Time-sharing of hybrid computers using electronic patching

by ROBERT M. HOWE
University of Michigan
RICHARD A. MORAN
THOMAS D. BERGE
Applied Dynamics
Ann Arbor, Michigan 48106

INTRODUCTION

Ever since the introduction of patchboards for allowing storage of analog-computer programs, the desirability of having a remotely controlled switch matrix to replace the analog patchboard has been evident. Only recently, however, has automatic patching received widespread interest and study *1, 2, 3*. One reason for this is the current widespread availability of hybrid computer systems, with the result that the automatic patching program can be stored and implemented through the general-purpose digital computer. Not only have hybrid computers made automatic patching of the analog subsystem more feasible, but also they have emphasized the desirability of having automatic patching.

Additional technological developments have made automatic patching feasible. The availability of low-cost, high-performance solid-state switches for implementing the necessary switching matrices is very important. Also, low-cost integrated circuit chips can be used to provide storage of switch settings. Finally, the availability and widespread use of digitally set coefficient devices for the analog subsystem allows high-speed setup of all the parameter values as well as component interconnections. This in turn allows complete problem reprogramming within milliseconds, which means that time-sharing of a single hybrid computer through remote stations is practical. The resulting increase in computer cost-effectiveness far exceeds the extra cost of the hardware and software necessary to implement automatic patching of the analog subsystem.

In the paragraphs to follow we will describe the details of an automatic-patching system for the AD/FOUR analog/hybrid computer shown in Figure 1. As of the writing of this paper more than 60 AD/FOUR computer systems are operating in the field, many of them interfaced with general-purpose digital computers. Because the design of this computer reduces the number of switch elements needed to mechanize an efficient automatic-patching system, it was decided to add this capability to the existing system as opposed to designing a completely new system from scratch. This has the advantage that AD/FOUR systems in the field can be updated to include automatic patching.

The next section describes the configuration of the automatic patching system. Following sections present an example application, discuss diagnostic considerations, and summarize the system capabilities when operating in a time-shared mode using remote terminals.

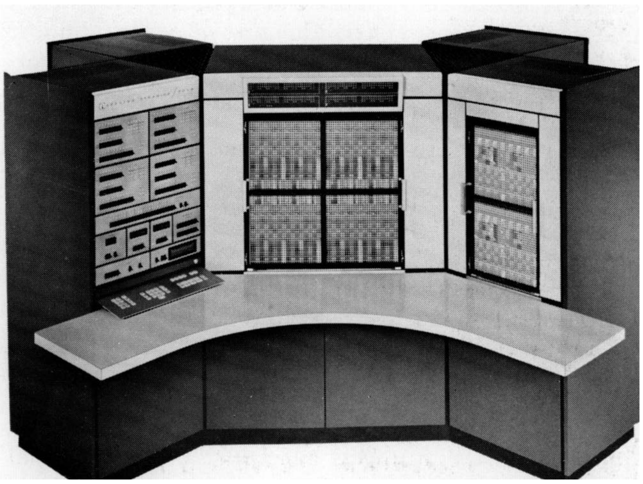
Figure 1 - AD/FOUR computer system

DESCRIPTION OF THE SYSTEM

In the AD/FOUR conventional analog programming is achieved using patchcords for interconnections between holes in the large removable patchboard in the center of the console shown in Figure 1. This patchboard is divided into four quadrants (hereafter called fields), with a matrix of $32 \times 30 = 960$ holes in each field. The fields are numbered 0, 1, 2, and 3, as shown in Figure 2, and the fields are normally filled with computing components in the order 2, 3, 0, 1. Thus field 1, in the upper right corner of the patchboard, is the last to be filled with analog components, and very few of the existing systems are expanded into field 1. For this reason it was decided to terminate the electronic-switch matrices for automatic programming in field 1. These are then patched to the appropriate holes in fields 0, 2, and 3 to mechanize the automatic patching. Thus a single analog patchboard is permanently wired with the automatic program configuration. Note that this patchboard can be replaced by a conventionally wired patchboard at any time when it is desired to operate the AD/FOUR in the normal patchboard mode.

As can be seen in Figure 1, there is a second patchboard on the right side of the console. This is just half the size of the central analog patchboard; it terminates the logic-level signals associated with the hybrid and digital computing elements in the computer. No attempt has been made to implement general-purpose automatic patching of the logic elements. Instead, it is proposed to accomplish all logic functions in the digital subsystem of the hybrid computer. The logic patchboard is wired to implement the necessary control and timing functions when the computer is operating in the automatic patching mode.

