AUGUST 1980

HEWLETT-PACKARD JOURNAL

HEWLETT-PACKARD JOURNAL
Technical Information from the Laboratories of Hewlett-Packard Company

AUGUST 1980 Volume 31 • Number 8

Contents:

In this Issue:

Audio frequencies are frequencies within the range of human hearing, roughly 20 Hz to 20 kHz. Frequencies on either side of this range are often loosely classified as "audio," too. Thus, the frequency range of the instrument shown on the cover, Model 8903A Audio Analyzer (page 3), is 20 Hz to 100 kHz. The 8903A is used for testing—among other things—many of the electronic devices that speak to us or play music, such as CB radios and high-fidelity systems. In our cover photo it's shown plotting the frequency response of the stereo amplifier on the left at different signal levels.

There are, of course, other ways of making the measurements in the 8903A's repertoire. What makes the 8903A better? First, it's a complete system that includes a low-distortion signal source, a counter, a voltmeter, and various filters and detectors. Second, all of this is under microprocessor control, automatically stepping through complicated sequences of measurements and computations. Third, it's extremely accurate; for example, it can measure total harmonic distortion (THD) down to 0.003% under normal conditions. Fourth, the 8903A has recorder outputs that make plotting results about as easy as it can be.

If you're not so fortunate as to have studied integral calculus in school, the integral of a function probably isn't the familiar and highly useful concept that it is to scientists and engineers. One way to think of an integral is this: draw the graph of the function as a meandering line on a piece of graph paper. Then the integral of the function is the area bounded by 1) the graph of the function, 2) the horizontal axis of the graph paper, and 3) two vertical lines called the upper and lower limits of integration. Sometimes the integral of the function can be expressed neatly in mathematical terms, but more often than not it can't. So various methods have been devised for estimating integrals using computers. Because most of these numerical integration programs run on very, very large computers, it seems like a small miracle that you can now carry a numerical integrator—a very good one—around in your pocket. Beginning on page 23, its designer tells us about its capabilities and limitations.

On page 18 is an article about the nine special integrated circuits that make the HP-85 Computer possible. This set of custom integrated circuit chips minimizes the cost of the electronics, eases the problem of cooling the computer, makes the small package possible, and provides features that couldn't have been included otherwise.

-R. P. Dolan

A Custom LSI Approach to a Personal Computer

by Todd R. Lynch

THE NEW HP-85 PERSONAL COMPUTER, which was featured in last month's issue, is a system totally integrated into a single package. Included in this system are a CRT, printer, tape drive, and keyboard. To control the I/O (input/output) devices and to interface to ROM (read-only memory) and RAM (random-access memory), the design uses LSI (large-scale integrated) circuits designed and fabricated by Hewlett-Packard.

The design of custom chips minimizes the cost of the electronics. Also, the power dissipation is reduced to a level that permits the use of air-convection cooling, eliminating the need for a fan. The LSI designs save large amounts of printed-circuit-board space, making a small system package possible. By designing LSI circuits dedicated to each mechanical subassembly, features can be added to the overall system that would be nearly impossible to incorporate with discrete logic designs.

HP-85 LSI Chip Set

There are nine custom LSI circuits in the HP-85. These circuits are interconnected as shown in Fig. 1. Eight of the LSI circuit designs use NMOS technology. The ninth circuit uses a bipolar technology with two layers of metallization.

The heavy emphasis on LSI design makes the HP-85 very compact. The electronics in the machine consists of these circuits plus a power supply, clock generator, CRT raster scan circuitry, and various motor and printhead drive circuits.

The system is partitioned as follows.

- CPU (central processing unit). The CPU commands the bus control lines and interfaces directly to the system operating commands (firmware) stored in the four ROMs.
- ROMs. Each ROM chip has the same basic custom design, differing only in the bit pattern permanently stored on each chip.
- Read/Write Memory. The user memory consists of eight commercially-available dynamic 16K×1 RAMs.
- RAM Controller. This chip is designed to interface between the CPU and the RAMs.
- Buffer. This chip is designed to provide the capability for system expansion by adding plug-in units in the rear of the machine.
- Keyboard Controller.
- Printer Controller.
- CRT Controller.
- Display Memory. Four commercial dynamic 16K×1 RAMs are used to store data for the display. Approximately one-fourth of this memory is used for the alphanumeric data and the remainder is used for graphic data.
- Cartridge Controller.
- Sense Amplifier. This bipolar chip interfaces the tape cartridge controller to the magnetic tape head.

Table I is a summary of the custom LSI chip sizes, number of pins (wiring connection points), and power dissipation.

The machine's sixteen-bit address allows direct access to 65,536 bytes of information. The memory map in Fig. 2 shows how this space is allocated. Note that 32K bytes of the address space are devoted to ROM while 16K bytes are for RAM. Additional RAM and ROM capacity may be added to the system. The I/O address space occupies the upper 256 bytes of memory. Each I/O device controller has from two to four dedicated addresses assigned to it.

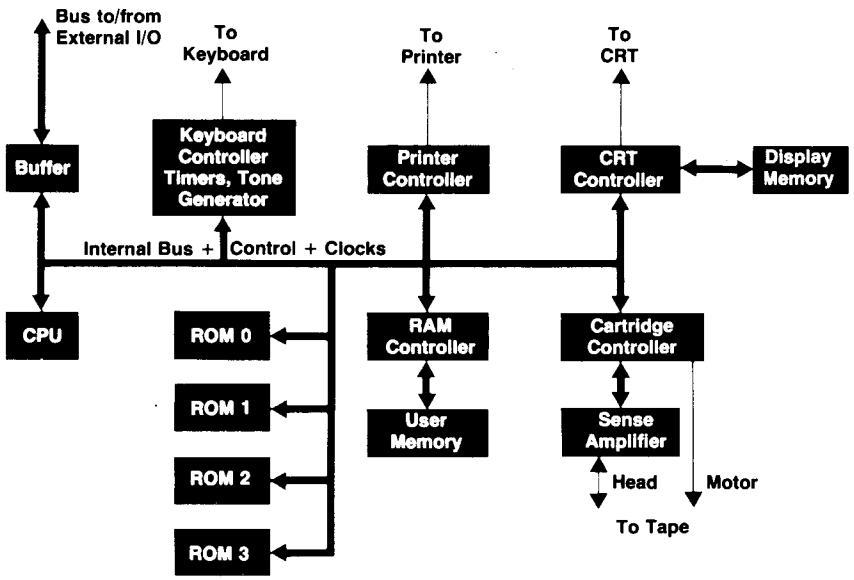

Fig. 1. *The HP-85 system block diagram contains nine custom LSI circuit designs. The user and display memories are the only parts in the diagram that use commercially available circuits. The development of custom circuits enabled the system to be contained in a compact package at low cost and without the need for a cooling fan.*

Table I

Custom LSI circuit characteristics. All of the circuits are fabricated with NMOS silicon gate technology except for the sense amplifier which is made with dual-layer-metallization bipolar technology.

Circuit:	Size (mm)	#Pins	Power (mW)
CPU	4.93×4.01	28	330
ROM	4.75×5.41	28	200
RAM Controller	2.67×3.56	40	220
Buffer	2.54×3.35	40	340
Keyboard Controller	3.78×4.39	42	200
Printer Controller	4.78×5.44	40	300
CRT Controller	4.14×5.51	40	200
Cartridge Controller	3.63×3.86	28	55
Sense Amplifier	1.50×1.55	16	150

CPU Design

Several commercially available CPUs were considered at the beginning of the project, but none could provide all the features needed to efficiently implement a powerful scientific BASIC language machine. BASIC requires high-precision arithmetic, which is best accomplished with decimal rather than binary numbers. Many different stacks are needed to parse (separate a statement into executable steps) and execute the language. The ability to handle variable-length data is required for variable-length tokens (bit sequences from one to several bytes in length), and multilevel vectored interrupts are needed to handle the I/O devices in real time. The design of the custom NMOS CPU incorporates all of these requirements plus many more.

Fig. 3 shows a block diagram of the HP-85 CPU. Major blocks are the 7000-bit PLA (programmable logic array), 64-byte register bank, and eight-bit ALU (arithmetic logic unit) and shifter. Each CPU instruction is decoded by a microprogram in the PLA that directs the rest of the chip to perform the desired function.

A very powerful feature of the CPU, which is incorporated into the PLA microprogram, is the ability to handle

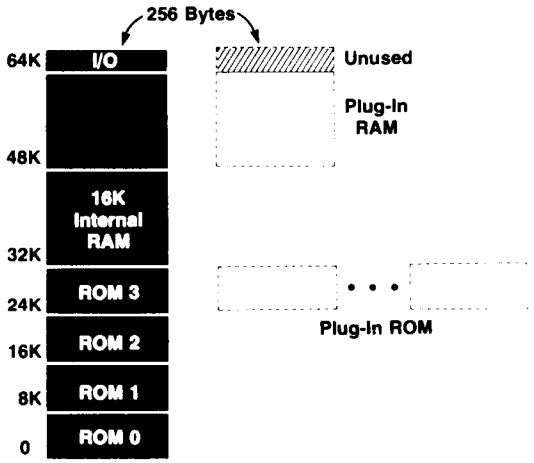

Fig. 2. *The HP-85 memory map allocates four 8K-byte system ROMs and one 16K-byte internal RAM. The RAM is used for user memory and 256 dedicated I/O locations. By using bank selection, up to six more 8K-byte ROMs may be added. The RAM may be expanded by 16K bytes if a plug-in module is added to the system.*

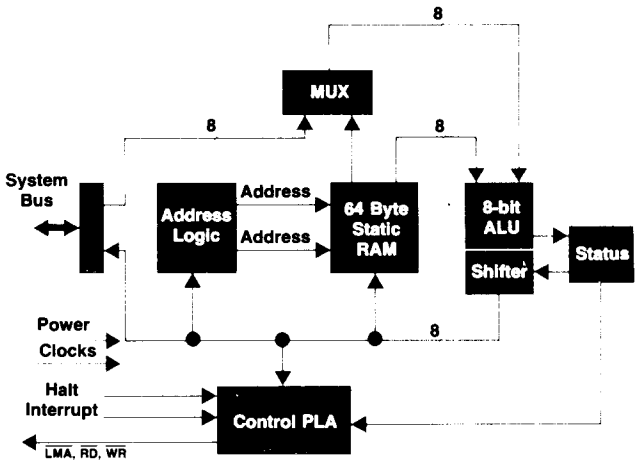

Fig. 3. *The CPU in the HP-85 is a custom NMOS circuit design. A 7K-bit programmable logic array (PLA) controls the interactions between the register bank, the ALU, and the rest of the HP-85 system.*

data ranging from one to eight bytes in length. This multibyte feature lets the programmer, for example, add two eight-byte mantissas, increment a sixteen-bit address, or load into the CPU a three-byte token from memory, each with a single instruction. With an off-the-shelf CPU, this would require setup and iteration within a software loop.

