Scientific Pocket Calculator Extends Range of Built-In Functions

Matrix operations, complex number functions, integration, and equation solving are only some of the numerous preprogrammed capabilities of HP's latest scientific calculator, the HP-15C.

by Eric A. Evett, Paul J. McClellan, and Joseph P. Tanzini

THE NEW HP-15C Scientific Programmable Calculator (Fig. 1) has the largest number of preprogrammed mathematical functions of any handheld calculator designed by Hewlett-Packard. For the first time in an HP calculator, all arithmetic, logarithmic, exponential, trigonometric, and hyperbolic functions operate on complex numbers as well as real numbers. Also, built-in matrix operations are provided, including addition, subtraction, multiplication, system solution, inversion, transposition, and norms.

The HP-15C also performs the **SOLVE** and $\int_y^x$ functions, which are very useful tools in many applications. The **SOLVE** operator numerically locates the zeros of a function programmed into the calculator by the user.[1] The $\int_y^x$ operator numerically approximates the definite integral of a user-programmed function.[2]

Design Objectives

The HP-15C was designed with the following goals in mind:

- Provide all functions of the HP-11C and HP-34C Calculators in the same slim-line package used for the HP-11C
- Provide additional convenient, built-in advanced mathematical functions which are widely used in many technical disciplines.

Achieving the first objective posed a keyboard layout problem. The nomenclature for the HP-11C functions filled every position on the keyboard. Since the display is limited to seven-segment characters, functions could not be removed from the keyboard and accessed by typing the function name as is done on the HP-41 Handheld Computers. Therefore, to free some space on the keyboard, only the two most common conditional tests are placed on the keyboard, **x=0** and **x≤y**. A **TEST** prefix is added to access the other ten

Fig. 1. *The HP-15C is an advanced programmable calculator with special functions that enable the user to solve problems involving matrices, integrals, complex arithmetic, and roots of equations. Its slim-line design fits easily in a shirt pocket.*

tests by executing **TEST 0**, **TEST 1**, . . . , **TEST 9**. A table on the back of the calculator indicates the correspondence between the digits and tests. This frees enough positions on the keyboard to add the **SOLVE** and $\int_y^x$ functions, plus a few more.

In striving to attain the second objective, we noted that nearly every text covering advanced mathematics for science and engineering includes chapters on complex analytic functions and matrix algebra. They are fundamental tools used in many disciplines. Furthermore, the complex functions and many of the matrix operations can be viewed as extensions of the functions already on the keyboard. This is an important consideration because of the limited number of key positions available. Thus, our goal was to extend the domain of some of the built-in functions to complex numbers and matrices in a natural way without altering how those functions operate on real numbers.

Complex Mode

A complex mode was introduced in which another register stack for imaginary numbers is allocated parallel to the traditional register stack for real numbers (Fig. 2). Together they form what is referred to as the complex RPN* stack.

The real X register is always displayed. A complex number $a+ib$ is placed in the X register by executing **a**, **ENTER**, **b**, **I**. The user may display the contents of the imaginary X register by executing **Re≶Im** to exchange the contents of the real and imaginary X registers. Or the user may hold down the **(i)** key to view the imaginary part without performing an exchange.

ENTER, **R↓**, **R↑**, **x≶y**, and **LST x** all operate on the complex stack, but **CLx** and **CHS** operate only on the real X register so that one part of a complex number can be altered or complemented without affecting the other. For example, the complex conjugate is performed by executing **Re≶Im**, **CHS**, **Re≶Im**.

The following functions include complex numbers in their domain: **+**, **−**, **×**, **÷**, **1/x**, $\sqrt{x}$, **x²**, **ABS** (magnitude), **LN**, **e^x**, **LOG**, **10^x**, **y^x**, **SIN**, **COS**, **TAN**, **SIN^{-1}**, **COS^{-1}**, **TAN^{-1}**, **SINH**, **COSH**, **TANH**, **$SINH^{-1}$**, **$COSH^{-1}$**, and **$TANH^{-1}$**. These functions assume the complex inputs are in the rectangular form, $a+ib$.

Often complex numbers are expressed in polar form: $re^{i\theta} = r(\cos\theta + i\sin\theta)$. In complex mode, the polar-to-rectangular conversion functions **→P** and **→R** provide a convenient means for converting between the polar and rectangular forms of a complex number.

*Reverse Polish notation, a logic system that eliminates the need for parentheses and "equals" keystrokes in calculator operations.

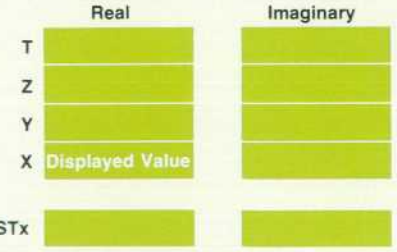

Fig. 2. *To handle complex numbers, the HP-15C uses another register stack in parallel with the traditional RPN stack. Only the contents of the X register in the real stack are displayed.*

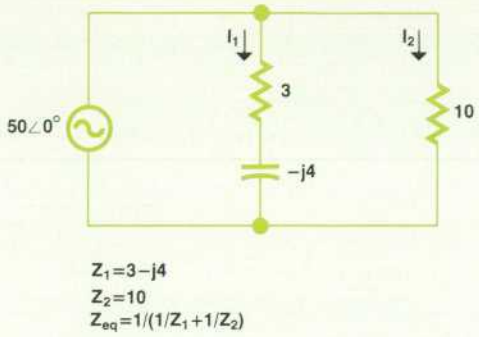

Fig. 3. *The complex arithmetic capabilities of the HP-15C make it easy to solve for the equivalent impedance of this parallel circuit (see text).*

Complex numbers are used extensively in electrical engineering. For example, to find the equivalent impedance in the parallel circuit shown in Fig. 3, perform the following steps on the HP-15C:

Keystrokes	Calculation
3 ENTER 4 CHS I	Z_1
1/x	$1/Z_1$
1 0	Z_2
1/x	$1/Z_2$
+	$1/Z_1 + 1/Z_2$
1/x	$Z_{eq} = 1/(1/Z_1 + 1/Z_2)$. Hold down **(i)** key to view imaginary part. $Z_{eq} = 2.9730 - 2.1622i$
→P	Convert to phasor form. $Z_{eq} = 3.6761 \angle -36.0274°$

This example is a very elementary application of the built-in complex function capability. Since complex operations can be used in conjunction with the **SOLVE** and $\int_y^x$ functions, the HP-15C can be programmed to carry out some sophisticated calculations such as computing complex line integrals and solving complex potentials to determine equipotential lines and streamlines.[3]

Matrix Descriptors

No set of matrix operations is complete without addition, subtraction, multiplication, system solution, and inversion. To provide these operations on the HP-15C, it seemed natural to extend the domains of the **+**, **−**, **×**, **÷**, and **1/x** functions to include matrix arguments. Since these functions operate on the stack contents, a means of placing a matrix name (descriptor) on the stack is essential. The set of alpha characters that can be represented in a seven-segment font is limited, but the letters A, B, C, D, and E have reason-

Fig. 4. *The seven-segment font used in the HP-15C's liquid-crystal display allows representations of the alphabetic characters A, B, C, D, and E as shown above for use in labeling matrices.*

able representations (Fig. 4).

