Project DARE: Differential Analyzer REplacement by on-line digital simulation

by GRANINO A. KORN

University of Arizona
Tuscon, Arizona

INTRODUCTION

While batch-processed applications of convenient, highly developed digital continuous-system simulation languages are now commonplace,[1,2] such systems do not provide the intimate man-machine intercourse cherished in analog/hybrid simulation. The DES-I system,[2] which combined a special simulation console and a digital plotter with an SDS 9300 (medium-sized) computer was, then, a pioneering effort, unfortunately abandoned by its manufacturer. The only commercially available interactive system appears to be the IBM CSMP 1130 system which, like its predecessor PACTOLUS,[2] can be programmed from a simple typewriter terminal. This is an interpreter system implemented on a small computer and thus yields relatively quite slow execution.

The writer has felt quite strongly for some time[6] that digital on-line simulation is ready to go—we do have simple simulation-language programming, plus very reasonably priced, fast digital computers, plus new graphic displays. All that would seem to be needed was a system design which would combine these items (Table I), with a good deal of human-factors engineering to make the operator happy as well as efficient. Project DARE (Differential Analyzer REplacement), sponsored by the National Science Foundation at the University of Arizona, is a continuing attempt to develop a series of such systems.

Project DARE demonstrates all-digital on-line simulation of dynamical systems. Each DARE system adds a very convenient but still relatively inexpensive simulation console to a small or large digital computer and can replace conventional analog computers in many applications. System equations or block-statements and input data are entered and conveniently edited on a cathode-ray-tube typewriter. Solutions or phase-plane plots appear on a second cathode-ray-tube display; system parameters and initial conditions are readily changed for successive runs; displayed data can be stored for comparisons; programs and results may be printed and plotted for hard-copy report preparation; and automatic iterative operation is possible. With a reasonably fast digital computer, man-machine interaction at the console is rather more comfortable than with even a modern analog/hybrid computer.

DARE I is a flexible CSSL-type floating-point system permitting relatively slow computation with the PDP-9 computer. DARE II is a block-diagram-based system which trades fixed-point operation for relatively very high speed on the small PDP-9, permitting, for instance, real-time flight simulation. DARE III and DARE IV are only in the planning stage and will implement economical and fast floating-point simulation on a time-shared CDC 6400.

A critical study of future possibilities indicates that DARE-type systems could permit flight simulations including 40 Hz frequencies by 1975, but that modern analog computers are still a hundred times faster. Actual present-day practical applications, however, employ really fast (and therefore relatively inaccurate) analog computation so rarely that much analog simulation could well give way to the more accurate, convenient, and often more economical digital methods demonstrated by Project DARE.

DARE I: *An on-line CSSL-type system*

DARE I software, written for the PDP-9 by J. Goltz as a Ph.D. dissertation,[5] produces a complete floating-point simulation system, including the basic monitor, editor, and loader used also by DARE II. DARE I source language is essentially similar to the SCI-sponsored CSSL.[1] Though basically equation-oriented, DARE I will also implement user-created analog or hybrid blocks as FORTRAN functions.

TABLE I—*A list of requirements for an on-line digital simulation system*

A useful on-line continuous system simulation system must provide for:

1. *Entry of system differential equations* (in equation and/or block statement form).
2. *Entry of data* (system parameters, initial conditions, function tables, etc.).
3. *Entry of simulation parameters* (frame time, communication interval or display sampling interval, maximum computation time, integration routine used, maximum tolerable error in variable-increment integration routines, choice of variables for display).
4. *Editing*, modification, and correction of the above entries.
5. *Display* of state variables vs. the independent variable (usually the time) and against each other (phase-plane plots).
6. *Preparation of hard copy for reports* in the form of printed tables, xy recorder plots, or strip-chart records.

In addition, a sophisticated simulation system must permit "simulation studies," viz.:

7. *Computations based on results from multiple differential-equation-solving runs* (statistics, cross-plots).
8. *Iterative computation*, i.e., repeated runs with system parameters and/or initial conditions recomputed on the basis of preceding runs for optimization, boundary-value problems).