The eight-bit ALU is capable of shifting and can do both decimal and binary arithmetic. The programmer sets a mode bit to specify the type of shifting or arithmetic desired. This, combined with the multibyte feature, lets the programmer easily work with signed, floating-point mantissas up to sixteen digits in length or two's-complement binary integers up to sixty-four bits in length.

Of the 64 eight-bit registers contained in the CPU, one pair is dedicated to the program counter, one pair to the stack pointer, and one pair to internal index calculations. The rest are general-purpose registers. One advantage of having a register as the program counter is that it can easily serve as one of the operands for any CPU instruction. If it is modified by an instruction, then an immediate jump to the new location occurs. The stack pointer points to the subroutine return address stack stored in memory. Additional stacks can be created with any other consecutive pair of registers in the CPU. Thus, the programmer can maintain many different stacks at any one time. Sixteen instructions are dedicated to manipulating data on the currently designated stack. Any pair of consecutive registers can also be used as a sixteen-bit index register. This lets the programmer index into several data arrays at the same time.

To handle real-time I/O devices, the CPU has multilevel vectored interrupt handling ability. Up to 127 interrupt vectors can be accommodated by the architecture. The CPU can also be halted by an external device. This lets that device control the system bus, so it can have direct memory access at high speeds.

No accumulator is present in the CPU. This is made possible by the design of the 64-byte register bank. The registers are constructed as a two-read/one-write memory. This means that, at a given time, any two bytes can be read

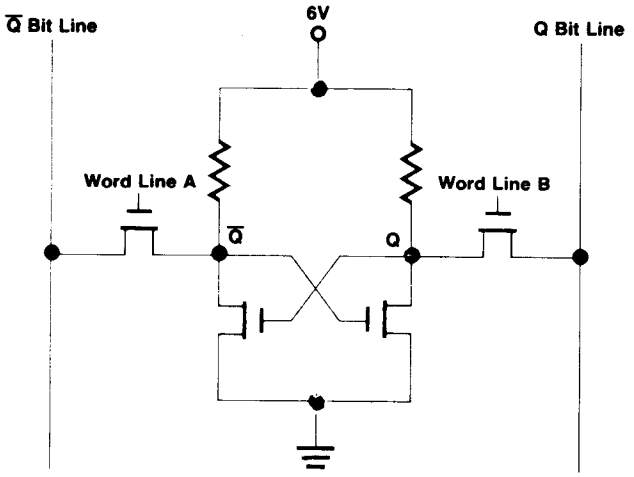

Fig. 4. *Independent control of each word line is used to achieve a cell that can be read out onto either bit line. The RAM made up with these cells can be read two words at a time.*

from the memory and operated on in the ALU. The result is returned to one of the accessed byte locations. This sequence takes one processor cycle (1.6 μs).

To design a two-read/one-write memory, a standard static RAM cell with a special variation is used (see Fig. 4). In a normal static RAM cell, a common word line enables both transmission gates between the bit lines and the cell. In a two-read cell, there must be two different word lines. Word line A enables complement information from one cell to be read out onto the $\overline{Q}$-bit line while word line B enables true information from another or the same cell to be read out on the Q-bit line.

The circuit design of such a cell must be done carefully. Since the bit lines are precharged before reading, it is possible that by enabling a single word line, a cell could be made to flip (change state) rather than be read. Inadvertent flipping is prevented by using a low voltage for the word line and a proper size ratio between the cell's pull-down transistor and the transmission gate.

The static RAM cell's pull-up device also received close scrutiny during the design phase. A cell with a low-power depletion-load pull-up requires an area larger than 6400 square micrometres (10 square mils) and a quiescent power of 120 microwatts. By using a polysilicon pull-up resistor, the RAM cell area is reduced by 40% and the power is reduced by 80%. The savings in area amounts to a reduction in size of more than 0.25 mm on each side of the die. The reduction in total power consumption is fifty milliwatts.

To obtain these reductions, a polysilicon resistor process was developed for the CPU. An additional masking step is required to define the regions on the chip where the polysilicon layer is lightly doped, hence creating the polysilicon resistors. The doping level was chosen for a sheet resistance of 10^7 Ω/□. An undoped layer could not be used because the sheet resistance was so high that the junction leakage currents at elevated temperatures resulted in unwelcome voltage drops across some resistors. It was also discovered that if the contact mask overlapped the polysilicon resistor area, the aluminum metallization could spread into the lightly doped resistor and lower its resistance.

System Control and Timing

Eight bus and three control lines leave the CPU. The eight-bit bus is used to time-multiplex a sixteen-bit address, instructions, and multibyte data quantities. The three control lines—$\overline{LMA}$ (load memory address), $\overline{RD}$ (read) and $\overline{WR}$ (write)—indicate to the system circuits what type of information is on the bus.

When $\overline{LMA}$ is low, all chips in the system know that one byte of a two-byte address will be placed on the bus. The $\overline{LMA}$ signals always come in pairs since all addresses are sixteen bits. Each chip in the system reads this address and compares it to the address range to which the chip is supposed to respond. Setting $\overline{RD}$ low indicates that the CPU wants to read the contents of the address most recently sent out. When $\overline{WR}$ goes low, the CPU wants to write into the location most recently addressed.

The multibyte feature is accomplished by sending out an address with two $\overline{LMA}$ signals followed by one to eight $\overline{RD}$ or $\overline{WR}$ signals. The circuit being read from, or written to, is expected to increment its memory address register every time it sees a $\overline{RD}$ or $\overline{WR}$. The CPU sometimes can fetch consecutive instructions from memory by simply sending out additional $\overline{RD}$ commands. This speeds up the instruction rate of the machine since sending out an address for every instruction and piece of data is not necessary.

Four nonoverlapping phases (Fig. 5) with 200 ns width and 200 ns spacing clock the circuits in the HP-85. In one cycle the system can access a preaddressed memory location, read it into the CPU, add it to a CPU register and store the result in the CPU register. An eight-byte add requires eight cycles plus the time to fetch the command, which is usually three cycles.

Before any chip writes on the bus, the bus gets precharged to a logic one. Therefore, the circuit desiring to put a one on the bus must merely continue to hold the bus high. If it wants a zero, it must discharge the bus. In NMOS technology it is easier to discharge a bus than to bring it high. Consequently, the circuits were designed with large devices to discharge the bus and relatively small devices to maintain a high level on the bus. This minimizes the chip area required for each circuit's driver logic.

The system bus has fewer spurious transitions using the

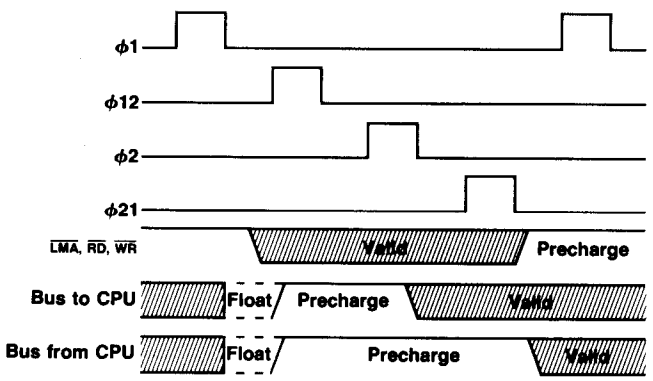

Fig. 5. *HP-85 system timing and control. The control lines are valid by the time the φ12 clock pulse occurs. If a system chip is to be read, it must respond during the φ2 clock pulse. Information then enters the CPU, goes through the ALU, and is stored during the next φ1 clock pulse. When the CPU sends out addresses or data, they are valid by the time the φ1 clock pulse occurs.*

precharged scheme. Timing is such that the driving circuit is not enabled until its data is valid and ready to go on the bus. It is not possible to discharge the bus inadvertently when the chip wants a logic one to be output.

ROM

The NMOS system ROM is organized as an 8K by eight-bit array. Because this circuit must respond to multibyte transfers, its memory address register is designed to increment as $\overline{RD}$ commands are given. The circuit can be enabled or disabled by a bank select command. When plug-in ROMs are used, it becomes necessary to selectively enable or disable ROMs in the address space from 24K to 32K (Fig. 2). Each ROM in this address space has a unique eight-bit bank number, and only one bank can be active at any given time.

The core of the ROM consists of 64-input, minimum geometry, NAND gate arrays. The principle of operation is to precharge the NAND stack from both ends. This requires 600 ns. At the beginning of phase $\phi 12$, one 64-device stack per bit of output is selected. If the stack discharges, the bus bit will be a zero, otherwise it will be a one. The transistors in the stack are preprogrammed during fabrication with the depletion mask. A depletion transistor is a logic zero, an enhancement transistor is a logic one. A high voltage on either transistor causes conduction, while a low voltage causes conduction only through the depletion device.

When trying to discharge a 64-device stack, 63 inputs are high and the selected bit is low. If a depletion device is at that location, the stack discharges, producing a zero.

RAM Controller

The NMOS RAM controller chip interfaces the CPU to eight 16K dynamic RAMs. Because it is a memory controller chip it has an incrementing memory address register. Also, the address space to which it responds is mask programmable. Therefore, a RAM controller chip different from the one in the mainframe of the HP-85 is used for the 16K RAM plug-in. This chip knows not to respond to dedicated I/O addresses in the upper 256 bytes of memory.