If completely flexible automatic patching were needed, then it would be necessary to be able to connect every analog element to every other element. Since this would require a prohibitively large number of switches, one invariably divides the analog system into modules when mechanizing an automatic patching system. Flexible interconnection of analog components within each module is then provided, along with programmable interconnections between modules. In the subject computer the module size for automatic patching is an analog field, which includes the following component count:

16 bipolar summer integrators

12 multipliers, each with a bipolar input buffer amplifier

64 digital coefficient units (DCU's)

In addition, each of the 16 summer integrators and 12 multiplier buffer amplifiers has a digital reference unit (28 in all). Function generation is performed using DCU's updated from the digital computer, with special dual comparators used to provide the necessary interrupts for DCU updating.

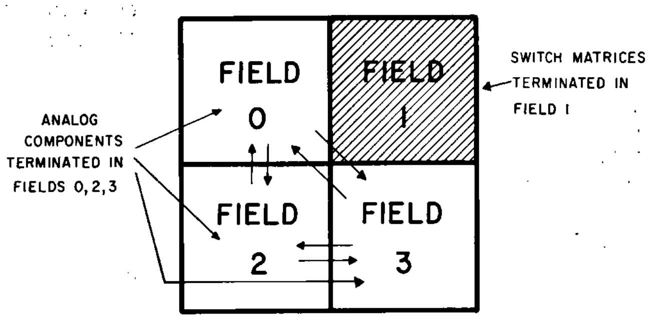
FIELD 0
FIELD 1
FIELD 2
FIELD 3
ANALOG COMPONENTS TERMINATED IN FIELDS 0,2,3
SWITCH MATRICES TERMINATED IN FIELD I

Figure 2 – Analog Patchboard of AD/FOUR computer

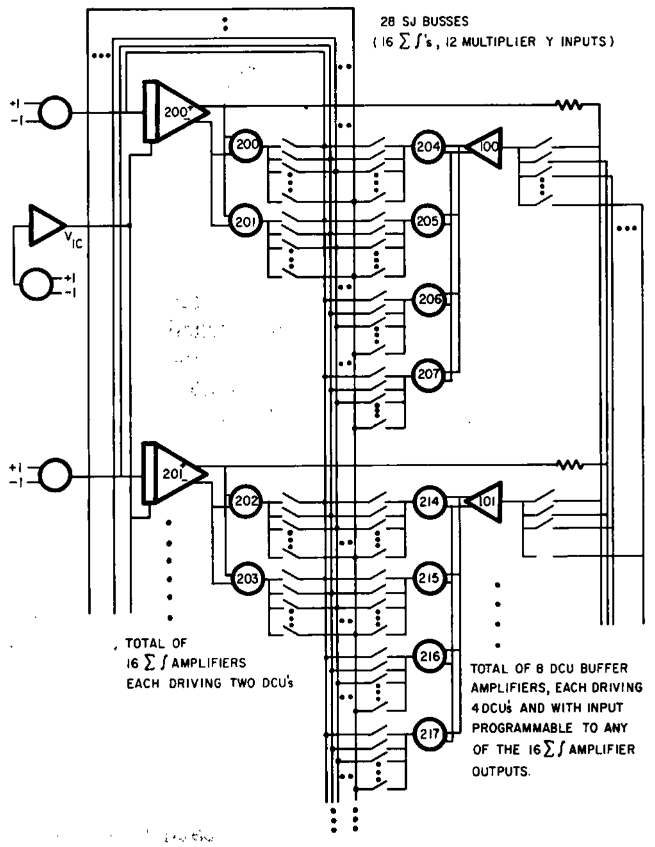
28 SJ BUSSES
(16 Σ∫'s, 12 MULTIPLIER Y INPUTS)
200
201
204
205
206
207
214
215
216
217
202
203
100
101
202
V_{IC}
TOTAL OF 16 Σ∫ AMPLIFIERS EACH DRIVING TWO DCU's
TOTAL OF 8 DCU BUFFER AMPLIFIERS, EACH DRIVING 4 DCU's AND WITH INPUT PROGRAMMABLE TO ANY OF THE 16Σ∫ AMPLIFIER OUTPUTS.

Figure 3 – Circuit schematic for DCU switch matrix

Figure 3 shows the circuit schematic within a given field for the summer-integrator and DCU switch matrix. Each bipolar output of the summer-integrators in the field is permanently patched to a pair of DCU's, utilizing $16 \times 2 = 32$ DCU's (amplifiers 200, 201, etc. in Figure 3). Each output of an additional 8 bipolar amplifiers (100, 101, etc. in Figure 3) is patched permanently to four DCU's, utilizing $8 \times 4 = 32$ DCU's. The input to each of these 8 amplifiers can be switched to any of the 16 summer-integrator outputs. Thus all summer integrators drive a minimum of 2 DCU's, and can be programmed additionally to drive a total of 6, 10, 14, etc., DCU's.