DARE I employs the FORTRAN compiler supplied with the digital computer and will be described in detail in a separate paper.[5]

DARE I accepts *system differential equations* in first-order (state-equation) form. These equations are simply *typed* in FORTRAN notation on the screen of a CRT typewriter at the right of the DARE console (Figure 1). An interactive CRT typewriter pro-

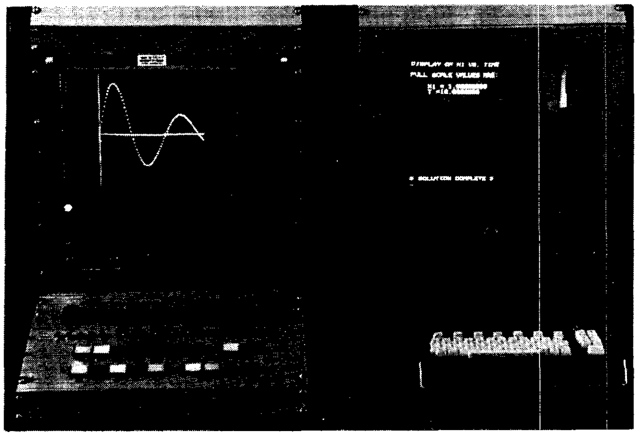

Figure 1—DARE simulation console for use with a PDP-9 or PDP-15 computer. Programs and data are entered, edited, and modified on the CRT typewriter at right. Up to four solution curves, or a phase-plane plot, are produced on-line on the output graphic CRT display at left. A simulation control panel underneath the output display controls simulation and display, with special push-buttons producing hard copy of programs, data, and solutions when desired. The teletypewriter and plotter used for this purpose are not shown.

Console switches (lower left) are sampled by the computer to provide control inputs:

Method Switch: A rotary switch used to select the integration routine.
DT, *TMAX*, *EMAX*: 4-decade thumbwheel switches in an adapted FORTRAN format.
The third decade reads from −5 through 0 to +5, and with the fourth decade indicates a power of 10.
Elapsed Time: A strip of 12 lamps to indicate the progress of computation, and to reassure the user that the computer is actually operating when computation exceeds a few seconds.
Sense Switches: 2 position switches for various functions, determined by program.
Trace Finder: Pushbuttons to identify one of 5 traces on scope display-probably by momentarily blanking it out.
Command Push-buttons (lower two rows):
Lighted pushbuttons, for purposes marked on buttons. "Type eqns," "type data," and "select display," are indicators only, *offering suggestions to the user from the computer.* Such suggestions can also appear on the alphanumeric CRT display.

gram proceeds to ask for *problem data* and *simulation parameters*. Of the latter, the frame time DT, the maximum computing time TMAX, and also the error EMAX for variable-increment integration, can be entered either with the CRT keyboard or by console digiswitches, whichever the operator prefers. Console buttons can *recall* selected program or data pages to the CRT screen for editing, or cause them to be *printed out* for report preparation.

As the differential-equation solution proceeds, all state-variable values are read onto DECtape once per "communication interval"[1] (typically every 10 to 50 DT). Thus any selected state variable can be brought back for single or multiple displays and printout; it is possible to compare a current solution with a selected earlier solution display. Permanent graphic records are obtained with an xy recorder and a four-channel stripchart recorder connected to the display.

The choice of *integration routines* for differential-equation solution has been discussed and rediscussed in many survey papers.[2,4] All DARE systems (like the better batch-processing systems[2]) offer a choice of integration formulas. With the on-line systems, console selection of integration routine and frame time (time increment DT) permits very convenient comparison of different integration methods in terms of stored solution displays.

The flexible and convenient *DARE CRT Editor program*[5,6] permits overwriting and correction, insertion of text, and automatic search for lines containing selected strings.

A SORT/EDIT program (precompiler) sorts the symbol string constituting the program and creates a FORTRAN differential-equation-solving program, which is then compiled and executed. After the first run, data such as system parameters and initial conditions may be changed on the CRT screen, and successive differential-equation solving-runs are obtained without recompilation. Iterative and statistical simulation studies can be programmed with FORTRAN statements.[5]

A new homemade graphic display[7] associated with our DARE console displays up to four variables against time, or selected phase-plane plots. The display uses one dual 9-bit (18-bit) word per display point to save memory and refresh time, can generate line segments for curve interpolation, and shares the processor memory through a standard PDP-9 data channel. This permits fast display refreshing with a minimum of time-wasting instructions.

