The ASYST Software for Scientific Computing

DAVID HARY, KOICHI OSHIO, STEVEN D. FLANAGAN

Scientists now use many software packages that have been adapted from mainframe and minicomputer systems to the personal computer (PC). Although such software packages were not developed specifically for the PC, these programs have withstood thorough scrutiny by users and developers over many years. ASYST (*1*), which was expressly developed for the PC, furnishes comprehensive and powerful procedures for mathematics, statistics, and graphics (module 1); signal analysis and advanced mathematics and statistics (module 2); signal acquisition (module 3); and control and measurement (module 4) (Table 1).

The multiple applications of ASYST are based on a comprehensive software algebra that not only facilitates programming but also provides a framework that preserves the parsimonious structure of mathematical formulas to ensure that algorithms are not only precise, coherent, and concise but also easy to test and correct. We have reviewed modules 1, 2, and 3 of ASYST (Table 2) and outlined the primary features of the system as well as some difficulties that we encountered in its use. Alternative software packages for scientific computing on the PC (for example, RS/1, Bolt Beranek and Newman, Cambridge) have been discussed elsewhere whereas some other packages (for example, DADISP, DSP Systems, Cambridge) are relatively new and have not been extensively reviewed (*2*).

Programming Structure

Software for scientific computing typically offers solutions for specific problems that require particular mathematical algorithms. Even when preexisting software libraries are used, software development often involves laborious construction. Scientific programming is further complicated by programming languages that require a scalar algebra with numerous equations for the implementation of relations that are often cast in matrix algebra. For example, the resultant force **F** on a ball due to gravity **G** and drag **D** is **F** = **G** + **D**. Usually, programs for implementing this equation require the separate addition of each of the x, y, and z spatial directions (for example, the vertical force is $F_z = G_z + D_z$). The simple vector addition equation is transformed into three scalar equations that depart from the original mathematical construct.

Program complexity increases if more intricate vector calculus is required. An example is the computation of the Magnus force **M** on a curve ball from the vector (or cross) product of the spin **S** (the vector normal to the plane of rotation of the ball) and velocity **V** vectors (*3*). In programming languages such as Fortran or BASIC, the vector product cannot be computed directly from the vector arguments but from the difference of products of the corresponding orthogonal scalar components. Thus the vertical component of the Magnus force M_z, which causes the "breaking" of the trajectory of a curve ball, is proportional to $S_x \cdot V_y - S_y \cdot V_x$.

Vectors and matrices are treated as single entities in ASYST; vector addition is as simple as scalar addition and vector products can be computed directly from the vector arguments. All of the procedures in ASYST are based on a matrix algebra that permits a coherent and concise mathematical formulation of computational algorithms. A rich repertory of commands exists for logical, statistical, and mathematical calculations. Scalar, matrix, and polynomial procedures in integer, real, and complex number types permit extensive calculations (for example, derivative, fast Fourier transform, or analysis of variance) with only a few command lines. These commands use mathematical instructions that are executed by a floating point processor (Table 2) with fast algorithms that conform to the Intel/IEEE standards (*4*). Also available are graphics, data communication (RS-232C), and digital and analog interfacing procedures. ASYST includes about a thousand preprogrammed procedures.

Applications for a particular user can be developed by combining preprogrammed procedures into new ones that can be executed either in "interpreted" (direct) mode for immediate test and analysis or in "compiled" (turnkey) mode for larger or real-time applications. These programming capabilities are enhanced by program development tools that can be used to edit, compile, and run programs; and by file management utilities to store, retrieve, copy, delete, rename, print, and type DOS (disk operating system) files, all of which are supported by an on-line help facility.

Data manipulations in ASYST are performed with a reverse Polish convention in which the operation appears after its arguments; for example, the equation $c = a + b$ is implemented by the command line: a b + c : = . Operations are performed either with a "numeric stack" designed for numeric calculations or a separate "symbolic stack" for logical and string manipulations. These stacks consist of a series of memory locations in which data are stored and retrieved on a last-in–first-out basis; the last element stored is placed on top of the stack and will be used first (Fig. 1). Extensive stack main-

Table 1. ASYST modules and prices.

