The role of flow charts in the early automation of applied mathematics

STEPHEN MORRIS

City University London

ORLENA GOTEL

Independent scholar, New York

The rapid spread of computers into almost all fields of commercial, intellectual, and social activity has obscured a great deal: the role of mathematicians in the early history of computing, the uses of early machines, the techniques developed by mathematicians to aid their use and the impact of automation on mathematics itself. Fortunately archives in both the United Kingdom and the United States have preserved some evidence of how mathematicians developed the techniques necessary to exploit early machines.

In the early days of computer programming flow charts were a fundamental aid to mathematicians as they automated computational processes. From the very start development of a diagram representing the proposed iterative process immediately followed the choice of formula or algorithmic approach to be used. The subsequent detailed coding depended explicitly on the structure of this flow chart and on the skilled exploitation of the limited memory capacity of machines at that time. Contemporary sources from both the UK and the US define this design process and mark the emergence of what became known as software engineering. Documentary evidence of actual use, rather than just written guidance as to intended use, is sparse, with some notable exceptions including those presented here. The use of flow charts as a general aid to programming and coding eventually fell out of sight as automated computation extended beyond the boundaries of the mathematical and scientific communities and programming came to be supported by structured methods and higher level programming languages. With the widespread use of these diagrams to define, or merely illustrate, general business processes the fundamental role that mathematicians and mathematics had played in making programming possible for non-mathematicians became obscured.

The use of diagrams as an aid to the design of computational processes has its roots in the earliest days of the first electronic computers. From that time it is possible to draw a clear line from fundamental early practices, in particular the use of flow charts, from Goldstine and von Neumann (1947) who were the originators of the modern form for this type of diagrammatic representation, through to the latest version of the Unified Modeling Language, the standard most recently developed for the assistance of software system design (Morris and Gotel 2006). Unfortunately, while the development of hardware and to a lesser extent the creation of programming languages have been well documented, little primary source material
appears to have survived detailing the use of early machines on a month by month or year by year basis, and even less documenting early program design documents and the programs themselves.

Indirect evidence of actual and expected usage of the new machines is more widespread. In the minutes of a meeting on 14 December 1949 of the National Research Development Corporation’s Advisory Panel On Digital Computers one comment highlights the significance of automated computation only to a relatively restricted mathematical community: ‘Scientists most accustomed to the use of large-scale universal machines are divided in their opinion as to whether anyone but a research mathematician could make effective use of such a machine.’ Such a conclusion is hardly surprising given the nature of the demands for automated computation which were at the time outstanding. In July and November of the following year the Ministry of Supply wrote to the Advisory Committee on High-Speed Calculating Machines (ACH-SCM) explaining the needs of all its research and development departments.

It is clear that there are a number of problems awaiting solution: these problems include the following: Anti-aircraft projectile trajectories. Problems relating to Ballistics, Control and Guidance of Guided weapons. Reduction and statistical analysis of trials of Fire Control Instruments. Analysis of multi-station aircraft tracking systems. Problems in the design of aerials, magnetrons, wave guides. Non-linear mechanics in armament problems. Quantum physics of semi-conductors. Problems associated with aero and hydro-ballistic research facilities. Aircraft flutter calculations (complex matrices). Fluid motion problems (partial differential equations of elliptic, hyperbolic and mixed types). Statistical and auto-correlation problems. Integral equations concerned with neutron transport problems. The reduction of sound ranging records of high air bursts. Hyperbolic partial differential equations in three variables. Bomb ballistics calculations. A number of these problems are of the highest urgency and service importance: solutions are being obtained in some cases by uneconomic means; in others by approximate methods of insufficient accuracy; in yet other cases solutions are not being obtained.

