Reasoning with computer-assisted experiments in mathematics

Liesbeth De Mol*

1 Introduction

According to some of the advocates of computer-assisted mathematics, "*computers [are] changing the way we do mathematics*". One significant such change is the development of a new branch of mathematics that is characterized by its method rather than by its subject, i.e., *experimental mathematics*. This new domain is still expanding, witness the increasing number of publications on experimental mathematics, the foundation of a new journal called *Experimental Mathematics* or the formation of new research groups like the *Centre for experimental and constructive mathematics*.
But, provided the computer is changing mathematics, what exactly is the scope of this change? In order to answer this question one needs to tackle a variety of different but related epistemological questions. Is a "computer experiment" nothing more than an old method in a new garment that can now be used by any mathematician who knows how to use Maple, or is there more going on? What exactly is a (computer) experiment in mathematics? Is it any different from an experiment in say physics and does it differ in any fundamental way from the way use was made of extensive computations in e.g. the work of Gauss and Euler? What kind of knowledge can or should be obtained through experimentation? How does one process and store the billions of computations and does this result in a fundamentally new kind of tools? These kind of questions are not new to the philosophical literature (see e.g. [1, 2, 17]). Rather than to directly engage into these questions, the approach of this paper is to trace the idea of "computer experiment" in mathematics in the last 60 years. By doing so I want to study if and in what sense experimentation in mathematics has and is changing since the rise of the computer. Such historically-oriented analysis can help to attack the above epistemological questions.

*Centre for Logic and Philosophy of Science, University of Ghent, Blandijnberg 2, 9000 Gent, Belgium, e-mail: elizabeth.demol@ugent.be. This research was supported by the Fund for Scientific Research – Flanders (FWO)

1.1 Experimentation as exploration: Derrick H. Lehmer's view on mathematics

Since the aim of this paper is to study the changes experimentation in mathematics underwent in the era of digital computing, I assume that there is in fact such a thing as an experiment in mathematics and that it existed long before the invention of the computer. Viz. the notion of "experimentation" is used here in the sense of the number theorist and computer pioneer Derrick H. Lehmer, i.e., as the *exploration of the universe of mathematics*.[1] Lehmer identifies two "schools of thought" in mathematics [10]:

> The most popular school now-a-days favors the extension of existing methods of proof to more general situations. This procedure tends to weaken hypothesis rather than to strengthen conclusions. It favors the proliferation of existence theorems and is psychologically comforting in that one is less likely to run across theorems one cannot prove. Under this regime mathematics would become an expanding universe of generality and abstraction, spreading out over a multi-dimensional featureless landscape in which every stone becomes a nugget by definition. Fortunately, there is a second school of thought. This school favors *exploration* [m.i.] as a means of discovery. [B]y more or less elaborate expeditions into the dark mathematical world one sometimes glimpses outlines of what appear to be mountains and one tries to beat a new path. [N]ew methods, not old ones are needed, but are wanting. Besides the frequent lack of success, the exploration procedure has other difficulties. One of these is distraction. One can find a small world of its own under every overturned stone.

For Lehmer it is the possibility of exploration that opens up the path of "*mathematics [as] an experimental science*", a view on mathematics that Lehmer had herited from his father, Derrick N. Lehmer, also a number theorist. Derrick H. Lehmer was also a computer pioneer. He was fond of these new machines and emphasized their significance for mathematics on several occasions. To Lehmer (D.H.), the computer was the perfect ally for the explorative-minded mathematician.

The view that experimental mathematics, if once accepts it exists, involves exploration, does not necessarily mean that the "experimental" reduces to the "explorative". In order to explore one needs something to explore. Since, in mathematics, one does not have the kind of physical world as one has in physics, that which is to be explored not only needs to be made accessible in some or the other way, it also needs to be generated. Indeed, to explore the universe of mathematics also means to generate it. In this sense, experimentation as exploration in mathematics involves at least two different kinds of actions: the generation of the data to be explored and the exploration. Experimentation

[1] See [15] for more details on Lehmer's views on mathematics.