ASYST module	Price
Basic system, analysis, and graphics (modules 1 and 2)	$1695
System with data acquisition (modules 1, 2, and 3)	$1995
System with GPIB/IEEE-488 (modules 1, 2, and 4)	$1995
System with data acquisition and GPIB/IEEE-488 (modules 1 through 4)	$2195
Extended support plan*	$225
Special classroom prices†	
Minimum of five systems (each includes modules 1 through 4)	$2500
System and documentation	$500
System without documentation	$250

*The extended support plan provides software upgrades as well as phone support by Adaptable Software Laboratories for 1 year. †The special classroom discount is available to eligible university instructors. The technical support plan is required for the special classroom purchase.

D. Hary and K. Oshio, Department of Biomedical Engineering, University of Southern California, Los Angeles, CA 90089.
S. D. Flanagan, Division of Neurosciences, Beckman Research Institute of the City of Hope, 1450 East Duarte Road, Duarte, CA 91010.

Fig. 1. Screen display from ASYST that shows a sequence of commands (boldface) and results that address the numeric stack. Comments were added to identify the commands and indicate the stack contents. For example, typing 1 2 3 in response to the ASYST OK prompt places 3 on top of the stack and 1 at the bottom of the stack as indicated by the notation: \ [1 2 3]. In this notation the brackets delimit the stack and the backslash serves as a comment command. Note that integers are of a double-precision default type.

ASYST commands and results	Explanatory comments
OK **1 2 3**	\ Place 1 2 3 on the numeric stack.
OK **?s**	\ Display stack contents from
3 (DP.INTEGER)	\ top, or first, element
2 (DP.INTEGER)	\ to bottom, or third, element.
1 (DP.INTEGER)	\ Our convention is: [1 2 3]
OK **dup ?s**	\ Duplicate the top element.
3 (DP.INTEGER)	\ [1 2 3 3]
3 (DP.INTEGER)	
2 (DP.INTEGER)	
1 (DP.INTEGER)	
OK * ?s	\ Multiply the top two elements.
9 (DP.INTEGER)	\ [1 2 9]
2 (DP.INTEGER)	
1 (DP.INTEGER)	

tenance operations, which are designed to facilitate stack-oriented calculus, are available for the numeric and symbolic stacks. These include commands to duplicate or drop the top stack element, swap the top two elements, or display the contents of these stacks (Fig. 1). Learning to manipulate the numeric stack with its reverse Polish convention is facilitated by a stack display mode where the four top elements of the stack are displayed. Logical operations [AND, NOT, OR, and XOR (exclusive or)] act only on the symbolic stack but allow logical algebra manipulation of numeric data that are exchanged with the symbolic stack through a set of binary mask procedures.

Command Structure-"Words"

Diverse numeric stack data manipulations from integer operations through matrix algebra use almost identical notation and are based on preprogrammed procedures and commands termed "words." Unary (single argument) operations (such as the square-root and logarithm functions) and binary (two arguments) operations (such as addition or subtraction) apply to complex, real, or integer numbers in either matrix, vector, or scalar form. Complex scalars, vectors, or matrices in Cartesian ($a + i{\cdot}b$) or polar ($r{\cdot}e^{i\theta}$) formats are manipulated with specific words to compute the real (a) or imaginary (b) parts, magnitude [$r = (a^2 + b^2)^{1/2}$], phase (θ), or complex conjugate ($r{\cdot}e^{-i\theta} = a - i{\cdot}b$).

Array comparison, concatenation, partitioning, and rearrangement words are included in the matrix algebra. This algebra is extended by two generalized products. First, the generalized outer product (⟨⟨ **binary, operation** ⟩⟩) is used to perform a binary operation on every pair of its matrix (or vector) arguments. For example, the command **a b** ⟨⟨ * ⟩⟩ performs the tensor product of the column vector (**a**) by the row vector (**b**) and yields a matrix (**c**) with elements $\mathbf{c}_{i,j} = \mathbf{a}_i{\cdot}\mathbf{b}_j$. Second, two consecutive binary operations are performed by using the generalized inner product (⟨⟨ **binary,operation,1 | binary,operation,2** ⟩⟩). For example, the inner (or scalar) product can be implemented to find the sum-of-squares of vector elements as **a a** ⟨⟨ * | + ⟩⟩ . The two generalized products permit a total of 182 different commands (that use +, −, /, *, maximum, minimum, modulo, and six comparison functions).