DARE II: A fast block-macro system with an efficient precompiler

The DARE I system demonstrates the convenience and power of a scale-factor-free, floating-point, equation-oriented, on-line simulation at relatively low computing speed. But *we also wanted to demonstrate a much faster on-line simulation system, which would permit true real-time flight simulation, still using the same small and inexpensive digital computer.* With the PDP-9, this meant giving up floating-point operation. DARE II machine equations must be scaled (much like those in analog computers) between -1 and 1 machine unit; with the PDP-9, ones-complement coding is employed. Overloads are detected and displayed by a special subroutine.

To provide high execution speed, DARE II uses the PDP-9 macro-assembler to create *macros corresponding to analog computing blocks*, an approach first used by Gaskill and McKnight in their batch-processed DAS system on the IBM 7090.[2] Our system permits especially convenient block programming, with each block named by type and by the actual output-variable name. The example of Figure 2 is represented by

```
SUM     F1, S1DOT, S2DOT
COS     COSA, A                          (1)
MULT    S1DOT, COSA, RDOT
```

where the first argument of each block-macro represents the block output. Note the convenient mnemonics used.

DARE II block-statements and data are entered on the dual-CRT console used also with DARE I and can be edited, modified, and printed out with the aid of the same string-processing editor.[6] DARE II simulations of many *small* systems (second to sixth order) are, however, so fast that *repetitive simulation and display* at two to 20 computer runs per second is possible. Keyboard entry of parameters is then too

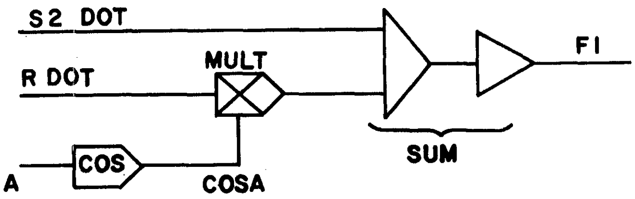

Figure 2—A block diagram

slow for CRT demonstration of parameter-change effects, and a "diddle knob" or joystick permitting rapid changes of a keyboard-addressed parameter will be added. The knob or joystick will control incrementation of an up-down counter holding the parameter value.

DARE II software incorporates substantial improvements over the DAS system. Block-macros may be typed in any order. An *optimizing precompiler* sorts statements like those in our example (1) before assembly, so that each block of the sorted program can operate on already computed quantities:

```
COS     COSA, A
MULT    SIDOT, COSA, RDOT          (2)
SUM     F1, SIDOT, S2DOT
```

This will then permit, say, integration of the output F1. *DARE II next employs conditional assembly*[10] *to completely eliminate the assembly of code for redundant store-fetch pairs corresponding to outputs and inputs of interconnected blocks.* Thus, the first macro COS COSA, A in (2) would ordinarily *end* with

```
STORE    COSA                      (3)
```

while the second macro MULT SIDOT, COSA, RDOT would *start* with

```
FETCH    COSA                      (4)
```

DARE II automatically cancels the redundant pair of instructions (3), (4), although (3) *would* be kept if it were needed elsewhere in the program. The pair

```
STORE SIDOT, FETCH SIDOT
```

will be similarly cancelled, unless SIDOT is needed elsewhere. *The DARE II precompiler program is specifically designed to permit elimination of as many track-store pairs as reasonably possible.* In addition, conditional assembly also eliminates code for unused multi-input-summer inputs and similar unused options. *As a result, DARE II produces code which is essentially as efficient as well-written PDP-9 machine-language code and permits relatively very fast execution* (Table II). If core storage is scarce, DARE II block macros can be subroutine calls to save core at the expense of some computing time.

