Programmed mathematics

Liesbeth de Mol

"PROGRAMMED MATHEMATICS"[1]

Liesbeth De Mol

I. Introduction

In 1976, the renowned historian of mathematics, Kenneth O. May, was invited as a keynote speaker to an influential conference on the history of computers that gathered an international set of well-known people from the field of computing including, amongst others, Stanislaw Ulam, Friedrich Bauer, Donald Knuth and Andrei Ershov. The aim was to present their thoughts on and contributions to the short history of the field of computing and to raise awareness of the significance of writing their histories. Since many in computing were still looking into the direction of mathematics and formal logic as a basic discipline (if not foundational) for their field, it made sense to invite a historian of mathematics and to hear about his views. His talk was titled *Historiography: A perspective for computer scientists* and one of its purposes was to emphasize the significance of the computer for the history of mathematics and to integrate the history of computing into the history of mathematics.[2]

Interestingly, May did not view the computer as just another technology belonging to the field of so-called applied mathematics. Quite on the contrary, in his understanding, "[t]*o say that […] [the computer is] one of the most important inventions in mathematics or in the history of mathematics would be an understatement*" (May 1980). This fitted with a view that opposed an idea of applied versus pure mathematics: according to May it was a myth, a construction, that there is some part of mathematics that is disconnected from the world and purely theoretical and a part that is only applied without theory. Instead he proposed another distinction: that between mathematical science and mathematical technology with both containing more theoretical and more practical developments. The latter included an algorithmic tradition (which he called the software side of mathematical technology) and one that concerns calculating devices (on the hardware side). It was then because of the computer that mathematical technology has gotten the place it deserves in the history of mathematics (May 1980):

> "[Mathematical technology] developed steadily in history without an elaborate literature, so that it doesn't appear in the historical record as a central thing. […] If we look at the history of mathematics this way, it seems to me that one of the effects of the coming of the electronic computer is that for the first time mathematical technology has become a consciously recognized discipline [...] instead of just being for practitioners who didn't seem to be in the mainstream, computing now becomes the Queen of Technology.

May's remarks go into two directions, attracting the history of computer science in the realm of the history of mathematics (which did not work out, see the last section), but also enlarging the focus of historians of mathematics to mathematical technology and computing. It is the aim of this chapter to focus on this second direction in order to evaluate the role of (modern) computing as a mathematical technology: how do the histories of computing and mathematics intertwine (or not) and what does this tell us about (the history of) mathematics itself and how it is perceived today?

Of course, it is not possible within the scope of one chapter to tell the complete histories and so the focus here will be on overall developments in the 20th and early 21st century of a (changing) relation between, on the one hand, mathematics – its practice, its self-perception and its results – and, on the other, computational technologies and the academic fields which are anchored in it historically. That history is very broad. In order to give focus in this chapter, I will concentrate mainly on the U.S. It should be emphasized here that this does not mean that there are no interesting histories to write in relation to other national and cultural contexts and their relations to the U.S. (see for instance, (Gerovich 2002) for Russia and (Mounier-Kuhn 2011) for France). The chapter

---

1 This draft is an earlier version of the chapter: Liesbeth De Mol, Mathematics and Technological change. Tom Archibald; David E. Rowe (eds.). *A Cultural History of Mathematics in the Modern Age, vol. 6*, Bloomsbury Academic, 2024

2May died unexpectedly in 1977 the paper was constructed by his colleagues on the basis of the recording of his talk at Los Alamos.

will develop along four major topics: the early history of computers in relation to mathematics (sec. 1); the use of computational technology within mathematics (Sec. 2); mathematics as a tool for computation (Sec. 3) and, finally, the changing disciplinary identity of mathematics from the perspective of computational technologies (Sec. 4).

Mathematical Tables and other Aids to Computation: a rapprochement between mathematics and computation

While the rise of the modern computer should not be reduced to developments in mathematics (Campbell-Kelly et al 2014), it is also clear that the field was an important driving force for the early history of large- scale machine computation. So, how is it that mathematics and mathematicians became involved with the making and use of this technology?

One important impetus came from an increased need for mathematics research in the context of military science (see also Chapter 2 of this volume). In that context it became clear that brute-force computation would be a necessary tool to develop and deploy ever more sophisticated weapons. Military science in the US became more important during WWI as is witnessed by the creation of the National Research Council in 1916, an initiative of the National Academy of Sciences, to serve as an advisory board for scientific matters relevant to the state. For mathematics specifically, there were a number of influential figures, like G.D. Birkhoff, O. Veblen and F.R. Moulton, who got involved with military applications (Archibald et al, 2014). During WWII the US mathematical community was mobilized to getting involved with the war effort amongst others through the establishment of the joint War Preparedness Committee of the *American Mathematical Society* and the *Mathematical Association of Americ*an and, later, the *Applied Mathematics Panel* (AMP). The latter was led by Warren Weaver with Mina Rees as his executive assistant. Its purpose was *`to bring mathematicians as a group more effectively into the work being carried on by scientists in support of the nation's war effort.'* (Bush, Conant, Weaver 1946, p. vii). That meant that most of the work of the AMP was requested by Army, Navy or other NDRC agencies, that is, mathematics in the service of others. By the end of the war, the AMP had undertaken around 200 studies, half of which were upon direct request from the military. That computation was considered a major method in this context is clear from the fact that from the four major topics mentioned in a series of three AMP summary reports from 1946, one is devoted to computational services. These include the making of tables and charts, the evaluation of integrals in several settings (going from the computation of ballistic trajectories to the shape of shells) and research into the potential of computing equipment.

Of particular interest here is the establishment of the Aberdeen Proving Ground (APG) which was the first U.S. Proving Ground and constituted just a few months after the US had entered World War I. It was here that weapons were being designed and tested and it played a major role in bringing closer mathematics and military applications. In 1918, Veblen was put in charge of experimental ballistics with the aim of producing firing tables. These tables were used in the field to aim fire. The trajectory of a shell is determined by a large number of factors like the density and temperature of the air, the wind velocity, the elevation of the gun, etc. Firing tables contained the necessary informations related to these different parameters which could then be used to aim the firing device correctly.

Such tables had to be computed for ever new weapon and so, as the military efforts of the US increased, there was a growing need for such tables. The computation of these firing tables at APG and, more specifically, its Research Department – which was renamed the Ballistic Research Lab (BRL) in 1938 – became a major contributor to computational technologies not just with respect to the technology itself but also with respect to the development of improved mathematical methods (Aspray 1990). Steadily it became a hub for work with and exploration of computing technology

and computational methods, attracting a diversity of mathematicians as consultants including, amongst others, Haskell B. Curry, John von Neumann and Isaac Jacob Schoenberg.