An extensive ensemble of commands permits the display and measurement of two- and three-dimensional data with video (*5*) or pen plotter graphics (Table 2). Preprogrammed words for Cartesian figures, histograms, pie charts, and axonometric and contour plots use a set of graphics window definitions, data scaling modes, axis and grid parameters, and color attributes that are accessible to the user, who can then develop specific graphics applications.

Other words can be used to scroll an array of data and to measure and store the coordinates (magnitude and index) of specific data points. The preprogrammed mathematical and integrated graphical words form a powerful yet flexible tool for the preliminary examination of data and allow interactive testing and development of new applications. Commands that execute simple algebraic expressions or sophisticated dot-products, polynomial fits, or fast Fourier transforms (Radix-2) (Fig. 2) can be used interactively for immediate study of experimental results.

Words, or colon definitions (so named because a colon begins the definition), which are defined by the user and use the stack and other previously defined variables and words, are needed for optimal use of ASYST. Words are essential because labeling of subroutines or the use of commands to branch to other labeled commands are not allowed in ASYST. Words are also important because iterative loops, conditional loops, IF-ELSE-THEN statements, and sequential case selections can only be

Table 2. ASYST versions, computers, operating systems, and peripherals tested. A math co-processor is required for the operation of ASYST. The 8087 is used with the IBM PC-XT and compatibles and the 80287 with the IBM PC-AT and compatibles. Suppliers and their locations: Analog Devices, Norwood, MA; COMPAQ Computer Corporation, Houston, TX; EPSON America, Torrance, CA; Hercules, Berkeley, CA; Intel Corp., Santa Clara, CA; International Business Machines, Boca Raton, FL; Practical Peripherals, Westlake Village, CA; Quadram, Norcross, GA.

Software or hardware	Versions or models
ASYST modules 1, 2 and 3	1.51, 1.51 upgrade, 1.53, 1.56, and 1.56 upgrade
Computers	IBM: IBM PC-XT and PC-AT
	COMPAQ: Portable, Portable Plus, and Portable 286
Operating systems	DOS versions 2.1, 3.0, 3.1, and 3.2
Random-access memory	640 kbytes
Interfaces	IBM parallel and serial adapters
Graphics adapters	IBM Enhanced Graphics with 256-kbyte memory module
	IBM Color Graphics
	COMPAQ graphics and COMPAQ Enhanced graphics
	Hercules Graphics Card
Display monitors	IBM Color Display
	IBM Enhanced Color Display
Printers	IBM: Proprinter and graphics printer
	EPSON: FX-80
Printer buffers	Quadram Microfazer
	Practical Peripherals Microbuffer-Mini
Plotters	IBM 7371 and HP 7475A
Data acquisition	IBM Data Acquisition and Control Adapter
	Analog Devices RTI-815

ASYST commands	Results and explanatory comments
P estimate.roots	\ [roots] Estimates the roots of the polynomial
	\ whose coefficients are in the array P.
M matrix.inv	\ [inverse.matrix] Inverts the matrix M.
M V solv.sim.eqs	\ [X] Solves the simultaneous equations: M·X = V.
X Y N leastsq.poly.fit	\ [poly] Fits the (x,y) data pairs in the arrays
	\ X and Y with a polynomial of degree N.
X fft	\ [Fourier transform of the array X]
M 2d.fft	\ [Two dimensional Fourier transform of matrix M]
M table.anova	\ [anova.table] Computes the two way analysis of
	\ variance table of the sample data
	\ stored in M

Fig. 2. Sample implementation of a few of the computationally intensive ASYST words. The commands are written as they would be used interactively at the ASYST prompt, in an ASYST program, or within ASYST words.

implemented through the use of words (Fig. 3).

