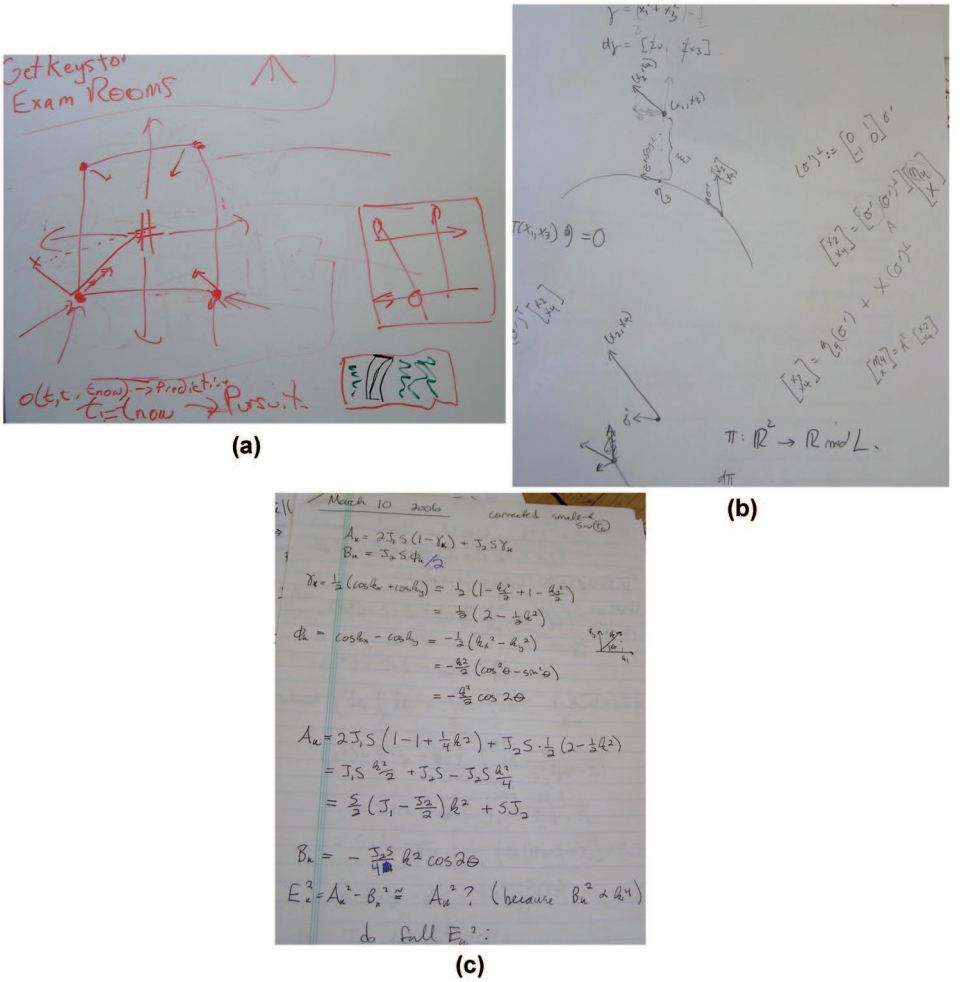
In the more applied domain of the engineers, problem solving shows the same progression from an initial, rough ideation phase to a more structured, evolution phase. Figure 4 shows paper-based artifacts that correspond to these phases of problem solving.

Like the theoreticians, the engineers also continually validate the correctness of their solution as it evolves. However, validation of their mathematical formulations tends to proceed through *model derivation*, and *simulation*, as opposed to formal mathematical argumentation. Model derivation is the process of developing a mathematical representation of the phenomena in question (e.g., circuits or flames) and involves manipulating and transforming mathematical entities to better approximate

[2]The slash could also represent the fact that the material is no longer needed, but in this case, it represents a dead end.

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

FIGURE 4. Example images illustrating a subset of the timeline present in the engineers' work. Similar to the theoreticians, we see the work progressing for a rough initial state (a) to a more structured derivation (c). Absent from the timeline for this group, however, are instances of formalized narratives within physical media. (Color figure available online.)



assumptions or experimental data. Simulation, on the other hand, involves testing the model under a range of cases and examining the output for correctness. This two-stage process has clear analogues to the theoretician's work in the sense that model derivation represents an informed conjecture (equivalent to the initial kernel of a solution for a theoretician), while simulation argues for the model's correctness (equating to the formal argumentation for a theoretician).

As engineers emerge from the evolutionary phase, we see a key difference between their workflow and that of the theoreticians: Although many theoreticians prefer to formalize their narrative using physical media prior to entering the

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

computational realm, we did not see evidence of argument formalization within the engineers' paper artifacts (see Figure 4 as a representative artifact timeline for engineers). The lack of argument formalization on paper can be accounted for by the engineers' need to enter the computational realm to analyze experimental data and to create simulations of the phenomenon under study. As noted by E14 earlier, model validation often involves complex simulations that are infeasible to do by hand. As such, computational tools play a more critical role for engineers than theoreticians. However, we later show that *both* groups have serious reservations about fully relying on computational tools to validate their work.

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

Summarizing Workflow: Commonalities and Differences

In both groups, we observed a common process of developing new mathematical formulations, though there were some notable differences in the tools and methods employed. Both groups begin with rough ideas that are sketched out using informal notation, prose, and diagrams. Our participants then evolve these informal sketches and arguments to the point where they represent either formal narratives (theoreticians), or mathematically sound models or controllers (engineers). Validation during the evolution phase differs between the two groups. For the theoreticians, it tends to be through argumentation alone, whereas for the engineers, there is often an additional simulation step, where the model/controller is simulated across a range of scenarios to demonstrate its correctness.

We also found overlap in the tools used in the different phases of work, with some crucial differences between the two groups, where these differences are largely attributable to the nature of the work being performed. Although both groups make heavy use of physical media for ideation, the role of computational tools during evolution and formalization differs between the two groups. Many theoreticians prefer to use physical media almost exclusively during these phases of work. Engineers, on the other hand, perform much of their model derivation on paper but begin to enter the computational realm for simulation, where hand computations are infeasible. The work artifacts suggest that this group appears to do more formalization within the computational realm than the theoreticians we studied.

From this overall description of work practices, we turn now to the specific roles computational tools play for both groups.

4.3. Computational Tools for Mathematics

The primary computational tools that our participants reported using are CAS for symbolic computation and matrix-based mathematics tools for performing mathematics, simulations, and data visualization. The two other computational tools mentioned during our interviews were Adobe FrameMaker [E10] and Microsoft Excel [E12].

Symbolic computation software is used by all of the theoreticians in our study and by three of the engineers. Of these 12 participants, 11 participants reported using

Maple and one participant indicated using Mathematica. The engineers in our study all use Matlab, matrix-based mathematics software.

In this study, we found that members of both groups use software to

- Manipulate, or verify the manipulation of, complex expressions.
- Search through large solution spaces for examples or counterexamples of potential solutions to a problem.
- Perform rapid experimentation.

In addition, engineers in our study use computer tools to

- Simulate models of the phenomenon under study.
- Make sense of the large data sets that arise from model validation.

Before describing these uses in more detail, we note that the *extent* to which computational tools are applied was much less than anticipated. In Section 5, we return to this latter point and explain why computational tools are reluctantly employed by our participants.