At the BRL, computation of firing tables relied mostly on human computers, aided by desk calculators, and, in the 1930s, computations coming from two differential analyzers and which were modeled after the differential analyzer by Bush and Caldwell developed at MIT. Today, we would characterize these as analog computers, that is, calculators that work with measurement rather than with numbers. By the late 1930s these had achieved a large generality in their ability to solve *any* differential equation that fitted the size of the machine and so they were well suited to solve the basic differential equations involved in the computation of ballistic trajectories.

However, at BRL, even with the help of the differential analyzers, one could not keep up with the demands for firing tables. Thus, when in 1943, John W. Mauchly from the Moore School of Electrical Engineering – an institute in close contact with BRL – made a first proposal to the army to build an electronic high-speed machine to compute firing tables, it was accepted quite quickly. This was the start of the construction of one of the first large-scale electronic computers that became known as ENIAC. It took until 1946 to complete the machine. Even though it was too late to serve its initial purpose, it was realized that the ENIAC could be used for a host of other applications. These included computations for number theory (by Derrick and Emma Lehmer), computations for weather prediction in the context of the Meteorology Project (led by Jules Charney) and the famous Monte Carlo calculations used in the design of nuclear weapons (led by John von Neumann who was deeply involved with the Manhattan project – see Chapter II of this volume). This approach relied on the use of random numbers to look at the (simulated) paths of a large number of neutrons determined by fission, scattering, absorption or escape in order to get a statistical picture of their behavior. As will be shown later, the introduction of high-speed discrete computation would have a major impact on the numerical methods used.

But while the increased relevance of computation for military science was certainly an important force that accelerated developments related to computational technology this was rooted in a more general realization of the potential of computational methods. Indeed, in the first half of the 20th century, one sees the rise of a number of (large-scale) scientific projects that relied on extensive computation, often through the production of numerical tables (Grier 2005) which need to be contextualized in a longer tradition of the making and using of numerical tables (Campbell-Kelly et al 2003). Within these projects one often relied on calculating and other technical devices that allowed to make more efficient the computational process. These included, amongst others, desk calculators, tabulators and assemblies of smaller interconnected calculators.

These mechanized computations often went hand-in-hand with computational work by (teams of) so-called “computers”, in some cases, organized by a principle for the division of labor (note that our current notion of “computer” thus derived from humans computing). Some well-known examples of larger projects are: the project at the *Nautical Almanac Office* in the UK which under the direction of Comrie in the 1930s introduced desk calculators; the *Mathematical Tables project* in New York led by Arnold Lowan and Gertrude Blanch and which relied almost exclusively on a division of labor principle with over 100 computers hired at a certain point; the astronomical work at the *Astronomical Computing Bureau,* led by Wallace J. Eckert and which later became the *J. Watson Scientific Laboratory for astronomical computations,* recognizing the support from IBM and Thomas J. Watson in particular. Once te first large-scale machines were built, these were often perceived as automations of the work of the human computer. For instance, as Maurice Wilkes pointed out in the context of the EDSAC (see below):

> “With the EDSAC the analogy between the machine and a computor is a close one in that the problem is presented to the machine in the form of a series of orders […] In a metaphoric sense the machine may be said to understand the same language as a computor” (Wilkes, M. 1949. “Programme design for a high-speed automaic calculating machine”, *Journal of Scientific instruments*, 26, pp. 217–220).

These kind of practices got more organized and contributed to establishing computing as a valuable scientific method. That process was aided by the establishment of organizations like the *Mathematical Tables Committee* in the UK or the *National Research Council Committee on the Bibliography of Mathematical Tables and other aids to Computation* (Grier 2005). Under the leadership of Raymond C. Archibald the latter founded the journal *Mathematical Tables and other aids to Computation* (MTAC) in 1943. Its purpose was to report on developments in mathematical tables and other computational techniques. Very soon, in the late 1940s, it also played a key role in reporting on work done in relation to the new large-scale computing machines. It even became the recommended journal of the *Eastern Association for Computing Machinery* (EACM), later the ACM, before it started its own journal in 1954. In the 1950s, interest in MTAC declined. This, ironically, was due, at least partially, to the rise of the modern computer: precomputed values in mathematical tables had become obsolete since, with these computers, one could quickly compute values as they were needed (see below) and so attention shifted more to the mathematics of computation itself. In 1959, MTAC was renamed and became *Mathematics of Computation.* Today, topics covered by the journal include: numerical analysis and computational discrete mathematics.

Despite the fact that the relevance of mathematical tables steadily declined with the rise of the modern computer, many of those involved with such table making and other computational projects became pioneers for the modern computer and computing. Archibald, through the establishment of MTAC, is one such example. But there are others who contributed to the construction and use of large-scale computing machines. Eckert, for instance, who already worked with IBM in the 1930s for his astronomical computations, became the director of the IBM SSEC project – a semi-electronic large-scale computer tat was built by IBM after Watson had a fallout with Howard Aiken.

Aiken was a Harvard graduate student of electrical engineering who realized the need for extensive computation of differential equations to study the behavior of electrons in vacuum tubes. Even though Harvard University – where there was a longer tradition of human computational practices via the Harvard Observatory – could pay salaries of human computers to attack such problems, Aiken believed that such help would not be enough for his problem and so he started to consider the possibility of building a machine to attack such problems. IBM decided to help in the construction of such a machine and so made available its machines and engineers to construct the ASCC, the Automatic Sequence Controlled Calculator. This referred to the fact that, unlike desk calculators, this machine should be able to automatically carry out a sequence of (coded) mathematical operations which were “read” by the machine from a punched tape. The ASCC soon became known as the Harvard Mark I and was basic to the history of the first computers. Its nickname was Bessie because it was used, amongst others, to compute tables of Bessel functions (some fifty volumes of Bessel functions were published). It was also used by the US Navy under the leadership of Grace Hopper, who got her PhD from Yale in mathematics in 1934 and who became a major contributor to automatic programming in the 1950s.

Jean Bartik was hired at BRL as a human computor and then became one of the many women involved with programming the ENIAC. She also made important contributions to the conversion of the ENIAC into a kind of stored-program machines (rather than wiring it directly it became possible to code porgrams relying on the order code) and later moved to work with UNIVAC – one of the first US commercial electronic computer.

Maurice Wilkes, director of the Cambridge Mathematical Laboratory was a committee member and later chairman of the British Mathematical Tables Committee. It was under his leadership that the EDSAC was built. That computer was, amongst others, important because of the programming practices that developed around it and which led to the publication of what is often considered the first programming handbook *Preparation of Programs for Electronic Digital Computers* authored by Wilkes, David Wheeler and Stanley Gill.