Data Acquisition and Storage

Multichannel data acquisition and control procedures are available for digital and analog input and output [analog-to-digital (A/D) and digital-to-analog (D/A) conversions, respectively], either in synchronous (internal or external clock interrupts) or sequentially delayed timing modes through a wide variety of interface boards (*6*). Concurrent (multitask) acquisition, display, and storage are also supported. Analog input channels are sampled serially according to a specified interchannel delay period (τ) termed the conversion delay. Data from all n channels can be successively stored in an n-dimensional array (Fig. 4). In the delayed timing mode, a longer conversion delay is inserted between channels to determine the sampling frequency f_0 per channel, which therefore depends on the number of channels sampled [$f_0 = 1/(n\tau)$]. Thus samples of multiple channels that are stored as "concurrent" in discrete time are in fact delayed by as much as $(n - 1)\tau$ second. This phase shift between "concurrent" channels linearly increases with frequency f such that the k-th channel leads the first channel by a phase of $\theta = 360f(k - 1)\tau$ degrees. For example, when four channels are sampled, channel 4 leads channel 1 by 135° at half the sampling frequency (Fig. 4). Slower (*7*) but more precise sampling is feasible in the synchronous timing mode by setting the conversion delay to a minimum and by setting the synchronization rate to the sampling frequency. With minimal effort we were able to program ASYST for (i) data acquisition from multiple channels, (ii) generation of a Poisson pulse train while concurrently acquiring four channels of electroencephalographic data at 250 Hz, or (iii) emulation of a metronome with small random fluctuations in rate (*8*).

Three different file types can be accessed; standard text (ASCII), standard binary, and ASYST binary files. ASYST files include an elaborate and flexible scheme for storing laboratory data as a sequence of comments (a laboratory "notebook") followed by a set of data arrays. Any number of arrays of different dimensions and number types can be stored or appended to existing ASYST files. The ASYST file structure is identified by an elaborate and accessible file template that is automatically modified as the file is appended (*9*). However, ASYST does not support subdirectory paths for its overlay, help, and messages files and does not include procedures to access subdirectories (*10*).

The matrix-oriented calculus, the combination of compiled and interpreted modes of operation, the extensive built-in mathematical, statistical, and engineering procedures, the intensive graphics commands, and the DOS commands make ASYST an innovative alternative approach for solving scientific problems. However, because of its ambitious features, ASYST is more prone to criticism than are other less comprehensive packages. Because many facets of scientific numerical analysis are encompassed in ASYST, no single user or small group of scientists can fully use its capabilities. Nevertheless, constructive critical review is crucial for future development of ASYST and we present an assessment of some of the difficulties we encountered with this package.

Areas for Improvement

ASYST variables have to be named and given a number type (integer, real, or complex) and a structure (scalar or array) before they can be used in the program. New variables cannot be declared within and confined to a colon definition. This limitation is further exacerbated because a few ASYST words (for example, those for array partitioning) do not readily accept stack data as arguments but instead require either defined scalars or constants, or interactive input of data. The definition of such nonspecific global variables hampers the building of independent libraries of words because such words cannot be safely nested within other words that might also use and modify identical variables (*11*).

Symbolic stack manipulations are considerably more difficult and less predictable than numeric stack calculations. This is because the symbolic stack combines strings and logical variables and apparent inconsistencies exist in the access to this stack. Whereas string commands selectively address string data, logical operations read a string as a sequence of logical variables. This can cause severe and hard-to-detect errors during program execution (runtime errors).

Perhaps the most attractive feature of ASYST is its comprehensive ensemble of preprogrammed words. Many numerical procedures such as matrix inversion, differ-

ASYST word	Equivalent BASIC subroutine
: +−*/ \ [a b op.code - a.op.b]	
case	1000 on op.code goto 1100,1200,1300,1400
1 of + endof	1100 c = a + b : goto 1500
2 of − endof	1200 c = a − b : goto 1500
3 of * endof	1300 c = a * b : goto 1500
4 of / endof	1400 c = a / b : goto 1500
endcase	1500 rem
;	1600 return
Interactive ASYST screen display	**Equivalent BASIC screen display**
OK **5. 2. 4 +-*/**	**10 a=5: b=2: op.code=4: gosub 1000**
OK **.**	**20 print c: end**
	run
OK 2.5000	2.5
OK	OK