Although the basic PDP-9 instruction set is quite limited (no byte manipulation, spare registers, or add-

TABLE II—Estimated computation times for a typical aerospace-vehicle simulation
(TIMES are in μsec except as noted)

OPERATION	NUMBER REQUIRED	DARE I PDP-9/FORTRAN (Floating-point)	DARE II PDP-9/Macro-assembler (Fixed-point)	DARE III/IV CDC 6400 (Floating-point)	197X System (Floating-point)
X + Y + Z	100	×1000 = 100,000	×5 = 500	×3.4 = 340	×0.2 = 20
XY	80	×700 = 56,000	×24 = 1920	×7 = 280	×1.2 = 96
AX	60	×700 = 42,000	×21 = 1260	×7 = 420	×1.2 = 72
F(X)	8	×4000 = 32,000	×52 = 416	×80 = 640	×10 = 80
SINX or COS X	10	×600 = 60,000	×60 = 6002	×100 = 1000	×15 = 150
TOTAL—ONE DERIVATIVE EVALUATION	→	290 msec	4.7 msec	2.7 msec	0.46 msec
Two Derivative Evaluations	→	580 msec	9.4 msec	5.4 msec	0.9 msec
RK2 Integration	12	×3000 = 36,000	×120 = 1440	×25 = 300	×4 = 48
Total Frame Time DT	→	616 msec	11 msec	5.7 msec	1.4 msec
Max. Frequency at 25 Frames/cycle	→	0.07 Hz	4 Hz	7 Hz	30 Hz

into-memory), many analog-computer blocks can be emulated quite nicely. As an example, a single-variable function with 256 uniformly spaced breakpoints can be formed by table lookup *and interpolation* in 50 μsec, and a two-variable function with 16 × 16 breakpoints can be formed in 120 μsec.[9] It is also readily possible to add to the DARE II macro-block repertoire; one can, for instance, create blocks which precisely correspond to the computing elements of any given analog computer.

Like DARE I, DARE II offers a choice of integration routines. Because PDP-9 lacks true index registers, the second-order Runge-Kutta routine[4]

$$\begin{aligned} {}^{k+1}X &= {}^{k}X + \tfrac{1}{2}(K_1 + K_2) \\ K_1 &= DT\,F({}^{k}X, k\,DT) \\ K_2 &= DT\,F[{}^{k}X + K_1, (k+1)\,DT] \end{aligned} \tag{5}$$

is probably the most useful, although it requires two evaluations of the derivative F(X, T) at each integration step. To implement Eq. (5), our program does not first evaluate all n K_1's and then proceed to add half of each to its ${}^{k}X$, as might be done with a real index register. The program instead computes each ${}^{k}X + \frac{1}{2} K_1$ and ${}^{k}X + K_1$ before the next K_1 is evaluated. When this is finished for all X, the program sets a tally switch to mark the second part of the Runge-Kutta routine, increments the independent variable, and uses the ${}^{k}X + K_1$ to produce the K_2 and the ${}^{k+1}X$ as each derivative is computed. All integrand accumulation is done in double precision to reduce roundoff-error effects.

With suitable interrupts from a real-time clock, a DARE II simulation could be readily linked to a hybrid-computer setup and/or to real system hardware (autopilot, operator positions). Note, in this connection, that the macro-assembler system would circumvent the reentrancy problems usually encountered in attempts to service multiple system interrupts with FORTRAN programs.[3]

A look into the future: DARE III and DARE IV

The DARE I and DARE II systems are expected to be completed in 1969. A useful and readily feasible next step could employ a modern 24 to 36 bit machine somewhat larger than our PDP-9 (e.g., SEL 840B, SDS Sigma 5, DEC PDP-10) to speed DARE I execution, or to add floating-point capability to DARE II. Such a system would cost between $120,000 and $200,000, which still matches the cost of a comparable analog-hybrid computer. Far more interesting from the point of view of economy as well as computing speed, however, is the possibility of *time-sharing* a substantially larger central digital computer, such as a CDC 6400. In fact, economical operation of even a medium-sized digital machine mainly intended for simulation should provide for time sharing with a "background" batch-processing program.

Our proposals for follow-on projects, then, envisage implementation of DARE I- and DARE II-like simulation systems with the University's CDC 6400, using the existing PDP-9/console combination as a remote user's station. 6400 activity would be restricted to very fast and efficient compiling and execution of differential-equation-solving programs, while the string-processing CRT editor, data entry and display, and also some iterative and statistics routines in slow simulations, would be performed by the small processor associated with the user's console. It is interesting to note that the simulation programs and data sent *to* the central computer involve only character strings transmitted at type-in rates. Alphanumerical data *from* the central computer do not require much higher rates; extensive numerical tables could be line-printed at the central installation. Each DARE CRT display, which is refreshed by the console processor, involves at most 2400 9-bit data samples. For typical 10 sec flight simulations, this would require transmission of 21,600 bits every 10 sec, or less than 2500 bits/second, so that a telephone line would do. Such operation is thus ideally suitable for remote time-sharing, provided that the 10-second-plus-overhead computer runs can be made available without excessive delays.