An important function of the RAM controller chip is refreshing the dynamic memory. An internal timer on the chip tells it when the next sequential location is due to be

The HP-85 Software Development System

by Nelson A. Mills

The HP-85 is based upon a new eight-bit custom processor. One of the first steps in the development of the machine was to provide a set of software development tools. The HP-85 software development system consists of an assembler, a hardware interface for the HP-85 simulator, and software debug system (see Fig. 1).

The assembler for the HP-85 processor was designed to run on an HP 1000 Computer. The assembler supports the full range of the HP-85 processor's instruction set and provides several useful pseudo-operations. Programs may be either absolute or relocatable and the program origin may be reset at any time. Several instructions are provided to facilitate data definition and may exist locally within the program, or may be retrieved from a file of global data definitions.

The hardware interface designed for the HP-85 software development system provides the ability to use an HP 1000 to control execution of the HP-85 simulator. The interface contains an on-board RAM which can be downloaded from the HP 1000 and is used to simulate the HP-85 ROM. Included are two breakpoint registers which can be used to halt execution of the HP-85 processor at either of two specified addresses. The processor can be made to execute steps in either a continuous-run mode, or in a single-step mode where execution halts after each instruction. The interface also provides the ability to halt the HP-85 processor at any time, and to read data from or write data to the system RAM or CPU registers.

The software debug system, provided as part of the development system, runs in the HP 1000 and performs four important functions. It includes a relocatable loader that is used to download the RAM on the interface board with the software under development. It controls execution of the HP-85 simulator via the interface board. It displays the current status of the breadboard, including CPU status, program counter, and the current contents of all CPU registers and any specified memory locations. Finally, it provides the means for system developers to modify the contents of memory, registers, status, or program counter, and to clear and set breakpoints.

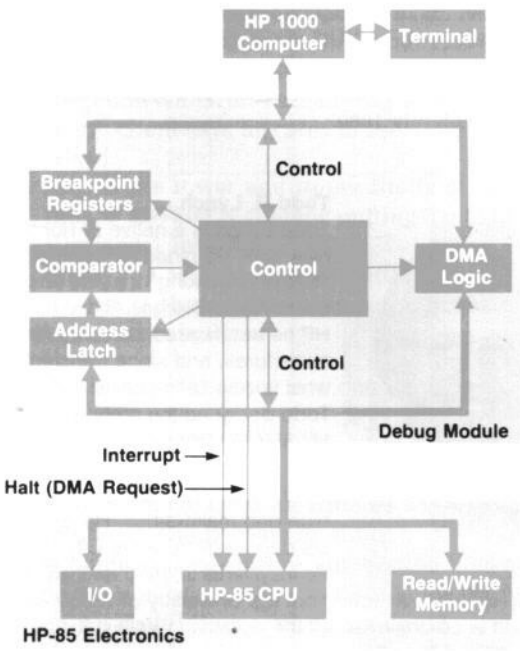

Fig. 1. *HP-85 software development system.*

Nelson A. Mills

Joining HP in 1976, Nelson Mills worked on the HP-85 operating system and interpreter. He is now the project manager for high-end firmware at the Corvallis Division. After receiving a BS in mathematics from Albion College, Michigan in 1961, Nelson spent five years in the U.S. Navy and then worked for ten years as a systems programmer. He and his family—wife and two children—live in Corvallis, Oregon. Outside of working hours Nelson enjoys photography, hiking, camping, and coaching basketball and AYSO soccer.

refreshed. The chip then waits for a cycle when memory is not being accessed to do the refresh. There is a two-element queue in case a vacant memory cycle does not come before the following location must be refreshed.

Keyboard Controller

The circuit needed just to interface to the keyboard would have been quite small. Consequently, four programmable timers were added to this NMOS chip. The timers can count up to 27 hours with one-millisecond resolution. Each has a maskable interrupt. When the timer gets to its preset count, it interrupts, resets and counts again.

The keyboard portion of the chip can scan an 8×10-key keyboard plus three dedicated keys—shift, control, and caps lock. The key debounce time is mask programmable from 1.67 to 11.69 ms. After the key has been debounced, the chip generates an interrupt. An internal ROM is used to convert the key position to its ASCII* equivalent.

Another feature of the chip is an output to a speaker for audio tones. Via firmware, a 1.2-kHz output tone can be obtained, or the frequency can be varied by periodically setting and clearing an internal flip-flop. This output can be accessed by the programmer using a command that can specify the frequency and duration of the tone.

CRT Controller

The NMOS CRT controller[1] interfaces the CPU to a 127-mm diagonal CRT and its dedicated display memory. The memory is four 16K×1 dynamic RAMs. This is enough storage to hold a full display of graphics information (256×192 dots) plus four displays of alphanumeric information (4×32 characters/line × 16 lines/display). The chip must make sure that this memory is properly refreshed during the vertical retrace. The interface to the video drive circuitry consists of a vertical sync pulse (60 Hz), a horizontal sync pulse (15.7 kHz), and a video line (4.9 MHz). An internal ROM provides a dot pattern translation of the stored ASCII characters.

Printer Controller

The NMOS printer controller[2] is the interface between the CPU and the thermal moving-head printer. To perform this function, the circuit must control a printhead drive motor, a paper advance motor and an eight-dot printhead. An internal 32-byte RAM allows firmware to buffer one print line of alphanumeric data at a time. An internal ROM translates an ASCII character into its appropriate dot pattern.

Tape Controller and Read/Write Amplifier

The NMOS tape controller and the bipolar read/write amplifier[3] control the tape unit in the HP-85. The read/write amplifier reads and writes data on the tape through a two-track magnetic head. The tape is formatted using a 1:1.75 delta distance code, in which 8-kHz flux reversals represent zeros and 4.6-kHz flux reversals represent ones.

The tape controller encodes digital information into this format and sends it to the read/write amplifier. When reading the tape, the signals from the read/write amplifier are decoded by the controller. The tape drive motor direction and speed are also controlled by the chip.

*American Standard Code for Information Interchange

Buffer

The NMOS buffer is the ninth chip in the system. With its help, the system bus can be expanded to include all of the plug-in I/O slots. It is capable of driving a 150-pF load. The delay between NMOS-level input and NMOS-level output is 50 ns. The signals that the buffer passes are eight bus lines, three control lines, and the interrupt and halt lines. The bus and control lines are designed to be bidirectional.

The clock lines are not buffered since this would cause unwanted skewing. The clock generator is capable of driving all internal as well as all external loads.

Acknowledgments

A great deal of credit must be given to the team that made this LSI chip set possible. Tim Williams designed the RAM controller and buffer. Donn Wahl designed the ROM. The keyboard controller was an effort from Jim Hutchins and Jerry Erickson. Jerry also designed the CRT controller. The printer controller was the work of Clement Lo. Implementation of the NMOS CPU was done by Jim Axtell. The cartridge controller design is that of Doug Collins. The sense amplifier is the result of the combined efforts of Mike Moore, Doug Collins and Mike Barbour. Special thanks must be given to Don Hale and Howard Shishido, who did the majority of the chip mask designs. Howard Honig contributed to the project by writing most of the test programs for the LSI circuits. Mike Pan, Rock Davidson, and Tom Kraemer also contributed to the chip set design effort. Bob Tillman and Steve Larsen helped get the NMOS process and polysilicon resistor development up to production standards. Payne Freret contributed some good ideas to the CPU design. Rick Bell did the design of the clock oscillator and generator.

Reference

1. J.F. Bausch, "A High-Quality CRT Display for a Portable Computer," Hewlett-Packard Journal, July 1980.
2. C.C. Lo and R.W. Keil, "A Compact Thermal Printer Designed for Integration into a Personal Computer," Hewlett-Packard Journal, July 1980.
3. D.J. Collins and B.G. Spreadbury, "A Compact Tape Transport Subassembly Designed for Reliability and Low Cost," Hewlett-Packard Journal, July 1980.

Todd R. Lynch

Todd Lynch is a native of Rochester, New York. He joined HP Laboratories in 1972 after working for a year on process control logic designs. Since coming to HP he has worked on computer architectures, and since transferring to what is now HP's Corvallis Division, Todd designed the architecture for the HP-85 CPU. He was the project manager for the integrated circuits in the HP-85 and is now project manager for high-end mainframes. He received a BS degree from Grove City College, Pennsylvania in 1970 and an MSEE from the University of Wisconsin in 1971. Todd is co-chairman for the Willamette Valley Junior Tennis Tournament and in addition to tennis enjoys woodworking, softball, and skiing. He lives in Albany, Oregon with his wife and new son.

Handheld Calculator Evaluates Integrals

The HP-34C is the first handheld calculator to have a key that performs numerical integration almost automatically. It may change your attitude towards what used to be regarded as a dreary tedious task.

by William M. Kahan

NUMERICAL INTEGRATION has been the subject of about two thousand books and learned papers, with a dozen or so "new" methods published every year. And yet the task in question has a simple geometrical interpretation seen in Fig. 1: given an expression f(u) and lower and upper limits y and x respectively, the value

$$I = \int_y^x f(u)\,du$$

represents the area under the graph of f(u) for u between y and x. Why so much fuss?

As I write this an electrical engineering colleague, Professor J. R. Woodyard, enters my office and asks to have

$$I_1 = \int_0^1 \left(\frac{\sqrt{u}}{u-1} - \frac{1}{\ln u} \right) du$$

evaluated on my HP-34C Calculator (Fig. 2). Let's do it.

Step 1. Key into the calculator under, say, label **A** a program that accepts a value u in the display (X register) and displays instead the computed value of the integrand

$$\sqrt{u}/(u-1) - 1/(\ln u)$$

Fig. 3 shows an HP-34C program to do this.

Step 2. Restore the calculator to **RUN** mode and set the display to, say, **FIX 5** to display five decimal digits after the point, which are as many digits of the integrand as my client says he cares to see. (More about this later.)

Step 3. Key in the lower and upper limits of integration thus, **0 ENTER↑ 1**, thereby putting 0 into the Y register and 1 into X.

Step 4. Press $\int_y^x$ **A**, wait 25 seconds until the display shows 0.03662, then press **x≤y** to see 0.00001. We have just calculated

$$I_1 = 0.03662 \pm 0.00001.$$

That was easy—too easy. Woodyard smiles as if he knew something I don't know. Could the calculator be wrong? How does the calculator know the error lies within ±0.00001?