Fig. 3. Command branching without labels. The "case" command is used within the word "+−*/" to branch to alternative command sequences thus selecting in this example one of the four mathematical operations (+, −, *, or /) according to the operation code (op.code) value on the stack. An equivalent procedure in BASIC, which uses line numbers as labels, is presented for comparison.

entiation, polynomial solutions, and random-number generation could have been implemented with any one of several algorithms that have different utility, domains of validity, computational errors, and speed. The following examples illustrate the limited range of validity of two of the numerical algorithms of ASYST. First, the algorithm for the power function does not have a singular point for the negative power of zero (it gives $0^{-1} = 0$). Second, the variance

$$\sigma^2 = \sum_{k=1}^{n} (X_k - \mu_x)^2 / n$$

of an n-element column array (X_k), where

$$\mu_x = \sum_{k=1}^{n} X_k / n$$

is implemented as

$$\sigma^2 = \sum_{k=1}^{n} x_k^2 / n - \mu_x^2$$

and occasionally yields negative results due to round-off errors. Where such results are not acceptable, other procedures should be used (*12*). Because it is not feasible to determine the full impact of an unknown algorithm by trial and error, the use of computationally intensive software such as ASYST requires a functional rather than only a numerical analysis of the computational errors and domain of validity of the programmed procedures. However, unlike other software packages (*13*), ASYST source code or details of the algorithms are not distributed. Although the documentation is quite extensive, it is insufficient for analyzing most of the algorithms. As a result the algorithms of ASYST cannot be fully assessed and procedures written in ASYST cannot be completely specified.

Because of its complex and highly interactive features, ASYST contained many flaws in its early versions. A recent provocative article (*14*) noted that, "With software products, it is usual to find that the software has major 'bugs' and does not work reliably for some users. These problems may persist for several versions and sometimes worsen as the software is 'improved'." Perhaps ASYST is an example of this phenomenon. There are difficulties that extend from documentation to more substantial problems in program development, compilation, and execution. Many of the earlier problems have been corrected (*15*), some still persist (*16*), and others have been introduced in the revisions. However, since the release of version 1.51 no documentation errors (*17*), and only a few software corrections were reported by the manufacturer (*18*). It is suggested here that mere correction of earlier versions is insufficient and that the responsibility of the publisher is to report such errors in detail. It is essential that calculations affected by errors be corrected and that new results be appropriately reported.

Error messages in ASYST are often difficult to decipher. A variety of errors (square root of a negative real or integer number, or an integer or real number overflow) cause a "system restarted" message on the IBM PC-AT and an "illegal 8087 operation" message on the IBM PC-XT (*19*). The first "is not really an error message" according to the documentation. For the illegal 8087 coprocessor operation, the manual simply states that the co-processor has encountered an unsuitable operand and that the square root of a negative number is a possible cause. Other possible causes for 8087 coprocessor errors are not listed nor is the user directed to the appropriate literature for a list of conditions that would generate this error. Moreover, numerical underflows occasionally, but not always, cause 8087 errors (*20*). In addition, error messages in ASYST do not include information about the procedure that caused the error. This makes long programs very difficult to correct. Also, recurring errors often overflow the DOS stack. This invariably halts the computer and requires a complete reset.

Software upgrades of ASYST are therefore essential; in fact, the developers have released new versions about twice a year. Still, experience with ASYST thus far appears to belie the developers' claim that ASYST "provides a complete error trapping system with easy to understand error messages."

Most of the 640-kbyte program and system memory of the IBM PC is required to run ASYST. The minimum size of the ASYST system is about 330 kbyte for modules 1 and 2. This includes the ASYST program, free memory for new words, and a stack of 32 kbyte, but does not include any space for arrays. Additional memory is consumed as memory is allocated for additional modules (*21*), strings, user-defined code, numeric stack, and arrays. Hence ASYST cannot coexist with memory-resident utilities that use significant amounts of memory (*22*). Memory rationing, which was significantly alleviated in version 1.53 (*23*), is substantially exacerbated by the use of double-precision numbers and arrays. Such arrays may be internally generated during execution of ASYST words (for example, logarithm in base 10), probably because integers (when not specifically terminated with a capital S) are in double-precision format (Fig. 1) and yield double-precision results when used as function operands. Thus the numeric stack tends to overflow at times for seemingly nonapparent and uncontrollable reasons unless the informed programmer has carefully determined the format of the outcome of each procedure used.