Manipulating Complex Expressions

One of the most common uses of computational tools by our participants is to solve, simplify, or integrate an expression that they find too long or too "tedious" to compute by hand. T1, T3, and E15 all speak to this type of use:

> Usually if it is a complicated expression that I can't resolve myself, . . . the kind of tedious work that is sort of boring and uninteresting but where it is easy to make mistakes. [T1]

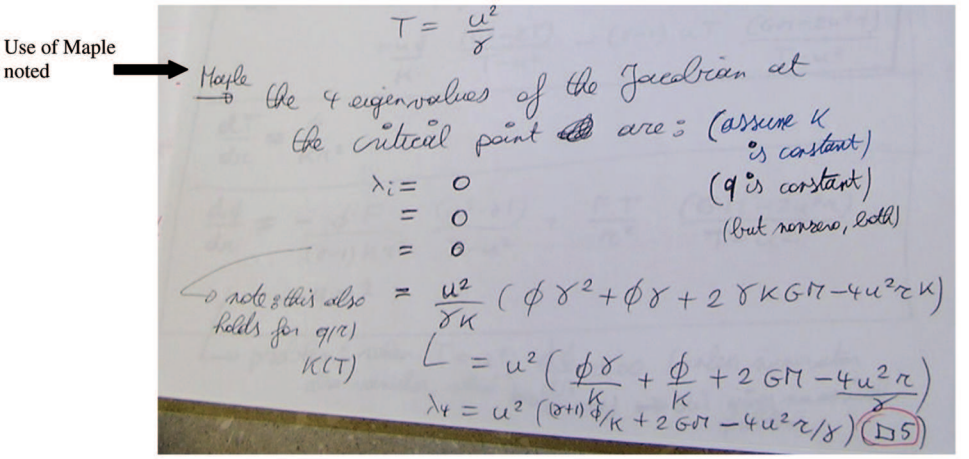
> If I have some horrible expression that I don't like, some large amount of tedious computation, "integrate this" or "reduce this giant mess to something useful," then sometimes I'll stick it in Maple to see if it can solve the problem for me. [T3]

> So supposed you have an ugly expression like this. You find the second derivative and you want to set it equal to zero. It was [a] huge expression, but [the symbolic toolbox in Matlab] simplifies that for you. [E15]

When solving these types of expressions, participants (both theoreticians and engineers) said that they use paper to formulate the expressions, use symbolic software to solve or simplify the expression (e.g., using Maple or the symbolic toolbox in Matlab), then transfer the result back to their paper work. Figure 5 illustrates an example of this process. In this snapshot, the participant has jumped to Maple to perform a calculation, copied the result back onto paper, and *explicitly noted* this use of Maple among their notes. Indicating that Maple was used can serve multiple purposes. First, it can act as a reminder of the path taken to arrive at the solution. It can, however, also relate to issues of trust in the software and its output, which is sometimes suspect. We explore this latter issue in a later discussion.

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

FIGURE 5. A portion of a participant's pen-and-paper work where he or she has used Maple and noted so directly within the narrative under construction. (Color figure available online.)



When participants choose to work with complex expressions by hand, they sometimes will use mathematics software to verify their work. E11 describes how he often ends up solving messy expressions by hand, which he then verifies using Maple:

> . . . especially when you are dealing with non-linear systems . . . you end up with these really ugly symbolic calculations that you have to do, that I typically make a mistake when I do them by paper. So for example, taking a bunch of derivatives of a function, sometimes I'll make a mistake, you know taking these derivatives. So I do find it is useful for me to do it in Maple because I can verify my calculations and check that I'm not making mistakes.

Thus, participants across both groups indicated that they work out the details by hand and then subsequently use the CAS to confirm hand-derived solutions. For the theoreticians, who tended to prefer to do the majority of their work using physical media, verifying hand-derived work or solving otherwise unsolvable expressions were by far the most frequently reported (or only reported) use cases.

Model Simulation and Making Sense of Large Data Sets

The most common use of computational tools for engineers is to test or validate a model through simulation. This process tends to involve computing the model's output for a large number of cases, as described by E19:

> So we actually perform the same simulation, we'll do it like a million times, or ten millions times. So it's obviously all done by computer. And then at the end we take those results and we graph them and compare them to measured data to see just how close we can get.

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

To validate their models, the engineers either use the computational tools to produce graphs that they can inspect visually (as just described by E19) or, as E11 indicates, use the tool to programmatically verify that all assumptions are held:

> So basically I—once I found the curve I used Matlab to check our assumptions. So that's what this script does . . . [is] show you the curve we're trying to follow and basically, um, numerically checking various conditions hold, that must hold for this theory to apply.

The most common tool used by our participants for performing simulations was Matlab. However, some indicated having to program the simulations from scratch using C or Fortran either because Matlab is too slow, the algorithms and functions it provides too generic, or because they preferred not to rely on proprietary software:

> And so in that sense, a lot of the Matlab, or these pre-packaged routines, isn't sufficient, because we need really tight memory control. You know, we essentially need to work at almost machine level in order to do this type of thing. So then we end up programming it ourselves, typically either in Fortran or in C. [E13]

> I think it's a question of speed and, as well, we don't want to rely too much on a commercial package . . . for example, we have a cluster with several processors, we won't have Matlab installed on all of these machines. It is more for flexibility, as well. . . . We don't need a specific package to do the work, but . . . we need compilers, obviously, but that's it. [E14]

Searching for Examples or Counterexamples

In addition to using computational tools to solve complex expressions, a small number of participants in both groups use the tools to perform exhaustive searches through a solution space. In particular, two participants (one theoretician and one engineer) reported using Maple or Matlab to search through a space of solutions for counterexamples that violate one or more mathematical properties:

> It's a matter of just testing all possible solutions to see if they are solutions or not. And the algorithms are really the fastest way I can test that. [T2]

> So we have a controller that I tested to make sure that it satisfied what we think the bounds are going to be, it worked and didn't exceed [the bounds]. So obviously the easiest way to disprove something is to find a counterexample. [E20]

In these cases, the computer's ability to quickly test potential solutions enables the mathematicians to verify their assumptions about the correctness of a potential solution, a task that is infeasible to do by hand.

Rapid Experimentation

In a few select cases, participants from both groups also reported using computational tools to help explore ideas through rapid experimentation. For example,

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

one of our participants (T1) indicated using Maple in the *Ideation* phase to experiment with the output of a number of similar expressions, whereas another participant (E12) indicated doing a similar type of rapid experimentation in Excel. A third participant (T2) showed us how he writes code to have Maple generate multiple plots, which he then examines visually to see if he can detect patterns. These uses cases, however, were not frequently discussed, with most participants across both groups primarily using physical media during the early phases of their work.

Summarizing the Use of Computational Tools

As noted, the extent to which computational tools assist with mathematics was less than anticipated. For many of the theoreticians, the only reported use case was manipulating complex expressions, and most reported doing so only infrequently. The engineers, in contrast, make greater use of computational tools, but primarily for creating simulations and analyzing data. It is important to note that these simulations are sometimes created using more general-purpose programming environments (as evidenced by the preceding quotes from E13 and E14), which suggests that the required capabilities of current mathematical tools are present in more general purpose programming environments.

In short, although mathematics software is clearly utilized by our participants, it is also not the focal point of the problem-solving process. In the next section, we delve into the reasons why mathematical tools play only a limited role for professional mathematics work.

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

5. LIMITATIONS OF COMPUTATIONAL TOOLS FOR MATHEMATICS

In our study, we discovered a number of issues in the design of current mathematical software that lead our participants to prefer working with physical media during many phases of the problem-solving process. These issues can be summarized as follows:

- A need for transparency in the problem-solving process.
- A lack of clear boundaries for a tool's capabilities.
- A need for free-form 2D representations.
- Transcription problems.
- A need for collaboration.