In the US then again, Derrick H. Lehmer (see also Chapter II of this volume) became one of the first ENIAC users. He was a number theorist who learned from his father (also a number theorist) that number theory requires exploration and, by consequence, extensive computation. To

this end, he had already constructed several special-purpose devices to help in the computation of tables of prime numbers (these were so-called prime sieves) and was a specialist in number-theoretical tables. In 1943, he became a member of the editorial board of MTAC and a co-editor with Archibald in 1944. During WWII he became involved with research for the Applied Mathematics Panel as an employee of the University of California. In that role, he constructed, amongst others, a special-purpose device to "simulate" random-bombing operations. This is probably how he became one of the members of the US Army Computations Committee that was to test, amongst others, the newly built ENIAC. Later, Lehmer would be an important advocate of so-called computer-assisted mathematics (see below).

The rise of the modern computer is certainly not just the result of mathematicians being mobilized for the war effort. There was already a more general evolution towards using computational methods not just in the military but in science, administration and engineering in general. Thus, when the US entered WWII there was already an important know-how and preparedness to rely on computation and exploration to attack certain problems. In such context, mathematicians were faced with a need for large-scale computation which required not just a good training in abstract mathematics but also a willingness to engage with computing machinery and engineering. The result of that process is that many a mathematician also became a computing pioneer who, alongside engineers and operators, shaped the field of electronic computing, its basic problems and, later, the identity of the disciplines that resulted from it.

III. Doing mathematics with the computer

As explained, mathematics was not only an important factor in the construction and use of the first computers, it was also realized that high-speed computation make it possible to study certain problems that were not solveable with traditional methods. The resulting mathematics was soon described as experimental, explorative or quasi-heuristic. By consequence, the apparent opposition between mathematics, on the one hand, which is viewed by many as being about proofs and theorems, and, on the other hand, experimentation, would haunt the field of so-called "experimental mathematics" and gave rise to occasional controversy between advocates and opponents.

One important set of problems concerned the mathematical analysis of non-mathematical phenomena (for instance, turbulence, nuclear reactions, problems of pattern bombing) that were too complex to understand or solve with classical methods of analysis (Galison 1997). Often, these were non-linear problems. In that context, even before the first high-speed computers, some had already shifted to alternative methods. For instance, there was von Neumann's work in relation to the Manhattan project which studied phenomena like implosion and chock waves relying on calculating aids like the punched-card machines at BRL or the Harvard Mark I. Another example relates to problems studied within the AMP to determine the probability of hitting a certain area with pattern bombing (the simultaneous release of all bombs carried by a formation of aircraft) and which were handled by so-called "model experiments" that aimed for statistical models by using randomized methods. However, while mechanical aids and new methods could help, they were limited by the speed of these processes. With high-speed electronic computers much more could be achieved. As has been emphasized by several advocates and pioneers of computer-assisted mathematics, the change of several orders of magnitudes should not be dismissed as being but a mere quantitative change – that would be like saying that modern automobiles and aeroplanes are no different than walking (Hamming 1963). For instance, as was emphasized on several occasions by von Neumann, it is only with high-speed computation that one can envision to computationally attack higher-dimensional non-linear problems like turbulence or meteorological forecasting. Moreover, the high-speed also allowed to consider problems that required a more complex and longer program structure like the earlier mentioned Monte Carlo computations which, at a later stage, also involved amongst others, a subroutine for generating random numbers as they were needed, rather than relying on a table of random numbers.

While, in the early years, these developments were still often regarded as developments in applied and mathematical physics, today they have resulted in specialized fields of computer

simulation and modeling (think of climate science for instance) where mathematics is more a tool to be used by others. However, besides the fact that these developments of course raised interest in the mathematics of computation (see below) it also resulted in a rejuvenating of older mathematical ideas where the computer now made possible new ways to revisit and study those ideas. One basic example here is research in non-linear dynamics and chaos (Aubin and Dalmedico 2002) – a field that revived due to the work of people like Ulam, Pasta, Fermi, von Neumann and Charney. Ulam and von Neumann, for instance, introduced and studied the special case $x_n = 4x_{n-1}(1-x_{n-1})$ of the logistic map in the context of their research into procedures for generating (pseudo-)random numbers and today known to be chaotic. Ulam, Pasta and Fermi started to explore non-linear dynamics with computational methods using the MANIAC computer at Los Alamos in 1955; Charney worked on the Meteorology project (using amongst others the ENIAC) and contributed in that context to the issue of long-term predictability which became a main result later of Lorenz who proved that this is practically impossible due to sensitivity to initial conditions (this is known as the butterfly effect – the idea that a small change like butterfly flapping can have large effects on the weather). Clearly, within that field, computers have played a rather basic role in that they made accessible certain properties through extensive computations and their visualizations. Well-known visualizations are the phenomenon of strange attractors or the Feigenbaum diagram (see Fig ??) which could then be further explored and studied.

But the first high-speed computers were not only used to study problems from other fields (typically problems of physics) but also to study problems of mathematics proper. Most notably here is the use of the computer in number theory. In that setting, there was already an older tradition of using computations to investigate open conjectures like the Riemann hypothesis, Fermat's theorem or Goldbach's conjecture or simply to explore properties of certain classes of numbers. Probably the earliest example is the number-theoretical computation on ENIAC by Derrick and Emma Lehmer which concerned the computation of a specific kind of tables (tables of exponents) that could help in the identification of so-called composite numbers, that is, numbers *b* for which $2^b \equiv 2 \, mod \, b$ with *b* not prime. That computation was a test problem which showed that a high-speed computer could be useful even for doing something as sophisticated as number theory. In the UK, Turing continued his earlier work on the Riemann hypothesis for which he had already constructed special purpose devices, on the Manchester Mark I. His ambition was to look for counterexamples that would invalidate the conjecture by computing more and more values of the Riemann zeta function. Lehmer did similar work on ENIAC and later on SWAC.

A recurring approach in that time and which is continued today, was to use the idle time of a given computer to compute more and more values of a given number-theoretical function. So, for instance, ENIAC was used in its idle time by Reitwiesner to compute expansions of π and e – later used by von Neumann as possible random sequences – and Fermat quotients which were relevant to proving Fermat's last theorem. The program was called *Slow Moses* and not only served number theory but was in fact aimed at illustrating the stability of the ENIAC hardware. On EDSAC in the UK, idle time was used to compute Mersenne primes (a Mersenne prime P_n is a prime of the form $2^n - 1$), a program that was then picked up also for the Pilot ACE which was faster than EDSAC.