Column and multidimensional arrays are limited to 64 kbyte; for example, a single-precision integer array that contains 32,678 elements can be defined and used. The exact procedure of array-specific stack manipulations, which is integral to the functioning of ASYST, is not fully documented. Apparently array calculations are actually performed on the memory locations of the array elements. When an array is duplicated on the stack the duplicated elements share the same memory locations. Thus every modification of a given element results in an identical change of all the duplicated elements on the stack. Unique memory can be allocated by copying the array rather than just duplicating it. This prevents the undesired propagation of data between stack elements.

Installation and Service

The developers of ASYST have genuinely attempted to support a variety of hardware for data acquisition, display, and printing.

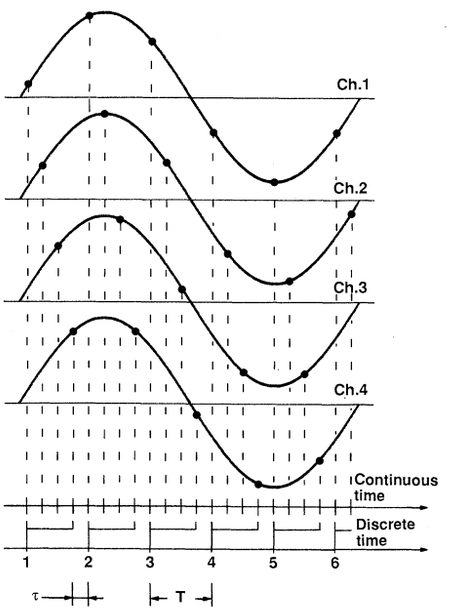

Fig. 4. Schematic representations of the acquisition of multiple channels in the sequentially delayed timing mode. All n channels (four in this figure) are connected to the same sinusoidal signal. Samples are delayed by a base interval of τ second as indicated in the continuous time scale. The sequentially delayed samples of all channels are stored with the same array index for each sample cycle as indicated in the discrete time scale which has a time unit T of $n\tau$ second. As the data are stored, "concurrent" samples within each cycle have as much as $(n - 1)\tau$ second lead relative to the first channel. Channel 4 leads channel 1 by 45° for this example in which the signal is sampled six times per cycle.

Although it is relatively simple to customize ASYST for specific plotters and acquisition adapters, accommodation of the graphics printing command to a particular printer and display-adapter combination is not trivial and often requires direct consultation with the Adaptable Laboratory Software personnel (*24*). The IBM PC standard software protocols that control data transfer to the printer (and to which ASYST seems to conform) limit the rate of data transfer to the printer. Thus standard printer buffers (Table 2) do not save substantial time in printing graphics (*25*).

ASYST is distributed on laser-hole-protected master and backup disks, a help disk, and two installation and demonstration diskettes (*26*). Because of its copy protection, the master diskette is required every time ASYST is loaded, which takes about 15 seconds on the IBM PC-AT. The documentation includes three manuals for the base and analysis systems and a manual for each additional module that is purchased. Maintenance of the ASYST software is annually contracted through the publisher and provided by Adaptable Laboratory Software (Table 1). This essential service entitled us to many hours of extensive and fruitful discussions with the knowledgeable and helpful consultants. Unfortunately, for those who cannot afford the maintenance fee, there are no formal mechanisms for reporting and discussing possible software errors after the 60-day initial period of free technical support. We recommend the establishment of such a procedure.

Conclusions

The acceptance of ASYST by the scientific community could dramatically change the way scientific data are handled and reduce the need for extensive in-house software development for many applications. However, there is no substitute for a well-conceived use of any software. We feel that the full acceptance of the ideas and concepts pioneered by Adaptable Laboratory Software and other software houses will depend on the ability of the scientific community to fully test and verify the procedures used by such products. Only then can the results produced by these software packages be subjected to confirmation which is crucial to rigorous scientific endeavor.