cannot be reduced to either one of both. It entails both.
It is not within the scope of this paper to provide arguments for the view on experimentation in mathematics as exploration. However, it is important to emphasize, first of all, that this definition of experimentation as exploration is not intended as a once-and-for-all definition. Rather it is proposed here is a *working definition*, a tool, which can be used to study this particular aspect of mathematics. Secondly, this view is not exclusively connected to the development of the computer. In fact, as is clear from Lehmer's quote, mathematics as an explorative science is just one of two schools of thought that co-exist. The idea that there has been a second stream in the history of mathematics where exploration rather than theorem-proving is the more important method has been argued for before [7, 8]. In this sense, I disagree with Sorensen's claim that mathematical experiment has changed from number-crunching and fact-gathering to explorative because of the existence of software like Maple: "*[o]nly with interactive experimentation and multi-purpose software [like Mathematica or Maple] could mathematical experimentation make a transition from number-crunching to exploration.*" [17]. The case studies of this paper provide evidence against the significance Sorensen attaches to Maple and Mathematica.

1.2 Human-computer interaction, internalization, time squeezing

If there is one characteristic of computer-assisted experiments in mathematics that separates it from mathematical experiments without computers, it is the fact that such experiments involve interactions and communications between a human mathematician and a digital computer. Hence, if one wants to understand computer experiments in mathematics, how they have evolved and how they differ from mathematical experiments without computers, studying changing mathematician-computer interactions is one good point to start from. This is exactly what this paper aims at. Evidently, changes in such interactions are strongly related to changes and advances in computer science and engineering. This paper will trace down relevant changes in these interactions by coupling them to a changing technology. I will consider four different cases of computer experiments: the first case is about ENIAC, the first electronic and programmable computer, announced to the public in 1946. The other three cases are situated in the late 70s, the 80s and the 90s. The historical gap between case I and case II–IV allow to show the changes that have happened during the history of digital computing more explicitly. I will first give a short description of some relevant features of the cases , and will then evaluate some of the characteristics of the mathematician-computer interactions by zooming in on the location of and tools for storing and processing information during the "experiments". It will be shown that due to what I will call "time squeezing" and "internalization", the "experiment" is more and more becoming an activity that is located in and during mathematican-computer interactions.

2 Description of the four cases

2.1 Case I: The ENIAC experience

Any historically-rooted study of computer-assisted mathematics should at least consider the mathematics done by and on ENIAC, the first electronic and programmable computer.[2] This machine, whose (historical, philosophical and scientific) significance is largely underestimated in the literature, has had a fundamental influence on the thinking of logicians, engineers and mathematicians (See [6, 16]). Among the people who were involved with these machine are Haskell B. Curry, the logician whose first name refers to a well-known functional programming language, Derrick H. Lehmer, the number theorist, John von Neumann, mathematician, logician and computer pioneer and Douglas Hartree, the father of numerical analysis. The machine was the first electronic computer used for an extensive number-theoretical computation and is most well-known for its numerical modeling of a fission device for an H-bomb [13].
ENIAC's speed was impressive for its time: it could add two numbers in 1/5000 of a second. It had however two major bottlenecks: its memory size and the labor involved in programming the machine. The only way to provide extra memory was through the external and analogue punched card mechanism, which seriously slowed down the computational process. Unlike modern computers, the ENIAC was not a stored-program computer. This meant that the programming needed to be done externally and physically. Add to that the fact that the machine was highly parallel and one understands Adèle Goldstine's characterization of the machine as a "son-of-a-bitch to program" [12]. Furthermore, since there was only one ENIAC and one needed a security clearance to visit the machine (because it was military) only few people had access to the machine.
I will mention one ENIAC computation here: the computation of exponents modulo 2 by Lehmer and his wife Emma. This computation was intended as a test of ENIAC's mathematical possibilities.

The idea was to use it to compute a list of exponents e of 2 modulo a prime p, i.e., the least value n such that $2^n \equiv 1 \bmod p$. Such exponents can be used to compute exceptions to the converse of Fermat's little theorem.
The set-up on ENIAC is interesting for several reasons. First of all, the main algorithm for computing exponents is developed from the machine's point of view, viz. not the most elegant and fast one from the perspective of human mathematicians, but the fastest one for the machine. Secondly, a truly parallel prime sieve was wired on ENIAC hence allowing it to compute its own primes rather than to feed such primes to the machine during computation. The sieve "screened out" all numbers having a prime factor ≤ 47. This meant that there was no guarantee that the numbers that passed that sieve were prime. In other words, this is an early example of what one calls a probabilistic algorithm. In

[2]The programmability of ENIAC is not well-known. Note however that it contained everything one needs for computing the basic arithmetic functions, iteration and conditional branching. Hence, in principle it could be programmed just as any modern computer can be programmed.

order to minimize the chance that a number passing the sieve was prime an extra test was internalized into the machine, and, finally, all were checked by hand at the end of the computation. Finally, it should be noted that ENIAC was only used for he computation of the exponents, not for the computation of the exceptions to Fermat's little theorem. This was done by hand, which was still a considerable labor [9]. The ultimate goal of the computation, the extension of existing prime tables to be used by the exploratory minded number-theorist was still a purely human work, far removed in time from the actual ENIAC computation.