Thus the decision was made to allow up to five matrices to reside in memory simultaneously, named **A**, **B**, **C**, **D**, and **E**. Their descriptors are recalled into the X register by the sequence **RCL MATRIX** followed by the appropriate letter. When the X register contains a matrix descriptor, the matrix name and dimensions are displayed. Matrix descriptors may be manipulated by stack operations and stored in registers just like real numbers, and certain functions accept matrix descriptors as valid inputs. For example, suppose **C** and **D** are 2-by-3 and 3-by-4 matrices, respectively, which are already stored in memory. To compute the matrix product **CD** and place the result in matrix **A**, the user parallels the steps required for real number multiplication, except that the result destination must be specified:

Keystrokes	HP-15C Display
RCL MATRIX C	C 2 3
RCL MATRIX D	d 3 4
RESULT A	d 3 4

At this point, the HP-15C's RPN stack contains the information shown in Fig. 5a. The matrix operands are in the stack, and the result matrix is specified. The user now executes × to compute the matrix product. When × is executed, the presence of the matrix descriptions in the Y and X registers is detected, the matrices are checked for compatible dimensions, the result matrix **A** is automatically dimensioned to a 2-by-4 matrix, the product is computed, and the matrix descriptor of the result is placed in the X register and displayed (Fig. 5b).

The operators + and − work similarly, and ÷ performs the matrix operation $\mathbf{X}^{-1}\mathbf{Y}$ if the X and Y registers contain matrix descriptors. This is useful for linear system solution, since the solution to the matrix equation $\mathbf{XR}=\mathbf{Y}$ is $\mathbf{R}=\mathbf{X}^{-1}\mathbf{Y}$. The **1/x** function key performs matrix inversion.

Other important matrix operations that are not natural extensions of functions on the keyboard are accessed by the prefix **MATRIX** followed by a digit. These include transpose, determinant, and matrix norms. A table on the back of the calculator indicates the correspondence between the digits and matrix operations.

Internal Format of Descriptors

Normal floating-point numbers are internally represented in the HP-15C by using a 14-digit (56-bit) binary-coded-decimal (BCD) format (Fig. 6).

The exponent e is given by XX if XS=0, and by −(100−XX) if XS=9. The value of the number is interpreted as $(-1)^{S}(\text{M.MMMMMMMMM})\times 10^{e}$. For example,

$$01234000000002 \text{ represents } 1.234\times 10^{2}$$

and

$$91234000000994 \text{ represents } -1.234\times 10^{-6}.$$

Matrix descriptors, on the other hand, are distinguished by a 1 in the mantissa sign digit and a hexadecimal digit corresponding to the matrix name in the most significant digit of the mantissa field. For example, the matrix descriptor C is represented internally as 1C000000000000.

When a matrix descriptor is detected in the X register, the matrix name is displayed, and the current dimensions of that matrix are fetched from a system memory location and also displayed.

Creating Matrices and Accessing Individual Elements

A matrix is dimensioned by entering the row and column dimensions in the Y and X registers of the stack, respectively, and then executing the **DIM** prefix followed by the matrix name. Individual matrix elements are accessed by executing the **STO** or **RCL** prefixes followed by the matrix name. The element accessed is determined by the row and column indexes stored in registers R0 and R1, respectively.

Matrix data is usually entered or reviewed from left to

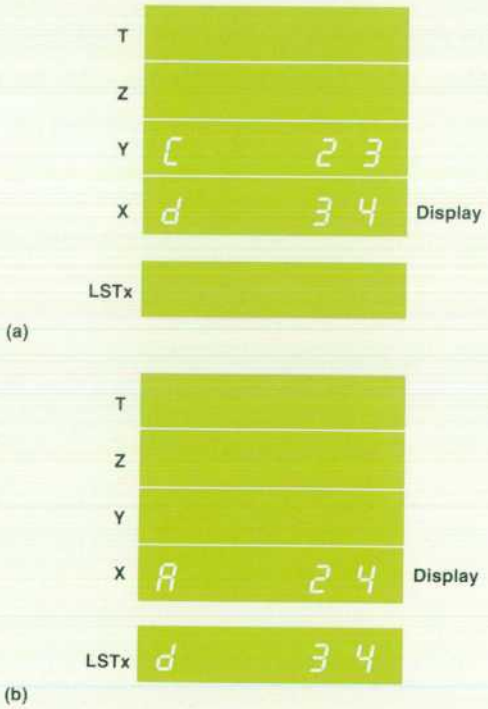

Fig. 5. *Before multiplying matrices* **C** *and* **D**, *the information in the RPN stack is located as shown in (a). After multiplication, the result, matrix* **A**, *is located as shown in (b) and the* **LSTx** *register contains the information for matrix* **D**.

right along each row and from the first row to the last. To facilitate this process, a user mode is provided in which the indexes are automatically advanced along rows after each **STO** or **RCL** matrix access operation. After the last element of the matrix has been accessed, the indexes wrap around to 1,1. As an added convenience, executing **MATRIX 1** initializes the indexes to 1,1.