As is clear from these few examples, one general use in number theory of computers (which continued an older tradition) was to compute larger values for certain functions to find counterexamples or to provide new insights into existing problems. The Mersenne prime program is one such example, the Riemann-Zeta computation is another one. This tradition has been continued up to today. One interesting example is the so-called Great Internet Mersenne Prime Search (GIMPS). It is a large-scale distributed computing project – a project where a large number of connected computers are used together to compute – to which anyone who installs the relevant software can contribute with their idle computer time. As was the case with the "Slow Moses" program on ENIAC, also here the project not only serves number theory but was also significant for advances in computing: it was one of the first major examples showing the potential of distributed computing. Also work on Fermat's little theorem was continued by amongst others, the

Lehmers and Vandiver on the SWAC (Corry 2009). Today, there are still several examples to be found, like the continuing search for exceptions to the Collatz conjecture also known as the *3n+1* problem, the ongoing search for prime gaps as well as research on prime constellations (eg twin primes that differ only by 2) which is relevant to several conjectures in number theory. Clearly, this kind of research is not restricted to the mere computation of cases but involves sophisticated methods that look for instance for patterns and so help to determine heuristically certain properties that can then assist in proving conjectures or formulate new ones. Interestingly, these kind of researches are not just relevant for the sake of number theory only, as a side effect they have resulted in new algorithms (think for instance of the Lehmer pseudo-random number generators) or insights into existing hardware and software. One relatively recent example in this context was the discovery of the famous FDIV Pentium bug by Thomas Niceley, a number theorist, and which costed millions of dollars to Intel. Niceley had found an inconsistency in his results while doing research on prime gaps and realized there was an error on the hardware level (a wrong value in a hardwired look-up table for division).

But it was not just the advances in hardware and software that resulted in improved ways for doing mathematics, also the internet enabled this. The Online encyclopedia for Integer Sequences (https://oeis.org/), initiated by Neil Sloane and which currently contains more than 300,000 integer sequences is one basic instance. It is used as an explorative tool and has resulted in several mathematical papers (see http://oeis.org/wiki/Works Citing OEIS for over 2000 papers that reference the encyclopedia in their work). One interesting feature of OEIS is that, attached to it, is a mail service which allows you to submit integer sequences to an algorithm called Superseeker which tries to find an explanation for your sequence by relying on OEIS combined with an extensive library of programs.

But computers have not only been used in pure mathematics to establish experimental results but are also used to prove theorems. One simple example was the “discovery” of the Bailey-Borwein-Plouffe formula for computing *pi* and which resulted in an improved algorithm. As the authors indicated, it was not found through formal reasoning but a combination of inspired guessing and extensive searching using the so-called PSLQ algorithm which is used to find integer relations between a set of real numbers.

But there are more controversial examples which are known as computer-assisted proofs. Such proofs usually involve a large number of particular cases that need to be determined and then verified, relying on a complex of computer programs. Probably the oldest such proof is by Emma and Derrick Lehmer, William H. Mills and John L. Selfridge which, amongst, proved and determined the finite set of primes which do not have a triplet of cubic residues. That proof went mostly unnoticed but there are more famous examples. Probably the most controversial was the four-color theorem proved by Kenneth Appel and Wolfgang Haken and which showed that any map can be colored using only four colors in such a way that no adjacent regions have the same color. The result was published in 1976 and was not considered a real proof by most of the mathematical community and this for a variety of reasons (MacKenzie 2004). Amongst others, its length and the complexity of the programs involved not only made the proof unsurveyable for human mathematicians but it was not guaranteed that there were no errors involved due to program mistakes or hardware failures. Moreover, it was also seen as a proof that does not provide mathematical understanding of why the theorem is true, amongst others, because not all details can be accessed by humans. A more recent and well-known result is Thomas Hales’ solution to the sphere packing problem also known as Kepler’s problem. Simply put, the problem Hales solved was that of determining the most efficient way to arrange (identical) spheres in a 3-dimensional space. It was published in *Annals of Mathematics* in 2005 after a review process by a team of reviewers who worked on it for several years. Even then, it was concluded that they were only 99% sure that the proof did not contain errors.

Uses of the computer in applied and pure mathematics discussed here share one basic feature: that is the apparent experimental and quasi-heuristic nature of the methods and results

involved. This was realized and emphasized on several occasions by those involved sometimes in quite provocative styles inviting discussion if not controversy.

In the context of applied mathematics and the kind of research done by von Neumann and Ulam, there has been a large debate over the experimental nature of their research and which still continues today in the history and philosophy of computer simulations (Galison 1997). In how far are computer simulations similar to physical experiments is one key question here. Von Neumann was quite clear in viewing computation as a kind of in-between, which is not too mathematical in the traditional sense but still closer to mathematics than experiments with wind tunnels (which he regarded as a kind of analog computers). He emphasized on several occasions that this approach allows to build up an intuition for certain problems and to use computers heuristically either in a traditional manner (testing hypotheses) or as simulations. Ulam spoke of "probabilistic experiments" with reference to Monte Carlo methods and also later "experiments on paper" with the help of the computer.

But also in the framework of pure mathematics one started to speak about experiments, exploration and heuristics even in the case of the computer-assisted proofs. Most outspoken here in the earlier years was Lehmer. As explained, he already inherited from his father a more explorative stance with respect to number theory and emphasized on several public occasions the significance of the computer for this kind of mathematics and drew a sharp contrast between, what he called, explorative mathematics and more traditional forms of doing mathematics. He was perhaps most explicit about this during his Gibbs lecture, a prestigious mathematical prize awarded by the AMS (Lehmer 1966):

> 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 as a means of discovery.

This rhetorics of contrasting experimental mathematics with more traditional forms of mathematics is recurrent and anchored in the fact that this form of doing mathematics has often been considered as a lower form of mathematics because of its lack of rigor and "real" proofs. Often the results are probabilistic and there is a lack of trust in the complex of programs and hardware used which might give rise to errors or hardware failures. Some advocates of experimental mathematics have, in fact, embraced their role as one that goes against the establishment. Féjès Toth, an important contributor to the sphere packing problem and one of the reviewers of Hales' proof compared these kind of computer-assisted proofs to experimental science where reviewers also cannot (always) certify the correctness of an experiment. Doron Zeilberger has perhaps been the most explicit advocate here with his plea for what he calls semi-rigorous mathematics that relies on probabilistic truths. Other leading figures in this setting are Stephen Wolfram, Jonathan Borwein and David Epstein. The latter also contributed significantly to the institutionalization of experimental mathematics through their two-volumed work on experimental mathematics (2006) and the establishment of the journal *Experimental mathematics* in 1992.