The output of each of the 64 DCU's in the computer in Figure 3 can be programmed to any of the 16 summer-integrator input summing junctions, or to any of the 12 multiplier Y-input buffer-amplifier summing junctions (see Figure 4). Thus a 64×28 summing-junction switch matrix for DCU outputs is represented by the circuit in Figure 3, as well as a 16×8 switch matrix for the DCU buffer amplifier inputs. Each DCU is the standard 15-bit AD/FOUR single-buffered MDAC, with the most significant bit providing polarity. Thus a given DCU can be programmed between a gain of −1.6384 and +1.6383 using two's complement logic and a minimum bit size of 0.0001. Since the DCU is a summing-junction output device, summing-junction switching can be used throughout the switch matrix.

A digital reference unit (DRU) is permanently patched to each of the 16 summer-integrator amplifiers, allowing a programmable bias. Also, the output, VIC, of an initial-condition amplifier is patched to all of the 16 integrator-summer input terminals. This allows initial conditions for the entire field to be set sequentially using the single DRU which drives the initial condition amplifier. The procedure starts with all integrators returned to the IC (Initial Condition) mode with the DRU set at zero (*i.e.*, VIC=0). This puts zero initial conditions on all integrators, which are then all switched to HOLD. Following this, each integrator is switched to IC sequentially with the DRU programmed to yield the desired initial condition for that particular integrator. The integrator is then returned to HOLD and the next integrator switched to IC, etc. After all initial conditions have been established in this manner, the integrators can all be switched to Operate.

Figure 4 shows the schematic for the connections to the 12 multipliers within a given field. The bipolar *X* inputs for each multiplier are permanently wired to specific summer-integrator outputs. The bipolar *Y* inputs are each programmed to the outputs of the 12 bipolar summers in the field. The summing junctions of these amplifiers are in turn switchable to the outputs of any of the 64 DCU's in the field (see Figure 3). In addition, there is a DRU permanently patched to the input of each of the 12 *Y* input buffer amplifiers, allowing a programmable bias into each multiplier *Y* input.

Extensive study of typical analog programs has shown that the *X* inputs to multipliers are seldom all independent. For this reason the multiplier *X* inputs are assigned to summer-integrator outputs using the configuration shown in Figure 5. The first four multipliers are assigned, respectively, to the first four summer-integrators. The fifth multiplier (no. 221) has its *X* input assigned to the fourth amplifier (no. 211), and the sixth multiplier (no. 222) has its bipolar *X* input patched to the common terminal of a 3-position, double-pole relay (nos. 220, 221). The position of this relay, part of the standard AD/FOUR equipment complement, is controlled by registers set from the digital computer. In this way the *X* input to multiplier 222 can be programmed to amplifier 210, 211, or 222. Thus the configuration of multiplier *X*