The following example illustrates some of these features by solving the following matrix equation for **C**:

$$\underset{\mathbf{A}}{\begin{bmatrix} 5 & -2 \\ 4 & 6 \end{bmatrix}} \underset{\mathbf{C}}{\begin{bmatrix} c(1,1) & c(1,2) \\ c(2,1) & c(2,2) \end{bmatrix}} = \underset{\mathbf{B}}{\begin{bmatrix} 8 & 3 \\ 2 & -6 \end{bmatrix}}$$

Keystrokes	Display	Comments
USER		Select **USER** mode.
MATRIX 1		Initialize indexes in registers R0 and R1 to 1.
2 ENTER DIM A	2.0000	Dimension matrix **A** to 2 by 2.
DIM B	2.0000	Dimension matrix **B** to 2 by 2.
5 STO A	5.0000	a(1,1)
2 CHS STO A	−2.0000	a(1,2)
4 STO A	4.0000	a(2,1)
6 STO A	6.0000	a(2,2). Indexes wrap around to 1,1.
8 STO B	8.0000	b(1,1)
3 STO B	3.0000	b(1,2)
2 STO B	2.0000	b(2,1)
6 CHS STO B	−6.0000	b(2,2). Indexes wrap around to 1,1.
RCL MATRIX B	b 2 2	Recall right-hand side
RCL MATRIX A	A 2 2	Recall coefficient matrix
RESULT C	A 2 2	Specify matrix **C** as result.
÷	C 2 2	Compute $\mathbf{C}=\mathbf{A}^{-1}\mathbf{B}$.
RCL C	1.3684	c(1,1)
RCL C	0.1579	c(1,2)
RCL C	−0.5789	c(2,1)
RCL C	−1.1053	c(2,2)

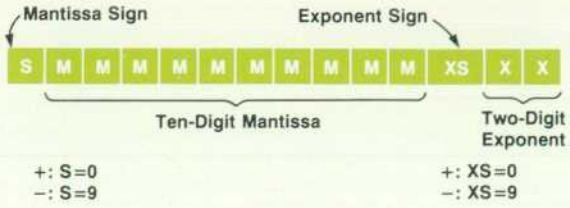

Fig. 6. *The internal representation for floating-point numbers in the HP-15C uses a 14-digit (56-bit), binary-coded-decimal format.*

Available Matrix Memory, Speed

A maximum of 64 matrix elements can be distributed among the five matrices. Since the HP-15C can invert matrices in place, up to an 8-by-8 matrix can be inverted. There is also enough memory to solve a 7-by-7 linear system of equations. Table I specifies the approximate time required to perform certain matrix operations.

Table I
Time in Seconds for Selected Matrix Operations

Order of Matrix	Determinant	Solving a System	Matrix Inversion
1	0.5	0.5	0.5
2	1.3	2.0	1.8
3	2.8	4.2	5.3
4	5.3	7.6	12
5	9.1	12	22
6	14	19	36
7	21	28	55
8	30	—	80

Designing the Complex Function Algorithms

After deciding to extend the real-valued functions and the RPN stack to the complex domain, our next step was to design the algorithms for complex arithmetic. Although their defining formulas are very simple, some disturbing examples made us question what accuracy should be achieved to parallel the high quality of the real-valued functions.

The real functions are generally computed with a small relative error (less than 6×10^{-10}) except at particular points of certain functions, where it is too costly in execution time or ROM space for the result to be computed that accurately.[3]

The relative difference R(x,y) between two numbers x and y is given by

$$R(x,y) = \frac{|x-y|}{|y|}$$

When X is an approximation of x, then we say R(X,x) is the relative error of the approximation X. Notice that the size of the relative error is related to the number of digits that are accurate. More precisely, $R(X,x) < 0.5\times10^{-n}$ implies that X

is an approximation to x that is accurate to n significant digits.

If we always wish to obtain small relative errors in each component of a complex result, then the outcome of the following example is very disappointing. For simplicity we will use four-digit arithmetic, instead of the 13 digits used internally to calculate the 10-digit results delivered to the X register of the calculator.

Example 1: Using the definition for complex multiplication,

$$(a + ib)(c + id) = (ac - bd) + (ad + bc)i,$$

consider the four-digit calculation of $Z \times W$, where $Z = 37.1 + 37.3i$ and $W = 37.5 + 37.3i$. We get,

$$\begin{aligned} Z \times W &= (1391 - 1391) + (1384 + 1399)i \\ &= 0 + 2783i \end{aligned}$$

Since the exact answer is $-0.04 + 2782.58i$, it is clear that accurate components are not always achieved by a simple application of this formula. The difference $a \times c - b \times d$ has been rounded off to result in a loss of all significant digits of the real part. The loss can be eliminated, but the calculation time would increase roughly by a factor of 4. Is it really worth this higher cost in execution time? For comparison we will consider an alternative definition of accurate complex results.

Complex Relative Error

As with real approximations we often want our errors small relative to the magnitude of the true answer. That is to say, we want $|$ (approximate value) − (true value)$|/|$(true value)$|$ to be small enough for our purposes. So relative error may be extended to the complex plane by $R(Z,z) = |Z-z|/|z|$. This extension may be applied to vectors in any normed space. A simple geometric interpretation is illustrated in Fig. 7. Approximations Z of z will satisfy $R(Z,z) < \delta$ if and only if the points Z lie inside the circle of radius $\delta|z|$ centered at z. This condition for complex relative accuracy is weaker than that for component accuracy. If the errors in each component are small, then the complex error is small. To show this, assume that $R(X,x) < \delta$ and $R(Y,y) < \delta$ where $z = x + iy$. Then,

$$\begin{aligned} R(Z,z) &= |(X - x) + i(Y - y)|/|z| \\ &\leq |X - x|/|z| + |Y - y|/|z| \\ &\leq R(X,x) + R(Y,y) \\ &< 2\delta \end{aligned}$$

Actually, R(Z,z) is less than δ, but this is slightly more difficult to show. On the other hand, however, a small complex error does not imply small component errors. Referring back to Example 1, we see that $R(ZW, zw) = 0.0002$, which is respectably small for four-digit precision, even though the real component has no correct digits.

It is not unusual for only one component to be inaccurate when the result is computed accurately in the sense of complex relative error. In fact, because the error is relative to the size $|z|$, and because this is never greatly different from the size of the larger component, only the smaller component can be inaccurate.