Interest in such problems and the possibility of obtaining solutions automatically and, by the standards of the time, exceptionally rapidly, was soon to have an impact on mathematics, as the work of Collatz suggests. At the time when the ACH-SCM was beginning its deliberations Collatz was finishing his first book about the numerical treatment of differential equations, written in order to give ‘an idea of the quantitative behaviour of the solution of a differential equation problem . . . obtained by numerical methods with nothing like the trouble and labour that the widespread prejudice of engineers, physicists and mathematicians would suggest’ (Collatz 1950, v). The second edition of his work appeared in 1954 and the changes of attitude caused him to note ‘the intense and active interest which is now being taken in the numerical solution of differential equations the world over’ (Collatz 1954, vi). He included brief biographies of both Carl Runge and Martin Kutta whose methods for the numerical solution of differential equations had provided the basis for some of the first sub-routines written both at Cambridge University for the EDSAC machine (Wilkes et al. 1951)

1 File DSIR 10/317, the National Archives, Kew.
2 File DSIR 10/319 First Annual Report (October 1949–September 1950), the National Archives, Kew.
3 File DSIR 10/317, the National Archives, Kew.
and at the National Physical Laboratory, Teddington for the Pilot ACE machine (see below).

The automated computation of algorithms, such as those based on the approach of Runge and Kutta, required a representation which would bridge the divide between mathematical formulae and the codes used to control the execution of machine operations. Goldstine and von Neumann solved this problem initially by making radical extensions to both the semantics and the syntax of the form of diagram already known as the ‘flow chart’ or ‘flow diagram’ (see Morris and Gotel 2006 for their new form and its antecedents). The new diagrammatical form then became an integral part of the process. In the words of a later Ferranti document now in the British Library (Ferranti 1955, 1.3):

The process may often be split into two stages. The first (‘programming’ in a narrow sense) is concerned with converting the numerical method into a series of steps, each of which may be, for example, the evaluation of a polynomial, a cosine or a logarithm, or the carrying out of a small group of arithmetical operations. The result of this stage is frequently expressed in the form of a ‘flow-diagram’ showing what operations are to be performed and how one operation leads to another. The second stage, often called ‘coding’, consists in writing down the actual orders which the machine will have to obey; this is relatively simple once the flow-diagram has been obtained.

Little evidence of actual programming practice survives from the era of individual machines prior to the commercial production of multiple examples complete with printed operation manuals and training materials. The papers of Grace Murray Hopper and Frances E Holberton in the US and those of Mike Woodger at the National Physical Laboratory (NPL) in the UK, all working as mathematicians, provide important evidence of how flow charts were actually used and became altered as practice developed. In April 1950 both Hopper and Holberton began teaching parts of a programming course for the engineers in their company, the Eckert-Mauchly Computer Corporation (E-MCC), and both retained copies of parts of the material used. The outline of the second lecture summarizes the role of the flow chart.  

A flow chart is the logical outline for the execution of a specific routine... based on the Goldstine and von Neumann paper: ‘Planning and Coding of Problems for an Electronic Computing Instrument’... developed to provide a convenient shorthand for representing logical functions for Binac and UNIVAC.

Figure 1 shows an example of the type of flow chart used in the E-MCC teaching material. It comes from a document produced just before the end of the course defining flow chart symbols. The rectangular box contains ‘operations necessary to process data’; the flag-like symbol is an ‘assertion box’ containing ‘a statement that “at this time this is true”’; the box with a double line to the left side is a ‘substitution box’ containing ‘the operations necessary for altering controls or instructions’; the box with rounded ends is a ‘decision box’ which ‘indicates a choice between two divergent

---

6.1.2 Determine the largest of three quantities: $a, b, c$

\begin{figure}
\centering
\includegraphics[width=\textwidth]{chart1.png}
\caption{Examples of basic flow charts as used by Eckert-Mauchly Computer Corporation, 1950 (reproduced from the Grace Murray Hopper Collection by courtesy of the Archives Center, National Museum of American History, Smithsonian Institution, Washington DC)}
\end{figure}

6.1.3 The \texttt{Q} instruction transfers control when the quantity in \texttt{rA} equals the quantity in \texttt{rL}.

\begin{figure}
\centering
\includegraphics[width=\textwidth]{chart2.png}
\caption{(rA):(rL)}
\end{figure}

6.1.4 Sum $x_i + y_i = z_i$ where $i = 0, 1, 2, ..., n$;

\begin{figure}
\centering
\includegraphics[width=\textwidth]{chart3.png}
\caption{Summation flow chart}
\end{figure}

paths of computation'; a circle is a point of entry to the routine and parentheses indicate the 'contents of' a memory location. Goldstine and von Neumann had introduced all these elements to the diagram excepting what had become known as the 'operation box' (for an example of this see Morris and Gotel 2006).