We elaborate on each of these issues in turn.

5.1. Need for Insight and Transparency

Through our interviews, we discovered that even when performing individual calculations, our participants seek more than just an answer from the system: They also

seek to further their understanding of the problem under study. For our participants, this deeper understanding is achieved by directly engaging with the problem by hand, and by being able to clearly see all manipulations of the evolving solution. We characterize these two themes as the need for *insight* and *transparency* in problem solving.

For the types of problems they solve, our participants feel they are able to gain better insight and can better detect patterns when manually solving problems, compared to using a computational tool. The following quotes from participants illustrate these perceptions:

> Computers are great for running through large amounts of examples, but you don't get the same insights. Whereas if you did something by hand, sometimes you just get more insight and can figure out the general pattern. [T2]

> You can notice patterns better if you've done it yourself rather than just the way Maple has grouped it. [T9]

Notably, these quotes echo some of the very same issues that have been identified in studies examining mathematical software use in educational contexts, where students report feeling that they learn more when solving problems by hand (Pierce & Stacey, 2001).

For E18 and E19, doing work by hand helps them understand the relationships between the different elements of the problem, and how the different pieces influence one another:

> The other aspect of this [work] is to actually see the math behind it. And it's well and good to produce the plots but it's also a good exercise to actually see the algebra and where these tones are actually coming from. And so for that, I've actually done some by hand. [E18]

> If something else does it for me, something like Maple, I don't—I find I don't learn the formulas as well, whereas if I do it by hand, I can see the direct relationship between everything. Like something I like to do, which I've done in here, I'll compute units by hand because I find it really helps me learn the formulas and see how the relationships between the different parameters work. [E19]

When doing work by hand, each of the steps taken in deriving the solution is visible. In contrast, when offloading tasks to the computer, much of the work being performed is not visible or tangible to the user. In short, there is a lack of transparency when performing tasks using a computer.

The lack of transparency in current tools leads to a fundamental distrust of computer-generated results for many of our participants:

> Sometimes the computer algebra, it skips steps, or you can't see, or in the end you have to go back . . . [T9]

> I tend to not trust the results from the symbolic toolbox. . . . Although it is very infrequent that the results are incorrect. [T6]

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

With respect to T6's comment, it is important to note that mathematical software, like any other piece of software, has bugs. Consequently, it *can* produce incorrect output through no fault of the user. (In fact, many websites document the various bugs present in current offerings.) Thus, although the software may rarely produce incorrect output, it is still a possibility that T6 must consider when performing work.

Because current tools do not list the operations performed to arrive at their output, our participants cite a need to verify results, as T1 and T2 indicate:

> Whenever you do something in Maple, you'd like to be able to re-produce it by hand. [T1]

> Sometimes the software package comes back with something even more horrible than you expected and it is hard to translate that back to something you understand. [T2]

T2's problem of interpreting the output mirrors issues identified in educational contexts, where students sometimes have difficulty understanding CAS output when it differs significantly from the expressions inputted (Artigue, 2002).

More generally, the need for our participants to be able to reproduce the work of the mathematics software is clear: For our participants, they must clearly communicate their results and argue for their correctness. Thus, although the mathematics software may be relatively quick in producing an answer (and relatively reliable as well), the participants must still invest time to re-create the operations performed to communicate these operations to others, as well as verify their correctness and appropriateness.

Although the lack of transparency is especially problematic for the theoreticians when doing symbolic manipulation, engineers also expressed concerns in relying too heavily on the output of mathematics software. For E20, issues of transparency and trust are amplified as more and more layers of third-party code are required to solve a problem:

> And they keep kind of adding new and new functions to do more and more complicated things. And those functions kind of call the lower functions and you keep kind of building up like that. And as you keep going up, it's like you are more "black-boxing it" over knowing what you're doing. And the more you go to that black box thing, the more you're putting your trust that this guy programming it has done the right thing, for your particular application. Which can be very difficult, right? Because as you keep building these things up, it is more and more likely that something in there won't necessarily work for that particular thing that you're doing. [E20]

In essence, as E20 expresses, as the complexity of the system increases, so too does the likelihood of errors in the system. These increases in software complexity also make it more and more difficult to achieve transparency in any practical sense.

In sum, although our participants regularly rely on mathematics software (particularly the engineers), they do so reluctantly and with reservation. Current tools can

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

often produce output very quickly, but it is nonetheless subject to verification, which adds to the overall costs of using these systems. Accordingly, manual work still has a very strong foothold in the problem-solving process for our participants.

5.2. Lack of Clear Boundaries for Tool Capabilities

The lack of transparency in current tools also makes it difficult for users to establish robust mental models of the *boundaries of the tool's capabilities*, that is, the boundaries that delimit the types of problems the tool can be relied upon to correctly solve, and the types of problems it is likely to fail on. E20 describes this experience of dealing with a problem that straddles the limits of the tool's capabilities:

> We know with Matlab, we certainly run into things where it screws up. And so, it's just like—to do it by hand you feel more confident, kind of thing, would maybe be the best way of putting it?. . . You're inverting matrices, and stuff like this, and it will spit out a number, when it maybe shouldn't be. It should probably be telling us this number is garbage, right? Just, you know, you're inverting like a very small matrix kind of thing, all of these like numerical sensitivities and stuff like that. . . . You get like some number times 10^{-32} or something like that. And [you think,] "ok, that's actually zero." . . . You can just kind of accept that [it is zero], as opposed to if you're trying to publish something at the end, you really have to be more careful and make sure that's actually zero and not just some small number. [E20]

This quote exemplifies the issue of fuzzy tool boundaries: The software is not always aware of its own limitations in calculating results, and the user must make an educated guess as to what the "true" solution is (which must be mathematically proved later). Stated another way, the tool is failing to distinguish between an expression or computation that falls within its domain of valid input (and for which it can produce a definitively correct result) and one that lies outside of its capabilities. The lack of appropriate feedback from the system about potential limitations in the validity of the output thus leads to difficulty in identifying the tool's boundaries, as T1 indicates: "I don't have a good understanding of what kinds of things [Maple] trips on" [T1].

Not having clear boundaries on valid input is particularly problematic when deriving new mathematical knowledge, where researchers are not easily able to distinguish between an error and an unexpected result, as what is "correct" is still being defined as part of the work. E20 deals with these uncertainties by trying to match tool output with his expectations, which can be difficult to do when exploring new phenomena:

> That's why I always like to try to—"what do I think I should be seeing"—and then compare that to what it gives me. . . . As a general rule of thumb with any computer system, I would say that they all have bugs. I believe that's a fairly fair statement. So how do you know that you're not finding that one case that's going to be a bug, and especially when you are doing math research where no one has necessarily done what you're doing before? [E20]

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

As E20 indicates, when the tool produces unexpected output, the researcher must both question his or her thinking and the tool itself, resulting in a significant amount of diagnostic work. More specifically, he must examine the model derivation, his implementation of the model within the tools, and finally the applicability and correctness of the tool's functionality given his specification of the problem:

> My first thought would have been to take my controller and make sure that it satisfies all the theorems that we have to show that it should be stable and then kind of going back in [to the tool] and saying "OK, it has got to be something that's causing this."