2.2 Case II: Mandelbrot and his set

As explained in the previous section, ENIAC was not a stored program computer. Stored programming was the material condition needed to make possible the development of intermediate languages between the programmer and the computer, viz. the development of programming languages. In the 50s and 60s the first real high-level programming languages like LISP, FORTRAN and Algol were developed. However, computers were still very expensive and hence also computer time. This meant that the computer was not available to many people. The development and commercialization in the late 60s of time-sharing systems was the next big step forward. Also displays and printers were becoming more and more advanced. In the meantime computers had become exponentially faster and were no longer slowed down by the analogue external storage devices used with ENIAC. It is in this technological context that Benoît Mandelbrot "discovered" in 1980 the set that was later named after him, viz., the Mandelbrot set.
These "experiments" were done on the DEC-VAX-11/780 a stored program computer with the OS VMS with GUI, graphic support and support for multiple programing languages. Mandelbrot could also "look" at visualizations on the display and print them with a printer.
In the late 70s Mandelbrot had developed an interest in the theory of rational maps of the complex plane. Based on his knowledge of the work by Fatou and Julia he started to "play around" with quadratic Julia sets which are defined as the boundary of the escapee sets of iterative maps:

$$z \to z^2 + c, \quad c, z \in \mathbb{C}$$

It was known through the work of Julia and Fatou that there exists a fundamental dichotomy for Julia sets: either a Julia set is connected or disconnected. It was also know that for certain values c, any point z converges to a finite stable cycle of size n. Mandelbrot aimed at classifying the connected Julia sets according to the size of such cycles n. In order to do see he decided to use another known fact: the prisoner set P_c for the map $z \to z^2 + c$ is connected iff. the orbit $0 \to c^2 \to c^2 + c \to \ldots$ remains bounded. This was the motivation behind his explorations of the map $c \to c^2 + c$ which lies at the foundation of

the Mandelbrot set M. Indeed:

$$M = \{c \in \mathbb{C} | c \to c^2 \to c^2 + c \to \ldots \text{remains bounded}\}$$

The idea of exploring this set should be taken quite literally: the map was visualized, printed, studied. The observations made led to new visualizations, blow ups of a certain detail of the set for example, etc. These first explorations led Mandelbrot to several important conjectures *during* the process of exploration. Many of these conjectures have been proved in the meantime. To give just one example recounted by Mandelbrot, in some of the first print-outs there seemed to be specks of dirt, and it was thought which could have been easily explained away as being caused by the quality of the Versatec printer. However, Mandelbrot saw a certain symmetry in these specks and wanted to see a blow-up (which meant writing a new program!) and saw that these were indeed not specks of dirt but smaller"islands" as Mandelbrot calls them, whose shape is like that of M. This observation ultimately resulted in the conjecture that M is connected, proven by Douady and Hubbard in 1982.
Mandelbrot's work on M can perhaps best be described as machine-aided, human-directed explorations. The computer and its computational power is used to render the Mandelbrot set observable: it generates the data needed, orders them and displays them in a format that is humanly practical. In this observational process, it is fundamental to point out that one picture is not enough. As Mandelbrot explains *When seeking new insights, I look, look, look and play with many pictures (One picture is* never enough)." [11]