Many other questions come to mind:

- Why is numerical integration impossible in general?
- Why do we persist in trying to do it anyway?
- How do we do it? How well do we do it?
- How does the $\int_y^x$ key compare with other integration schemes?
- What can go wrong and how do we avoid it?
- What else have we learned?

These questions and others are addressed in the following pages.

Tolerance and Uncertainty

Integrals can almost never be calculated precisely. How much error has to be tolerated? The $\int_y^x$ key answers this question in a surprisingly convenient way. Rather than be told how accurately $I = \int_y^x f(u)du$ should be calculated, the HP-34C asks to be told how many figures of f(u) matter. In effect, the user is asked to specify the width of a ribbon drawn around the graph of f(u), and to accept in place of I an estimate of the area under some unspecified graph lying entirely within that ribbon. Of course, this estimate could vary by as much as the area of the ribbon, so the calculator estimates the area of the ribbon too. Then the user may conclude from Fig. 4 that

I = (area under a graph drawn in the ribbon)
± (½ area of the ribbon)

The calculator puts the first area estimate in its X register and the second, the uncertainty, in the Y register.

For example, f(u) might represent a physical effect whose magnitude can be determined only to within, say, ±0.005. Then the value calculated as f(u) is really f(u) ± Δf(u) with an uncertainty Δf(u) = 0.005. Consequently **FIX 2**, which tells the calculator to display no more than two decimal digits after the point, is used to tell the calculator that decimal digits beyond the second cannot matter. Therefore the calculator need not waste time estimating $I \pm \Delta I = \int_y^x (f(u) \pm \Delta f(u))du$ more accurately than to within an uncertainty $\Delta I = \left|\int_y^x \Delta f(u)du\right|$. This uncertainty is estimated together with $I \pm \Delta I$, thereby giving the calculator's user a fair idea of the range of values within which I must lie.

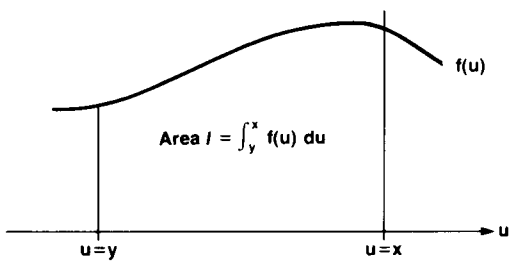

Fig. 1. *An integral interpreted as an area.*

Fig. 2. *HP-34C Calculator has keys to solve any equation and to compute integrals.*

The uncertainty Δf(u) is specified by the user via the display setting. For instance, **SCI 5** displays six significant decimal digits, implying that the seventh doesn't matter. The HP-34C allows the user's f-program to change the display setting, thereby providing for uncertainties Δf(u) that vary with u in diverse ways. But users usually leave the display set to **SCI 4** or **FIX 4** without much further thought.

By asking the user to specify Δf(u) instead of Δ*I* the HP-34C helps avoid a common mistake—wishful thinking. Other integration procedures, which conventionally expect the user to specify how tiny Δ*I* should be, blithely produce estimates of *I* purporting to be as accurate as the user wishes even when the error Δf(u) is far too big to justify such claims to accuracy. The HP-34C does not prevent us from declaring that f(u) is far more accurate that it really is, but our attention is directed to the right question and not distracted by questions we cannot answer. Whether we specify Δf after a careful error analysis or just offer a guess, we get estimates $I \pm \Delta I$ that we can interpret more intelligently than if we got only *I* with no idea of its accuracy or inaccuracy.

A Survey of Integration Schemes

Students are taught the fundamental theorem of calculus:

$$I = \int_y^x f(u)\,du = F(x) - F(y) \quad \text{provided } \frac{d}{du}F(u) = f(u)$$

This means that one could calculate *I* if one could discover somehow an expression F(u) whose derivative is the given expression f(u). Students are taught integration as a process, applied to expressions, that starts with f and ends with F. But in professional practice that process hardly ever succeeds. A compact expression F(u) is almost always difficult or impossible to construct from any given f(u). For instance, neither

$$\int_{-\infty}^{x} \exp(-u^2/2)du/\sqrt{2\pi} \quad \text{nor} \quad \int_0^{\infty} \exp(-u + x \ln u)du$$

possesses a closed form, that is, an expression involving only finitely many elementary operations (+, −, ×, ÷, ln, exp, tan, arctan, ...) upon the variable x. Nevertheless, both integrals can be approximated arbitrarily accurately by aptly chosen formulas. So often do statisticians and engineers need values of those integrals that formulas for them, accurate to ten significant decimal digits, can now be calculated in a few seconds by pressing a key on certain handheld calculators. (Press **Q** on the HP-32E to get the first integral, the cumulative normal distribution; press **x!** on the HP-34C to get the second integral, the gamma function Γ(1+x), whether x be an integer or not.)

Almost every rare integrand f(u) whose indefinite integral $F(x) = \int^x f(u)du + c$ is expressible in a compact or closed form can be recognized by a computer program that accepts the string of characters that defines f and spews out another string that represents F. (Such a program is part of the MACSYMA system, developed at MIT, that runs on a few large computers—two million bytes of memory—at several universities and research labs.) Perhaps the terms "compact" and "closed form" should not be attached to the expression F(x), since usually, except for problems assigned to students by considerate teachers, the integral

LBL A	Begin with u in the X register
ENTER↑	Save u in the stack
$\sqrt{x}$	... $\sqrt{u}$
LST X	Recall u
1	
−	
+	... $\sqrt{u}/(u-1)$
x≷y	Recall u again
LN	
1/x	... 1/ln(u)
−	
RTN	Display $\sqrt{u}/(u-1) - 1/\ln u$

Fig. 3. *This program makes the HP-34C calculate the integrand* $\sqrt{u}/(u-1) - 1/\ln u$ *when the argument u is in the X register and key* **A** *is pressed. Labels B, 0, 1, 2, or 3 would have served as well as A.*

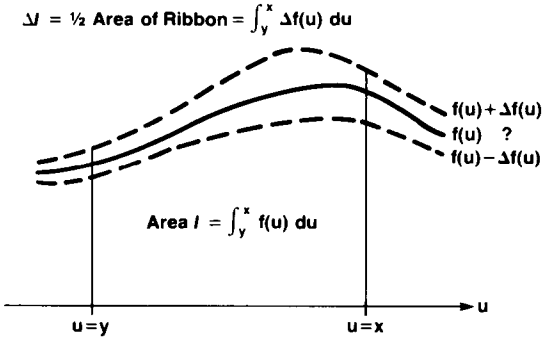

Fig. 4. *The graph of an uncertain integrand f(u) ±Δf(u) can run anywhere in the ribbon bounded by the dashed lines. The area under such a graph, I ±ΔI, is uncertain by ±ΔI, which is one-half the area of the ribbon. The HP-34C displays its estimate of I ±ΔI in its X register and holds an estimate of ΔI in its Y register.*

F(x) far exceeds the integrand f(u) in length and complexity. Shown in Fig. 5 are two compact forms and one closed form for F(x) when $f(u) \equiv 1/(1+u^{64})$. The extent to which F(x) is here more complicated than f(u) is atypically modest out of consideration for the typesetter. The formulas in Fig. 5 will remind many readers of hours spent on calculus problems, but they do not provide economical ways to calculate F(x) for any but very big or very tiny values of x. When I use the HP-34C's $\int_y^x$ key to calculate $F(1) = \int_0^1 du/(1+u^{64}) = 0.989367 \pm 0.000004$ the answer appears in 200 seconds including 20 seconds taken to enter the f-program plus 180 seconds for a result (in **SCI 5**). Calculating F(1) from any formula in Fig. 5 takes at least about ten times longer, not including the time taken to deduce the formula. Engineers and scientists have long been aware of the shortcomings of integration in closed form and have turned to other methods.

Perhaps the crudest way to evaluate $\int_y^x f(u)du$ is to plot the graph of f(u), like Fig. 1, on uniformly squared paper and then count the squares that lie inside the desired area. This method gives numerical integration its other name: numerical quadrature. Another way, suitable for chemists, is to plot the graph on paper of uniform density, cut out the area in question, and weigh it. Engineers used to measure plotted areas by means of integrating engines called planimeters. These range from inexpensive hatchet planimeters of low accuracy to Swiss-made museum pieces costing hundreds of dollars and capable of three significant decimals. (For more details see reference 1). Nowadays we reckon that the computer will drive the graph plotter so it might as well integrate too.

Today's numerical integration techniques are best explained in terms of averages like

$$A = I/(x-y) = \int_y^x f(u)du/(x-y)$$

which is called "the uniformly weighted average of f(u) over the interval between x and y." Another kind of average,

$$A = \sum_{j=1}^{n} w_j f(u_j) \quad \text{where} \quad w_j > 0 \text{ and } \sum_{j=1}^{n} w_j = 1,$$

is a finite weighted average of n samples $f(u_1), f(u_2), \ldots, f(u_n)$. Provided the sample arguments $u_1, u_2, \ldots, u_n$, called nodes, all lie between x and y the sample average A will approximate, perhaps poorly, the desired average A, and hence provide $I = (x-y)A$ as an approximation to $I = (x-y)A$. Statisticians might be tempted to sprinkle the nodes u_j randomly between x and y—that is what Monte Carlo methods do. But randomness is a poor substitute for skill because the error $A - A$ tends to diminish like $1/\sqrt{n}$ as the number n of random samples is increased, whereas uniformly spaced and weighted samples provide an error that diminishes like $1/n^2$. Other more artful methods do even better.

Different numerical integration methods differ principally in the ways they choose their weights w_j and nodes u_j, but almost all have the following characteristics in common. Each average A is associated with a partition of the range of integration into panels as shown in Fig. 6. Each panel contains one node u_j whose respective weight is

w_j = (width of panel j)/(width of range of integration).

