The ENIAC at 70

Details of the Euler-Heun Computation

BRIAN J. SHELBURNE

This article is an addendum to the article "The ENIAC at 70" in the February 2017 issue of *Math Horizons*. It presents a more detailed description of the accumulator and master programmer units and details how the ENIAC could be programmed to execute the Euler-Heun calculation presented in the article.

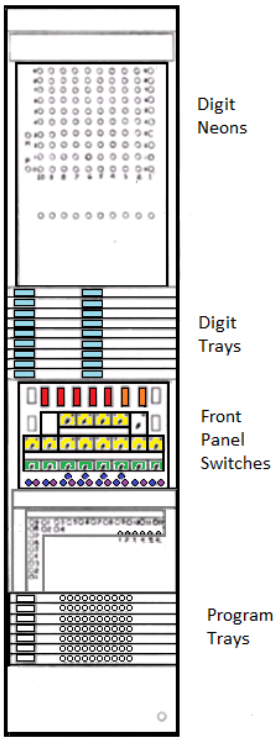

Figure 1. The front view of an accumulator.

Details of the Accumulator

Unlike today's computers, in which storage (main memory), computation (ALU), and program control are implemented as logically distinct units, the ENIAC's 20 accumulators combined memory with computation and program control.

Figure 1 shows the general layout of an accumulator (and other computational units). *Digit neons* displayed the accumulator's contents. *Digit trays* functioned as the ENIAC's common data bus. Cables connected the digit tray terminals (light blue blocks) with the five digit-input (red) and two digit-output (orange) terminals for the accumulator.

The 12 sets of front panel switches (yellow) on the front

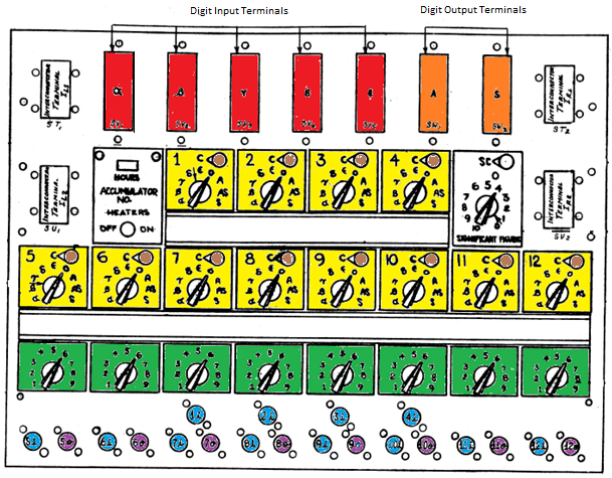

Figure 2. An accumulator's front panel.

of each accumulator were used to locally program the accumulator. To trigger the local programming of an accumulator, cables for control pulses (similar to digit pulses) were connected to terminals below the front panel switches from program lines in the *program trays*. When an accumulator had completed its operation, it could emit a control pulse to trigger the operations of other accumulators. Thus, a sequence of accumulator operations could be chained together.

An Accumulator's Local Controls

Figure 2 shows the front-panel switches of an accumulator. We see the various digit inputs and outputs, switches, and program input/output ports used for local program control.

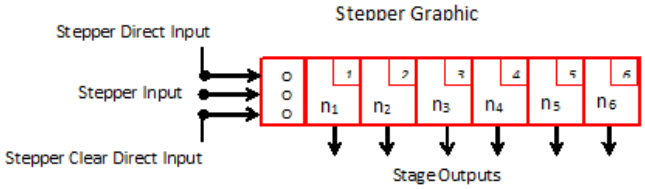

Figure 3. The front panel of the master programmer.

The row of five (red) rectangles on the upper left, labeled α, β, γ, δ, and ε, were the input ports used to receive from the digit lines; the two on the right (orange) labeled A and S were the output ports used to transmit either additively or subtractively (or both).