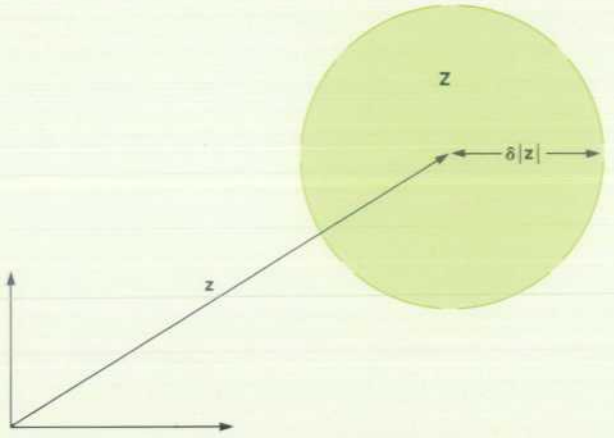

Fig. 7. *A simple geometric representation of complex relative error $R(Z,z) < \delta$.*

To show this we shall assume, without loss of generality, that $|x|$ is the larger component. Then

$$|z|/|x| = \sqrt{1 + |y/x|^2},$$

which implies that $1 \leq |z/x| \leq \sqrt{2}$, since $|y| \leq |x|$ by assumption. Thus $|x|$ and $|z|$ do not differ greatly. The important part is that $|x| \geq |z|/\sqrt{2}$. This gives

$$|X-x|/|x| \leq |Z-z|/|x| \leq \sqrt{2}|Z-z|/|z| < \sqrt{2}R(Z,z)$$

So the relative error of the larger component (assumed to be x here) is very nearly as small as the complex relative error bound R(Z,z). It also follows that the smaller component is accurate relative to the larger component's size (i.e., $|Y-y|/|x| \leq |Z-z|/|x| \leq \sqrt{2}R(Z,z)$.

This provides a quick way to determine which digits of a calculated value can possibly be incorrect when it is known that the calculated value has a certain complex error. By representing the smaller component with the exponent of the larger component, the complex error indicates the number of correct digits in each component.

For instance, in Example 1 we obtained the approximation $Z = 0 + 2783i$ of the true answer $-0.04 + 2782.58i$. Since the larger component is 2.783×10^3 we will represent the first component with the same exponent (0.000×10^3) to obtain $Z = 0000.0 + 2783i$. These components must be accurate to nearly four digits since $R(Z,z) = 0.0002$.

Perhaps the zero component of Z confuses the issue here, so another example may be appropriate. First, let

$$Z = 1.234567890 \times 10^{-10} + 2.222222222 \times 10^{-3}i$$

Then think of Z as

$$Z = 0.0000001234567890 \times 10^{-3} + 2.222222222 \times 10^{-3}i$$

If the complex relative error indicates 10-digit accuracy, i.e., $R(Z,z) < 0.5 \times 10^{-10}$, then this implies that the first 10 digits are correct, that is,

$$Z = 0.000000123 \times 10^{-3} + 2.222222222 \times 10^{-3}i$$

Error Propagation

We have seen that computing the product of two complex numbers in the straightforward manner does not necessarily result in a small error in each component (Example 1). However it can be shown that the product does have a complex relative error bound of roughly 10^{-n} whenever n digits of precision are used in the calculation. Moreover, small relative errors in the input values give rise to relative errors nearly as small in the output values. This is not true for small component errors. One acceptable rounding error in an input value may produce an inaccurate component, even when the multiplication is exact. This is illustrated by the following example.

Example 2: Let $z = (1 + 1/300) + i$ and $w = 1 + i$, then using four-digit precision we have

$$Z = 1.003 + 1.000i$$
$$\text{and } W = 1.000 + 1.000i$$

Therefore,

$$ZW = 0.003 + 2.003i$$
$$= 3.000 \times 10^{-3} + 2.003i$$

exactly, yet

$$zw = 3.333 \times 10^{-3} + 2.003i$$

to four digits. The single rounding error of $1+1/300 \rightarrow 1.003$ in the component of the input Z was magnified from a relative error of 0.0003 to 0.1.

So, in general, computing accurate components will not improve the result of a chain calculation because intermediate input values are often inexact (this is the idea of backward error analysis and is explained more fully in reference 3). It is important to realize that this is not, in itself, a good reason to forsake accurate results based on the assumption that the input values are not exact. For example, if we assume that X has an error in its eleventh digit and thus decide that sin(X) for $X > 10^5$ degrees, say, need not be computed accurately, then we would have failed to provide a useful result for those special cases where we know that the input value is exact.

As a simple illustration consider accurately calculating the value sin(1,234,567,899.1234567890) where the argument is in degrees. Using

$$\sin(1{,}234{,}567{,}899) = 0.9876883406$$

is grossly inaccurate. Instead, let $x = 1.234567899 \times 10^9$ and $y = 0.123456789$, then evaluate

$$\sin(x+y) = \sin(x)\cos(y) + \cos(x)\sin(y).$$

Here we know x is exact, and since sin(x) and cos(x) are computed accurately by the HP-15C, the final result $\sin(x+y) = 0.9873489744$ is very accurate.

The point here is that clean results (in particular accurate components) are desirable, but in our estimation the cost of adding ROM and increasing execution time was too high on this machine to provide complex arithmetic that is accurate in each component. However, accurate components are delivered in those functions where it is more practical. This is discussed further in the following section.

In general, the HP-15C delivers complex results that satisfy $R(Z,z) < 6 \times 10^{-10}$, except where functions involving trigonometric calculations (in radians) are evaluated at very large arguments or near transcendental zeros such as multiples of π. This inaccuracy is embedded in the real-valued functions and is an example of an error that is too costly to correct completely.[3,4]

Some Specific Complex Functions

For complex arithmetic we obtained accurate results (i.e., small complex relative errors) from the standard formulas used to define each operation. But, in general, defining formulas are usually not accurate for computers. In this section we will single out two particular functions, sin(z) and $\sqrt{z}$, and very briefly focus on some difficulties that arise.

- **Sin(z).** A typical defining formula for the complex sine function is given by

$$\sin(z) = \frac{e^{iz} - e^{-iz}}{2i} \qquad (1)$$

If this is used to compute sin(z) for small $|z|$, the two exponential terms will be nearly equal and thus cause a loss of accuracy. This will result in a large complex relative error even though each step of the calculation is very accurate. If equation (1) is replaced by

$$\sin z = \sin(x)\cosh(y) + i\cos(x)\sinh(y) \qquad (2)$$

where $z = x+iy$, then the relative error problem for small $|z|$ will be solved, and furthermore the components will become accurate (except for the trigonometric difficulty with large angles mentioned earlier). To observe the striking difference in results, we calculate

$$w = \sin(1.234567 \times 10^{-5} + 9.876543 \times 10^{-5} i)$$

for each formula. The outcome is represented below.