2.3 Case III: Brady and Busy Beavers

Our next case study concerns what one would nowadays call a computer-assisted proof, i.e., the proof that the Busy Beaver function for Turing machines with 4 states is 13. This result was established by Brady [5] in the early 80s and by Kopp at around the same time. Evidently, the computer technology Brady could rely on is comparable to that Mandelbrot could rely on.
The Busy Beaver problem as originally formulated by T. Radó, is the problem to determine for any class of Turing machines $TM(m, 2)$ with m states and 2 symbols the maximum number of 1s left on the tape by some $T \in TM(m, 2)$ that halts when started from a blank tape. Even though this problem is recursively unsolvable in general, one can try to solve it for specific m. This is what Brady did for values of $m = 4$. However, since the number of Turing machines with m states is equal to $(4m + 1)^{2m}$ one needs to check no less than 6,975,758,441 machines for $m = 4$. This is why the computer is a necessary tool here. The general idea of the proof is to decide the halting problem for all machines in a given class when started with a blank tape, i.e., to exclude those machines that end in an infinite loop. In every stage of a proof, the machines for which it is not yet known whether or not they will halt are called the holdouts.
In order to tackle this problem, Brady was able first to reduce the 6,975,758,441 machines to only 5820 by way of tree normalization and backtracking (see [14]

for more details). The resulting set of holdouts is further reduced by way of a heuristic and experimental process of identification of different types of infinite loops (through exploration of hundreds of printouts of the behavior of different machines) and the automated detection of the several types of loops found. Each Turing machine is "sent" through what one could call a flowchart of flowcharts. I.e., a flowchart that is a composition of the different types of algorithms (which could also be represented as flowcharts) developed for detecting the different kind of loops. Several of the programs used in the process are "heuristic". For example, one of the main "filtering" procedures called BBFILT *tentatively* classifies a machine in one of three classes. Brady described BBFILT as follows: *[I]t must be remembered that the filtering [BBFILT]was a heuristic technique based upon experimental classification.*
Besides the several different loop detecting programs (12 in total):

> [m]ore than 18 other programs were written, for various housekeeping purposes, simulating and displaying machine behavior, exploring other reduction and filtering possibilities etc. In all, at least 53 files were created and maintained for the project. Keeping track of what resembled a large scientific experiment became a major task in itself

2.4 Case IV: Wolfram, Mathematica and "A new kind of science"

The last case concerns Stephen Wolfram's work on cellular automata. Roughly speaking, his research can be divided into two periods of time. A first period going from 1982-1988 in which Wolfram published his main results and ideas on cellular automata (CA) and the period after 1988 when *Mathematica*, a specialized software package for scientific research comparable to *Maple*, was first released. It was also around that time that Wolfram started to write on his *A new kind of Science*, a voluminous book published in 2002 as [19].
For his earlier work in the 80s on CA Wolfram used several different computers and programming languages and developed his own "software" for the special needs of his research. His research on CA was published in the early 80s and later published as the collection [18]. Similar to Mandelbrot, Wolfram established several observational results of which some were proven later. One important result, based on thousands of runs of different CA, is the classification of the behavior of CA into four classes of behavior. Wolfram's observations also led him to the formulation of interesting philosophical points of view, relating his theoretical results to the physical world. For instance, the idea that the complexity and randomness we see in nature can be explained from within the framework of CA.
Interestingly, Wolfram had already found most of his basic results on CA before the release of *Mathematica*. It were these results from computer science that lay the foundation for the development of a more general theory described in *A new kind of science*. It was also the extensive programming involved, that led him to the development of *Mathematica*. Indeed, he first conceived of *Mathematica*

because he needed it himself. On the website of *Mathematica* one finds the following description:

> [T]he visionary concept of Mathematica was to create once and for all a single system that could handle all the various aspects of technical computing – and beyond – in a coherent and unified way

The idea behind Mathematica is to offer a platform that integrates and centralizes the many different methods and techniques one needs for scientific computing: visualizations, statistics, random generators, algebraic functions, etc are all offered as one package. There is no need to lose time by writing your own visualization algorithm, you merely have to know the right name for it in Mathematica.
In 2002 Wolfram's long awaited book was finally published. It presents his results on CA into one coherent theory and extends this theory to problems in physics, cognitive science, etc He also builds up an argument in favor of his methodology of computer experimentation and visualization, criticizing the traditional methods of mathematics.