But while the field has been taken more serious in the last decades it still gives rise to debate often in a context of opposing two different views on mathematics: one that is led by an idea of rigor and formal proof, another, which believes that mathematics is not just about formal truths and is perhaps closer to the sciences than one used to think. A fairly recent clash of these viewpoints can be found in the Jaffe-Quinn debate that was instantiated by Arthur Jaffe and Frank Quinn in a paper published in 1993 in the *Bulletin of the AMS*. In that paper they distinguish between rigorous mathematics and so-called theoretical mathematics. The former refers to proof-oriented mathematics, the latter to speculative and intuitive work. When the paper was published a sequence of responses was published, most notably one by William P. Thurston who argued that the Jaffe-

Quinn paper was very much about the projection of the sociology of mathematics onto a one-dimensional scale (contrasting speculation with rigor).

Interestingly and perhaps ironically it was exactly in the context of such debates that Thomas Hales embarked on a new project known as the Flyspeck project: given the many uncertainties that surrounded the sphere packing proof, he decided to provide a formal proof, that is, a proof that is formally verified by a proof assistant and thus, again, computer-assisted. The project has been finished and in the meantime also formal proofs of the four-color theorem have been provided. These kind of projects led to important collaborations between computer scientists, logicians and mathematicians constructing software that not only formally verifies proofs but also existing programming tools. Most notable here is the CompCert project which works on the formal verification of the C compiler (the C compiler is perhaps the most basic compiler for programming languages). As will be shown later, research into the formal verification of programs is an important branch in the history of programming and which led, in the 1980s to the important formal correctness debates.

Today, computer-assisted mathematics is more accepted than it used to be in the 1950s and 1960s. The possibilities only grew as memory and speed increased. That process went hand-in-hand with an easier access to the computer and improved peripherals (which made possible, amongst others, better visualizations). In that process the computer became much more than a simple brute force of computation – it became a heuristic tool that not only provided data but also did (and does) much of the analysis on those data by providing visualizations, graphics, etc. This resulted in important cross-fertilizations between mathematics and the computing field. The Flyspeck example, GIMPS or the discovery of the Pentium bug are clear examples here. But it is also known that the development of software packages like Maple and Mathematica has contributed significantly to the popularization of computer-assisted mathematics: the Borwein-Epstein book relies entirely on Maple and it was Wolfram's experimental research in the 1980s and 1990s on cellular automata that led him to the development of *Mathematica*. One major new problem arising in those contexts is a problem of accessibility: both Maple and Mathematica are closed and proprietary which means that the many programs used in this software are not accessible to the mathematicians using them.

Mathematics of computation

Mathematics was and is not only a field of application for high-speed computation. It is also a basic field for developing new or improved algorithmic and programming methods to gain amongst others a higher efficiency in speed and memory and improved programming tools. Moreover, once it became clear that high-speed computation gives rise to a number of different problems that were there to stay, a new field started to develop that allowed to *study* computation and the computer as a theoretical topic in and by itself. In that sense, mathematics is not only a telos for computation that allows to gain new mathematical insights, it also shapes the way computations are done (methods) and how they can be understood (theory). The computer is not just a tool for mathematics, mathematics itself is a tool for the computer. It is thus not surprising that as computers became more broadly available, the role of mathematics for computing did not remain restricted to mathematical research departments at universities but also had its significance in other departments (most notably, computer science). Moreover, mathematics also found its way more easily into business and industry. As such, work within what we here call the mathematics of computation often found itself at the borderline between a number of fields (mathematics, computer science, engineering) and so strict borderlines between disciplines cannot easily be drawn.

The modern computer soon resulted in the realization that old methods would have to be reconsidered and that new approaches would be needed. This development fits in a broader transition of the steady replacement of human computational work by machines and a history of automation. This meant that methods had to be adapted to the specifics of the technology rather

than to humans and this against the background of the basic philosophical question of what can be automated.

In the early years especially, it was pointed out by several pioneers that methods that work well for a human do not necessarily work well for a machine and vice versa. As was pointed out for instance by Douglas Hartree, a UK mathematician who became involved with the construction and use of calculating machines (amongst others, different differential analyzers and the ENIAC) and wrote one of the first historical overviews titled *Calculating instruments and machines* (published in 1949) wrote that:

> "in programming a problem for the machine, it is necessary to try to take a "machine's-eye view" of the operating instructions, that is to look at them from the point of view of the machine which can only follow them literally, without introducing anything not expressed explicitly by them, and try to foresee all the unexpected things that might occur […] it is quite difficult to put oneself in the position of doing without *any* hints which intelligence and experience would suggest to a human computer"

Besides this general change of perspective, the specifics of a changing computing technology also required different approaches. First of all, the fact that the modern computer works digitally, implies that one needs to state a mathematical problem in a discrete algorithmic form and to replace non-discrete mathematical operations (eg transcendental operations like integration) by elementary arithmetical operations and explicit steps. This implies a more constructive and finitist approach to mathematical problems and a move towards what one could call discrete mathematics (see chapter 1).

Moreover, increases in speed and, later, memory also had basic effects on the methods used and studied and resulted in new fields, or, the thriving of older fields. An early example is the rather swift transition from using mathematical tables in a computation – which was the human way of working – to algorithms that allow to compute values in tables as they are needed. That shift was anchored in the fact that, at the time, the technology for electronic memory was lagging behind on high-speed computation. This implied that, if during a computation one needed additional data, one had to rely on much slower electro-mechanical devices like punched cards implying a significant slow-down of the computation. Hence, it was far more interesting and achievable to compute values in tables as they were needed. A rather well-known example here is the replacement of cards of random numbers by von Neumann's pseudo-random generator in the ENIAC Monte Carlo computations. That is, rather than using numbers that were generated by some outside random device and then stored in some table (see for instance the work by the RAND corporation who published a table of one million random numbers generated with a roulette wheel) they used an algorithm. In these kind of examples mathematical tables (data) were replaced by more sophisticated programming techniques – a trade-off that was quite common in the early years.