Eqn.	W (10-digit calculation of w)	R(W,w)
(1)	$1.234567006 \times 10^{-5} + 9.876530000 \times 10^{-5} i$	10^{-6}
(2)	$1.234567006 \times 10^{-5} + 9.876543015 \times 10^{-5} i$	10^{-10}

The HP-15C's internal calculation is based on equation (2), with minor modifications that exploit the relationships between the real functions to eliminate redundant computation.

- $\sqrt{z}$. The most common definition of the principal square root is

$$\sqrt{z} = \sqrt{|z|}\, e^{i\theta/2} \qquad (3)$$

where θ is the Arg (z), satisfying $-\pi < \theta \leq \pi$.

This formula is accurate with respect to complex relative error, but not accurate in each component. This can be seen by working through the calculation of $\sqrt{a}$, where $a = -1 + (-1 \times 10^{-15} i)$, with 10-digit precision. Here $\theta/2$ rounds to precisely 90 degrees, thus causing $\sqrt{a} \rightarrow 0 - i$, while the true

value rounds to $5\times10^{-16}-i$. The complex error is small but certain information in the real component is lost. The fact that $\sqrt{a}$ lies on the right side of the imaginary axis can be critical when computing near discontinuities called branch cuts. For example, $\ln(-i\sqrt{a}) \approx -i\pi/2$, but the inaccurate component of $\sqrt{a}$ will cause it to evaluate to $i\pi/2$ since $-i\sqrt{a}$ is near the branch cut of ln(z). More will be said about branch cuts in the next section.

It turns out that $\sqrt{z}$ can be computed with accurate components and without loss in execution time. This function, along with the inverse trigonometric and hyperbolic functions, is computed on the HP-15C with accurate components. Their algorithms are not described by a simple formula as with sin(z) in equation (2), but rather are described in terms of their components. These accurate components are achieved by recognizing and eliminating errors such as those described above.

Principal Branches

The function $\sqrt{z}$ is an inverse function of $f(z) = z^2$. As is often the case with defining inverses, we must select from more than one solution to define the principal branch of the inverse. This is done for the real function by selecting the non-negative solution of $x^2 = a$ and denoting it by $\sqrt{a}$. Because of the branch point at 0, any branch for $\sqrt{a}$ must have a discontinuity along some slit (branch cut). In equation (3) above, it is along the negative real axis. Notice in Fig. 8 that values below the negative real axis map to values near the negative imaginary axis, while above the slit, values map near the positive imaginary axis. Since it is traditional to have $i = \sqrt{-1}$ we must attach the slit (negative real axis) to the upper half plane, making it continuous from above and not from below, that is, $-\pi<\theta\leq\pi$. One will occasionally see $\sqrt{z}$ defined for $0\leq\theta<2\pi$, which places the discontinuity along the positive real axis. We have avoided doing something like this in the branches of all of the complex inverse functions so that each will be analytic in a region about its real domain. This is important since complex computation is often performed in a region about the real domain in which the function's values are defined by the analytic continuation from the real axis.

The placement of the branch cuts and the function values along the slit are fairly standard for $\sqrt{z}$ and ln(z), but the inverse trigonometric and hyperbolic functions have not, as yet, become standardized. However, by following a few reasonable rules there is not much room for variation.

The first rule, analyticity about the real domain, has already been mentioned. Secondly, we have tried to preserve fundamental relationships such as the oddness or evenness of functions (e.g., $\sin(-z) = -\sin(z)$) and the computational formulas relating functions to the standard principal branches of ln(z) and $\sqrt{z}$ (e.g., $\pi/2-\sin(z) = g(z)\sqrt{1-z}$ where g(z) is analytic at 1, that is, a power series in $z-1$).

The determination of formulas involving a choice of branches is often quite complicated. W.M. Kahan has presented a very enlightening discussion[5] of branch cuts and has pointed out to us that the HP-15C branch cuts should satisfy certain simple formulas relating them to the principal branch of ln(z). These formulas are satisfied and are reproduced below.

$$\ln(z) = \ln(|z|) + i\,\mathrm{Arg}(z)$$

and

$$\sqrt{z} = \exp(\ln(z)/2)$$

with $-\pi < \mathrm{Arg}(z) \leq \pi$ and $\sqrt{0} = 0$

$$\begin{aligned}
\mathrm{arctanh}(z) &= [\ln(1+z) - \ln(1-z)]/2 \\
&= -\mathrm{arctanh}(-z) \\
\arctan(z) &= -i\,\mathrm{arctanh}(iz) \\
&= -\arctan(-z) \\
\mathrm{arcsinh}(z) &= \ln(z + \sqrt{1+z^2}) \\
&= -\mathrm{arcsinh}(-z) \\
\arcsin(z) &= -i\,\mathrm{arcsinh}(iz) \\
&= -\arcsin(-z) \\
\arccos(z) &= \pi/2 - \arcsin(z) \\
\mathrm{arccosh}(z) &= 2\ln\left[\sqrt{(z+1)/2} + \sqrt{(z-1)/2}\,\right]
\end{aligned}$$

These are not intended as algorithms for computation, but as relations defining precisely the principal branch of each function.

Matrix Calculations

As mentioned earlier, the HP-15C can perform matrix addition, subtraction, and multiplication. It can also calculate determinants, invert square matrices, and solve systems of linear equations. In performing these last three operations, the HP-15C transforms a square matrix into a computationally convenient and mathematically equivalent form called the LU decomposition of that matrix.

LU Decomposition

The LU decomposition procedure factors a square matrix, say **A**, into a matrix product **LU**. **L** is a lower-triangular square matrix with 1s on its diagonal and with subdiagonal elements having values between −1 and 1, inclusive. **U** is an upper-triangular square matrix. The rows of matrix **A** may be permuted in the decomposition procedure. The possibly row-permuted matrix can be represented as the matrix

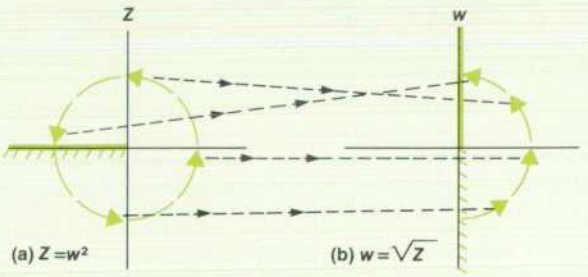

Fig. 8. *The complex function $Z = w^2$, shown in (a) has an inverse function $w = \sqrt{Z}$, shown in (b), which maps the Z plane onto the right half plane of w with a branch cut along the negative real axis of the Z plane.*

product **PA** for some invertible matrix **P**. The LU decomposition can then be represented by the matrix equation $\mathbf{PA} = \mathbf{LU}$ or $\mathbf{A} = \mathbf{P}^{-1}\mathbf{LU}$.