Figure 2 shows that this version of the notation at E-MCC allowed for its elaborate use. The flow of control runs from left to right and from top to bottom. The use of assertions and substitutions was already relatively uncommon and they were soon to fall from the notation as it became standardized. The use of exit and
Figure 2. Flow chart showing section of matrix manipulation program for BINAC, 1949 (reproduced from the Grace Murray Hopper Collection by courtesy of the Archives Center, National Museum of American History, Smithsonian Institution, Washington DC)
re-entry points (marked by circles and numbered 8.1–8.5) avoids a tangle of lines but makes the logic less easily comprehended.

An accompanying report explains the purpose of the whole program, of which this flow chart forms a part, and gives detailed operating instructions. The full flow chart for the routine represents a program divided into sections, each drawn on a separate sheet and each with multiple points of exit from and return to the main flow of control. Nevertheless its overall structure remains a linear sequence in one dimension, a single routine which does not call any sub-routine. The accompanying report refers to it as a routine for solving five different matrix equations. In order to overcome the problem of restricted memory in the BINAC (Binary Automatic Computer) the program employs the method of partitioning large matrices into sub-matrices of appropriate size.

In anticipation of increases in memory capacity, sub-routines were in preparation for all machines at this time. A later document from the Mathematics Division of the NPL written in October 1953 discusses all aspects of the preparation of sub-routines for the DEUCE machine in expectation of its availability in 3–6 months. It gives a ‘first priority list’ of sub-routines to be produced for the DEUCE, fifty nine in all, of which fifty had already been prepared for the Pilot ACE. A ‘Runge–Kutta method’ was one of three differential equation routines written for both machines. Figure 3 shows an example of its use as part of a program written by Woodger for ‘G E Gadd (Aero) on a non-linear equation for shockwave boundary layer separation, studies of heat transfer’. Both the program design documentation and the punched cards for this program survive in the NPL archive. The boxes contain not only specific operations but also punching instructions for the interim outputs demanded by limited memory capacity and warnings such as ‘Fail: Buzzer’ at a particular condition check. This flow chart is in some ways an alternative representation of the operational instructions given in the E-MCC document mentioned above, although now the complexities of the possible flows defy expression in natural language and demand an intimate knowledge of both machine and technique. The difficulty of interpretation gives credence to the later criticism of how the use of flow charts promoted ‘spaghetti’ programming (Dijkstra 1968).

At the same time as Woodger was working on this program he was also the sole UK representative on the committee defining ALGOL, the first attempt to standardize a higher level programming language, and work had started on the third version of the Lyons Electronic Office, the LEO machine (Hally 2005, 103–134). As the text in the manual following the example shown in Figure 4 emphasizes, the uses of the flow chart would now rapidly expand:

7 NPL Archive Series H Folder N15, T Vickers, DEUCE Sub-routines.
8 NPL Archive Series H Folder N9, Simplified programming systems for ACE, M Woodger work 1959, Incl. complete DEUCE prog. for shockwave boundary layer interaction, Ma4532, 1957–59 and Series H Ref II.
Figure 3. Higher level program flow chart by Woodger, 1958 (copyright Queen’s Printer and Controller of HMSO, reproduced from the NPL Archive by courtesy of The Science Museum Library and Archive, Wroughton)
Flow charts are of very general usefulness, and can range from showing the sequence of operations in a data-processing system as a whole, to more-or-less detailed programmers' charts prepared immediately prior to coding. Flow charts are often used as a means of communication between programmers, and the final version may be drawn up for this purpose after the program has been written.

In the new era of commercial computing and outside the domain of pioneering mathematicians, the specifically algorithmic purposes of flowcharts were becoming obscured, their uses generalized, their forms again made simple and their functions sometimes separated from program design, even used just as explanation after the event.
Bibliography


Frances E Holberton Papers, Charles Babbage Institute, Minneapolis MN.


Grace Murray Hopper Collection, Archives Center, National Museum of American History, Washington DC.