3 Changing mathematician-computer interactions. Evaluation of the cases

If we compare the four cases of Sec. 2, it becomes apparent that there is a world of difference between the ENIAC "experiments" and Wolfram's "experiments". In order to get a firmer grip on these changes in mathematical experimentation, it is interesting to evaluate the processes of mathematician-computer interactions for these different cases in relation to the location and the methods for storing and processing information during the "experiment".
Turning to ENIAC, it is interesting to point out that the actual machine computations – the computation of exponents – are long computations that do not involve what one could call "responsive interaction". I.e., what one typically has is that the ENIAC makes one very long computation and the mathematician awaits the result (although sometimes a human intervention is needed, for instance, to reverse the punched cards). Once the result is there, the mathematician does the rest of the work and the exploration of the data can be started. One could wonder in what way then the ENIAC computations discussed here are different from hand-made computations of exponent tables or digits of π or e that were published before? First of all, there is the fact of the speed of ENIAC – whereas it would normally take weeks to compute these first 2,000 digits, it now took about two days. The significance of that kind of "*time squeezing*" should not be underestimated. Secondly, there is the fact that even though there is just one long computation, this computation has many different stages that are all *internalized* and *centralized* into the machine making up one general program/flowchart. I.e., the machine is wired such as to "combine" and "centralize" different kinds of computations into one long computation. This

sequencing and combining of smaller algorithms into one would challenge many a logician and computer scientists in the years to come, Haskell B. Curry being one of the first to think about the *composition of programs* [16]. Thirdly, the algorithms underlying the computation are *machine-adapted*.The fact that Lehmer chose to let the machine generate its own primes because of the speed gained or his design of the "idiot" exponent routine are clear examples of that. Finally, motivated in part by the slowness of the analogue punched card input, one sees a steady replacement of storing large amounts of data which can be used in the course of a computation by what one could call the "*call-by-value*" method, the idea that the machine generates values when they are needed rather than to look them up. This is a partial *internalization* and *compression* (space squeezing) of data: instead of storing the data, the algorithm that generates them is stored. However, when data are needed that are not easy to generate with the machine, they are still stored externally to the machine. This slows down the process of computing and also often implies the need for integrating human action into the algorithm.

These features indicate that even though the ENIAC "experiments" are very different from before, the experimental process – involving the generation and exploration of data – is (relatively) one-dimensional and discontinuous, with clearly separated and consecutive phases in time, distributed discontinuously between the mathematician's work and the computer's work. The mathematician (in collaboration with one or more ENIAC programmers/engineers) translates a problem to the ENIAC, the ENIAC computes the different aspects of the computation (sometimes interrupted by a human action to make the process going) and then the mathematician can start to work on the data, processing and exploring them. The "experiment" is (relatively) strictly separated into computer phases and a human phases.

This is very different if one looks at mathematical computer-assisted experiments in the 70s and beyond, when the computer technology has changed significantly. To start with Mandelbrot's early generations and explorations of M, it is clear that a more advanced technology results in important changes on the level of the processing and storage of data. First of all, contrary to ENIAC, thousands of data are now stored *internally* into the computer, a necessity for the visualization of M. Also, whereas in the case of ENIAC the amount of data outputted were still manageable for humans, these data now need to be preprocessed before they are "delivered" to the human mathematician, they are ordered into a 2D surface to provide a finite approximation of the visual representation of M. This creates a need for the development of feasable visualization algorithms. The one used in the early explorations of M is probably the encirclement algorithm which works through iterations on internally stored data. It is also worth noting that Mandelbrot's explorations resulted in the use of concepts and a terminology that is directly inspired by the visual appearance of M (like the use of "islands") In this sense, the pictures of M could be regarded as an interface between the mathematician and the computer. These different features result in a changed interaction between the mathematician and the computer and hence a changed way of "experimenta-

tion". Most importantly, it is now no longer the case that the computer makes one long computation after which the mathematician takes over and does the exploration. Rather one now has a whole series of interchanges between the computer computing parts of M and representing them visually to the mathematician and the mathematician studying and exploring these visualizations, formulating conjectures, finding proofs and translating his requests and questions back to the machine. Indeed, as Mandelbrot emphasized: "One picture is never enough!" This process of back-and-forth computing and exploring develops in time. These explorations and computations as a flow of information between the mathematician and the computer, are squeezed into a "reasonable" time frame. The exploration and generation of data are not strictly separated into two phases, but are becoming part of the interaction, interfaced by the visualizations and the programming. This results in an increased involvement of the machine in and during the experimental process itself.The "experiment" is now much more a mixing of exploration, computation, communication, interaction and interpretation.

The example of the Busy Beaver proof leads to a similar conclusion. However, there is one fundamental difference with the Mandelbrot case: whereas in that case, the machine only needed to do one thing – visualize approximations of M – the computer now does many different things: it simulates and displays machine behavior, it detects and explores the behavior of the different Turing machines, etc. This results in what one could call a flowchart of flowcharts.