The computer has motivated the development, adaptation and improvement of a broad range of different algorithmic methods with a diversity of applications, ranging from cryptography to methods for computer graphics and sound analysis. The steady evolution of the computer from a difficult to access technology to one that is ubiquitious has increased the need for such methods and has thus resulted in the thriving of domains that were before more marginal to mathematics. A prominent example here is the field of numerical analysis which is mostly concerned with approximation methods for analytical problems, like differential equations. The field has a long history and was already gaining in significance before the introduction of high-speed computation with the increased need for mathematical tables such as the firing tables mentioned before. Those required to solve two basic differential equations through a step-by-step method of successive approximations since they could not be solved by classical methods. The computer however, gave it a new set of problems. An indicative example here is the problem of rounding-off errors and error estimates. If one is working with discrete machines, one cannot represent reals exactly and one needs to work with finite approximations that have only a certain number of places used to represent a given number. For instance, with 10 places, one could represent pi as 3.141592654. If then, for instance, we would have two 10-place numbers that are finite approximations of existing

numbers and they are multiplied, one needs to round-off to another 10-place number resulting in a certain error. This problem became much more significant with the rise of the electronic computer: to put it roughly, if one can do 10000 computations rather than 100 then the potential effects of rounding off become much more dramatic, as was shown, for instance, by Ulam's work on the logistic map where small differences in the initial condition grow exponentially with every new iteration. The problem of estimating the errors arising from rounding-off has become one classic subfield of numerical analysis. Since many problems in computing required numerical analysis, the need for improved methods grew significantly with the steady spread of the computer in science and society and its changes on the technological level. It was for instance the work by Ingrid Daubechies (see also Chapter 1) on wavelets which allowed, amongst others, to improve methods of image compression – wavelets are, for instance, used in the jpeg compression algorithm; another more recent example is the reliance on optimization techniques in the setting of so-called Deep Learning, an AI technology which, if one were to believe its promotors (especially those with commercial interests) is yet again revolutionizing our world.

While the field of numerical analysis is concerned with numerical solutions to problems and so concerns *calculation*, the computer soon also required another type of algorithms that are able to attack what one could call non-numerical problems and identified as *nonnumerical analysis* by the renowned mathematician Donald E. Knuth. In a series of books titled *The art of computer programming*, and known colloquially as the bible of programming, Knuth introduces this term to indicate the scope of his multi-volumed series and also speaks in this context of *analysis of algorithms.* Some basic classes of algorithms studied and analyzed in that series are: sort and search algorithms, data structures and pseudo-random generators (which are identified as semi-numerical). This kind of work is a clear marker of the transition from calculation to computation and has extended significantly the realm of what can be automated. They lie at the borderline between mathematics, programming and computer science.

But where mathematical methods that are used to improve on computational technologies could be understood as a more practical development, such developments have often also stimulated more theoretical studies. The Knuth book series mentioned above, is one example here: besides providing practical methods, they are also analyzed according to, amongst others their time and space efficiency – a practical problem but one that is also core to what is know as computational complexity theory and which, today, is mostly a theoretical field (see below). A prototypical example in that setting is the Traveling salesman problem: given a set of cities and the distances between pairs of cities, find the shortest path in such a way that every city is visited only once and that you return to your point of origin. While this problem is a problem of operations research, it is also perhaps the most famous example of so-called NP-hard problems as studied within computational complexity theory. Another example is the field of computability over the reals which is motivated by numerical analysis but mostly a pure theoretical field.

As explained, mathematics has been basic to developing computational methods adapted to the computer and its widespread use. However, it was also realized that there was a large number of problems that appeared to be specific to computation and programming itself and soon, mathematics found itself facing a newly emerging discipline known as computer science (Mahoney 2011). Computer science became an established field in the 1950s and 1960s and went hand-in-hand with the establishment of computer science departments at the universities. In the US the first computer science department was established in 1962 at Purdue university. From the start, the field was haunted by debates concerned with its disciplinary identity (see the next section), its content and scope and even its name (think of the use of "Informatique" in France or cybernetics in Russia). Despite these disciplinary debates one can identify some clear theoretical developments. We focus on two principal developments: first the use and study of formal models for computing machinery, second, the use of formal logic in programming.

When the first computers were being built, the few pioneers who were also familiar with developments in mathematical logic, realized that a connection could be made with developments

from the 1930s in the setting of the foundational debates in mathematics and, more specifically, work on so-called decidability problems (De Mol 2018). In the early 20th century, several such problems had been proposed, most notably perhaps, Hilbert's 10th problem. That is, the problem of whether or not there exists a uniform method (today, we would speak of an algorithm) to decide for any given Diophantine equation, whether or not it has an integer solution. Another example is the so-called *Entscheidungsproblem* for first-order logic: to decide with a (uniform) finite procedure for any proposition in first-order logic whether or not it can be derived in that logic. In the 1920s, David Hilbert was convinced that, in fact, it would be possible to provide such procedures as is clearly captured in the phrase "Wir müssen wissen, wir werden wissen" (we must know, we shall know). Others, including von Neumann were more critical and understood that if ever such a procedure would be found, it would imply the existence of a mechanical set of rules to solve *any* mathematical problem (Gandy 1988). In the 1920s and 1930s such problems were solved in the negative and independently by Alonzo Church (1936), Emil Post (1921) and Alan Turing (1936). In that work one finds, what one would nowadays call, different abstract models of computability which are logically equivalent. If one agrees that these models capture the intuitive notion of computability, then it follows that, amongst others, the *Entscheidungsproblem* and any other logically equivalent problem is uncomputable – that is, there exists *no* algorithm that will decide every instance of the problem. Variants of this statement are often identified as the Church-Turing thesis, Church's thesis, Turing's thesis or Post's thesis. Most well-known, Turing showed how computability of any real number reduces to computability by what came to be known as a Turing machine. The implication is that any number that cannot be computed by a Turing machine cannot be computed in general.

The Turing machines were derived by Turing by providing an informal analysis of all the possible (human) processes that can be carried out in computing a number and which resulted in a basic set of operations to be carried out by an abstract machine. That machine is always in a given state and works on a linear tape consisting of squares containing symbols. When in a given state, the machine reads the symbol on the square it is scanning and depending on its value, makes a movement to an adjacent square (left or right), prints some symbol and goes to the next state. Most importantly for later, Turing also constructed a so-called universal machine: an abstract machine that is capable to simulate the operations of any other Turing machine and was used as a model for the modern computer. It was that machine which would be used (and is still used) as a model for the general-purpose modern computer: just like the universal Turing machine, so goes the argument, the computer is also able to compute anything that is computable. Moreover, once one has a certain number of primitive operations, it is not required to add any further complications in order to compute more. A few operations suffice.

In the 1940s von Neumann and Turing picked up this model to reflect on the possibilities and limitations of modern computers. In the 1950s this and other models were used in a context of what came to be known as *automata studies* and which, at least initially, could be seen as a rapprochement between logic and computer engineering. The universal Turing machine was then used by, amongst others, mathematically-oriented engineers like Shannon or Moore who concluded that only a few operations suffice to built a general-purpose computer and they pursuid a project of determining the smallest possible numbers of operations needed to build a computer. Moreover, logicians like Martin Davis and Hao Wang derived improved models for the modern computer like the Post-Turing machine and which, amongst others, resulted in another model known as register machines which were considered to be closer to the modern computer than the Turing machine. This tradition of developing and reasoning with formal models of computation is continued until today, following and affecting new computational technologies. So, for instance, today we have formal models for concurrent computation (e.g. the pi-calculus), quantum computation (e.g. quantum Turing machines) or biocomputation (e.g. petri-nets).