Cables used to connect accumulators contained 11 lines for the 10 *decades* and the sign of the number (*PM*). Special *shifter cables* could be employed to shift the digits to the left or right for multiplication or division by a power of 10 before being added or subtracted. Thus, a calculation like $y_n = y_{n-1} + y_{n-1} \cdot \Delta x$ was easily accomplished for $\Delta x = 0.01$ using a –2 shifter cable that shifted the digits of y_{n-1} to the right two places and then added the quantity to y_{n-1}.

In the middle were 12 local program control switches (yellow) arranged in two banks that were labeled α, β, γ, δ, ε, ○, A, AS, and S to receive on inputs α through ε, to do nothing, or to transmit additively only, additively and subtractively, or subtractively only, respectively. Each local program control switch had a *clear-correct* switch (the brown circle in the upper right) to optionally clear the accumulator at the end of an operation.

Underneath was a row of eight repeat switches (green) for local program control of switches five through 12. They allowed the corresponding operation to be repeated up to nine times.

Multiple accumulators could receive the contents transmitted by one accumulator over the same digit line (obviously multiple accumulators could not simultaneously transmit over the same digit line) or mutually exclusive sets of accumulators could be used at the same time to transmit using different digit lines—both being forms of parallel processing.

On the lower part were 12 input ports (blue), one for each of the 12 switches; they could receive a control pulse that would trigger the corresponding operation. The eight output ports (purple) corresponded to switches five through 12; they could forward a control pulse to the next unit when the operation was completed.

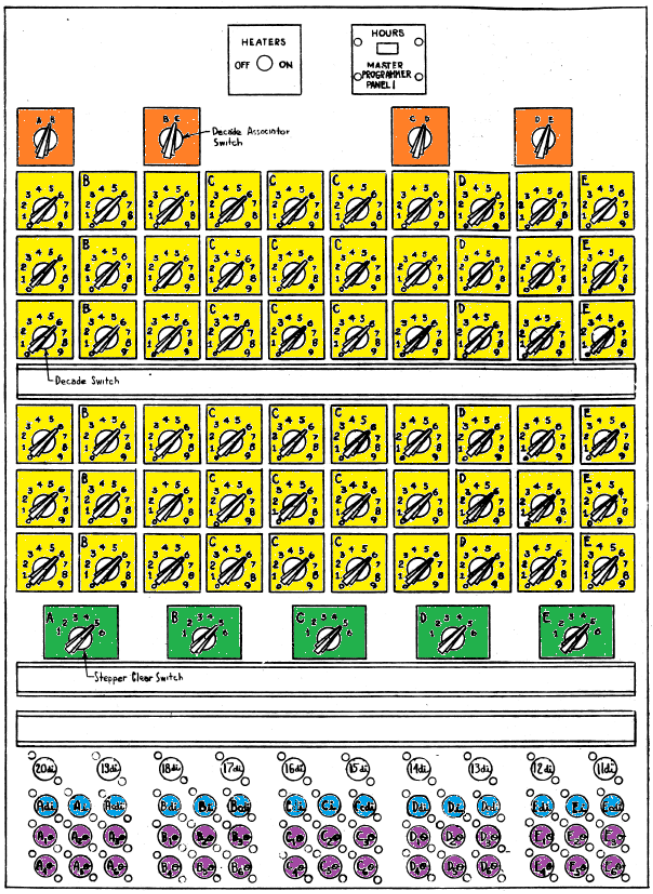

Figure 4. Stepper graphic used in ENIAC programming diagrams.

The Master Programmer

The master program provided a means of repeating a sequence of accumulator operations, and when a preset limit was reached, initiating a different sequence of accumulator operations. It consisted of 10 units called *steppers,* each with associated decades to count control pulses.

Figure 3 shows the left panel of the master programmer for steppers A through E, and figure 4 shows the schematic for a stepper. Each stepper had three input ports (blue) and six output ports (purple), one for each *stage*. Switches on the face of the master programmer (yellow) were used to fix a count, n_k, for each stage k. Control pulses received by the *stepper input port* (middle blue port) incremented that stepper's decade(s) sending a control pulse through the output port (purple) of the current stage (like a transceiver).