The HP-15C uses the Doolittle method with partial pivoting to construct the LU decomposition. It constructs the decomposition entirely within the result matrix. The upper-triangular part of **U** and the subdiagonal part of **L** are stored in the corresponding parts of the result matrix. It is not necessary to save the diagonal elements of **L** since they are always equal to 1.

Partial pivoting is a strategy of row interchanging to reduce rounding errors in the decomposition. The row interchanges are recorded in the otherwise underused XS format fields of the result matrix's diagonal elements. The recorded row interchanges identify the result matrix as containing an LU decomposition and the result matrix's descriptor includes two dashes when displayed.

The determinant of the decomposed matrix **A** is just $(-1)^r$ times the product of the diagonal elements of **U**, where r is the number of row interchanges represented by **P**. The HP-15C computes the signed product after decomposing the argument matrix **A** into the result matrix.

The HP-15C calculates the inverse of the decomposed matrix using the relationship

$$\mathbf{A}^{-1} = \left[\mathbf{P}^{-1}\mathbf{LU}\right]^{-1} = \mathbf{U}^{-1}\mathbf{L}^{-1}\mathbf{P}$$

It does this by inverting both **U** and **L**, computing the product of their inverses, and then interchanging the columns of the product in the reverse order of the row interchanges of **A**. This is all done within the result matrix.

Solving a system $\mathbf{AX}=\mathbf{B}$ for **X** is equivalent to solving $\mathbf{LUX}=\mathbf{PB}$ for **X**, where $\mathbf{PA}=\mathbf{LU}$ denotes the LU decomposition of **A**. To solve this system, the HP-15C first decomposes the matrix **A** in place. The calculator then solves the matrix equation $\mathbf{LY}=\mathbf{PB}$ for matrix **Y** (forward substitution) and finally $\mathbf{UX}=\mathbf{Y}$ for matrix **X** (backward substitution), placing the solution **X** into the result matrix.

The LU decomposition is returned by a determinant or system solution calculation and can be used instead of the original matrix as the input to subsequent determinant, matrix inverse, or system solution calculations.

Norms and the Condition Number

A norm of a matrix **A**, denoted by $\|\mathbf{A}\|$, is a matrix generalization of the absolute value of a real number or the magnitude of a complex number. Any norm satisfies the following properties:

- $\|\mathbf{A}\| \geq 0$ for any matrix and $\|\mathbf{A}\| = 0$ if and only if $\mathbf{A} = 0$
- $\|a\mathbf{A}\| = |a| \times \|\mathbf{A}\|$ for any number a and matrix **A**
- $\|\mathbf{A}+\mathbf{B}\| \leq \|\mathbf{A}\| + \|\mathbf{B}\|$ for any matrices **A** and **B**
- $\|\mathbf{AB}\| \leq \|\mathbf{A}\| \times \|\mathbf{B}\|$ for any matrices **A** and **B**.

One measure of the distance between two matrices **A** and **B** is the norm of their difference, $\|\mathbf{A}-\mathbf{B}\|$. A norm can also be used to define a condition number of a square matrix, which measures the sensitivity of matrix calculations to perturbations in the elements of that matrix.

The HP-15C provides three norms. The Frobenius norm of a matrix **A**, denoted $\|\mathbf{A}\|_F$, is the square root of the sum of the squares of the matrix elements. This is a matrix generalization of the Euclidean length of a vector.

The HP-15C also provides the row (or row-sum) norm. The row norm of an m-by-n matrix **A** is the largest row sum of absolute values of its elements and is denoted by $\|\mathbf{A}\|_R$:

$$\|\mathbf{A}\|_R = \max_{1\leq i\leq m} \sum_{j=1}^{n} |a_{ij}|$$

The column (or column-sum) norm of a matrix **A** is denoted by $\|\mathbf{A}\|_C$ and is the largest column sum of absolute values of its elements. It can be computed as the row norm of the transpose of the matrix **A**.

For any choice of norm, a condition number K(**A**) of a square matrix **A** can be defined by

$$K(\mathbf{A}) = \|\mathbf{A}\| \times \|\mathbf{A}^{-1}\|$$

Then $K(\mathbf{A}) = \|\mathbf{A}\| \times \|\mathbf{A}^{-1}\| \geq \|\mathbf{A}\mathbf{A}^{-1}\| = \|\mathbf{I}\| \geq 1$ for any norm. The following discussion assumes the condition number defined by the row norm. Similar statements can be made for the other norms.

If rounding or other errors are present in matrix elements, these errors will propagate through subsequent matrix calculations. They can be magnified significantly. Consider, for example, the matrix product **AB** where **A** is a square matrix. Suppose that **A** is perturbed by the matrix $\Delta\mathbf{A}$. The relative size of this perturbation can be measured as $\|\Delta\mathbf{A}\| / \|\mathbf{A}\|$. The relative size of the resulting perturbation in the product is then

$$\begin{aligned}\|\Delta\mathbf{A}\,\mathbf{B}\| / \|\mathbf{A}\,\mathbf{B}\| &= \|\Delta\mathbf{A}\,\mathbf{A}^{-1}\mathbf{A}\,\mathbf{B}\| / \|\mathbf{A}\,\mathbf{B}\| \\ &\leq \|\Delta\mathbf{A}\,\mathbf{A}^{-1}\| \\ &\leq \|\Delta\mathbf{A}\| \times \|\mathbf{A}^{-1}\| \\ &= K(\mathbf{A})\ \|\Delta\mathbf{A}\| / \|\mathbf{A}\|\end{aligned}$$

with equality for some choices of **A**, **B**, and $\Delta\mathbf{A}$. Hence K(**A**) measures how much the relative uncertainty of a matrix can be magnified when propagated into a matrix product.