It was also in this setting of automata studies that focus shifted to the efficiency of time and space for algorithmic methods that solve a specific problem and their upper and lower bounds. These problems are studied theoretically in the field of computational complexity theory where one has determined a rather large number of different classes of computational problems classified according to their so-called space and time complexity. In that setting, the Turing machine is still one of the most basic models used. Most well-known here is the so-called P=NP problem and which asks whether these two classes are identical or not. It is one of the seven Millenium Prize problems proposed by the Clay Mathematics Institute of Cambridge Massachusetts – the one who can solve it gets a 1,000,000$ prize. The fact that a problem that belongs to (theoretical) computer science is considered as a basic problem in mathematics is symptomatic of a more general development in which mathematics is no longer perceived as the (only) queen of the sciences (see the next section).

While automata studies and, in general, models of computation, are concerned with the theoretical study of computation, its limits and possibilities, another basic role of formal logic and mathematics was to be played in the context of computer programs and software (Priestley 2011). It was realized very soon that the design of programs and programming for a high-speed computer gives rise to a number of basic challenges which became a major problem in and by themselves resulting in, amongst others, the so-called software crisis in the late 1960s. The use of formal logic in that context is not too surprising: formal devices such as those developed by Church or Post were understood as an abstract formal game played with marks on a piece of paper, following a set of rules that did not require an understanding of what is being done. That, of course, fitted well with a computer which was assumed to not have any insight and understanding (see in this context the quote of Hartree on p. ??).

Given the high-speed of the machine, programs need to be, in a sense, ahead of time and function as a control over the computational process to be. It is here that formal logic could play a basic role as is explicit from the following quote by von Neumann:

> [C]ontemplate the prospect of locking twenty people for two years during which they would be steadily performing computations. And you must give them such explicit instructions at the time of incarceration that at the end of two years you could return and obtain the correct result for your lengthy problem! This dramatizes the necessity for high planning, foresight, and consideration of the logical nature of computation. This integration of logic in the problem is a consequence of the high speed.

Together with Hermann Goldstine, von Neumann then developed a step-wise process for planning and coding a computer. Central to their idea was the use of so-called flow diagrams which, in a sense, capture the structure of a high-speed computation. They would remain a basic program technique for many decades (Ensmenger 2016).

when the first computers were built, it was clearly understood that programming a machine using a low-level machine code turned out the be extremely hard and the work by von Neumann and Goldstine fits in that context. This became much more problematic once computers were commercialized (late 1940s and early 1950s) and so used by a broader group of people who wanted to do more with computers than just scientific computation and use them, for instance, in accounting.

Programming had to be less difficult. This resulted in the 1950s in what was known as automatic programming or automatic coding and entailed at least two developments. First, the development of a notation that is easier to work with than the machine code and, second, the automatic “translation” of such notations to machine code (by a compiler or an interpreter). In the late 1940s the logician Haskell Curry had already worked on that problem and developed a so-called theory of program composition which was much affected by his logical work on combinators but did not result in an actual implementation. That work, however, went largely unnoticed and it was in the 1950s that most progress was made with the construction of the first compilers that resulted in the first high-level programming languages like FORTRAN and LISP. Since most people involved with

programming had a background in mathematics, also the notations were mathematical and/or logical in nature. John Backus who worked for IBM and made major contributions to FORTRAN spoke about an algebraic notation; John McCarthy who is the inventor of LISP relied on Church's models of effective calculability – lambda-calculus and general recursive functions – in defining the LISP language. Most notably was the work around the ALGOL language, influential for later languages and which was an international effort originating in the general need for a *universal programming language* which, so it was hoped, would become the main programming language at a time when a multiplicity of programming languages was already developing (Nofre et al 2014). Just like LISP also ALGOL was conceived of as a formal language and, around it, one can find a number of different aspects of how formal logic could be used into this context. For instance, one of Post's model for generating sequences of finite words – known today as Post production systems – were used in the formulation of the so-called Backus-Naur form that was used for the definition of the formal syntax. These systems had also entered the field of programming via another path: in the 1950s Noam Chomsky and Marcel-Paul Schützenberger had worked and classified formal grammars into a hierarchy of grammars. Post production systems played a basic role here and would enter the theory of compilers via that path.

But ALGOL also had important in giving programming a more mathematical identity (see also the next section). Amongst others, ALGOL stands for ALGOrithmic Language and in one of the main magazines of computer science, *Communications of the ACM*, a new section was established on Algorithms in which approved "algorithms," expressed in the ALGOL language (of course, an ALGOL program is a program, not an algorithm), would be published on a regular basis and a specific standard was introduced for the publication of these algorithms. In connection to this, it is also interesting to point out that algorithms also became a major focus of theoretical research in Russia where, amongst others, Andrei Markov, Andrei Kolmogorov and Vladimir Uspensky worked extensively in determining an abstract model for algorithms.

The need for reviewed algorithms however was also anchored in another problem that was becoming more basic as more people were using a computer: many a computer program contained errors – a basic problem even today. The problem of error went hand-in-hand with a larger issues related to the fact that as computers were used by more people in a variety of ways, the system of programs designed and programmed to use the machine was become more and more complex up to the point that providing good software in time was considered to be a major problem in and by itself. This problem is known as the software crisis. One reply to these issues came from a number of mathematically minded programmers and engineers: to provide a mathematical basis for programming. In that context, several more logical and mathematical techniques were developed. Perhaps most famous from that context was research that aimed at the formal verification of programs which prove that a given program is "correct" with respect to its formal specification and so, supposedly, does not contain any errors and is doing what it is supposed to be doing. The main idea was to provide a formal proof showing that a program is correct using some formal model of the program at hand, for instance, a (formal) semantics of a programming language. The idea of the correctness of programs became very popular in the 1970s even though it was realized at the time that it was not realistic for a number of reasons to prove the formal correctness of every program. This ultimately resulted in the late 1970s and 1980s in a clash between those who were convinced of the need of such high standards for programs and those who did not believe this was the right path to follow or, worse, if it was even possible (MacKenzie 2004, Tedre 2014)

Interestingly, as explained in the previous section, even though the popularity of correctness proofs waned, today there is a renewed interest in formal verification which can now be used more extensively and efficiently thanks to the development of so-called proof assistants like COQ and programming system which have their historical roots both in mathematical logic (most notably, work on the so-called Curry-Howard isomorphism and Martin-Löf type theory) and earlier work from the 1950s on automated theorem proving (aimed at, amongst others, the automatic generation of theorems in formal logic). The use of formal verification is considered key to one of two basic approaches to safety-critical systems like self-driving cars. As explained in the previous section, it is

also in this same context that mathematicians like Thomas Hales have found a new interest since this kind of mathematical technology can and has also been used in providing formal proofs for providing a certified proof for specific problems like the sphere-packing problem.