Based on initial DARE II experience, smaller simulation problems would be solved much more rapidly, say in 0.1 sec of central-processor time. Repetitive console displays demonstrating parameter-change effects would *not* be possible with reasonable data-transmission rates (nor would many such demonstrations be economically feasible)! Our proposed time-sharing scheme is, however, ideally suited to *fast iterative simulation or statistics-taking by the central processor* In this type of operation, only successive criterion-function values, accumulated statistics, or similar *numbers*, need to be transmitted and displayed during the iteration runs, and low transmission rates would again suffice.

In a console simulation system specifically designed for remote time sharing, our PDP-9 is really unnecessarily elaborate and could very effectively be replaced by the less costly 8K PDP-15, with DECtape but without extended arithmetic. Such a system, including very reasonable display facilities, would cost well under $50,000. An even less expensive system could be readily based on an even smaller 12- to 16-bit computer. This would save another $10,000; but the 18-bit word length of the PDP-15 is especially efficient for display-refreshing purposes and adds to the stand-alone capabilities of the console. Note, in this connection, that our own PDP-9-based console could employ DARE I for complete problem debugging before ever using CDC 6400 time.

With the large central computer and its relatively efficient compiler available, the proposed DARE III and DARE IV systems corresponding the the FORTRAN-based DARE I and the assembler-based DARE II, may well merge into each other. The multiple indexing needed for efficient implementation of integration routines may well be done best by the CDC 6400 FORTRAN compiler, while derivative computations would probably still be executed more efficiently by an assembler-based system employing conditional assembly, as in the DARE II scheme.

Digital vs. analog/hybrid simulation: *Computing-speed considerations*

Table II lists detailed estimates for various digital computation times required in a typical medium-sized aerospace simulation. Our example involves 12 state-variable-derivative integrations, 100 three-term additions, 140 products, and 18 functions of one variable. The DARE I and DARE II systems are implemented on a Digital Equipment Corporation PDP-9 (one μsec cycle time). This machine was chosen because it has an 18-bit rather than a 16-bit word length, although some of the newer 16-bit machines have much better instruction sets. The PDP-9 FORTRAN compiler appears to be designed mainly to save core storage and produces relatively very slow execution. At a reasonably conservative 25 frames (time increments DT) per period, the resulting 616-msec frame time for our aerospace simulation would permit the DARE I system to produce sinusoidal oscillations at 0.07 Hz. Speedwise, we see that the only differential analyzer our DARE I system replaces is an old-fashioned Bush or General Electric mechanical differential analyzer!

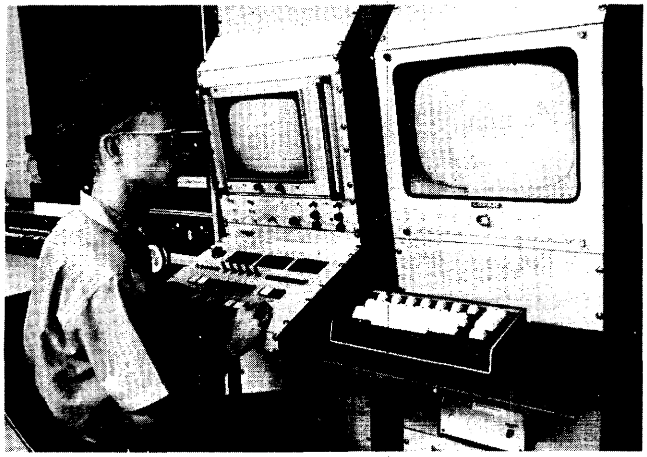
Figure 3—DARE console in operation with the PDP-9

A notable and inexpensive improvement in this situation is afforded by the fact that several PDP-9-sized digital computers are already available with hardware floating-point arithmetic. No such option is available with the PDP-9, but we ourselves have designed a current-mode logic, floating-point arithmetic unit for the PDP-9 which, if and when installed, would yield a speed improvement by a factor of at least 15 for the DARE I system, so that our simulated aerospace vehicle could wiggle at about 1 Hz, floating-point.