The formula given above for A amounts to approximating the area in each panel under the graph of f(u) by the area of a rectangle as wide as the panel and as high as the sample $f(u_j)$. The simplest method is the midpoint rule, whose nodes all lie in the middles of panels all of the same width. Other methods, like the trapezoidal rule and Simpson's rule, vary the panel widths (weights) and nodes in ways designed to exploit various presumed properties of the integrand f(u) for higher accuracy. Which method is best? If this question had a simple answer there would not be so many methods nor would we need texts like "Methods of Numerical Integration" by P.J. Davis and P. Rabinowitz,[2] which contains 16 FORTRAN programs and three bibliographies with well over 1000 citations.

For example, consider Gaussian quadrature. This method is widely regarded as "best" in the sense that it very often requires fewer samples than most other methods to achieve an average A that approximates the desired A to within some preassigned tolerance. But the weights and nodes of Gaussian quadrature take quite a while to calculate. Programs to do so, and the resulting tables of weights and nodes for various sample counts n, have been published.[3] Had we chosen Gaussian quadrature for the $\int_y^x$ key we would

$$F(x) \equiv \int_0^x f(u)\,du \quad \text{where} \quad f(u) \equiv 1/(1+u^{64}).$$

$$F(x) = x\sum_{k=0}^{\infty} (-x^{64})^k / (64k+1) \quad \text{if} \quad x^2 \le 1$$

$$= \frac{\pi}{64}\csc\left(\frac{\pi}{64}\right)\operatorname{sign}(x) + x\sum_{k=1}^{\infty} (-x^{-64})^k / (64k-1) \quad \text{if} \quad x^2 \ge 1$$

$$= \frac{1}{32}\sum_{k=1}^{16}\left(\sin\theta_k \cdot \arctan\left(\frac{2x\sin\theta_k}{1-x^2}\right) + \tfrac{1}{2}\cos\theta_k \cdot \ln\left(1 + \frac{2}{\frac{x+x^{-1}}{2\cos\theta_k} - 1}\right)\right)$$

$$+ \left(\frac{\pi}{64}\csc\left(\frac{\pi}{64}\right)\operatorname{sign}(x) \quad \text{if} \quad x^2 > 1\right) \qquad \text{where } \theta_k = (k-\tfrac{1}{2})\pi/32$$

Fig. 5. *Formal integration transforms many a simple expression f(u) into messy formulas F(x) of limited numerical utility.*

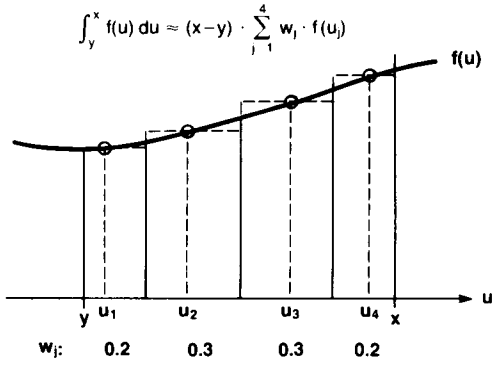

Fig. 6. *The integral, regarded as an area, is here divided into four panels each of whose areas is approximated by the area of a rectangle as wide as the panel and as high as a sample.*

have had to store at least as many nodes and weights as we could expect to need for difficult integrals, amounting to at least several hundred 13-digit numbers, in read-only memory. But that would have left no space in the HP-34C for anything else, so a different method had to be found.

The $\int_y^x$ key could not use a method that generates just one average A because that gives no indication of how accurately it approximates A. Instead we looked only at methods that sample repeatedly and with increasing sample counts $n_1 < n_2 < n_3 < \ldots$ to produce a sequence of increasingly accurate averages $A_1, A_2, A_3, \ldots$. Provided that sequence converges to A so fast that each $|A_{k+1} - A|$ is considerably smaller than its predecessor, the error $|A_k - A|$ can be approximated accurately enough by $|A_k - A_{k+1}|$, and the last average A_{k+1} can be accepted in lieu of A as soon as $|A_k - A_{k+1}|$ is tolerably small.

How small is "tolerably small"? That depends upon the area of the ribbon discussed above under "Tolerance and Uncertainty." Since the integral $I = \int_y^x f(u)du$ inherits an uncertainty $\Delta I = \left|\int_y^x \Delta f(u)du\right|$ from the uncertainty $\Delta f(u)$ in the integrand, so does $A = I/(x-y)$ inherit an uncertainty $\Delta A = \Delta I/|x-y|$, which may be approximated by

$$\Delta A = \sum_{j=1}^{n} w_j \Delta f(u_j)$$

in the same way as A is approximated by A. Indeed, A and ΔA can be computed together since they use identical weights and nodes. And so the sequence $A_1, A_2, A_3, \ldots$ is accompanied by a sequence of respective uncertainty estimates $\Delta A_1, \Delta A_2, \Delta A_3, \ldots$. Now "tolerably small" can be defined to mean "rather smaller than ΔA_{k+1}."

The foregoing argument provides an excuse for accepting A_{k+1} in lieu of A whenever two consecutive estimates A_k and A_{k+1} agree to within ΔA_{k+1}, but it provides no defense against the possibility that convergence is not so fast, in which case A_k and A_{k+1} might agree by accident and yet be both quite different from A. The $\int_y^x$ key waits for three consecutive estimates A_k, A_{k+1}, and A_{k+2} to agree within ΔA_{k+2}. Only the most conservative integration schemes wait that long. While this conservatism strongly attenuates the risk of accidental premature acceptance of an estimate, the risk that three consecutive estimates might agree within the tolerance and yet be quite wrong cannot be eliminated. Later, under "How to Deceive Every Numerical Integration Procedure," some such risk will be proved unavoidable, but the risk now is so small that further attenuation is not worth its cost.

The combination of ignorance with conservatism is surprisingly costly. Had we known in advance that A_k would be accurate enough we would have calculated none of the other averages. Instead, waiting for three consecutive averages to agree could easily cost some methods almost 6.25 times as many samples as if only A_k had to be calculated, and more than that if the sample counts n_1, n_2, n_3, ... are not chosen optimally. For the $\int_y^x$ key we chose $n_k = 2^k - 1$ and we chose a method whose successive averages each share almost half of the previous average's samples, thereby preventing the cost of ignorance from much exceeding a factor of 4.

Memory limitations precluded the use of another family of methods known as adaptive quadrature. These methods attempt to distribute nodes more densely where the integrand f(u) appears to fluctuate rapidly, less densely elsewhere where f(u) appears to be nearly constant or relatively negligible. They succeed often enough that the best general-purpose integrators on large computers are adaptive programs like Carl de Boor's CADRE; this and others are described in reference 2. Alas, adaptive programs consume rather more memory for scratch space than the twenty registers available in the HP-34C.

What Method Underlies the $\int_y^x$ Key?

The HP-34C uses a Romberg method; for details consult reference 2. Several refinements were found necessary. Instead of uniformly spaced nodes, which can induce a kind of resonance or aliasing that produces misleading results when the integrand is periodic, the $\int_y^x$ key's nodes are spaced nonuniformly. Their spacing can be explained by substituting, say,

$$u = \frac{3}{2}v - \frac{1}{2}v^3$$

into

$$I = \int_{-1}^{1} f(u)du = \int_{-1}^{1} f\left(\frac{3}{2}v - \frac{1}{2}v^3\right)\cdot\frac{3}{2}(1-v^2)dv$$

and distributing nodes uniformly in the second integral. Besides suppressing resonance, the substitution confers two more benefits. One is that no sample need be drawn from either end of the interval of integration, except when the interval is so narrow that no other possibilities are available, and consequently an integral like

$$\int_0^3 \frac{\sin u}{u}\,du$$

won't hang up on division by zero at an endpoint. Second, $I = \int_y^x f(u)du$ can be calculated efficiently when $f(u) = g(u)\sqrt{|x-u|}$ or $g(u)\sqrt{(x-u)(u-y)}$ where g(u) is everywhere a smooth function, without any of the expedients that would otherwise be required to cope with the infinite values taken by the derivative f'(u) at u = x or u = y. Such integrals are encountered often during calculations of areas enclosed by smooth closed curves. For example, the area of a circle of radius 1 is

$$\int_0^2 \sqrt{u(4-u)}du = 3.14159 \pm 8.8\times10^{-6}$$

which consumes only 60 seconds when evaluated in **SCI 5** and only 110 seconds to get $3.141592654\pm1.4\times10^{-9}$ in **SCI 9**.

Another refinement is the use of extended precision, 13 significant decimal digits, to accumulate the sums that define A_k, thereby allowing thousands of samples to be accumulated, if necessary, without losing to roundoff any more information than is lost within the user's own f-program. The last example's 10 significant decimal digits of π could not have been achieved without such a refinement.

How Does the $\int_y^x$ Key Compare with Other Integrators?

What most distinguishes the HP-34C's $\int_y^x$ key from all other schemes is its ease of use. No step-size parameters, no plethora of error tolerances, no warning indicators that "can usually be ignored." Only the minimum information needed to specify $\int_y^x (f(u)\pm\Delta f(u))du$ has to be supplied. And because the $\int_y^x$ key is effective over so wide a range of integrals it ranks among the most reliable procedures available anywhere. Usually it is far faster than simpler procedures like the trapezoidal rule or Simpson's rule commonly used previously on calculators. For integrands defined by programs that fit comfortably into a mid-sized handheld calculator that can hold at most 210 program steps, the $\int_y^x$ key is comparable in speed (count the number of samples) with the integrators available on large computers. For much more complicated integrands the best adaptive integrators on large computers are appreciably faster.

One of the HP-34C's most important components is its Owner's Handbook. It is for most owners the first guide to the foothills of an awesome range of new possibilities. Two chapters are devoted to $\int_y^x$. The first is introductory, and allows the user to evaluate simple integrals effortlessly and confidently. The second chapter is a longer explanation of the power and the pitfalls, concerned mainly with numerical integration generally rather than with the HP-34C in particular. This chapter had to be included because its explanations and practical advice are not yet to be found in any text likely to be consulted by an owner, nor are they supplied by the instructions that accompany other integrators on other computers or calculators. This second chapter is part of the educational burden that must be borne by innovators and pioneers. The Owner's Handbook provides no formulas for the nodes and weights used by the HP-34C because they are not needed to understand how the $\int_y^x$ key works; instead they can be deduced from information in this article.