> You want to look at what you're doing and you want to come up with an idea of what you expect to see happen. And if Matlab gives you something that you don't expect to see happen, then you need to—you can't just go—clearly, you can't just jump to the conclusion that "what I thought should happen doesn't actually happen." You can look at how you've coded that. . . . And I'd say the more layered—the more things the [Matlab] function you're trying to use is doing, the more careful you need to be as well. [E20]

Clearly related to the aforementioned issues of transparency and software complexity, the inability to define clear operational boundaries for these tools increases the burden of their use by requiring users to verify the correctness of the system's output, especially when it deviates from expected results.

5.3. Lack of Free-Form 2D Representational Forms

Throughout problem solving, both groups of participants make use of expressions, diagrams, free-form annotations, and prose to represent and manipulate the evolving solution. Our participants found physical media, such as paper, well suited to working with these various items, in contrast to current computational systems, which provide very restrictive and highly structured 2D representational forms for mathematical entities, diagrams, and text. In this subsection, we describe how the properties of physical media and physical space are employed by our participants and the difficulties in achieving similar results with computational tools.

When interacting with physical media, we found numerous examples of participants directly interacting with the objects of interest: items are annotated, embellished, edited, and crossed out in-place. This in-place interaction is an important feature of the problem-solving process, as it documents the process of transforming the initial state into the more desirable end state. It also serves to document the approaches that *don't* work. T7 comments on this process:

> And I don't even necessarily work down the page. . . . I just sort of have everything all in one spot. Obviously it's not very neat or easy to deal with, but just having everything on one page kind of makes a big difference. . . . I think it's easy having everything all in one spot. It just stops me from forgetting anything. [T7]

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

Participant T2 also describes this iterative process, and the benefits of using physical media to support the work:

> So it is sort of an iterative process. . . . So at first you figure out how you might approach a problem. You try it and it either works or it doesn't. . . . I think this [paper artifact] went through a couple more refinements before it turned into an actual argument. [T2]

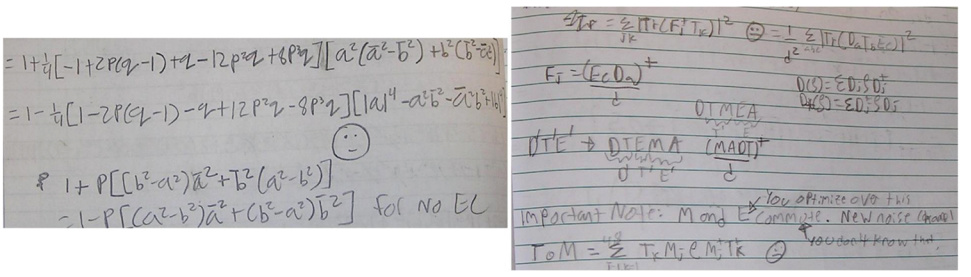
With the physical media, we saw a range of embellishments, including individual terms crossed out in a derivation; entire proofs crossed out that didn't work; and a three-level rating scheme with happy, sad, and neutral faces (see Figure 6). In contrast to physical media, making these types of free-form annotations with computational tools requires workarounds, such as marking up a printout of the output, as shown in Figure 7.

Despite being marked with annotations and embellishments, in some cases it is possible to discern the progression of the solution from one form to another through the symbols and prose on paper, particularly when expressions are written sequentially, in a top-down fashion (e.g., Figure 2C). In other cases, the flow of the work is less obvious, as in the cases of diagrams, which may include many in-place modifications as the understanding of the problem evolves.

We also found physical space itself serve as a tool for our participants: Space is used to communicate information through the arrangement of elements within a single document, large surfaces (such as physical tables) are used to lay out multiple sheets of paper to obtain an overview of the entire problem, and related papers are grouped into folders and special notebooks. As an example, E18 uses the whiteboard because the large surface enables him to view all information relevant to the current problem simultaneously:

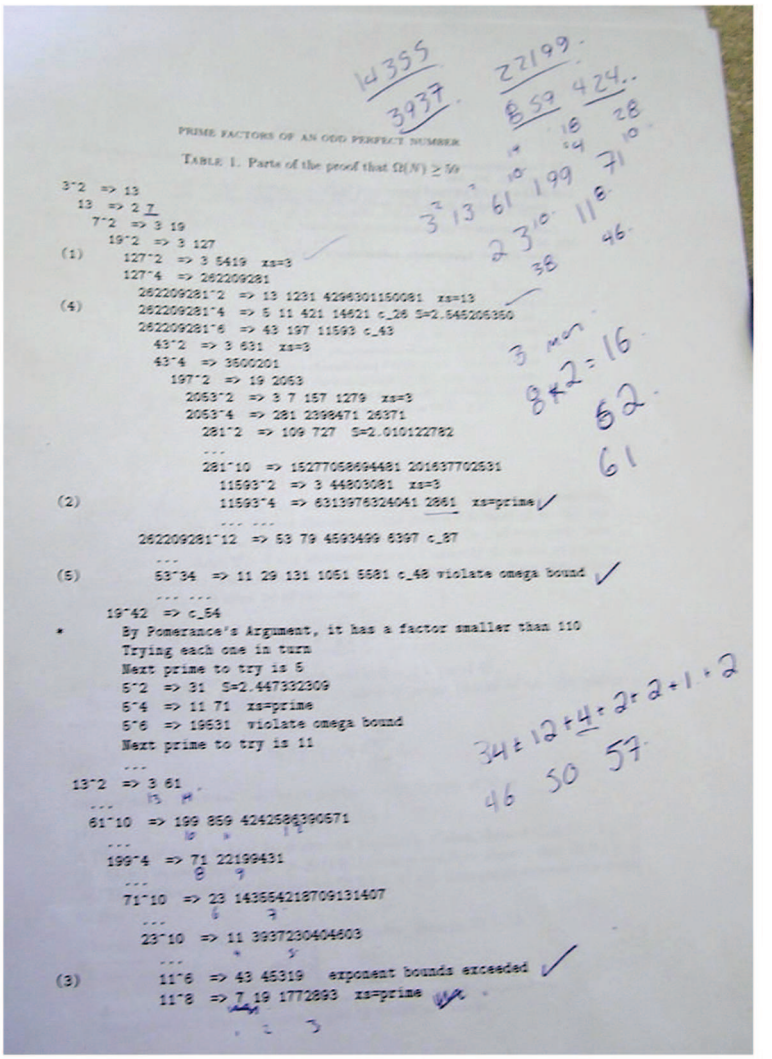
> And what it helps to do is I can make sure I don't lose any key features of what I'm talking about. [E18]

FIGURE 6. Participant E16' s three-level rating scheme for his work: happy faces for successful approaches (left), a sad face for unsuccessful approaches (right, bottom of page), and a neutral face when "I don't know whether what I've just done is relevant" (right, top of page). (Color figure available online.)



Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

FIGURE 7. Participant T3's embellishment of a printout of Maple's output. (Color figure available online.)



One participant was particularly enthused about the features one notebook offered her to assist with the organization of her work:

> I'm very excited about the notebook. This is why: so these ones, they're from France, and the pages are removable. So they are like a binder, so that you can take the pages out and back in. And one thing that's nice is I do different things and I can group together stuff and if I finish a notebook and I haven't quite finished the project then I can take the stuff and put it back in. [T9]

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

In contrast to the free-form, unstructured nature of paper and physical space, commonly available computational tools tend to enforce a highly linear, rigid structure on data. Even tools that allow a mixture of text, mathematics, and figures (such as Mathematica) still rigidly divide the space into discrete areas that follow a top-down organization. As a result, users cannot use space to organize or encode information with current software offerings, and they are likewise very limited in how they can annotate and embellish information in the system.