Disciplining mathematical technology (1953 – 2021) – changing roles and disciplinary identities

A short summary of what will be written here
The purpose of this concluding and shorter section is to bring together what was written in the introduction and the main sections: in May's view, the idea was still to bring the history of computing into the history of mathematics. That view can be seen as a confirmation of a then still popular view that the computing field is very much about mathematics. However, today the history of computing is very distinct from the history of mathematics – there are only few who really engage with that connection (De Mol and Bullynck 2019). Why is that the case? This is at least partially anchored in a changing relation between mathematics and the field of computing. When it became clear that computation and the computer gave rise to a set of new questions, it was not clear what this new field should look like. This gave rise to the search for a disciplinary identity that would allow to establish the field as a respected discipline alongside others. In that setting, many looked into the direction of mathematics (eg Knuth and Dijkstra) as a foundation for the field. In fact, as is clear from the introduction to the book of the Los Alamos conference mentioned in the introduction, it was believed that theory should be the captain and application the soldier. In the meantime, however, things have evolved significantly and this due to the widespread use of the computer in the sciences and in society. Today, the identity of the computing field is still under debate but the relation between mathematics has significantly changed. The field has matured and is perhaps steadily taking over the role of mathematics as something that is basic to any other science. At the same time, the significance of computing in mathematics has also changed and hence mathematical technology is steadily gaining in significance within the broader field of mathematics.

References

Aspray, William. 1990. *John von Neumann and the origins of modern computing*, Cambridge, Massachusetts: MIT Press.

Archibald, Thomas, Dumbaugh, Della and Kent, Deborah, "U.S. mathematicians in World War I," in: (Aubin and Goldstein 2014), pp. 229-271.

Aubin, David and Goldstein, Catherine (eds.). 2014. *The war of guns and mathematics. Mathematical practices and communities in France and its Western Allies around World War I*, American Mathematical Society.

Aubin, David and Dahan Dalmedico, Amy. 2002. "Writing the History of Dynamical Systems and Chaos: *Longue Durée* and Revolution, Disciplines and Cultures" *Historia Matematica*, 29(3): 1-67.

Campbell-Kelly, Martin, Croarken Mary, Flood, R and Robson E. 2003. *The history of mathematical tables. From Sumer to Spreadsheets.* Oxford: Oxford University Press

Campbell-Kelly, Martin, Aspray, William, Ensmenger Nathan and Yost, Jeffrey. 2014. *Computer . A history of the information machine*, third edition, Bouler: Westview Press.

Corry, Leo. 2008. “Number crunching vs. number theory: computers and FLT, from Kummer to SWAC (1850-1960), and beyond.” *Archive for history of exact sciences*, 62: 393–455.

De Mol, Liesbeth and Bullynck, Maarten. 2018. “Making the History of Computing. The History of Computing in the History of Technology and the History of Mathematics.” *Revue de Synthèse* 139(3–4): 361–80.

De Mol, Liesbeth and Durand-Richard, Marie-José. Forthcoming. “Calculating machines and numerical tables – a reciprocal history,” in Dominique Tournès (ed.), *The history of mathematical tables*, Springer.

De Mol, Liesbeth. 2018. "Turing Machines", *The Stanford Encyclopedia of Philosophy* (Winter 2019 Edition), Edward N. Zalta (ed.), URL = <https://plato.stanford.edu/archives/win2019/entries/turing-machine/

Ensmenger, N. 2016. “The mutliple meanings of a flowchart”, *Information and Culture*, 51: 321-351.

Galison, P. 1997. *Image and logic. A material culture of microphysics.* Chicago: The university of Chicago Press.

Gandy, R. 1988. “The confluence of ideas in 1936”, Herken, R. (ed.), *The universal Turing machine. A half-century survey.* Oxford: Oxford university Press.

Gerovich. Slava. 2002. *From Newspeak to Cyberspeak: A History of Soviet Cybernetics.* Cambridge, Massachusetts: MIT Press.

Grier D.A. 2001. “The rise and fall of the committee on mathematical tables and other aids to computation.” *IEEE Annals of the History of Computing* 23(2):38–49

Grier, David Allen. 2005. *When computers were human*, Princeton: Princeton University Press.

Hamming, Richard. 1963. “Intellectual Implications of the Computer Revolution.” *The American Mathematical Monthly*, 70(1): 4–11.

Lehmer, Derrick H. 1969. “Computer technology applied to the theory of numbers.” Leveque, William J. (ed.), *Studies in Number Theory* (W.J. Leveque, ed.), MAA Studies in Mathematics, vol. 6, Prentice Hall.

MacKenzie, Donald. 2004. *Mechanizing proof. Computing, risk and trust.* Massachusetts: MIT Press.

Mahoney, Michael. 2011. *Histories of computing.* Edited and with an introduction by Thomas Haigh, Cambridge, Massachusetts: MIT Press.

May, Kenneth O. 1980. “Historiography: a perspective for computer scientists”, in: Metropolis, N., Howlett, J. and Rota, Gian-Carlo. *A history of computing the the twentieth century*, New York: Academic Press, pp. 11-18.

Mounier-Kuhn, Pierre. 2010. *L'informatique en France de la seconde Guerre Mondiale au Plan Calcul. L'émergence d'une science.* Paris: Presses de l'Université Paris-Sorbonne.

Nofre, David, Priestly, Mark and Alberts, Gerard. 2014. “When technology became language”, *Technology and Culture*, 55(1): 40-75.

Priestley, Mark. 2011. *A science of operations. Machines, logic and the invention of programming*, Berlin: Springer.

Siegmund-Schultze, Reinhard. 2004. “The ideology of applied mathematics within mathematics in Germany and the U.S. until the end of Wold War II.” *Llull: Revista de la Sociedad Española de Historia de las Ciencias y de las Técnicas*, 27(60): 791–812.

Tedre, Matti. 2014. *The science of computing. Shaping a discipline.* Boca Raton: CRC Press.

Rees, Mina. 1980. “The mathematical sciences and world war II,” *American Mathematical Monthly*, 87(8): 607–621.

von Neumann, John. 1948. “Electronic methods of computation,” *Bulletin of the American Academy of Arts and Sciences*, 1:2-4.