REFERENCES AND NOTES

1. ASYST was developed by Adaptable Laboratory Software and published by Macmillan Software Company, New York, NY 10022.
2. The data acquisition aspects of ASYST were reviewed by P. Wirth and L. E. Ford [*Byte* **11** (no. 7), 303 (1986)]. RS/1 and ASYST were discussed by S. A. Borman [*Anal. Chem.* **57**, 983A (1985)]. RS/1 was presented by C. H. Russell [*Res. Dev.*, **27**, 46 (December 1985)], and reviewed by G. S. Decherney [*Digital Rev.* **2**, 79 (August 1985)]. DADISP was presented by M. S. Conner [*EDN* **31** (no. 15), 54 (1986)].
3. W. R. Bennett, Jr., *Scientific and Engineering Problem-Solving with the Computer* (Prentice-Hall, Englewood Cliffs, NJ, 1976), pp. 49 and 218.
4. The utility of the floating point co-processor was reviewed by S. S. Fried [*Byte* **9** (no. 9), 197 (1984)].
5. Supported by ASYST are color graphics (CGA) with up to 640 horizontal by 200 vertical picture elements (pixels), enhanced graphics (EGA) (640 by 350 pixels), Hercules graphics (720 by 350 pixels), and HP MultiMode and AT&T 6300 graphics (640 by 400 pixels).
6. Among the data acquisition interfaces supported by ASYST are adapters by Analog Devices, Cyborg, Dataq, Data Translation, IBM, Keithley, Metra-Byte, and Scientific Solutions (Tecmar).
7. In the repeated and sequentially delayed sampling mode, up to 15,000 samples per second (sps) can be acquired with the IBM Data Acquisition and Control Adapter (IBM DACA) (two channels, 7500 sps per channel). For proper operation of this interface, the DELAY output must be connected to the A/D control-enable input. The data acquisition buffer (DAS buffer) should be configured to match the sample count. Sampling rates of less than 2400 sps (2 IBM DACA channels) were achieved in the synchronous mode. A minimum delay of 100 μsec should be set between IBM DACA channels to yield an interchannel flutter of 50 to 200 μsec. See DACA-UTL.ASY in the Technical Database (*8*).
8. The program and text files cited in this review are located in an archive file (ASYST.ARC) on the Thousand Oaks Technical Database, Thousand Oaks, CA. Tel. (805)493–1495.
9. ASYST files can be generated in BASIC with the aid of the ASYS-SUB.BAS BASIC subroutine library on the Technical Database (*8*).
10. ASYST is programmed to find its overlay, help, and data files in the default directory. ASYST can be directed with the IBM File Facility (IBM Personally Developed Software, Wallingford, Connecticut) to search for overlay and help files in a given list of directories. ASYST can then be called from any directory or drive. Additional simplification of program development is gained by defining subdirectories that include ASYST data files and programs as virtual devices (for example, D:, E:, and so forth) with the SUBST (DOS 3.1 and higher) command. See AUTOEXEC.ASY and CONFIG.ASY on the Technical Database (*8*).
11. To alleviate this problem and write independent ASYST words, an additional number stack can be used for local variables [STACK.ASY on the Technical Database (*8*)]. Although multiple stacks can be defined in ASYST, these stacks cannot remain concurrently open and therefore cannot serve to store local variables.
12. A word for the power function that tests for the singular zero point and returns the largest real is included in the MATH-UTL.ASY file on the Technical Database (*8*). Also included is a procedure for the variance that is computed as the average sum of squares of deviations from the mean. Although slower than the ASYST variance, this procedure does not produce negative variance.
13. BASICA Scientific Subroutine Library (Wiley, NY 1985). Turbo Pascal Numerical Methods Toolbox (Borland International, Scotts Valley, CA, 1986).
14. The software requirements for the Strategic Defense Initiative have been discussed by D. L. Parnas [*Am. Sci.* **73**, 432 (1985)].
15. Some of the errors already corrected in recent revisions are: DOS device messages could not be trapped, which resulted in program termination; the smoothing procedure returned double-precision arrays that limited the maximum array size; ASYST files, binary files, or text files can be opened or accessed simultaneously only in specific combinations; an empty symbol stack returned a "TRUE" value and an error message; and the concatenation of scalars to arrays yielded unpredictable results.