When the decade count reached the limit for that stage, the stepper reset the stepper decades to zero and advanced to the next stage, $k+1$. This is like a for-

loop in which the loop terminates after the limit, n_k, is reached. In this way, after a sequence of accumulator operations was executed a fixed number of times, another sequence could be executed.

A *stepper clear switch* (green) fixed the number of stages used by the stepper (up to six) so that the stages wrapped around to stage 1 when the last stage completed. In addition, each stepper had a *stepper direct input port* (left blue port) that immediately advanced the stepper to the next stage regardless of the current count (used for conditional branching) and a *stepper clear direct input port* (right blue port) that cleared (reset) the stepper to the first stage.

Each of the two panels for the master programmer had 10 sets (columns) of decades that could be combined horizontally to increase the capacity of the stages for each stepper. For steppers A through E in figure 3, columns 2, 4, 5, 6, 8, and 10 were allocated to steppers B (2), C (4, 5, and 6), D (8), and E (10). Decades in columns 1, 3, 7, and 9 could be allocated to the stepper on the left or right, as determined by the corresponding *decade association switch* (orange)

The ENIAC could also do conditional branching (if-then-else control statements). Since digit and control pulses were the same (except occurring at different times within the 200 microsecond addition time), a special cable could connect the PM line of the A output of an accumulator to the control input of another accumulator (which in turn emitted a control pulse) or to the *stepper direct input* terminal of the stepper in the master programmer (which immediately advanced the stepper to the next stage).

The PM output was either zero (positive) or nine pulses (negative) so a control pulse was sent if the source accumulator contained a negative value. (Alternately the PM line of the S output was nine pulses if the accumulator was positive.) Because of the different times at which digit and control pulses occurred, an accumulator might be used (called a *dummy program*) to convert the digital pulse into a true control pulse.

Wiring the ENIAC

In the *Math Horizons* article we described how the ENIAC could be used to solve the differential equation $\frac{dy}{dt} = y$ with initial condition $y(0) = 1$ using the Euler-Heun recurrence equation

$$y_n = y_{n-1} + h \cdot \frac{y_{n-1} + (y_{n-1} + h \cdot y_{n-1})}{2},$$

with $y_0 = 1$. (A derivation of the Euler-Heun recursion equation is given at the end of this article.) Our purpose here is to describe the wiring of the ENIAC that would carry out these calculations.

Since the ENIAC was programmed by rewiring units and setting switches, graphic wiring diagram were used for program setups. The 1946 article by Hermann and Adele Goldstine [4] presents a very nice wiring diagram for a simple ENIAC calculation of squares and cubes of integers (see also [1] and [3]). The diagram of the Euler-Heun calculation in figure 5 is based on this graphic but using color.

In the wiring diagram, digit lines on top connect AC17 statically to the printer (blue dashed line) and link the four accumulators and the constant transmitter with a common digit line (green line). Note that AC19 and AC20 are connected by -1 and -2 shifter cables, respectively.

Next are the computational units: the initiating unit, the four accumulators, and the constant transmitter. Rectangles within each are local programming control settings: T for *transmit*, C for *clear*, and R for *receive*. Numbers indicate repeats. We do not show the printer since it was accessed through the initiating unit.

Program lines (numbered 1 through 10) with connecting red and black arrows link the various units together through their local programming controls. A red arrow triggers the local programming control of a unit; a returning black arrow passes a control pulse back, which in turn triggers the local program control of the next unit(s) in the programming sequence.

For example, line 3 triggers AC17 to transmit and AC18 and AC20 to receive (red arrows). When AC17 is done, it transmits a control pulse (black arrow) to line 4.

Finally, steppers C and D control looping with stage 1 of stepper C counting 10 iterations and stage 2 triggering stage 1 of stepper D to eventually restart stage 1 of stepper C (for 50 iterations).