Our block-oriented DARE II system, also running on the PDP-9, was specifically designed to demonstrate relatively high-speed, real-time flight simulation on the inexpensive computer. The price paid for this is fixed-point operation, *but DARE II's efficient execution and 11-msec frame time permits about 4 Hz in the aerospace-simulation example.*

An improved 18- to 24-bit stand-alone computer of the future could probably produce comparable *floating-point* simulation at 4 Hz. As we have noted, though, the DARE III/IV systems will implement the economically much more important goal of time-shared operation with a large central digital computer, in this case the CDC 6400. As we have seen, very efficient and still relatively machine-independent execution will be obtained by FORTRAN integration and macro-assembler implementation of derivative computations, although many operators may prefer an entirely equation-oriented approach. In either case, *Table I indicates estimated frame times of the order of 5.7 msec, thus permitting about 7 Hz operation at 25 frames per cycle. Note that this system would provide floating-point aerospace-vehicle simulation in real time.*

The last column of Table II extrapolates the DARE III/IV system to a hypothetical 1970X digital computer permitting an approximately fivefold increase in computing speed through faster hardware and/or multiprocessing, instruction look-ahead, or hard-wired subroutines. *This is in no sense a way-out extrapolation, since digital-computer projects now on the drawing boards already plan for a fifty-fold speed increase.* Probably the most time to be gained in simulation calculations would be through the availability of fast scratchpad memories or multiple registers, which would permit derivative computations with as few core-memory references as possible; this will already be approximated in the assembler version of our CDC-6400 simulation program. Additional computing bandwidth would readily be obtained with computer systems employing parallel multiple processors, which would fit nicely into differential-equation solving schemes. Note, however, that no manufacturer of large digital computers would even consider a special design for continuous system simulation, so that all improvements must make, as it were, incidental usage of developments in large-scale scientific and business computers.

Let us now consider the computing-speed situation on the analog/hybrid computer side. One or two analog computers available for sale in 1970 will offer not only 0.02 percent of half-scale static accuracy, but also 0.1 percent of half-scale error in linear computations at frequencies up to 1 KHz; multiplication and function generation are somewhat less accurate. In applications where such component accuracies suffice, even existing analog computers are thus seen to have a 20:1 speed advantage over the fastest digital-simulation systems. *This bandwidth advantage is moreover, not likely to decrease within the next ten years;* since 1965, improved ± 10-V hybrid computers developed in our laboratory have operated with errors below 0.2 percent for linear and one percent for nonlinear operations up to *10 KHz*, at perfectly reasonable cost.[11,12]

Digital versus analog/hybrid: Economics

Our DARE system is implemented on about $90,000 worth of PDP-9 and simulation console; another $25,000 could be very advantageously spent on a disk to speed compilation. When implementing the fixed-point DARE II language, our stand-alone system is roughly comparable to a modest 150-amplifier hybrid computer of 1960 vintage, say, an Electronic Associates 231-R together with a small digital computer used for potentiometer setup, static checking, and some function generation.

At a more or less comparable price, the on-line digital system is incomparably more convenient to program, check out, and operate (this is, of course, doubly true of the floating-point system). We also have, of course, all the possibilities of the 16K PDP-9 with dual display and can produce floating-point check solutions with DARE I.

Our PDP-9 installation is, however, mainly intended as a demonstration. A more useful stand-alone installation, based perhaps on the SDS Sigma 5, would roughly double our cost, but would permit real-time floating-point flight simulation, plus some foreground-background time sharing. Although such a system would be economically competitive with a 1970 analog/hybrid computer in many applications, *the full economic potential of on-line digital simulation will be realized only in a time-sharing system.* The tremendous advantage of the time-sharing system is, simply, that the central processor is free for other business while the simulation user looks at his console-refreshed display, or simply scratches himself. We have already seen that the communication requirements for time-shared simulation are quite small.