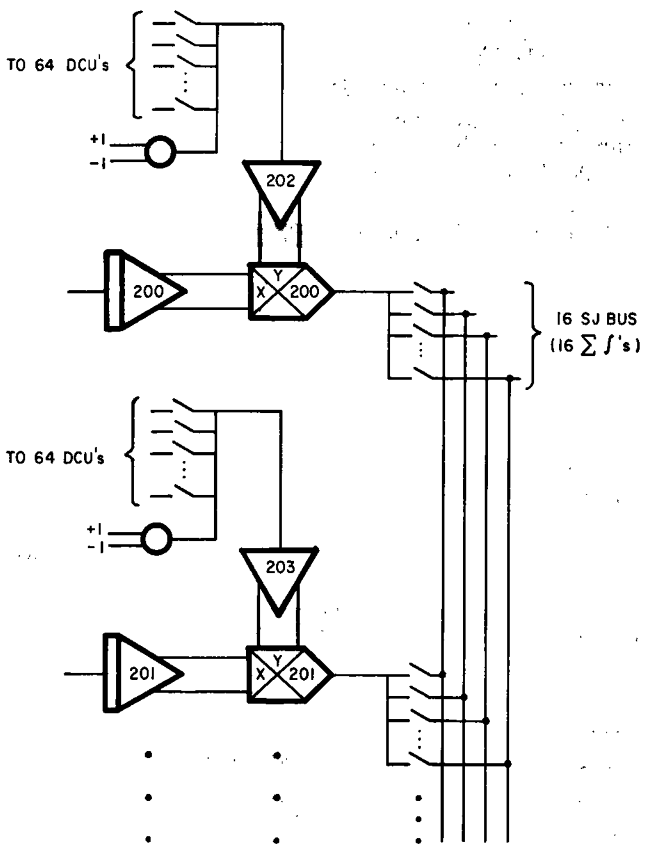

Figure 4 – Circuit schematic showing multiplier switch matrix

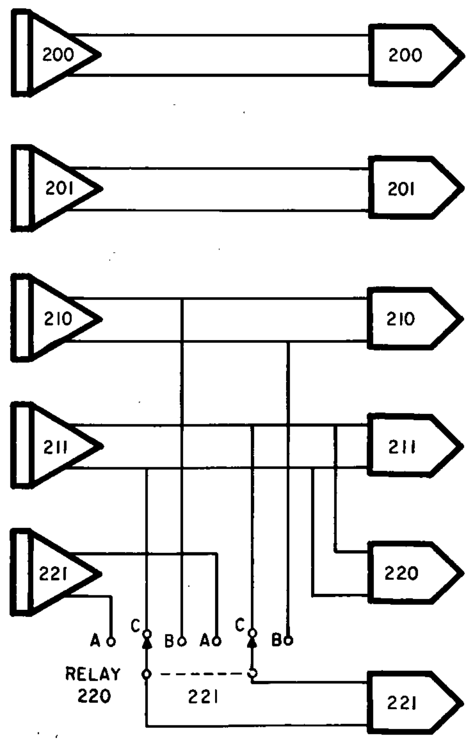

Figure 5 – Schematic showing assignment of amplifier outputs to multiplier X inputs in one-half of field 2

inputs assigned, respectively, to summer integrators in half a field can be programmed to be 1, 1, 1, 2, 1; 1, 1, 1, 3; or 1, 1, 2, 2. Later examples show the utility of this scheme.

Four hard feedback-limiters are permanently programmed, respectively, around four of the 16 summer-integrators in each field in the automatic patching system. DRU (digital reference unit) outputs are permanently programmed in pairs to each hard limiter to allow digitally controlled setup of the + and − limits. In the subject computer each summer-integrator is normally in the summer configuration. By grounding, respectively, either one of two patchboard holes associated with each summer integrator, the amplifier can be changed to an integrator or to a high-gain configuration. In the automatic-patching system these amplifier configuration holes in each field are patched to corresponding holes in field 1. These field-1 holes, under register control from the digital computer, provide program-controlled electronic grounds. Thus any of the 16 summer-integrators in each field can be configured as summers, integrators, or high-gain amplifiers.

FUNCTION GENERATION

Generation of precision functions of one or more variables on analog computers has always presented major difficulties. In fact, one of the principle tasks of the digital subsystem in many hybrid problems has been the generation of multi-variable functions for use by the analog sybsystem. However, this approach has serious dynamic limitations. Since it is desirable to have the automatically patched analog computer operate at relatively high speed in order to permit time sharing, pure digital function generation is undesirable. Instead, a hybrid method analogous to that first proposed by Rubin *4* is utilized.

The graph in Figure 6 illustrates the function-generation method when applied to a function of one variable. Between breakpoints *xi* and *xi*+1 the function *f(x)* is represented as having intercept *ai* and slope *bi, i.e., f(x)=ai+bix.* The mechanization is achieved by terminating two DCU's in amplifier 1, as shown in the figure. Whenever *x* crosses over into a new zone, *e.g.,* between *xi*+1 and *xi*+2, the two DCU's are updated to represent *ai*+1 and *bi*+1, respectively, the intercept and slope in the new zone.

High-speed detection of the zone in which *x* is located is achieved by a special dual comparator with digitally-programmed bias inputs *xi* and *xi*+1. Whenever *x* passes outside the zone bounded by *xi* and *xi*+1, the gate shown in Figure 6 throws the analog system into HOLD. By sensing the state of comparators A and B the digital computer determines whether *x* now lies in the *i*−1 or *i*+1 zone. It then looks up the values for intercept *a* and slope *b* in the new zone and sets the corresponding DCU's terminated in amplifier 1 to the new values. It also sets the bias inputs into comparators A and B to the corresponding values for the new zone. After completing these operations the digital computer restores the analog system to the OPERATE mode.