Uncertainties in the square system matrix **A** or the matrix **B** of the system of equations $\mathbf{AX}=\mathbf{B}$ will also propagate into the solution **X**. For small relative uncertainties $\Delta\mathbf{A}$ in **A**, say $\|\Delta\mathbf{A}\| / \|\mathbf{A}\| << 1/K(\mathbf{A})$, the condition number is a close approximation to how much the relative uncertainty in **A** or **B** can be magnified in the solution **X**.[6]

A matrix is said to be ill-conditioned if its condition number is very large. We have seen that errors in the data—sometimes very small relative errors—can cause the solution of an ill-conditioned system to be quite different from the solution of the original system. In the same way, the inverse of a perturbed ill-conditioned matrix can be quite different from the inverse of the unperturbed matrix. But both differences are bounded by the condition number; they can be relatively large only if the condition number is large.

Singular and Nearly Singular Matrices

A large condition number also indicates that a matrix is relatively close to a singular matrix (determinant = 0). Suppose that **A** is a nonsingular matrix.

$$1/K(\mathbf{A}) = \min(\|\mathbf{A}-\mathbf{S}\|/\|\mathbf{A}\|)$$

$$\text{and } 1/\|\mathbf{A}^{-1}\| = \min(\|\mathbf{A}-\mathbf{S}\|),$$

where each minimum is taken over all singular matrices **S**.[6] $1/\|\mathbf{A}^{-1}\|$ is the distance from **A** to the nearest singular matrix. $1/K(\mathbf{A})$ is this distance divided by the norm of **A**.

For example, if

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 1 & 0.9999999999 \end{bmatrix}$$

then

$$\mathbf{A}^{-1} = \begin{bmatrix} -9{,}999{,}999{,}999 & 10^{10} \\ 10^{10} & -10^{10} \end{bmatrix}$$

and $\|\mathbf{A}^{-1}\| = 2\times10^{10}$. Therefore, there should exist a perturbation matrix $\Delta\mathbf{A}$ with $\|\Delta\mathbf{A}\| = 5\times10^{-11}$ that makes $\mathbf{A}+\Delta\mathbf{A}$ singular. Indeed,

$$\Delta\mathbf{A} = \begin{bmatrix} 0 & -5\times10^{-11} \\ 0 & 5\times10^{-11} \end{bmatrix}$$

has $\|\Delta\mathbf{A}\| = 5\times10^{-11}$, and

$$\mathbf{A}+\Delta\mathbf{A} = \begin{bmatrix} 1 & 0.99999999995 \\ 1 & 0.99999999995 \end{bmatrix}$$

is singular.

In principle, because the HP-15C's matrices are bounded in size, exact arithmetic and exact internal storage could be used to ensure 10-digit accuracy in matrix calculations. This was considered prohibitively expensive, however. Instead, the HP-15C is designed to perform arithmetic and store intermediate calculated values using a fixed number of digits.

Numerical determinant, matrix inversion, and system solution calculations using a fixed number of digits introduce rounding errors in their results. These rounding errors can be conceptually passed back to the input data and the calculated results interpreted as exact results for perturbed input data $\mathbf{A}+\Delta\mathbf{A}$. If the norm of the conceptual perturbation $\Delta\mathbf{A}$ is comparable to $1/\|\mathbf{A}^{-1}\|$, the original nonsingular input matrix **A** may be numerically indistinguishable from a singular matrix.

For example, a square matrix is singular if and only if at least one of the diagonal elements of **U**, the upper triangular matrix in the LU decomposition of **A**, is zero. But because the HP-15C performs calculations with only a finite number of digits, some singular and nearly singular matrices cannot be distinguished in this way.

The matrix

$$\mathbf{B} = \begin{bmatrix} 3 & 3 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1/3 & 1 \end{bmatrix}\begin{bmatrix} 3 & 3 \\ 0 & 0 \end{bmatrix} = \mathbf{L}\,\mathbf{U}$$

is singular. Using 10-digit accuracy, the calculated LU decomposition is

$$\mathbf{L}\,\mathbf{U} = \begin{bmatrix} 1 & 0 \\ 0.3333333333 & 1 \end{bmatrix}\begin{bmatrix} 3 & 3 \\ 0 & 10^{-10} \end{bmatrix}$$

which is the decomposition of the nonsingular matrix

$$\mathbf{D} = \begin{bmatrix} 3 & 3 \\ 0.9999999999 & 1 \end{bmatrix}$$

Hence the calculated determinants of **B** and **D** are identical. On the other hand, the matrix

$$\mathbf{A} = \begin{bmatrix} 3 & 3 \\ 1 & 0.9999999999 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 1/3 & 1 \end{bmatrix}\begin{bmatrix} 3 & 3 \\ 0 & -10^{-10} \end{bmatrix} = \mathbf{L}\,\mathbf{U}$$

is nonsingular. Using 10-digit accuracy, the calculated LU decomposition is

$$\mathbf{L}\,\mathbf{U} = \begin{bmatrix} 1 & 0 \\ 0.3333333333 & 1 \end{bmatrix}\begin{bmatrix} 3 & 3 \\ 0 & 0 \end{bmatrix}$$

which is the LU decomposition of the singular matrix

$$\mathbf{C} = \begin{bmatrix} 3 & 3 \\ 0.9999999999 & 0.9999999999 \end{bmatrix}$$

The calculated determinants of **A** and **C** are also identical.

Because the calculated LU decompositions of some singular and nonsingular matrices are identical, any test for singularity based upon a calculated decomposition would be unreliable. Some singular matrices would fail the test and some nonsingular ones would pass it. Therefore, no such test is built into the HP-15C.

Instead, if a calculated diagonal element of **U**, which we call a pivot, is found to be zero during the LU decomposi-

tion, rather than aborting the matrix calculation and reporting the input matrix to be singular, the HP-15C replaces the zero pivot by a small positive number and continues with the calculation. This number is usually small compared to the rounding errors in the calculations. Specifically, it will be about 10^{-10} times the largest absolute value of any element in that column of the original matrix. If every element in that column of the original matrix has an absolute value less than 10^{-89}, the value 10^{-99} is used instead.

An advantage of replacing zero pivots by nonzero pivots is that matrix inversion and system solution calculations will not be interrupted by zero pivots. This is especially useful in applications such as calculating eigenvectors using the method of inverse iteration. Example programs calculating eigenvalues and eigenvectors can be found in reference 3.

The effect of rounding errors and possible intentional perturbations causes the calculated decomposition to have all nonzero pivots and to correspond to a nonsingular matrix usually identical to or negligibly different from the original matrix.

Complex Matrix Calculations

The HP-15C only operates on real matrices, that is, matrices with real elements. However, it is possible to represent complex matrices as real matrices and to perform matrix addition, subtraction, multiplication, and inversion of complex matrices and to solve complex systems of equations using these real representations.