Many computational tools for mathematics (in particular, CAS) also offer minimal (if any) support for free-form diagramming. When diagramming is possible, creating and editing diagrams is typically not a lightweight procedure, and there is usually very limited control over how the diagrams are integrated with the text.

Finally, current systems have limited means by which one can choose an appropriate level of formalism when representing content computationally. For example, using a CAS to manipulate expressions requires some terms, like matrices, to be formally defined. Short-hand notation cannot be used (as is done for the matrix in Figure 3). Instead, tools such as CAS impart their own level of formalism that cannot readily be adjusted to suit the current problem-solving context.

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

5.4. Transcription Problems

Although we noticed a clear preference for physical media, our participants sometimes must transfer their work to computational forms. This transcription process can introduce complications and otherwise disrupt the flow of work.

One particularly challenging transcription problem is transcribing equations into a form that allows the computational tool to manipulate them. If done imperfectly, unexpected results can arise. When unexpected results do arise, our participants must determine whether there was actually a transcription error, or whether the unexpected results are due to other factors, such as a potential error in the system itself (as described in Section 5.2):

> I'll type in an expression, I'll have spent an hour trying to figure out what it means and what the results are, and then I realize I've made an error typing. [T1]

> The only concern is that sometimes you end up having too many brackets. Although [Matlab] has [parentheses matching] I still find it sometimes tricky and it is very easy to make mistakes, stupid mistakes. [T6]

> [Expression entry] is such a barrier, to do any kind of manipulation. I mean I actually am really excited because I think our new printer at home has OCR, and can recognize the hand-written stuff. [E12]

Part of the transcription problem is due to the reduction in dimensionality of the information: Participants must reduce a 2D expression into a 1D representation inputted via a keyboard. Another problem, however, is the inability for the system to perform sophisticated error-checking on the input. In contrast to natural language,

where tools such as spell checkers and grammar checkers can help detect errors, error detection in mathematical input is primarily limited to rudimentary syntax checking, such as checking for missing parentheses. Mathematician themselves may have difficulty performing error-checking, as they are still becoming familiar with the problem and thus less tuned to what the expression "should" look like.

When considering transcription problems, it is important to realize that errors can arise due to the design of the system (e.g., complex syntax requirements), or simply through human error, as E12 indicates:

> This would all be derived by hand, all hand-derived . . . usually 3 or 4 times because you get to the end and you say "Hmm. It says [the planes] have to be 1000 miles apart. I don't think I've done this right." And you go through and you can find "OK, I had something squared and I dropped it from one line to the next." That part's frustrating.

Some participants indicated that current systems' expression syntax overhead is prohibitive, or that expression entry can be error prone or unnatural when using current mathematical software. However, although transcription problems arose as an issue with current mathematics software, we had expected it to be a more dominate issue for our participants. We were thus surprised at the rarity of these types of comments, particularly in relation to the other issues identified through our interviews. Between the two groups, transcription issues appeared to be more problematic and a greater barrier for theoreticians than engineers. This result may in part be due to the need for engineers to develop simulations, which requires them to deal with programming languages and syntax more frequently than theoreticians.

5.5. Lack of Support for Collaboration

A final limitation of current computational tools for mathematics is limited support for collaboration. Many of our participants, and the majority of the theoreticians, commented on the colocated collaborative nature of their work. Colocated collaboration is particularly prevalent during *Ideation* as described by participant E13:

> So we do, we do a lot of sharing of ideas, sort of on the blackboard and stuff like that. And when I'm teaching my students in here. You know lots of times, you know, we derive the models, talk aloud about the algorithms on the chalkboard, and stuff like that.

When collaborating, participants tend to gather around a large surface, typically a whiteboard, and manipulate content in a highly interactive fashion. Collaboration around these physical media is fairly lightweight, flexible, and fluid. In contrast, colocated collaboration with computational tools is far more constrained. For example, participants must either take turns using a single input device or designate one participant to enter expressions. Both of these strategies require far more coordination

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

than colocated collaboration around physical media and can interrupt the flow of new ideas.

We also observed a tendency to use different colored ink and markers to distinguish between contributors. Current mathematical software, on the other hand, provides very limited, or no, support for tracking and viewing changes by various participants.

Although it is not as large of a deterrent to use as the issues described in previous sections, collaboration plays a significant role within mathematics work, and as such, the lack of any real support in current mathematical software offerings means they tend not to be used during collaborative activities.

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

6. DISCUSSION

In the previous sections, we detailed the ways in which mathematical tools help or hinder problem solving in professional contexts. In this section, we compare our findings with previous results from educational research and draw design implications from our findings. Finally, we describe limitations of our study.

6.1. Comparison to CAS Use in Educational Settings

As mentioned, prior human-centered work on computational tools for mathematics has focused mainly on CAS use in educational settings. We found a number of similarities in how our participants view CAS when compared to these educational perspectives, despite the differing levels of mathematical experience.

Past research in educational contexts has found that some students feel they are able to learn more by doing mathematics work by hand (Pierce & Stacey, 2001). We were surprised to see such intellectual advantages expressed by our expert mathematicians, because these users have mastered the basic concepts. Despite their high levels of expertise, our participants still feel that they gain more insight into the problem domain by solving expressions by hand.

Our participants also indicate that they have difficulty making sense of CAS output when it doesn't match expectations, as has been found with students (Artigue, 2002).

Finally, past research in educational settings has found that students feel that "real mathematics" is done by hand, not by computers (Pierce & Stacey, 2001). The descriptions of our participants' work practices suggest that expert mathematicians might share this perception: The type of tasks relegated to a CAS were often described by our participants as the tedious, "ugly" tasks where one was unlikely to gain insights into the nature of the problem. Our participants preferred to do as much work by hand as possible, to gain confidence in the correctness of the solution.

There were a few notable differences between these settings. First, whereas educational research has suggested that CAS can enable higher level understandings

of problems through rapid experimentation, this type of usage was not frequently mentioned by our participants. Instead, our participants use CAS for more targeted purposes, such as simplifying an individual expression. Although one might assume that the simulations created by the engineers are providing a form of rapid experimentation, this use of the software can still be considered a very targeted use of the tools intended to confirm mathematical models, rather than explore possibilities. In short, the tools are not used by the experts in our study to engage in "what-if" scenarios. In a similar vein, past educational research has also suggested that CAS is well suited to promote higher level problem understanding by taking care of low-level calculations (e.g., Artigue, 2002). However, as alluded to previously, our participants feel that it is precisely these low-level operations that lead to insights and a deeper understanding of the mathematical entities under manipulation. Thus, although "tedious" work is relegated to CAS, our participants still prefer to do as much work by hand to better understand the problem.

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

6.2. Implications for Design

Physical media such as pen and paper, whiteboards, and blackboards are free-form, unstructured media that enable a wide range of expression in problem solving. In our study, we found participants leverage these media's free-form characteristics to sketch, write expressions, manipulate expressions in place, and write explanatory prose to describe and clarify concepts. Using these media, individual elements can be freely arranged and organized, and one can be as formal or informal as desired in expressing one's ideas. However, these media provide no explicit support for mathematical problem solving.

Computer-based tools, on the other hand, offer the promise of providing assistance by actually manipulating mathematical entities. However, these capabilities currently come at the cost of conforming to a rigid, formal syntax and a highly structured medium.