Every numerical integrator like $\int_y^x$, which executes a user-supplied program to get the integrand's value f(u), imposes constraints upon that program. Some constraints, like requiring f to have a smooth graph on the interval of integration, are practically unavoidable. Others are nuisances like

- Begin the f-program with a special label, say A'.
- Do not use certain memory registers, say #0 - #5.
- Do not use certain operations, say = and **CLR**.

The $\int_y^x$ key is encumbered with no such nuisances. The f-program may begin with any of several labels, so several different integrals can be calculated during one long computation. The f-program may use memory registers freely and may use any operation key except $\int_y^x$ itself. One of

Evaluation in RUN Mode	Integrand in PRGM Mode		Equation in PRGM Mode	
FIX 5	LBL A		LBL B	
CLEAR REG	x ≷ I	. . . save u	I	. . . get u
0	SOLVE B	. . . for v	×	. . . uv
ENTER↑	x ≷ I	. . . save v, get u	LST x	
1	I		e^x	. . . e^u
$\int_y^x$ A	÷	. . . u/v	×	
Wait for answer / to be displayed	RTN		1	
			+	. . . $1+uve^u$
			ln	
			I	
			−	
			+	
			RTN	. . . $v-u+\ln(1+uve^u)$

Fig. 7. *A program to evaluate* $I = \int_0^1 udu/v(u)$ *where* $v = v(u)$ *satisfies* $v-u+\ln(1+uve^u)=0$.

those keys is the HP-34C's powerful **SOLVE** key.[4] Consequently this calculator is currently the only one that can evaluate conveniently integrals of implicit functions.

For example, let $v = v(u)$ be the root of the equation

$$v - u + \ln(1 + uve^u) = 0.$$

Then

$$\int_0^1 u\,du/v(u) = 1.81300 \pm 0.000005$$

results from a program rather shorter than on any previous calculator; it is exhibited in Fig. 7.

Furthermore, $\int_y^x$ may be invoked, like any other function, from within a program, thereby permitting the HP-34C to **SOLVE** equations involving integrals. For example, solving

$$\int_0^\pi \cos(x \sin\theta)d\theta = 0$$

for x = 2.405... takes a short program contained in the Owner's Handbook, and exhibits the first zero of the Bessel function $J_0(x)$.

How to Deceive Every Numerical Integration Procedure

Such a procedure must be a computer program—call it P—that accepts as data two numerical values x and y and a program that calculates f(u) for any given value u, and from that data P must estimate $I = \int_y^x f(u)\,du$. The integration procedure P is not allowed to read and understand the f-program but merely to execute it finitely often, as often as P likes, with any arguments u that P chooses. What follows is a scheme to deceive P.

First ask P to estimate I for any two different values x and y and for $f(u)\equiv 0$. Record the distinct arguments $u_1, u_2, \ldots, u_n$ at which P evaluates f(u). Presumably when P finds that $f(u_1) = f(u_2) = \ldots = f(u_n) = 0$ it will decide that $I = 0$ and say so. Next give P a new task with the same limits x and y as before but with a different integrand

$$f(u) \equiv ((u-u_1)\cdot(u-u_2)\cdot\ldots\cdot(u-u_n))^2.$$

Once again P will calculate $f(u_1)$, $f(u_2)$, ..., and finding no difference between the new f and the old, P will repeat exactly what it did before. But the new integral I is quite different from the old, so P must be deceived.

The HP-34C's $\int_y^x$ key can be hoodwinked that way. Try to evaluate $\int_{-128}^{+128} f(u)du$ using first $f(u) \equiv 0$ programmed in a way that pauses (use the **PSE** key) to display its argument u. The calculator will display each sample argument it uses, namely 0, ±88, ±47 and ±117. Next program

$$f(u) \equiv (u(u-88)(u+88)(u-47)(u+47)(u-117)(u+117))^2$$

and evaluate $\int_{-128}^{+128} f(u)du$ again. The calculator will say that both integrals are 0, but the second polynomial's integral is really 1.310269×10^{28}. That polynomial's graph, shown in Fig. 8, has the sharp spikes that characterize integrands troublesome for every numerical integration procedure. To calculate the integral correctly, reevaluate it as $2\int_0^{128} f(u)du$, thereby doubling the spikes' width compared with the range of integration.

The threat of deceit impales the designer of a numerical integrator upon the horns of a dilemma. We all want our integrators to work fast, especially when the integrand f(u) is very smooth and simple like $f(u)=3u-4$. But if the integrator is too fast it must be easy to deceive; fast integration means few samples $f(u_j)$, implying wide gaps between some samples, which leave room for deceitful misbehavior. Figs. 9a-9e illustrate the dilemma with two estimates of $\int_y^x f(u)du$. The first estimate is based upon the three samples drawn at the white dots, the second upon seven samples including those three white plus four more black dots. Fig. 9a shows why all sufficiently smooth graphs f(u) that agree at all seven samples have nearly the same integrals, but Fig. 9b shows how two integrands could provide the same samples and yet very different integrals. The coincidence in Fig. 9b is unlikely; successive estimates based upon increasingly dense sampling normally would reveal the difference as in Fig. 9c. However, situations like those illustrated in Figs. 9d and 9e are very likely to deceive.

Textbooks tell us how to avoid being deceived: avoid integrands f(u) among whose first several derivatives are some that take wildly different values at different places in the range of integration. Or avoid integrands f(u) that take wildly different values when evaluated at complex arguments in some neighborhood of the range of integration. And if wild integrands cannot be avoided they must be tamed. We shall rejoin this train of thought later.

Improper and Nearly Improper Integrals

An improper integral is one that involves ∞ in at least one of the following ways:

- One or both limits of integration are ±∞, e.g.,

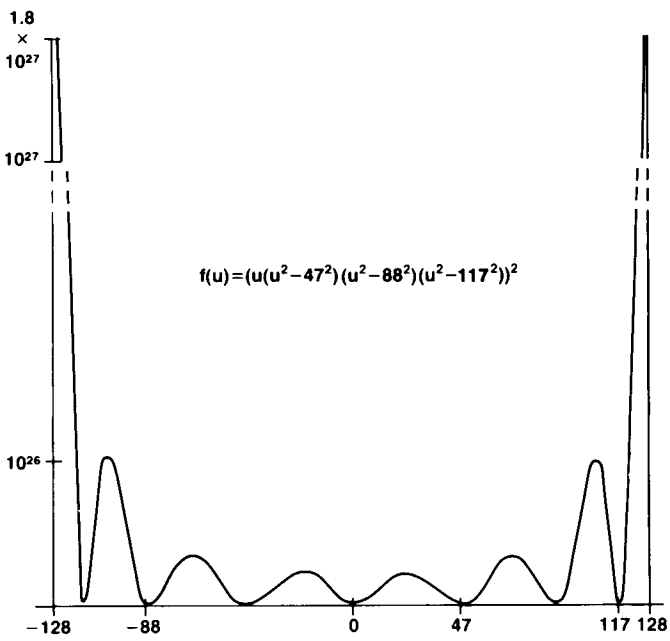

Fig. 8. *The polynomial f(u) was devised to deceive the HP-34C into miscalculating its integral as 0 instead of 1.31×10^{28}. This spiky graph is typical of integrands that can baffle any numerical integrator. 73% of the area under the graph lies under two spikes whose widths span less than 4% of the area of integration.*

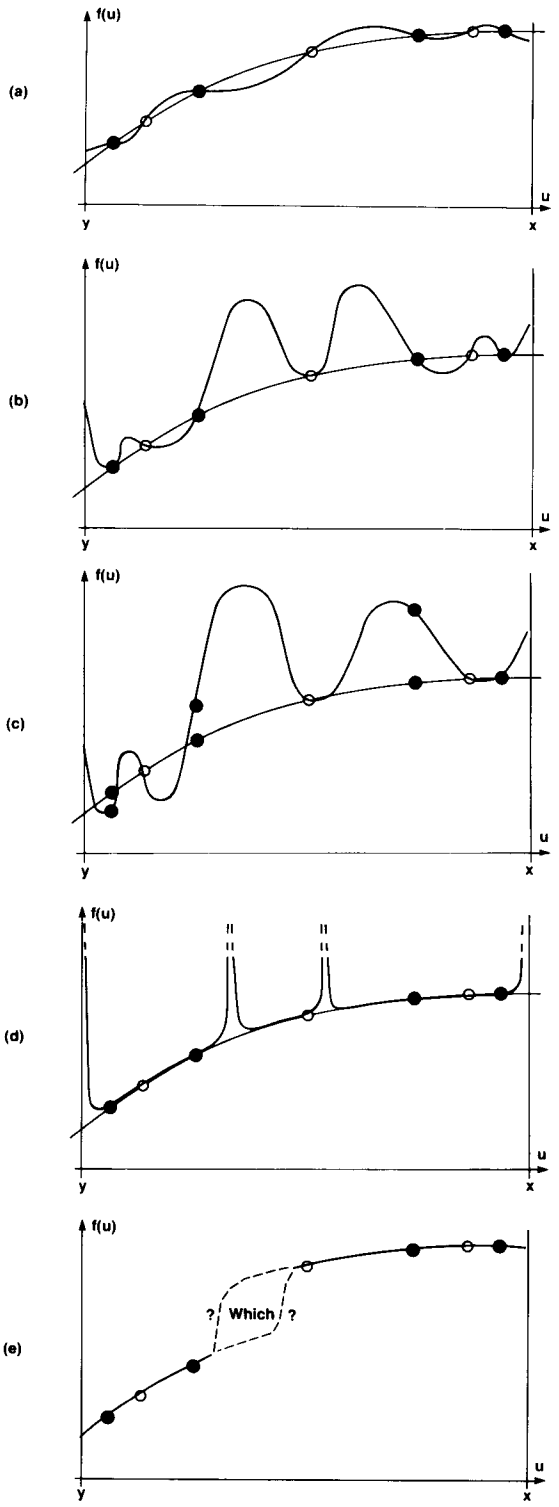

Fig. 9. *Few samples (open circles) mean fast integration but a large possibility of error. More samples (solid dots plus open circles) usually mean more accuracy, but not always, as in (b), (d), and (e). (a) Which is the graph of f(u)? No matter; both have almost the same integral. (b) Which is the graph of f(u)? They have very different integrals. (c) Here two graphs that coincide on the first samples ○ are distinguished by a significantly different outcome after second samples ● are drawn. (d) If the graph of f(u) has a few sharp and narrow spikes, they will probably be overlooked during the estimation of the integral based on finitely many samples. (e) If the graph of f(u) has a step that was not made known during the estimation of the integral, then the estimate may be mistaken.*

$$\int_{-\infty}^{\infty} \exp(-u^2)\,du = \sqrt{\pi}/2.$$

- The integrand tends to $\pm\infty$ someplace in the range of integration, e.g.,

$$\int_0^1 \ln(u)du = 1.$$

- The integrand oscillates infinitely rapidly somewhere in the range of integration, e.g., $\int_0^1 \cos(\ln u)du = 1/2$.