Let $\mathbf{Z} = \mathbf{X} + i\mathbf{Y}$ denote a complex matrix with real part $\mathbf{X}$ and imaginary part $\mathbf{Y}$, both real matrices. One way to represent $\mathbf{Z}$ as a real matrix is as the partitioned matrix

$$\mathbf{Z}^{\mathrm{P}} = \begin{bmatrix} \mathbf{X} \\ \mathbf{Y} \end{bmatrix}$$

having twice the number of rows but the same number of columns as $\mathbf{Z}$. Complex matrices can be added or subtracted by adding and subtracting such real representations.

Another computationally useful real representation for $\mathbf{Z}$ is

$$\widetilde{\mathbf{Z}} = \begin{bmatrix} \mathbf{X} & -\mathbf{Y} \\ \mathbf{Y} & \mathbf{X} \end{bmatrix}$$

having twice the number of rows, and columns as $\mathbf{Z}$. The HP-15C's built-in matrix operation **MATRIX 2** performs the transformation

$$\mathbf{Z}^{\mathrm{P}} \rightarrow \widetilde{\mathbf{Z}}$$

The operation **MATRIX 3** performs the inverse transformation

$$\widetilde{\mathbf{Z}} \rightarrow \mathbf{Z}^{\mathrm{P}}$$

Suppose $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{C}$ are complex matrices and $\mathbf{A}$ is invertible. Then complex matrix multiplication, inversion, and system solution can be performed with real matrices and built-in HP-15C operations using the relationships:

$$(\mathbf{AB})^{\mathrm{P}} = \widetilde{\mathbf{A}}\mathbf{B}^{\mathrm{P}},$$

$$\widetilde{(\mathbf{A}^{-1})} = (\widetilde{\mathbf{A}})^{-1},$$

$$\mathbf{A}\,\mathbf{C} = \mathbf{B} \rightarrow \mathbf{C}^{\mathrm{P}} = (\widetilde{\mathbf{A}})^{-1}\mathbf{B}^{\mathrm{P}}.$$

These procedures are illustrated in the *HP-15C Owner's Handbook*.

Matrix Transpose

The operations **MATRIX 2** and **MATRIX 3** perform their transformations using a matrix transpose routine. The rows and columns of a matrix are interchanged to form the transpose of that matrix. The transformation is performed in place, replacing the original matrix by its transpose. This routine is available to the user as **MATRIX 4**. Consider the following example:

$$\begin{bmatrix} a & b & c \\ d & e & f \end{bmatrix} \rightarrow \begin{bmatrix} a & d \\ b & e \\ c & f \end{bmatrix}$$

Here the elements of the matrices have been displayed in a two-dimensional format. However, they are stored in a one-dimensional sequence within the calculator's memory. For this example, the transpose operation changes the ordering of the elements within the calculator memory as

a b c d e f → a d b e c f.

The **MATRIX 4** operation moves the elements according to

These movements form disjoint loops. The first value in the sequence is the first candidate for moving. As a value is copied into its destination, that destination is tagged in its XS field. The previous value at that location is the next candidate for moving. Movement along a loop continues until a destination is encountered that is already tagged. The content of the tagged destination is not changed and the current loop is terminated. The value in the location immediately following that tagged destination is the next candidate for moving.

This operation continues moving values along loops until the sequence is exhausted, at which point all destination tags are removed. Finally, the recorded dimensions of the matrix are switched.

Accuracy of Matrix Calculations

Accuracy specifications for all matrix operations are given in reference 3. These specifications are stated in terms of both backward and forward error analysis. Reference 3 includes a general rule of thumb for the number of significant digits in a calculated matrix inverse or system solution. It also includes descriptions of techniques to improve upon the accuracy of calculated system solutions and to reduce the ill-conditioning of systems of equations.

Acknowledgments

Numerous individuals made valuable contributions to the HP-15C software effort. As the software project manager, Rich Carone helped formulate some of the original design concepts and kept the software effort on track. Diana Roy, Robert Barkan, and Hank Schroeder wrote the *HP-15C Owner's Handbook*. We would like to give special thanks to Professor William Kahan, who contributed many design ideas, provided strong guidance in developing the mathematical algorithms, and wrote a portion of the *HP-15C Advanced Functions Handbook*. His unbounded enthusiasm for the product helped keep us going, especially when we still had features to implement and no ROM space left.

References

1. W.M. Kahan, "Personal Calculator Has Key to Solve Any Equation f(x)=0," Hewlett-Packard Journal, Vol. 30, no. 12, December 1979.
2. W.M. Kahan, "Handheld Calculator Evaluates Integrals," Hewlett-Packard Journal, Vol. 31, no. 8, August 1980.
3. *HP-15C Advanced Functions Handbook*.
4. D.W. Harms, "The New Accuracy: Making $2^3=8$," Hewlett-Packard Journal, Vol. 28, no. 3, November 1976, p. 16.
5. W.M. Kahan, "Branch Cuts for Complex Elementary Functions," working document for IEEE Floating-Point Standards Subcommittee, September 9, 1982.
6. E. Atkinson, *An Introduction to Numerical Analysis*, John Wiley & Sons, 1978, pp. 461-463.

Paul J. McClellan

Paul McClellan started working part-time at HP in 1979 while he was a graduate student at Oregon State University. He has a BS degree in physics and mathematics awarded by the University of Oregon in 1974 and an MS degree in statistics from Oregon State. He plans to complete the requirements for the PhD degree in statistics this year. Paul began full-time work at HP in late 1982 and worked on the **SOLVE** and matrix computation routines for the HP-15C. He is a member of the American Statistical Association and coauthor of the *HP-15C Advanced Functions Handbook*. When not busy with work or his studies, Paul enjoys rock climbing, mountaineering, cross-country skiing, and visiting Portland, Oregon. He is single and lives in Corvallis, Oregon.

Joseph P. Tanzini

Joe Tanzini was born in Trenton, New Jersey and attended Trenton State College, receiving a BA degree in mathematics in 1973. He continued his mathematics studies at Lehigh University and earned the MS and PhD degrees in 1975 and 1982. Joe started at HP as a software engineer in 1980 and wrote firmware for calculators, including coding the integrate algorithm and complex functions for the HP-15C. He also coauthored the *HP-15C Advanced Functions Handbook*. Joe enjoys bicycling and walking during his leisure time. He is married, has two daughters, and lives in Corvallis, Oregon.