I believe that the foregoing considerations clearly indicate the area of future analog/hybrid vs. digital simulation competition. *In applications where analog/hybrid and digital simulation systems compete at equal computing speeds, i.e., in most real-time or "slow" simulation, the new digital systems will win overwhelmingly both on economic and on human-engineering grounds.* Since, on the other hand, reasonably complex nonlinear digital simulations will not be able to run at frequencies much in excess of 100 Hz, *faster simulation will still belong on analog/hybrid computers.*

A crucial question confronting the simulation community (and specifically the analog-computer industry) is, then, this: *where, and how large, are the application areas of really fast analog/hybrid computation?* The most immediately important would seem to be:

1. *Parameter and functional optimization,* including trajectory optimization.
2. *Random-process simulation,* including optimization of statistics, communication-system simulation, and parameter-tolerance studies.
3. *Solution of partial differential equations,* including techniques requiring multiplexing of analog computing elements.

It is in precisely these applications that the very large number of computer runs needed may give the analog/hybrid computer a measure of economic advantage even over digital batch processing. Even here, only important and frequent applications could tilt the balance away from time-shared digital simulation, which saves much analog-computer scaling, setup, checkout, and "head-scratching" time, not to speak of computer amortization. Cost estimates for different simulation methods sometimes omit these "hidden" costs.

I wonder, finally, how much practical high-speed analog/hybrid computation is really done in the aerospace, chemical and nuclear-energy industries, which are, at this time, the principal consumers of continuous-system simulation. Our own laboratory's work on the design and applications of very fast analog/hybrid computers,[11,12] for instance, has always elicited much polite interest, but very little imitation. By contrast, much current aerospace work involves "slow" or real-time hybrid simulation of aerospace systems, with the digital computer doing housekeeping functions such as static checking, plus function generation and, perhaps, some accurate trajectory integration. The resulting accuracy and software problems combine all the *worst* features of both analog and digital computation; the main reason for employing hybrid simulation at all is either the existence of actual hardware in the loop or some 20- to 50-Hz components due to hydraulic servos and/or aeroelasticity. This type of hybrid simulation can be swallowed by future on-line digital systems like Jonah by the whale. For the 1970s, the simulation community would be well advised to include on-time digital simulation in its planning, together with some careful reconsideration of faster analog/hybrid techniques.

ACKNOWLEDGMENTS

The writer is grateful to the National Science Foundation for supporting Project DARE under NSF Grant GK 1860, and to Dr. R. Mattson, Head, Electrical Engineering Department, The University of Arizona for contributing University facilities. Project DARE software and hardware are being developed by a group of graduate students in the Electrical Engineering Department, including H. M. Aus, D. Chinnock, J. Goltz, T. Liebert, J. Puls, and A. Trevor. Professor J. V. Wait is co-principal investigator.

REFERENCES

1 SCI SOFTWARE COMMITTEE
The SCi continuous-system simulation language
Simulation Dec 1967

2 R D BRENNAN R N LINEBARGER
A survey of digital simulation
Simulation Dec 1964

3 B JOHNSON
Real-time digital simulation
Proc IBM Symposium on Digital Simulation 1964

4 P R BENYON
Review of numerical methods for digital simulation
Simulation Nov 1968

5 J GOLTZ
The DARE I on-line continuous-system simulation system
ACL Memo 169 Electrical Engineering Dept
The Univ of Ariz 1969

6 *A PDP-9/Cathode-ray-typewriter editor*
ACL Memo 164 Electrical Engineering Dept The Univ of Ariz 1968

7 G A KORN *et al*
A new graphic display/plotter for small digital computers
Proc SJCC 1969

8 A TREVOR J V WAIT
DIFFE: An on-line differential-equation solving routine with automatically scaled display
ACL MEMO 153 Electrical Engineering Dept the Univ of Ariz 1968

9 H M AUS G A KORN
Table-lookup/interpolation function generation for fixed-point digital computations
IEEETEC August 1969

10 M D McILROY
Macro-instruction extensions of compiler languages
C ACM April 1960

11 G A KORN
Progress of analog/hybrid computation
Proc IEEE Dec 1966

12 B K CONANT
A new solid-state iterative differential analyzer making maximum use of intergrated circuits, Proc. FJCC *1968.*