Improper integrals are obviously troublesome. Equally troublesome, and therefore entitled to be called nearly improper, are integrals afflicted with the following malady:

- The integrand or its first derivative changes wildly within a relatively narrow subinterval of the range of integration, or oscillates frequently across that range.

This affliction can be diagnosed in many different ways. Sometimes a small change in an endpoint can render the integral improper, as in

$$\int_{0.0001}^1 \ln(u)du = -0.99898\ldots \rightarrow \int_0^1 \ln(u)du = 1.$$

Sometimes a small alteration of the integrand can render the integral improper, as in

$$\int_{-1}^1 dx/(x^2 + 10^{-10}) = 314157.2654\ldots \rightarrow \int_{-1}^1 dx/x^2 = \infty.$$

Sometimes the value of the integral is nearly independent of relatively huge variations in one or both of the endpoints, as is $\int_0^x \exp(-u^2)\,du \approx \sqrt{\pi}/2$ for all $x > 10$. Regardless of the cause or diagnosis, nearly improper integrals are the bane of numerical integration programs, as we have seen.

During the HP-34C's design a suspicion arose that most integrals encountered in practice might be improper or nearly so. Precautions were taken. Now that experience has confirmed the suspicion, we are grateful for those precautions. They were:

1. Avoid sampling the integrand at the ends of the range of integration.
2. By precept and example in the Owner's Handbook, warn users against wild integrands, suggest how to recognize them, and illustrate how to tame them.

The second precaution ignited controversy. Against it on one side stood fears that its warnings were excessive and might induce paranoia among potential customers. Who would buy a calculator that he thinks gets wrong answers? Actually wrong answers were very rare, thanks in part to the first precaution, and many attempts to vindicate dire predictions about mischievous improper and nearly improper integrals were thwarted by unexpectedly correct answers like

$$\int_0^1 \ln(u)\,du = 0.9998 \pm 0.00021$$

in 2 minutes at **SCI 3**. Or

$$\int_0^{30} \exp(-u^2)\,du = 0.886227 \pm 0.0000008$$

in 4 minutes at **SCI 5**. If the wages of sin be death, O Death, where is thy sting?

On the other side stood a number of embarrassing examples like

$$\int_0^{400} \exp(-u^2)\, du$$

miscalculated as 0.0 ± 0.0000000005 in 14 seconds. Another, had we known it then, would have been Woodyard's example at the beginning of this article; the correct answer

$$\int_0^1 \left(\frac{\sqrt{u}}{u-1} - \frac{1}{\ln u} \right) du = 0.03649 \pm 0.00000007$$

in 23 minutes at **FIX 7** differs from **FIX 5**'s wrong answer 0.03662 in the worst way; the error is too small to be obvious and too large to ignore. Adding to the confusion were examples like

$$A(x) = x^{-1} \int_0^x \sqrt{-2 \ln \cos(u^2)}\, du/u^2 = 1 + x^4/60 + x^8/480 + \ldots$$

for which computation in **SCI 4** produced ridiculous values like $A(0.1) = 0.95742 \pm 0.00005$, $A(0.01) = 0.58401 \pm 0.00003$, and $A(0.001) = 0$, all impossibly smaller than 1. This example appears to condemn the $\int_y^x$ key until the integrand $f(u)=\sqrt{-2 \ln \cos(u^2)}/u^2$ is watched for small arguments u and seen to lose most of its figures to roundoff, losing all of them for $|u| \leq 0.003$, despite an absence of subtractions that could be blamed for cancellation. Then the example appears to condemn the whole calculator. Who wants responsibility for a calculator that gets wrong answers?

Don't panic! The answers are wrong but the calculator is right.

How to Tame a Wild Integral

Forewarned is forearmed. Every experienced calculator user expects to encounter pathological examples like some of those above, and expects to cope with them. The question is not "whether" but "when"? And that is when attention to detail by the calculator's designers is rewarded by the user's freedom from petty distractions that can only complicate a task already complicated enough. But like the dog that did not bark,* the absence of distracting details may fail to be appreciated. That is why the examples explained below have been chosen—to illustrate the advantages of liberated thought. Work them on your calculator as you read them; don't skim them like a novel. Then you may come to think of your calculator the way I think of mine, as a trusted friend who stays with me when I need help.

The integral $A(x)$ above contains an integrand $f(u) = \sqrt{-2 \ln \cos(u^2)}/u^2$ that loses its figures when u becomes tiny. The problem is caused by rounding $\cos(u^2)$ to 1, which loses sight of how small u^2 must have been. The solution compensates for roundoff by calculating f(u) as follows:

```
Let y = cos u^2 rounded.
If y = 1 then let f(u) = 1
        else let f(u) = √(-2 ln y)/cos^-1 y.
```

The test for y = 1 adds four steps to the f-program and, provided ln and $\cos^{-1}$ are implemented as accurately as on all recent HP calculators, the problem goes away.

*See the last few paragraphs of the Sherlock Holmes story "Silver Blaze" by Conan Doyle.

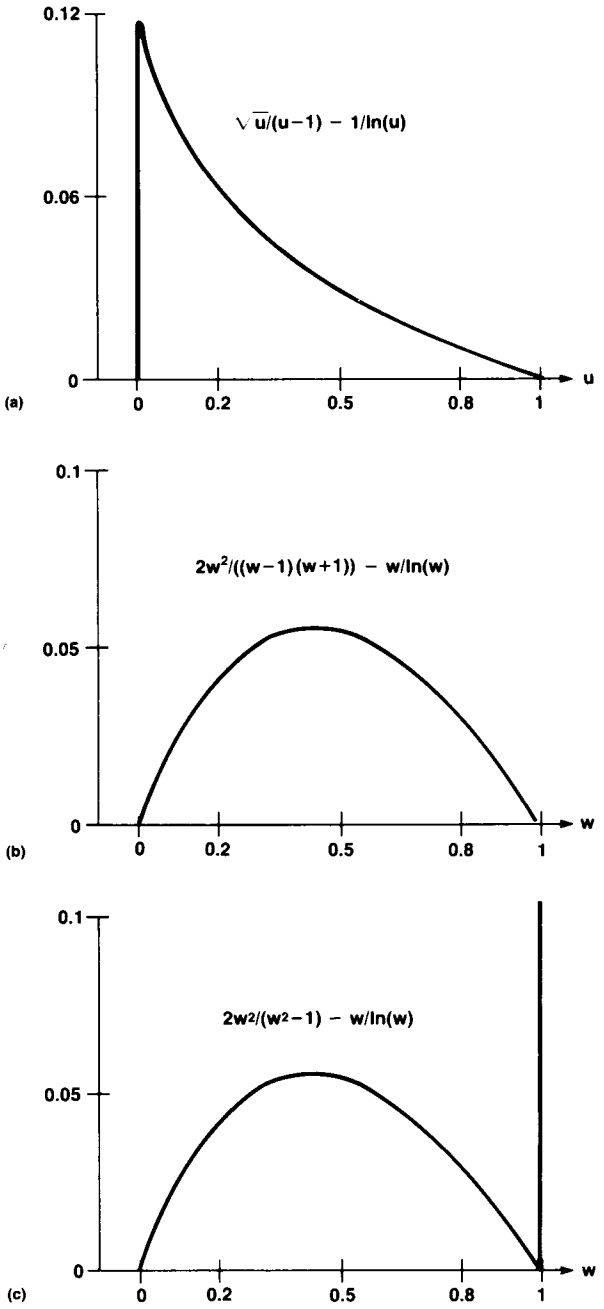

Fig. 10. *Substituting w^2 for u turns the wild graph (a) into the easy one (b). But do not replace ((w−1) (w+1)) by (w^2−1) because roundoff errors introduce a spike, as shown in (c).*

Woodyard's example I_1 has an integrand f(u) whose derative $f'(u) \to \infty$ as $u \to 0$. The graph of f(u) shown in Fig. 10a looks like a lovers' leap. Stretching the u-axis near u=0 by substituting $u = w^2$ turns the precipice into the hummock shown in Fig. 10b and transforms the integral into an easy calculation:

$$I_1 = \int_0^1 \left(\frac{2w^2}{(w-1)(w+1)} - \frac{w}{\ln w} \right) dw$$

The HP-34C computes this as 0.03649 ± 0.000005 in 100 seconds at **FIX 5** or 0.0364900 ± 0.00000008 in 200 seconds

at **FIX 7**. Do not replace $(w-1)(w+1)$ by (w^2-1) because the latter loses to roundoff half of its significant digits as $w \rightarrow 1$ and introduces a gratuitous spike into the integrand's graph shown in Fig. 10c, which was plotted on an HP-85. Do not worry about $w = 0$ or $w = 1$ because they don't happen, but do worry that as $w \rightarrow 1$ the integrand approaches the unreliable expression $\infty - \infty = 0$. This means that **FIX 7** displays about as many digits as could possibly be correct for all $w < 0.999$, beyond which the $\int_y^x$ key draws few if any samples because it converges so fast.

The graphs of $\exp(-u^2)$ over $0 \leq u \leq 300$ and of $1/(u^2 + 10^{-10})$ over $-1 \leq u \leq 1$ both resemble huddled mice with very long tails stretched out hundreds or thousands of times as long as their bodies. Plotting the graphs on a page of normal width is futile because the bodies get squashed into vertical whiskers.