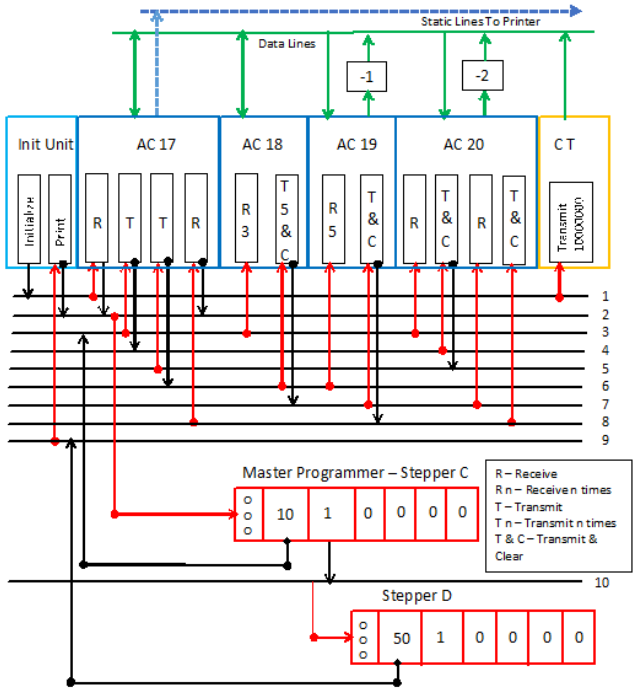

Figure 5. The setup for the Euler-Heun calculation $\frac{dy}{dt} = y$.

The calculation has three parts: an initialization, an integration step repeated 10 times to compute the next value of e^t, and a print step to print e^t for increments of 0.1.

Initialization: The initialing unit clears the accumulators, readies the printer, and then uses line 1 to transmit 10,000,000 from the constant transmitter to AC17, which is set to receive. AC17, now containing y_0, then transmits a control pulse to stepper C via line 2 to begin the main sequence.

Integration: The program uses lines 3 and 4 to compute $y_{n-1} + h \cdot y_{n-1}$, which is stored in AC18. To do so, AC18 and AC20 are both initialized to y_{n-1}, then the -2 shifter cable from AC20 transmits $0.01 \cdot (y_{n-1})$ to AC18.

Line 5 then transmits AC17 to AC18, so it now contains

$$y_{n-1} + (y_{n-1} + h \cdot y_{n-1}).$$

Note that line 3 triggers AC18 to receive three times before transmitting its contents to AC19 five times (via line 6).

To divide by 2 (thus avoiding the use of the ENIAC's divider/square rooter), the contents of AC18 is transmitted to AC19 five times (via line 6), then line 7 transmits AC19 times 0.1 (using a -1 shifter cable) to AC20.

Line 8 then transmits 0.01 times AC20, that is,

$$h \cdot \frac{y_{n-1} + (y_{n-1} + h \cdot y_{n-1})}{2},$$

back to AC17, which now contains the next value of y_n,

$$y_{n-1} + h \cdot \frac{y_{n-1} + (y_{n-1} + h \cdot y_{n-1})}{2}.$$

The program control pulse from AC17, via line 2, increments stage 1 of stepper C of the master programmer, which repeats the previous sequence until the stage 1 limit of 10 is reached. At this point, the decades in stepper C reset to 0 and stepper C advances to stage 2. However, stepper C transmits a last stage 1 control pulse (to complete the 10th iteration), and since stepper C is in stage 2, the next control pulse goes via line 10 to stepper D. Stage 2 of stepper C also reaches its limit (1) so it clears stepper C's decades to 0 and wraps around to stage 1.

Printing: Stepper D sends a control pulse via line 9 to the print input terminal on the initiating unit, which causes the contents of AC17 (y_n) to be output. At the end, a control pulse on line 2 to stepper C restarts the integration steps for 10 iterations. After reaching its limit of 50, stepper D advances to the next stage, which has no control pulse output, and the calculation halts.

And that demonstrates how the ENIAC was programmed. Whew!

Addendum: The Euler-Heun Method

Euler's method is a familiar algorithm for numerically solving a differential equation. The Euler-Heun method is not as well known. So, we will give a brief introduction here for the special case of our differential equation $\frac{dy}{dt} = y$.