In this section, we consider how the advantages of both media could be synthesized to create a better computational tool for expert mathematics work. We structure our discussion by considering the specific limitations of current tools uncovered by our study.

Increasing Transparency

When users enter an expression into most mathematical software, the system responds with a result but no explanation of how that result was obtained. In our study, we found that more information is generally required to provide insight into the problem space. Consequently, a promising area of future work is designing systems to provide greater transparency in communicating intermediate steps to the user.

Although no support for transparency is included within Maple (the CAS used by most of our participants), this is not the case for all commercially available CAS. A previous version of Mathematica, for instance, provided some transparency within

their student edition, showing the steps of certain derivations. Wolfram Alpha (a web-based service that provides functionality similar to Mathematica) includes a "show steps" feature that shows the intermediate steps required to obtain a result. Whether this functionality currently scales to the types of problems encountered by expert mathematicians is unclear, but it is a step in the right direction. Finally, the Derive system can also display the steps of a simplification, along with the rules used in the transformation (http://www.chartwellyorke.com/derive.html). Our findings suggest these efforts will be welcomed by both students and expert users of such systems.

One of the challenges in exposing the internal workings of a computational tool for mathematics may be that the methods employed do not easily lend themselves to display to end-users. That is, the heuristics used may not easily translate to clear explanations for end-users. As such, one barrier to fully realizing these concepts may simply be the additional work required to develop usable explanations.

There is also a question as to how much commercial software companies wish to reveal their underlying algorithms, given that they may be considered trade secrets and a part of their competitive advantage. However, there are powerful open source equivalents to proprietary CAS and matrix-manipulation software that could be used to research these concepts. In particular, Maxima is a CAS with a decades-long history of development, and the open source matrix software Octave clones much of Matlab's language and core functionality. Both of these systems provide means for researching these concepts without any concern toward intellectual property.

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

Increased Awareness of Tool Boundaries

Better communicating the boundaries of a tool's capabilities presents a number of open research challenges because of the number of different ways errors can be introduced into results. However, there are a number of potential paths forward.

One of the most obvious ways to assist users with this problem is to provide better documentation for the functionality provided, clearly describing the domain of input and types of problems each feature can reasonably operate on. Although simply stated, one of the challenges of this approach is foreseeing all of the different ways end-users may want to apply the software. To combat this issue, the documentation could be put online, with the capability for users to comment on the documentation. This practice is becoming more commonplace as it allows the community to refine and clarify the documentation over time (see, e.g., documentation for the open source databases MySQL or PostgreSQL, both of which allow users to add comments to the online documentation). Providing easily accessible and searchable FAQs and "knowledge bases" are additional ways to help communicate the boundaries of tool capabilities.

Sophisticated uses of computer algebra systems and matrix-based mathematics software share some similarities with programming in general, as evidenced by many of the engineers in our study who translate their work to general-purpose programming environments when speed and efficiency are important. Parallels can thus be drawn to the domain of programming, where understanding a system's boundaries is akin to

understanding the limitations of a programming API. Consequently, there is the opportunity to translate research results that examine how to support software developers to the domain of mathematics software. For example, recent work has examined how search engines can be utilized to help software developers find relevant code snippets or to debug compiler errors and run-time errors (Brandt, Guo, Lewenstein, Dontcheva, & Klemmer, 2009; Hartmann, MacDougall, Brandt, & Klemmer, 2010). Similar assistance may be possible for mathematics software, helping users find the best method for solving a particular type of problem, or helping them understand whether the system is potentially at fault when an unexpected result is encountered.

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

Support for Free-Form Input

One of the most obvious design implications arising from our study is that there is a need for more informal, unstructured modes of input when interfacing with mathematical software. As we found when examining work artifacts, our participants regularly employ freehand diagramming; utilize multiple colors of pens to distinguish elements; and align, tag, and cross-reference items in the documents they create. Existing mathematics software, on the other hand, supports very few of these operations. Among commercial offerings, Mathematica arguably provides the most support for mixing prose, diagrams, and mathematics in its notebooks, but the notebooks are still highly structured and compartmentalized. As such, they are less suitable for the early stages of problem solving. Accordingly, there is a clear opportunity to provide more unstructured, flexible spaces for mathematicians to work through problems by hand, with the services of a CAS or matrix-based tool available when necessary.

Pen-based research prototypes for CAS are beginning to address some of these needs by providing support for free-form input. For instance, the designers of MathPad^2 have investigated integrating free-form diagrams with text (LaViola & Zeleznik, 2004), whereas the designers of MathBrush have explored the possibility of recognizing common short-hand representations of matrices, typically employed in early phases of problem solving (Tausky, Labahn, Lank, & Marzouk, 2007). It is not difficult to imagine expanding on these ideas to create a system in which users can define their own short-hand templates that expand to full sets of terms when recognized.

Also of note is MathJournal, whose flexible environment supports a wide range of annotations (e.g., free-form diagrams, different colors, alignment; http://www.xthink.com/MathJournal.html). MathJournal's backend, however, is not as powerful as a CAS and, therefore would not likely be sophisticated enough for our participants.

One obvious advantage of pen-based systems is that users interface with the system in a way most akin to how they currently do mathematics—with a pen. Utilizing a pen as input has the promise of more easily entering the 2D entities of mathematics, whether they are equations or diagrams. In addition, pen-based input has the potential to alleviate transcription errors that can arise when transferring mathematics from other media to the more formal syntax required of current systems (though recognition errors are still a factor).

Notably, multiple options exist for pen-based input: One could use physical media (e.g., an Anoto pen or a whiteboard with sensing technology) or directly interface with a computer using a tablet or a Tablet PC. Given some participants' desire to use physical space as a tool (e.g., using space to group related documents or arranging them on a large surface), a Tablet PC might not suit all users' needs. Instead, something similar to Paper Augmented Digital Documents (Guimbretiere, 2003) might be more appropriate. With Paper Augmented Digital Documents, users could continue to use paper but also have access to computational power by docking the pen when CAS functionality is needed. At this point, the user could enter into a dialogue with the system to define terms more formally, if necessary, and select the appropriate manipulations. The system could also verify the sequence of operations the user undertook, looking for potential errors at each step. In this way, the system could support a practice already engaged in by our participants—using CAS to validate work done by hand.

Additional inspiration for ways to support flexible input with access to computational power can be found in research on computational tools to support designers, whose work practices share a number of similarities to those observed here. For example, Gross and Do (Do & Gross, 2001; Gross & Do, 1996) found that, similar to our participants, architects go through a phase of exploring ideas followed by an iterative refinement phase, with sketches playing an important role in architects' workflow. They also found that imprecision and abstraction were critical to the exploration and advancement of designs. Their solution, the Electronic Cocktail Napkin, allows architects to continue working in this imprecise, abstract, and free-form fashion. At the same time, the tool also attempts to recognize constraints between diagram entities and maintain these constraints as the architect refines his or her sketch.

Bailey and colleagues (Bailey & Konstan, 2003; Bailey, Konstan & Carlis, 2001) studied designers of interactive multimedia applications, finding reluctance to move to the computational realm despite difficulties in communicating an application's behaviour using physical media alone. Their DEMAIS system allows designers to create free-hand sketches but also to design executable behaviors through a set of gestures. In the area of software design, Damm, Hansen, and Thomsen (2000) found that when creating UML diagrams, programmers want a continuum of formality to choose the level of formality appropriate to the current stage of the design. Their Knight system allows users to work in a free-form sketch mode, with the option of switching to a UML mode. In UML mode, strokes are recognized by the system and converted into formal UML elements. These types of dual-mode solutions are promising avenues for future exploration within the domain of mathematics as well.