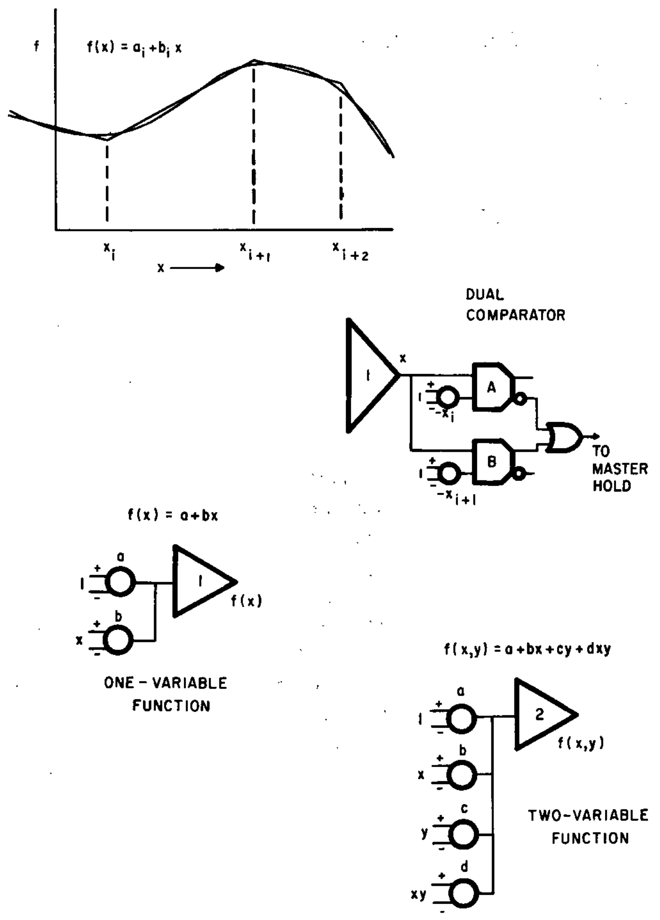

Figure 6 – Circuit schematic for function generation

Because the analog computer is in the HOLD mode while the digital computer is accomplishing the necessary DCU and DRU updatings, any dynamic errors which would otherwise result from the time required by the digital computer to accomplish the updatings are eliminated. The only significant sources of dynamic error include the lag in dual comparator response (the order of one microsecond), the delays in HOLD mode control switches (the order of one microsecond), and differential timing errors in activation of both HOLD and OPERATE mode control (less than one microsecond). Also, offsets caused by cycling the integrators back and forth between HOLD and OPERATE must be considered. In the subject computer each such mode cycle causes the order of 0.5 millivolt of additional integrator output offset for integrators programmed with a gain of 1000 (typical for high-speed problem solution). At foreseeable HOLD-OPERATE cycle rates in implementing the function generation as described here, the equivalent steady offset represents only from 0.01% to 0.1% of full scale. Even with these offsets there will be no significant effect except where open-ended integrations are involved. Study has shown that integrator drift during HOLD is completely negligible over the time required for updating DCU's and DRU's.

Generation of functions of two variables is implemented using the formula $f(x,y)=a+bx+cy+dxy$. The circuit is shown in Figure 6 using 4 DCU's terminated in amplifier 2. The DCU settings correspond to the function $f(x,y)$ for $xi\ \ x\ \ xi+1$, $yi\ \ y\ \ yi+1$. When either x or y moves into a new zone, as detected by the respective dual comparator, the 4 DCU's are updated while the analog computer is in HOLD. The resulting $f(x,y)$ analog function is equivalent to linear interpolation in x and y. A function of three variables can be generated using the formula $f(x,y,z) = a + bx + cy + dz + exy + fxz + gyz + hxyz$. As before the 8 DCU settings needed to generate the three-variable function correspond to the appropriate x, y, and z zones.

Since each summer-integrator in the automatic patching system can terminate any number of DCU's and has a permanently assigned DRU (digital reference unit), each summer integrator can be used to generate any multivariable function or any sum of multivariable functions. Reference to Figure 4 shows that each Y-input bipolar buffer amplifier for multipliers terminates any number of DCU's as well as an assigned DRU. Thus it can be used to generate the sum of multivariable functions, which in turn are multiplied by the other multiplier input, X.

In the computer's automatic patching system, 8 dual comparators, each with an assigned DRU pair as shown in Figure 6, are available in each analog field. Inputs for these 8 dual comparators can be assigned to any of the 16 summer integrators in the field, using switches driven by digitally set registers. Fixed function generators, *e.g.*, sin x, cos x, and log x function generators, can be terminated in multiplier locations in the computer. In this case the general I/O format shown in Figure 4 for multipliers is preserved. For example, the Y-input bipolar buffer amplifier is used to terminate the sin x and cos x fixed dfg (still with summing-junction output and hence output switches).

INTERFIELD PATCHING

Up to now we have only described the circuit configuration needed to patch automatically the connections within one field. It is, of course, also necessary to implement interconnections between each of the fields (three-fields maximum; see Figure 2). This is accomplished in the following two ways:

I. The first of the two DCU's permanently assigned to the output of each summer integrator (*e.g.*, DCU 200, 202, etc., in Figure 3) has its output programmable to the summing-junction input of any summer-integrator in the other two fields as well as summing junctions in its own field.

II. The input to each quad DCU buffer amplifier can be switched to any summer-integrator amplifier output in the other two fields as well as in its own field (see Figure 3).

The effectiveness of the above automatic interfield patching capability is best appreciated by studying example problems. Extension of II above to amplifier outputs trunked in from adjacent consoles provides effective interconsole trunking capability.