Also in this case, phases of computation and exploration are squeezed into the process of the mathematician-computer interaction. However, the inspection and exploration is now no longer strictly located with the human. Hence, the mixing of exploration and computation is further increased. The result is a highly continuous human-computer interaction, the exploratory activities now being distributed between the two in the flow of time. The involvement of both the computer and the mathematician *during* the process of experimentation intertwines the human and the machine contribution to the proof which is "found" in and during the interaction.

The case of Wolfram is in a certain sense very similar to and very different from the Busy Beaver case. If one considers Wolfram's work on CA from the 80s one also sees that the computer is used for more than just the visualization of CA. It is used in many different phases of the research on CA and for many different things going from the simulation of CA to statistical analyses. There is one important difference however: whereas in the Busy Beaver case, the computer was used to prove a Busy Beaver case, Wolfram uses the computer for the development of a theory of CA and ultimately another kind of science which is characterized by a method of computer-assisted explorations.

Wolfram's attempt to formulate a unified and integrated theory as one finds it described in *A new kind of science* heavily relies on *Mathematica*. The parallel between *Mathematica* as a software package that uses one "language" to deal with several aspects of computing in a unified and coherent manner and the idea of a single framework/language (that of CA) to unify and integrate the various aspects of science (cognitive science, physics, biology, etc), is striking.

This becomes the more striking when relating this to the proposal by Bailey and Borwein of a "new kind of" mathematics, viz. experimental mathematics presented in their two volumes [3, 4] and the extensive use they have each made and make of Maple in their research.
In these software packages one sees an increased internalization of the methods used by the mathematician into the machine: statistical analysis, methods of symbolic algebra, simulation of computational models, visualization, etc are all techniques that are turned into algorithms that can be called by their name and need not be known. The mathematician now has a tool box without the need to know the techniques and knowledge behind the tools. This affects the process of mathematician-computer interaction and experimentation: first of all, the time squeezing is further increased since one no longer needs to waste time in developing and programming algorithms that allow to use these "tools". This results in faster interactions and a more direct interaction with the "(simulated) object" or problem studied. I.e., the length of time during and after the generation of the object by the computer (for instance M) and its computer-assisted inspection (for instance by zooming in and out) is becoming smaller and smaller, the most extreme case being "real-time" explorations in which the computer is completely hidden away. The wider accessibility and integration of algorithmic tools ultimately leads to the possibility of formulating integrated and general theories on the basis of computer-assisted experiments. Hence, the process of time-squeezing and internalization that started with ENIAC and which is to some extend still quite similar to "experimental" mathematics before the rise of the computer, has resulted in a way of experimentation that is located more and more *in* the mathematician-computer interaction, where the generation and exploration of the "universe of mathematics" is more and more squeezed into a mixed process of mathematician-computer interaction.

4 Discussion

As is clear from the case studies, one cannot claim, as Sorensen does [17], that mathematical experimentation only became truly explorative once software like Maple or Mathematica became available. Indeed, there is nothing fundamentally new that explains a transition from fact-gathering or number-crunching to "truly explorative" mathematics. Cases III and IV, evidently show that the computer was already used for much more than mere number-crunching long before the development of these packages. Furthermore, the point of view of this paper is that experimental mathematics involves two different actions: the generation of data (fact-gathering?) and the exploration of these data. Seen from that perspective, there is no reason to assume that there was no exploration before the development of Maple or even the development of the digital computer.
It was shown here that if one goes from "experimentation" on ENIAC to software like Mathematica, one cannot deny that there have been fundamental changes in the interactions between the mathematician and the machine. Whereas with

ENIAC the interaction is still highly discontinuous and linear, the (machine) computation phase being strictly separated from the (human) phase, this interaction becomes a mixed ensemble of computation and human and machine explorations, the typical features of the more experimental sides of mathematics like conjecturing or concept formation being mixed into the interaction happening at some interface. These changing interactions are considered here as one way to characterize mathematical experiments in the era of digital computing.
Two major concepts have turned out very useful in this study of the impact and significance of the computer for experimental mathematics: time squeezing and internalization, two concepts that are rooted in the fundamental characteristic of experimental mathematics on the computer: mathematician-computer interactions. The fact that one only needs hardly a second nowadays to compute something that would have taken weeks or months combined with the possibility of a feasible computer-enhanced exploration as a consequence of an increased internalization, are major advances in mathematics and their impact can and should not be underestimated. It squeezes the universe of mathematic and its exploration into a feasible amount of time.
Perhaps the computer is inspiring a steady return to what Lehmer called the second school of thought in mathematics. However, if this is the case, this is not because of the development of Maple and Mathematica, but rather because of the ongoing process of time squeezing and internalization together with the wider availability of the computer to the mathematician. The fact that this possible return is accompanied by what one could call a "reinjection" of time into mathematics turns a study of the impact of the computer on mathematics into a true challenge not only for the historian but also for any philosopher of mathematics who wants to study mathematics as a product of the human mind (and not as absolute).