Support for Collaboration

Collaboration during the early phases of problem solving is a common practice among our participants. Many results from the fields of computer-supported collaborative work and ubiquitous computing are applicable to address these collaboration

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

needs, with capture and access systems particularly well suited to supporting initial problem-solving work. For example, large surfaces that allow multiple, simultaneous input would be useful to support existing whiteboard work. Coupling capture and access capabilities to these surfaces would help document and share work with others. Commercially available systems such as SmartBoards, or research systems such as the ZombieBoard (which uses a camera and special glyphs to initiate capture; Saund, 1999), are examples of existing systems that could provide such services.

Electronic document sharing and synchronization systems such as revision control systems may also be useful for later stages of problem solving. Although we did not specifically ask participants whether they currently use such systems, an interesting area of future research is to consider how these systems could be catered to the specific needs of mathematical work. For example, there is a question of how best to represent differences between mathematical formulae. Although there is a mature body of research that examines how to display differences between natural language documents and source code, we are not aware of any work that specifically targets the problem of visualizing the differences between mathematical equations.

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

6.3. Study Limitations

There are two aspects of our study design that could impact the generalizability and validity of our findings. The first pertains to the size and composition of our participant group, and the second pertains to our data collection methods.

In this study, we interviewed a group of 20 professional mathematicians who conduct research in a university setting. Although we interviewed two different classes of mathematicians within this setting, it is possible that the work practices and attitudes of our participants are not representative of all expert mathematicians. For example, in industrial settings, users might be less focused on argumentation and more interested in obtaining answers to individual questions. Consequently, there may be less of a need for tool transparency for these users because of a diminished need to formally defend one's solutions. On the other hand, transparency may be equally important in industry to justify a given solution among alternatives, with respect to cost and/or safety concerns.

We also did not encounter any mathematicians through our recruiting who specifically use computers to prove theorems. It is quite possible that mathematicians who do this type of work are more trusting and welcoming of computational tools.

Finally, our sample size was not large enough to systematically study the impact of individual characteristics, such as age, computer experience, or prior training with computational tools for mathematics.

A second limitation of our study is the self-reported nature of the interview data. To address this limitation, we also collected work artifacts to provide evidence that our participants' statements are reflective of their actual work practices. We found the artifacts useful in documenting the existence of different phases of mathematical work and to illustrate the extent to which participants rely on free-form 2D representations with physical media. Other findings, however, rely solely on self-reports, such as

frequency of and reasons for tool use. In-situ observations, experience sampling, and software instrumentation could be used to more precisely quantify and qualify these self-reports.

Given the aforementioned limitations, our study should be considered a first step in understanding use and perceptions of computational tools in professional work practices. Additional studies are required to determine the prevalence of the themes uncovered here.

7. SUMMARY AND FUTURE WORK

Whereas most prior work on computational support for mathematics has focused on novice users still learning basic mathematics techniques, this research has examined the work practices of a collection of mathematical researchers who are experts in their field. The goals of these two different groups—students and professionals—are obviously quite different, which leads to differing requirements from the tools they use. The results of this research are thus important as they suggest distinct use cases and requirements for mathematics software when used by experts: Whereas students are attempting to learn and integrate new knowledge and problem-solving skills, professionals are seeking to *generate* new knowledge. Thus, although there are some commonalities in perceptions of current tools across students and professionals, these commonalities arise for different reasons. For example, both students and professionals would benefit from increased transparency from the tools to better understand how a result was derived. For the students, this transparency helps them to better learn the concepts being taught. For the professionals, the transparency helps them verify the output and demonstrate its correctness to others. Designs intended to support transparency can thus be developed to separately satisfy these two divergent sets of needs.

To summarize the results of our study, we found that computational tools are used very reluctantly and sparingly by the theoretical mathematicians we studied, and more frequently by the engineers. For the engineers, computer-based tools enable them to build and validate models, typically through computational simulation. Computer-based tools also allow them to analyze large data sets, a task that would not be feasible by hand. However, despite these uses of computational systems, we found *both* groups express reservations about relying solely on mathematical software to perform their work. Both groups tend to prefer to prove the mathematical concepts by hand using traditional techniques and argumentation; for neither group is it acceptable to use a result from a computational system as the definitive and only proof of correctness for a particular argument. Instead, results from computational systems are used to complement work done by hand and to provide further evidence of the correctness of one's work. These select uses of mathematics software are borne out of limitations in current offerings—numerical instability, the lack of clear operational boundaries for functionality provided, and the existence of bugs all

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

contribute to a tendency to prefer mathematics performed by hand, rather than through a computer-based system.

For both groups, physical media are also the preferred media for problem solving, as they afford a number of advantages not found in current software offerings. More specifically, current systems lack the ability to create rich annotations, they require a degree of formality and structure unsuitable to early phases of work, and they hide intermediate steps when producing a result.

To address the issues identified in current mathematics software, there are a number of promising avenues for future research in this field, some of which are already being pursued by researchers developing pen-based interfaces to mathematical systems. We also note the potential for research supporting software developers to be fruitfully applied to this problem domain. Like the engineers in our study, software developers are tasked with developing large, complex systems. As such, it is likely that many results from this field of inquiry could be applicable to the design of mathematics software.

In addition to exploring new design possibilities, there are additional areas in need of further research in this space. As discussed in the previous section, little is known about users' needs in industry. Although we expect there to be some commonalities between users' needs in industry and the needs of the engineers in our study, there are likely to be some key differences due to the forces and constraints at play in a commercial environment. Understanding these differences will help develop a richer picture of what types of features are needed for these tools' users. Also, although we did not see any obvious effects of age, gender, or computer training on computational tool usage and perception, a larger study is needed to tease out the role of individual differences. In a similar vein, it would be interesting to see whether providing training with computational tools to illustrate their potential capabilities would lead to more positive perceptions and higher usage.

Finally, it would be beneficial to gather more detailed information about actual software usage across all user groups. For example, open source systems such as Octave or Maxima could be instrumented to collect data to understand common usage patterns. For similar reasons, it would be interesting to collect these types of data from pen-math systems, such as MathBrush or MathPad^2. These data would provide more concrete, quantitative descriptions of use and provide benchmarks for comparing designs and their impact. As an example, instrumentation data could be used to determine whether the increased flexibility provided by pen-based systems leads to more use across the various phases of problem solving. Similarly, if a system provided mechanisms to increase the transparency of the software's intermediate steps, instrumentation data could be used to determine whether users increased usage of the software, or performed fewer tests to verify the correctness of the outputted solution. An observational study could also provide further insight into when and why breakdowns occur both with computational tools and physical media.

Returning to the discussion that introduced this article, computational tools are clearly enabling engineers to do work that would not otherwise be possible to do by hand. For the theoretical mathematicians we studied, current mathematics software

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

plays a more minor role in their work. However, across both groups, much of the fundamental mathematical work is still performed by hand. Although current systems are a boon to some tasks (e.g., simplifying expressions, checking work done by hand), they do not play an overwhelmingly dominate role in mathematicians' work. As such, current mathematical tools both realize and fall short of visionaries' ideals for computer-based tools: They excel at working with large sets of data and modeling complex systems but lack the affordances, features, and robustness necessary to be relied upon for more purely mathematical work.