Most people who integrate such functions numerically cut off the tails. Thin tails can be cut almost indiscriminately without much degrading the accuracy or the speed of integration. Such is the case for $\int_0^x \exp(-u^2)du$, which $\int_y^x$ evaluates in less than, say, 4 minutes at **SCI 5** provided that x, if bigger than 4 or 10, is cut back to something between 4 and 10. But $\int_{-x}^{x} du/(u^2 + 10^{-10})$ has too thick a tail to cut without losing accuracy or patience when x is large. That is why Procrustean methods are not recommended. Better to shrink the tail via an artful substitution like $u = \lambda + \mu \tan v$ where λ lies within the body of the mouse and μ is roughly that body's width. Doing so with $\lambda = 0$ and $\mu = 1$ changes $\int_0^x \exp(-u^2)\, du$ into

$$\int_0^{\arctan x} \exp(-\tan^2 v)\,(1 + \tan^2 v)\, dv$$

which $\int_y^x$ evaluates in three minutes at **SCI 5** even when x is as big as 10^{10}. Don't worry about $\tan \pi/2$ because it can't happen on a well-designed calculator.

$$\int_{-x}^{x} du/(u^2 + 10^{-10})$$

benefits miraculously from the foregoing substitution when $\lambda = 0$ and $\mu = 10^{-5}$, but values near those do almost as well.

Another technique might be called "subdivide and conquer." It subdivides the range of integration into subintervals upon each of which the integrand f(u) is tame, although f(u) may look wild on the range as a whole. For example, $f(u) \equiv \sqrt{u^2 + 10^{-10}}$ has a V-shaped graph practically the same as that of $|u|$. Evaluating $\int_{-3}^{5} f(u)\, du$ accurately takes a long time if done with one press of $\int_y^x$, but subdividing the integral into

$$\int_{-3}^{0} f(u)\, du + \int_0^5 f(u) du$$

takes two presses of $\int_y^x$ and one of $\Sigma+$ but much less time.

Subdivide and conquer works best when combined with apt substitutions. For example, if the formulas in Fig. 5 were unavailable how would $F(\infty) = \int_0^\infty du/(1 + u^{64})$ be calculated?

$$F(\infty) = \int_0^1 du/(1 + u^{64}) + \int_1^\infty du/(1 + u^{64}) \quad \text{... subdivided}$$

$$= \int_0^1 du/(1 + u^{64}) + \int_0^1 w^{62} dw/(w^{64} + 1) \quad \text{... } u = 1/w$$

$$= \int_0^1 (1 + u^{62})\, du/(1 + u^{64}) \quad \text{... merged via } w = u$$

$$= 1 + \int_0^1 (u^{62} - u^{64})\, du/(1 + u^{64}) \quad \text{... some algebra}$$

$$= 1 + \tfrac{1}{8} \int_0^1 (1 - v^{1/4})\, v^{55/8}\, dv/(1 + v^8) \quad \text{... } u = v^{1/8} \text{ to shrink a tail}$$

$$= 1.000401708155 \pm 1.2 \times 10^{-12}$$

in 10 minutes at **SCI 8**. Thus we have calculated $F(\infty) = (\pi/64)\csc(\pi/64)$ to 13 significant decimals on a ten-significant-decimal calculator.

Oscillatory integrals like $\int_0^1 \cos(\ln u)\, du$ sometimes succumb to stretching substitutions like $u = v^2$ that damp the oscillations, but generally oscillatory integrals cannot be calculated accurately and quickly without sophisticated tricks beyond the scope of an article like this. A simple trick worth trying when the period of oscillation is known in advance is called *folding*, though it is really another instance of subdivide and conquer. Here is a didactic example.

$$I_3 = \int_0^{600\pi} \frac{\sin^2 u}{\sqrt{u} + \sqrt{u+\pi}}\, du = \text{still running after over three hours at SCI 5}$$

$$= \sum_{n=0}^{599} \int_{n\pi}^{n\pi+\pi} \frac{\sin^2 u}{\sqrt{u} + \sqrt{u+\pi}}\, du$$

$$= \sum_{n=0}^{599} \int_0^{\pi} \frac{\sin^2 v}{\sqrt{v+n\pi} + \sqrt{v+n\pi+\pi}}\, dv$$

after being subdivided and with $u = v + n\pi$. Exchanging $\int$ and $\sum$ produces

$$I_3 = \int_0^{\pi} \sin^2 v \cdot \sum_{n=0}^{599} \frac{1}{\sqrt{v+n\pi} + \sqrt{v+n\pi+\pi}}\, dv.$$

At this point a program should be written to calculate the sum, but because the example is didactic the sum collapses to yield

$$I_3 = \int_0^{\pi} \frac{600 \sin^2 v}{\sqrt{v} + \sqrt{v+600\pi}}\, dv = 21.10204 \pm 0.00007$$

in 5 minutes at **SCI 5**.

Now for a final example drawn from life:

$$V = \int_0^\infty \frac{du}{(a^2+u)\sqrt{(a^2+u)(b^2+u)(c^2+u)}} \quad \text{for } a = 100, b = 2, c = 1.$$

This integral pertains to the electrostatic field about an ellipsoidal body with principal semiaxes a, b, c.[5] The ellipsoid is needle-shaped like an antenna or a probe. The classical approach transforms V into a standard form called an elliptic integral of the second kind and interpolates on two variables in published tables to get a numerical value. The following approach takes less time.

First transform the improper integral ($\int_0^\infty$) into a proper

one by substituting, say, $u = (a^2-c^2)/(1-v^2) - a^2$ to get

$$V = \lambda \int_{\mu}^{1} \sqrt{(1-v^2)/(v^2+\alpha)}\, dv$$

where

$$\lambda = 2/((a^2-c^2)\sqrt{a^2-b^2}) = 2.00060018 \times 10^{-6}$$

$$\mu = c/a = 0.01$$

$$\alpha = (b^2-c^2)/(a^2-b^2) = 3.001200480 \times 10^{-3}$$

Now, as always happens when $a >> b > c$, the integral is nearly improper because α and μ are both so nearly 0. We suppress this near impropriety by finding an integral in closed form that sufficiently resembles the troublesome part of V. One candidate is

$$W=\lambda \int_{\mu}^{1} dv/\sqrt{v^2+\alpha} = \lambda \ln (v+\sqrt{v^2+\alpha}) \Big|_{v=\mu}^{1}$$

$$= \lambda \ln ((1+\sqrt{1+\alpha})/(\mu+\sqrt{\mu^2+\alpha}))$$

$$= 8.40181880708\times10^{-6}$$

Then

$$V = W + \lambda \int_{\mu}^{1} (\sqrt{(1-v^2)/(v^2+\alpha)} - 1/\sqrt{v^2+\alpha})\, dv$$

$$= \lambda \int_{\mu}^{1} \left(\frac{W/\lambda}{1-\mu} - \frac{v^2}{(1+\sqrt{1-v^2})\sqrt{v^2+\alpha}}\right) dv$$

$$= 7.78867525\times10^{-6} \pm 1.3\times10^{-14}$$

after seven minutes at **FIX 8**. Don't worry about $\sqrt{1-v^2}$ as $v \to 1$ because the figures lost to roundoff are not needed and its infinite derivative doesn't bother the HP-34C.

Conclusion

A powerful mathematical idea has been placed at the disposal of people who will invoke it with fair confidence by pressing a button marked $\int_{y}^{x}$ without having to understand any more about its internal workings than most motorists understand about automatic transmissions. Integrals that might previously have challenged the numerical expert and a big computer now merely amuse the scientist or engineer, and tomorrow they will be routine. And now those engineering students who do attend classes in numerical analysis need no longer be expected to memorize the names nor the remainder terms of quadrature formulas but may instead be taught to use integration wisely.

Acknowledgments

Stan Mintz was the first to request an $\int_{y}^{x}$ key for the HP-34C. Dennis Harms helped to select the algorithm and microprogrammed it into the calculator. Robert Barkan wrote the Handbook's two chapters on $\int_{y}^{x}$. Then Dennis helped to shorten this article. Working with these people has been a pleasure.

References

1. F.A. Willers, "Practical Analysis," translated by R.T. Beyers, Dover, New York, 1948, Chapter 3.
2. P.J. Davis and P. Rabinowitz, "Methods of Numerical Integration," Academic Press, New York, 1975.
3. A.H. Stroud and D. Secrest, "Gaussian Quadrature Formulas," Prentice-Hall, New Jersey, 1966.
4. W.M. Kahan, "Personal Calculator Has Key to Solve Any Equation f(x)=0," Hewlett-Packard Journal, December 1979.
5. J.A. Stratton, "Electromagnetic Theory," McGraw-Hill, New York, 1941, pages 201-217.

William M. Kahan

William Kahan is professor of mathematics and computer science at the University of California at Berkeley. An HP consultant since 1974, he has helped develop increasingly accurate arithmetic and elementary functions for the HP-27, 67/97, 32E, and 34C Calculators and the HP-85 Computer, financial functions for the HP-92 and 38E/C, and other functions for the 32E and 34C, including $\int$ and SOLVE for the 34C. A native of Toronto, Canada, he received his BA and PhD degrees in mathematics and computer science from the University of Toronto in 1954 and 1958, then taught those subjects at Toronto for ten years before moving to Berkeley. A member of the American Mathematical Society, the Association for Computing Machinery, and the Society for Industrial and Applied Mathematics, he has authored several papers and served as a consultant to several companies. He is also co-author of a proposal to standardize binary floating-point arithmetic that has been adopted by several microprocessor manufacturers and is soon to be promulgated by the IEEE. He is married and has two teenage sons.

HEWLETT-PACKARD JOURNAL

AUGUST 1980 Volume 31 • Number 8

Technical information from the Laboratories of Hewlett-Packard Company

Hewlett-Packard Company, 1501 Page Mill Road
Palo Alto, California 94304 U.S.A.
Hewlett-Packard Central Mailing Department
Van Heuven Goedhartlaan 121
1180 AM Amstelveen The Netherlands
Yokogawa-Hewlett-Packard Ltd., Suginami-Ku
Tokyo 168 Japan