References

[1] Alan Baker, *Experimental mathematics*, Erkenntnis **68** (2008), no. 3, 331–344.

[2] Jean-Paul Van Bendegem, *What, if anything, is an experiment in mathematics*, Philosophy and the many faces of science (D. Anapolitanos, A. Baltas, and S. Tsinorema, eds.), Rowman & Littlefield, 1998, pp. 172–182.

[3] Jonathan Borwein and Davis Bailey, *Mathematics by Experiment. Plausible reasoning in the 21st century*, AK Peters, Wellesley, MA, 2003.

[4] Jonathan Borwein, Davis Bailey, and Roland Girgensohn, *Experimentation in Mathematics: Computational paths to discovery*, AK Peters, Wellesley, MA, 2004.

[5] Allen H. Brady, *The determination of the value of Radó's noncomputable function σ for four-state Turing machines*, Mathematics of Computation **40** (1983), no. 162, 647–665.

[6] Maarten Bullynck and Liesbeth De Mol, *Setting up early computer programs. D. H. Lehmer's ENIAC computation.*, Archive for Mathematical Logic **49** (2010), 124–146.

[7] Leo Corry, *Hunting prime numbers from human to electronic computers*, The Rutherford Journal - The New Zealand Journal for the History and Philosophy of Science and Technology **3** (2010).

[8] Catherine Goldstein, *How to generate mathematical experimentation and does it provide mathematical knowledge?*, Generating Experimental Knowledge, PMPIWG, Max Planck Institute for the History of Science, pp. 61–85.

[9] Derrick H. Lehmer, *Maurice kraitchik, recherches sur la Th'eorie des Nombres, v. 1, paris, 1924 (errata)*, Mathematical tables and other aids to Computation **2** (1947), no. 19, 313.

[10] Derrick H. Lehmer, *Mechanized mathematics*, Bulletin of the American Mathematical Society **72** (1966), no. 5, 739–750.

[11] Benoit Mandelbrot, *Fractals and chaos. the mandelbrot set and beyond*, Springer, 2004.

[12] Scott Mccartney, *Triumphs and tragedies of the world's first computer*, Walker & Company, 1999.

[13] Nicholas Metropolis, *The beginning of the monte carlo method*, Los Alamos Science (Special Issue, Stanislaw Ulam 1909-1984) **15** (1987), 125–130.

[14] Liesbeth De Mol, *Looking for Busy Beavers. a socio-philosophical study of a computer-assisted proof*, College Publications, Accepted for publication in: K. Francois, B. Löwe, T. Müller and B. van Kerkhove (eds.), Foundations of the formal sciences VII.

[15] Liesbeth De Mol, *Doing mathematics on the ENIAC. Von Neumann's and Lehmer's different visions.*, Mathematical Practice and development throughout history (E. Wilhelmus and I. Witzke, eds.), Logos Verlag, 2008, pp. 149–186.

[16] Liesbeth De Mol, Maarten Bullynck, and M. Carlé, *Haskell before Haskell. Curry's contribution to programming (1946-1950)*, CIE10 (E. Mayordomo L.-M. Gomes F. Ferreirea, B. Löwe, ed.), LNCS, vol. 6158, Springer, 2010, pp. 108–117.

[17] Henrik K. Sorensen, *Exploratory experimentation in experimental mathematics: A glimpse at the pslq algorithm*, PhiMSAMP (Benedikt Löwe and Thomas Müller, eds.), College Publications, 2010, pp. 341–360.

[18] Stephen Wolfram, *Cellular automata and complexity. collected papers*, Addison-Wesley, XX, 1994.

[19] Stephen Wolfram, *A new kind of science*, Wolfram Inc., Champaign, 2002.