DIAGNOSTICS

Diagnostics, both to determine proper component functioning and to debug analog programs, is appreciably simpler with the automatic-patching system than with a conventional analog patchboard program. First of all, the patchboard on which the automatic-patching program is wired also serves as the diagnostic patchboard, *i.e.*, no special diagnostic patchboard is required. Second, because of the fixed configuration, the complete computer control of all component modes and interconnections, the presence of programmable initial conditions and bias inputs for every bipolar amplifier, etc., it is extremely straightforward to write the software for checking every analog amplifier, DCU, DRU, and nonlinear element as well as every matrix switch.

For example, proper functioning of every DCU and DCU output switch can be implemented by setting a one-machine-unit initial condition on the integrator driving each DCU, with all other summer-integrators programmed as summers. Each bit of the DCU can be checked individually by programming the DCU back to the driving-amplifier input and monitoring that amplifier input. Next the DCU can be set at, say, unity gain and its output switched successively to every amplifier summing junction (both summer-integrators and multiplier Y-input buffers). Proper matrix-switch functioning is assured by checking the respective amplifier outputs. Equally straightforward checks can be implemented for other components, including a rate test for all integrators using the programmable bias (DRU) input. The thousands of individual checks making up an overall three-field diagnostic will take only seconds, with easily-identified printout of any malfunctions.

A similar scheme is proposed for program verification, where, successively, a given initial condition (usually one machine unit) is applied to every integrator with the output (or summing-junction input) to all amplifiers measured and stored. This allows checking of the setting and input assignment of every DCU, as well as its output assignment. By programming unit X and Y inputs to each multiplier, successively, and monitoring all summer outputs and integrator inputs, multiplier output assignments can be checked. Function generator setup can be checked by observing amplifier outputs for successively programmed values of the input variable. It is believed that this type of program verification is even more powerful and easily debugged than the more conventional static check.

TIME-SHARED OPERATION WITH TERMINALS

It is estimated that complete setup time for all component configurations, switch-matrix registers, and DCU and DRU settings will be approximately 10 milliseconds for the AD/FOUR automatic-programming system. With integrators programmed at nominal gain settings of 1000, this implies solution times of perhaps 10 to 100 milliseconds for typical systems of nonlinear differential equations. Such rapid program turnaround time, in turn, suggests that it should be quite feasible and extremely cost effective to time-share a single hybrid system among a number of users stationed at remote terminals.

A relatively simple terminal system suitable for time-sharing is shown in Figure 7. This is the AD Dynamics Terminal, originally developed primarily for the educational market in order to allow individual terminal operators to time-share a single problem programmed on the AD/FOUR or AD/FIVE hybrid computer. Across the top of the front panel of the terminal are eight 3-digit-plus-sign parameter-entry thumbwheel switch sets which are assigned, respectively, to 8 problem parameters. To the right are 8 pushbuttons which control singly or in combination logic signals on the hybrid system, which in turn control problem configuration. On the lower panel are channel selector, gain, and bias controls for the x and y axes of the memory scope used for solution readout. Also on the lower panel of the terminal are the computer mode-control switches.

A number of terminals (up to 16 or more) can be connected to a single AD/FOUR hybrid computer. The computer interrogates each terminal in sequence to see whether the operator has requested a solution since the last interrogation of his terminal. If he has, the computer sets his parameter values and proceeds to generate a solution, which is stored on the memory scope and takes, perhaps, 50 milliseconds. If the operator has not requested a solution, the computer wastes no time and skips to the next terminal for interrogation. Under these conditions each terminal operator usually receives a response to his solution request in a fraction of a second, and can obtain and display multiple solutions for different parameter settings about as fast as he can reset the parameter-entry wheels and push the solution button.

When operating with this automatic programming system, the Dynamics Terminal will be used to call up various programs using the 8 logic pushbutton switches (256 combinations). Several of the pushbuttons can be used to assign the 8 parameter inputs to different groups of problem parameters. If a given terminal operator calls for a solution, his problem is programmed on the computer upon interrogation of his terminal. If the problem has been stored in core (roughly 500 words required for a typical large problem), then the program setup takes only about 10 milliseconds. The net result is an access time essentially comparable to that now enjoyed with the Dynamics Terminal System, except that each user receives his own problem when he obtains control of the computer.

When the relatively simple Dynamics Terminals as described above are used for time-sharing, initial setup and debugging of each problem must be done using the conventional hybrid I-O and not through the terminal. Obviously, it would be advantageous to have a more sophisticated terminal which also allows problem setup and debugging, in addition to problem running. This