Euler's method replaces the differential equation with the recursive difference equation $\frac{\Delta y}{\Delta t} = y$, or $\Delta y = \Delta t \cdot y$. From this we obtain

$$y_n = y_{n-1} + \Delta y = y_{n-1} + \Delta t \cdot y_{n-1},$$

or

$$y_n = y_{n-1} + h \cdot y_{n-1},$$

where y_0 is an initial condition and h is some small value (we used $y_0 = 1$ and $h = 0.01$).

For the Euler-Heun method, we use

$$y_n = y_{n-1} + h \cdot \frac{y_{n-1} + y_n}{2},$$

which is similar to the more efficient trapezoid rule for integration. However, because y_n is not known, we approximate y_n by $y_{n-1} + h \cdot y_{n-1}$ to obtain the Euler-Heun formula

$$y_n = y_{n-1} + h \cdot \frac{y_{n-1} + (y_{n-1} + h \cdot y_{n-1})}{2}.$$

Further Reading

There are many excellent resources on the ENIAC, a few of which are presented below.

The following article gives a detailed description of the ENIAC, including a number of excellent and detailed diagrams. Arthur Burks was on the design team for the ENIAC.

[1] Arthur Burks, Alice Burks, The ENIAC: First general-purpose electronic computer, *IEEE Annals of the History of Computing* **3** no. 4 (1981) 310–399.

The following reference contains photocopies of the original 1946 documents, so some of the type is hard to read. It also contains a set of beautifully drawn illustrations of the ENIAC units with comments explaining what each switch and terminal plug was used for.

[2] Arthur Burks, Harry Huskey, *ENIAC Operating Manual*. Reprint of the 1946 original published for the Ordnance Department, U.S. Army by Periscope Film LLC (2012) ISBN 978-1-937684-67-9.

The following technical manual was written by the mathematician Adele Goldstine.

[3] Adele Goldstine, *ENIAC Technical Manual.* Reprint of the 1946 original published for the Ordnance Department, U.S. Army by Periscope Film LLC (2012) ISBN 978-1-937684-66-2.

If you are going to read only one article about the ENIAC, we suggest the following one. It's an excellent introduction to the ENIAC that was published in 1946, soon after the announcement of the ENIAC.

[4] Hermann Goldstine, Adele Goldstine, The Electronic Numerical Integrator and Computer (ENIAC), *Mathematical Tables and Other Aids to Computation* **2** (1946) 97–110. Reprinted in *The Origins of Digital Computers—Selected Papers 3rd Ed.,* Ed. by B. Randell, Springer-Verlag, Berlin, 1982; and in *IEEE Annals of the History of Computing* **18** no. 1 (1996) 10–16.

[5] Hermann Goldstine, *The Computer from Pascal to von Neumann*, Princeton University Press, Princeton, NJ, 1972.

The following new book is an excellent history of the ENIAC, from its conception and design to how it was subsequently used at the BRL. Highly recommended!

[6] Thomas Haigh, Mark Priestley, and Crispin Rope, *ENIAC in Action: Making and Remaking the Modern Computer*, MIT Press, Cambridge, MA, 2016.

The following book deals less with the technical aspects and more with the human drama that brought the ENIAC into existence. It covers the subsequent careers of Eckert and Mauchly, who left the Moore School in 1946 and formed the Eckert-Mauchly Computer Company, which produced the UNIVAC-I, the first commercial computer in the United States. The appendix contains the complete "First Draft of a Report on the EDVAC" by von Neumann.

[7] Nancy Stern, *From ENIAC to UNIVAC: An Appraisal of the Eckert-Mauchly Computers*, Digital Press, Educational Services, Digital Equipment Corporation, Bedford, MA, 1981.

This book contains an excellent and *modern* description of the technical aspects of the ENIAC.

[8] J. Van der Spiegel, J. Tau, T. Ala'ilima, L. P. Ang, The ENIAC: History, operations, and reconstruction in VLSI in *The First Computers, History and Architectures,* Ed. by R. Rojas and U. Hashagen, MIT Press, Cambridge, MA, 2000.

Brian J. Shelburne is a professor of mathematics and computer science at Wittenberg University in Springfield, Ohio.