NOTES

Background. This work is based on an earlier work: *Friend or Foe? Examining CAS Use in Mathematics Research*, in CHI 2009 (ACM, 2009), http://doi.acm.org/10.1145/1518701.1518740

Authors' Present Addresses. Andrea Bunt, E2-445 EITC, Department of Computer Science, University of Manitoba, Winnipeg, Manitoba, Canada, R3T 2N2. E-mail: bunt@cs.umanitoba.ca. Michael Terry, David R. Cheriton School of Computer Science University of Waterloo, Waterloo, Ontario, Canada N2L 3G1. E-mail: mterry@cs.uwaterloo.ca. Edward Lank, David R. Cheriton School of Computer Science University of Waterloo, Waterloo, Ontario, Canada N2L 3G1. E-mail: lank@cs.uwaterloo.ca.

HCI Editorial Record. First manuscript received July 23, 2010. Revision received April 1, 2011. Final manuscript received November 4, 2011. Accepted by Ruven Brooks. — *Editor*

REFERENCES

Anthony, L., Yang, J., & Koedinger, K. (2005). Evaluation of multimodal input for entering mathematical equations on the computer. *Proceedings of CHI 2005 Conference on Human Factors in Computing Systems*. New York, NY: ACM.

Artigue, M. (2002). Learning mathematics in a CAS environment: The genesis of a reflection about instrumentation and the dialectics between technical and conceptual work. *International Journal of Computers for Mathematical Learning, 7*, 245–274.

Bailey, B. P., & Konstan, J. A. (2003). Are informal tools better? Comparing DEMAIS, pen and paper and authorware for early multimedia design. *Proceedings of CHI 2003 Conference on Human Factors in Computing Systems*. New York, NY: ACM.

Bailey, B. P., Konstan, J. A.. & Carlis, J. V. (2001). DEMAIS: Designing multimedia applications with interactive storyboards. *Proceedings of the ACM Conference on Multimedia*. New York, NY: ACM.

Beyer, H., & Holtzblatt, K. (1998). *Contextual design: Defining customer-centered systems*. San Francisco, CA: Morgan Kaufmann.

Borwein, J., Bailey, D., & Girgensohn, R. (2004). *Experimentation in mathematics: Computational paths to discovery*. Natick, MA: A K Peters.

Brandt, J., Guo, P. J., Lewenstein, J., Dontcheva, M., & Klemmer, S. R. (2009). Two studies of opportunistic programming: interleaving web foraging, learning, and writing code. *Proceedings of CHI 2009 Conference on Human Factors in Computing Systems*. New York, NY: ACM.

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

Bunt, A., Terry, M., & Lank, E. (2009). Friend or foe?: Examining CAS use in mathematics research. *Proceedings of CHI 2009 Conference on Human Factors in Computing Systems*. New York, NY: ACM.

Bush, V. (1945, July). As we may think. *The Atlantic Monthly*.

Damm, C. H., Hansen, K. M., & Thomsen, M. (2000). Tool support for cooperative object-oriented design: gesture based modelling on an electronic whiteboard. *Proceedings of CHI 2000 Conference on Human Factors in Computing Systems*. New York, NY: ACM.

Do, E. Y.-L., & Gross, M. D. (2001). Thinking with diagrams in architectural design. *Artificial Intelligence Review, 15*, 135–149.

Engelbart, D. C. (1968). *Augmenting human intellect: A conceptual framework. A research center for augmenting human intellect* (Summary Rep. AFOSR-3233). Menlo Park, CA: Stanford Research Institute.

Gross, M. D., & Do, E. Y.-L. (1996). Ambiguous intentions: A paper-like interface for creative design. *Proceedings of UIST 1996, Symposium on User Interface Software and Technology*. New York, NY: ACM.

Guimbretiere, F. (2003). Paper augmented digital documents. *Proceedings of UIST 2003, Symposium on User Interface Software and Technology*. New York, NY: ACM.

Hartmann, B., MacDougall, D., Brandt, J., & Klemmer, S. R. (2010). What would other programmers do: Suggesting solutions to error messages. *Proceedings of CHI 2010 Conference on Human Factors in Computing Systems*. New York, NY: ACM.

Heid, M. K. (1988). Resequencing skills and concepts in applied calculus using the computer as a tool. *Journal for Research in Mathematics Education, 19*, 3–25.

Labahn, G., Lank, E., Marzouk, M., Bunt, A., MacLean, S., & Tausky, D. (2008). MathBrush: A case study for pen-based interactive mathematics. *Proceedings of SBIM, Eurographics Workshop on Sketch-Based Interfaces and Modeling*. Geneva, Switzerland: Eurographics Association.

LaViola, J. (2007). An initial evaluation of MathPad^2: A tool for creating dynamic mathematical illustrations. *Computers and Graphics, 31*, 540–553.

LaViola, J., & Zeleznik, R. (2004). MathPad^2: A system for the creation and exploration of mathematical sketches. *Proceeding of SIGGRAPH 2004, International Conference on Computer Graphics and Interactive Techniques*. New York, NY: ACM.

Leinback, C., Pountney, D., & Etchells, T. (2002). Appropriate use of a CAS in the teaching and learning of mathematics. *International Journal of Mathematical Education in Science and Technology, 33*, 1–14.

Maplesoft. (2005). Adoption of Maple 10 into academic market exceeds all Expectations. Retrieved from http://www.marketwire.com/press-release/Maplesoft-563405.html

Oviatt, S., Arther, A., & Cohen, J. (2006). Quiet interfaces that help students think. *Proceedings of UIST 2006, Symposium on User Interface Software and Technology*. New York, NY: ACM.

Oviatt, S., & Cohen, A. O. (2010). Toward high-performance communications interfaces for science problem solving. *Journal Science Education and Technology, 19*, 515–531.

Pierce, R., Herbert, S., & Giri, J. (2004). CAS: Student engagement requires unambiguous advantages. *Proceedings of the 27th Annual Conference of the Mathematics Education Research Group of Australasia*. St. Lucia, Australia: MERGA.

Pierce, R., & Stacey, K. (2001). Observations on students' responses to learning in a CAS environment. *Mathematics Education Research Journal, 13*, 28–46.

Quinlan, J. E. (2007). *Profile of software utilization by university mathematics faculty* (Unpublished doctoral dissertation). Ohio State University, Columbus, Ohio.

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013

Robinson, T., & Burns, C. (2009). Computer Algebra Systems and their effect on cognitive load. *Proceedings of the 9th Bi-Annual International Conference on Naturalistic Decision Making.* New York, NY: ACM.

Ruthven, K. (2002). Instrumenting mathematical activity: Reflections on key studies of the eductional use of Computer Algebra Systems. *International Journal of Computers for Mathematical Learning, 7*, 275–291.

Saund, E. (1999). Bringing the marks on a whiteboard to electronic life. *Proceedings Workshop on Cooperative Buildings, Integrating Information, Organization, and Architecture*. New York, NY: Springer.

Schon, D. (1983). *The reflective practitioner: How professionals think in action*. London, UK: Temple Smith.

Tausky, D., Labahn, G., Lank, E. & Marzouk, M. (2007). Managing ambiguity in mathematical matrices. *Proceedings of SBIM, Eurographics Workshop on Sketch-Based Interfaces and Modeling.* Geneva, Switzerland: Eurographics Association.

vanMerrienboer, J., & Sweller, J. (2005). Cognitive load theory and complex learning: Recent developments and future directions. *Educational Psychology Review, 17*, 147–177.

Downloaded by [University of Manitoba Libraries] at 08:44 21 March 2013