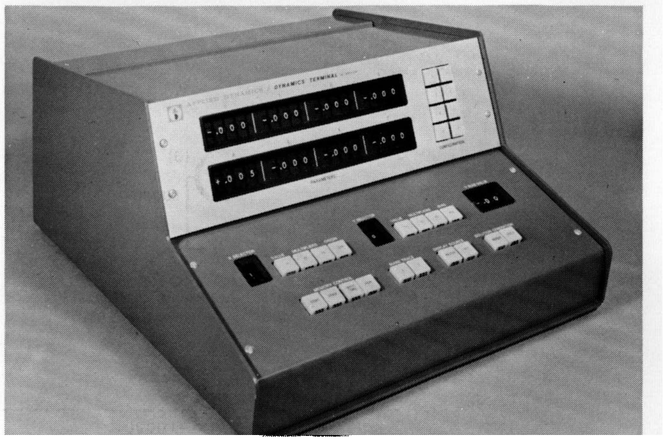
Figure 7 – Dynamics Terminal

more sophisticated terminal will probably require a keyboard, alpha-numeric display, and perhaps even a mini-digital computer. In any case, if problem setup and debugging is to be achieved through terminals while time-sharing the hybrid computer with other terminals, a very extensive and powerful software system for time-sharing will have to be available for the digital subsystem.

A SIMPLE EXAMPLE

As a simple example for the programming of a nonlinear differential equation, consider the Vander Pol equation:

$$x = (1 - x^2)x - x$$

A circuit for solving this equation using the automatic patching setup is shown in Figure 8. Since the solution is known to approach a limit cycle of amplitude 2, we have indicated that $x/2$ is computed. Thus $x = 2$ corresponds to $x/2 = 1$ (one machine unit) at the output of integrator 201. Since DCU's can be set to gains between -1.6384 and $+1.6384$, the value of the parameter , as set on DCU 212, can range up to $1.6383/4 = 0.4096$.

Although the gain (time scale) of the two integrators is under digital program control, integrator gain constants of 1000 would normally be used for a high-speed solution. Under these conditions the resulting oscillation would have a period of approximately 6 milliseconds, which means that for five cycles to be displayed the solution would take about 30 milliseconds. The problem can be rescaled to allow higher values of by simply reducing each integrator gain by the same factor. For example, if the gain of each integrator is reduced by a factor of 4, DCU's 200, 203, and 212 would be reset to 0.25, -0.25, and , respectively. Now can range up to 1.6383, and the computer solution is one-fourth as fast as before. Thus about 120 milliseconds are required for some five cycles of the solution. Or by programming the basic integrator time scales to X10,000 instead of X1000, about 12 milliseconds is required for five cycles of the solution.

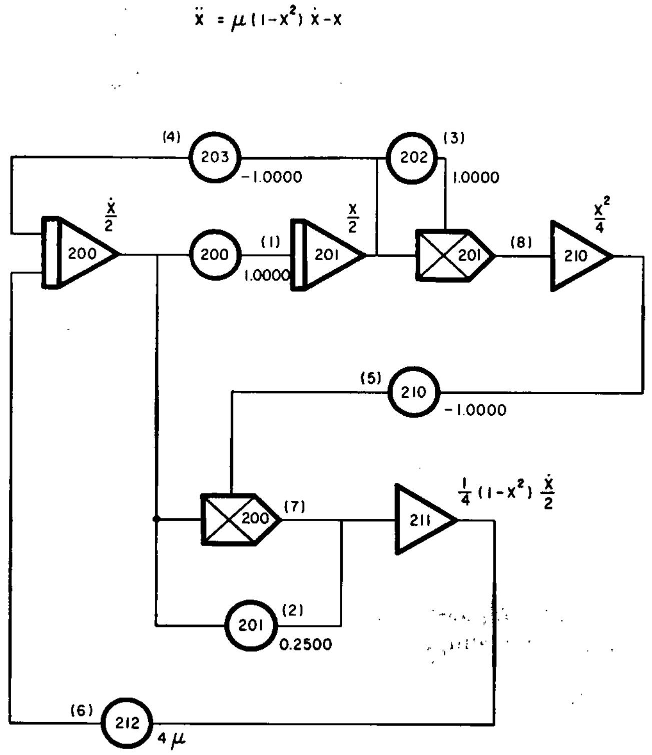

Figure 8 – Automatic patching circuit for van der pol's equation

It should be noted that the address and data format currently used for setting DCU's in the AD/FOUR is used for setting the switch registers needed to program the connections shown in Figure 8. The address indicates the device to which the switch common is connected. The data word indicates the component to which the device is connected. In the address fields 4, 6, and 7 are used instead of fields 0, 2, and 3 in order to distinguish the address of switch registers from the address of actual components, as used for readout purposes or DCU setup purposes. Hence the field address of all switch commons in Figure 8 is 6 instead of 2. The first 3 bits of the data word give the field to which the switch is to be connected, the next 3 the area, the next 3 the number. In each case the switch goes to an amplifier. Thus the following list would be required to implement the switch settings in Figure 8.