16. In addition to the discussion of programming and computational difficulties detailed in the text, we found that: attempting to load ASYST with memory-resident programs that do not leave ASYST with enough space may hang the system and require a complete restart; user-defined function keys do not fully execute unless terminated with the Enter key; file access is prohibited while a text file is open; multidimensional arrays of maximum size (65,356 bytes) cannot be viewed; a large number of words are not documented; error messages clear the symbol stack; use of the deferred copy command sequence: "FILE1" DEFER> COPY TO FILE2 clears the symbol stack. Some problems depend on the particular system as well as the system software that is stored in read-only memory (BIOS ROM). For example, division by zero yields zero on a TANDY 3000 (BIOS ROM date of October 1985). Also, COMPAQ BIOS ROM Version F and higher is required for proper 80287 error-trapping in a COMPAQ 286 that does not have a COMPAQ fixed disk. The IBM PC-XT and PC-AT tested have BIOS ROM dates of 8 November 1982 and 10 January 1984, respectively. This date can be read from memory with the BASIC program BIOSDATE.BAS on the Technical Database (*8*).
17. Among the more disturbing documentation errors we found: a mismatch between the description and implementation of the fast Fourier transform; the procedures OUT>FILE.ON and Z=RE&IM are not recognized as valid ASYST words; an incomplete description of the real number round-off procedure; and a mismatch between the description and the actual ASYST file structure. For a list of documentation errors, refer to the TYPOES.DOC document file on the Technical Database (*8*).
18. *ASYST Appl. Newsl.* **2** (no. 1), 3 (1986); *ibid.* **3** (no. 1), 3 (1987).
19. In early versions (before 1.53), these errors would hang the IBM PC-AT and require a complete restart.
20. *ASYST Appl. Newsl.* **3** (no. 1), 2 (1987).
21. Users of modules 3 or 4 or both who write memory-intensive analysis procedures may benefit from a streamlined version that includes modules 1 and 2 for analysis purposes only. This releases about 44 kbyte of memory in version 1.53 and 41 kbyte in 1.56.
22. Memory-resident programs stay resident in memory when they are first run. Once in memory, these programs take control in response to a special key combination. For example, the DOS GRAPHICS.COM program stays resident and prints a graphics screen in response to the keys Shift-PrtSc. Because of the limited access to subdirectories (*10*), it is useful to use a limited-size version of the memory-resident Sidekick notebook (Borland International, Scotts Valley, CA) to read data from the screen or "export" data directly to ASYST in a way that emulates keyboard entry.
23. In the first release of version 1.56 (10 December 1986) excessive unusable memory was allocated to the ASYST system and increased the minimum system size from 330 to about 363 kbyte (modules 1 and 2) and 384 kbyte (modules 1, 2, and 3). This unusable memory space increases from 4% of the minimum system (version 1.53, modules 1 and 2) to 16% and is about 12% with all three modules installed. In a later release, some of the unused memory (8%) was freed (modules 1 and 2 only). Still, improved memory rationing by the developers could free between 16 kbyte (modules 1 and 2) to 30 kbyte (modules 1, 2, and 3) of usable memory.
24. Modification of the EGA graphics print-screen utility is required in recent revisions (1.53 and higher) because of a possible software flaw that causes a diminished vertical range (348 instead of 350 lines). This had been casually reported in *ASYST Appl. Newsl.* **2** (no. 3), 2 (1986).
25. To remedy this problem we are developing a device-specific memory-resident program that transmits an entire graphics screen through an external printer buffer (Table 2) in less than 10 seconds.
26. ASYST is operational as delivered. Users can install specific data acquisition devices, reconfigure parts of the memory allocation, program function keys, set color attributes for a variety of text windows, and configure graphics screens. INITASYS.ASY and KEYS.ASY on the Technical Database (*8*), include initialization procedures to invoke interrupt 5 (for graphic screen dumps), define function keys, to set color attributes, and to set the graphics screen for the enhanced graphics adapter. For our applications we configured the stack (referred to as "heap" by the configuration procedure) size to three to four times the size of largest defined array.

27. This work was supported in part by NIH grants RR-01861 and NS-18854-05 and by the Faculty Research and Innovation Fund of the University of Southern California.