Setting			Data Word		
No.	Component	Address	Field	Area	Number
1	DCU	600	010	000	001
2	DCU	601	010	001	001
3	DCU	602	010	000	011
4	DCU	603	010	000	000
5	DCU	610	010	000	010
6	DCU	612	010	000	000
7	MULT	600	010	001	001
8	MULT	601	010	001	000

For clarity the switch setting numbers 1 thru 8 are shown in Figure 8 to allow correlation between the above table and the settings. Actually, in implementing the switch settings the digital computer merely thinks it is addressing and setting DCU's, so that existing HCR's (hybrid communication routines) can be used. Although the example in this section is very simple, it illustrates the implementation of the automatic patching scheme for solving a nonlinear differential equation. The actual switching configuration described in this paper evolved from considering the program for much larger problems, *e.g.*, nonlinear partial differential equations, six-degree-of-freedom flight equations, complex control systems, etc.

SWITCHING SYSTEM

There is a tradeoff between hardware and software in any automatic patching system. By choosing a system of only three large modules of sixteen integrators each we have enormously simplified the setup procedure, allowing a direct programming technique without the need for any fancy interpreter, *i.e.*, standard HCR's are utilized to control the interconnections. Indeed the setup procedure is so direct that there is no additional training burden in implementing this system.

The number of switches in the system is as follows:

Switches per Field:	
64 DCU's to 28 S.J.'s	1792
12 MULT S.J.'s to 16 S.J.'s	192
8 QUAD DCU AMPL to 16 Outputs	128
8 DUAL COMP to 9 AMPL	72
	2184
Interconnection between Two Fields	
(16 DCU's × 16) × 2	512
(8 QUAD DCU AMPL to 16 Outputs) × 2	256
Total Switches	768
One Field System (16 Integrators)	2184
Two Field System (32 Integrators)	2184
	2184
	768
	5136
Three Field System (48 Integrators)	2184
	2184
	2184
	768
	768
	768
	8856

These numbers compare very favorably with those of other systems previously cited. Reference 3 points out the advantage of a current-patched computer in an automatic patching system. However, the profound advantage of implementing such patching in a computer equipped with only digital coefficient devices and controlling the interconnections ("patching") with the same software which sets the coefficient devices has been overlooked.

It can be seen from the switch quantities that the system will add about 20% to the cost of the hybrid system, a figure which can be substantially recovered in the saving in patching hardware alone.

The system is flexible. Since it is implemented on a patchboard, the patchboard can be modified easily. Moreover this patchboard can be removed and the computer can be used in the traditional way. Also large numbers of consoles can be easily and economically retrofitted with this system.

Finally, the system uses modern electronic switches. This allows true hybrid time-sharing, providing the rather large software investment is made. Even without such an investment, the system will provide time-sharing of the analog console with complete repatching between successive runs.

REFERENCES:

1 STARR D A   JOHNSON J J
The design of an automatic patching system
SIMULATION June 1968 pp 281-288

2 HANNAUER G
Automatic patching for analog and hybrid computers
SIMULATION May 1969 pp 219-232

3 GRACON T J   STRAUSS J C
A decision procedure for selecting among proposed automatic analog computer patching systems
SIMULATION September 1969 pp 133-145

4 RUBIN A I
Hybrid techniques for generation of arbitrary functions
SIMULATION December 1966 pp 293-308

RICHARD A. MORAN is a graduate of the United States Naval Academy and holds an M.S. Degree in Electrical Engineering from Stanford University, majoring in Systems Techniques. For six years prior to joining Applied Dynamics in 1967, he was on the faculty of the U.S. Air Force Academy where he was Director of the Analog Computer Laboratory. He has also taught in the engineering and science area for Baylor University and the University of Colorado. His consulting activities have included several government agencies and analog computer manufacturers. Mr. Moran is Manager of Marketing Services at Applied Dynamics and is responsible for the generation of documentation for the product line, direction of training courses offered to customers, operation of the Hybrid Computation Facility, and Advertising. He is past Chairman of the Rocky Mountains Simulation Council and member of the SCI Board of Directors. Mr. Moran is a Senior Member of the I.E.E.E. and I.S.A., a Life Member of SCI and a registered professional engineer.

THOMAS D. BERGE is a graduate of the University of Michigan at Ann Arbor where he received his B.S. and M.S. Degrees in Electrical Engineering as a General Motors scholar. He is currently doing course work toward his professional EE degree. In 1966 Mr. Berge joined Applied Dynamics where he became a Product Manager in Computer Development. His circuit design experience includes hard limiters, high speed electronic switches, and digital to analog converters. Recently, he managed development of an on-line hybrid computer system to reduce the data obtained in analyzing automobile exhaust emissions. He is the project leader for the development of the electronic patching system for the AD/Four computer. Mr. Berge is a member of Tau Beta Pi, Eta Kappa Nu, Phi Kappa Phi, and the Instrument Society of America.

*Also published in *Proceedings of the Summer Computer Simulation Conference*, Denver, June 10-12, 1970. Republished with permission of copyright owners.