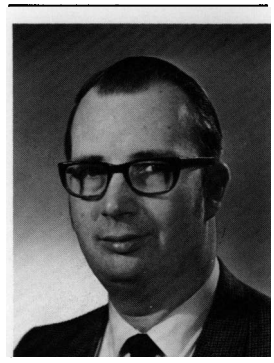
Example of function optimization via hybrid computation



by Richard A. Volz
Department of Electrical and Computer Engineering
University of Michigan
Ann Arbor, Michigan 48104

RICHARD A. VOLZ received the BS degree in electrical engineering from Northwestern University in Evanston, Illinois in 1960 and pursued his graduate studies there under a National Science Foundation fellowship, earning the MS and PhD degrees in 1961 and 1964, respectively. In 1964 he joined the staff at the University of Michigan, Ann Arbor, where he has worked on optimal discrete control, radio navigation, multipath problems in high-frequency radar systems, computer-aided design, and hybrid computation. His current interests include computational methods of optimization and hybrid computation. At the present time he is involved with two computer-aided design and simulation projects. The first is a general-purpose control system simulator based on the time-shared use of a hybrid computation system which is used in undergraduate control laboratories. The second is a digital iterative graphic computer-aided design package for control systems based around the use of a graphics terminal such as the Computek. He is currently an associate professor in the Department of Electrical and Computer Engineering and chairman of the Systems Engineering Laboratory at the University of Michigan.

ABSTRACT

Iterative techniques for function optimization have been considered extensively for use in all-digital computation. Relatively little has been done to take advantage of the much higher integration speed of hybrid computation systems. This paper demonstrates application of one simple procedure in a hybrid environment and compares the results to those obtained by an efficient digital procedure. Even though a much more efficient procedure was used on the digital, time-saving factors between 8 and 2 were obtained via the simpler hybrid implementation. Since the dollar cost of the hybrid is much less than that of the digital, the hybrid has a large advantage per solution.

INTRODUCTION

There has been a great deal of research and literature on necessary and/or sufficient conditions for function optimization problems. Yet, in many cases an explicit means of doing the calculation is unknown. Typically one must resort to iterative techniques in which one generates a sequence of trial solutions which (hopefully) converge in some sense to the true solution. A large majority of function optimization work has been done for all-digital computation. In this paper, a simple steepest-descent technique via a hybrid computation system is considered and compared with purely digital techniques.

For the class of function-optimization problems considered, each trial requires the solution of one or more sets of differential equations. On a digital computer this can be very time-consuming, particularly if the equations are nonlinear and of high order. An analog computer, on the other hand, can compute the solutions to differential equations very efficiently but is substantially lacking in the ability to perform the logical operations necessary for complicated iterative algorithms, even if equipped with an array of logic elements to make it a hybrid computer. Also, analog computers and many hybrid computers do not have adequate function-storage and playback capabilities for some of the methods one might like to consider. Use of a hybrid computation system consisting of both an analog-hybrid type computer and a separate digital computer offers the advantages of both machines. The analog would be relegated to the task of solving the differential equations, and the digital computer would be given the task of controlling program flow through the algorithm used.

This approach has been studied in various forms by several investigators. Halbert, Landauer, and Witsenhausen[1] used the maximum principle to determine the optimum control in terms of the adjoint variables and then did a systematic search on the initial conditions of the adjoint variables, using the hybrid to solve the system and adjoint equations rapidly (many times faster than real time). Steinmetz[2] made a similar use of the maximum principle to determine the optimum temperature profile of a reactor. In this case he used a high-speed analog loop to maximize the Hamiltonian.

Anderson and Gupta[3] have considered a method of converting terminal constraints to a modified criterion function and solving the entire problem, both state and adjoint equations, backward in time, again using the maximum principle. Mayback[4] also used the maximum principle to determine his control in terms of the adjoint vectors. In his case, however, he used a random search to determine the correct initial conditions on the adjoint. Miura, Tsuda, and Iwata[5] have used a somewhat different approach based upon the maximum principle in which they search out a portion of the set of obtainable states reachable with constant cost. By increasing the cost until the reachable set intersects their target, they find the optimal control.

Several of these methods require integration of both the state and adjoint equations in the same direction in time. When this is done, one or the other will generally be unstable, which can lead to severe accuracy problems. Gilbert[6] avoids this problem for linear time-optimal control problems by integrating system equations forward in time and the adjoint equations backward in time.

The above methods of solving the function-optimization problem are all indirect in that they convert the problem from the original optimization problem to a secondary problem, such as a two-point boundary-value problem or a problem of determining reachable states. It is also possible to approach the problem by a direct method in which the control function is varied directly by the iterative algorithm. Gradient methods of this nature have been successfully used by Bryson and Denham[7] and others with a digital computer. While hybrid use of such techniques has been suggested by Bekey and Karplus[8] and Howe,[9] to the knowledge of the author implementation in a hybrid environment has not, until now, been reported.

A similar direct-descent process in which the gradient is obtained by re-solving the system differential equations a number of times with small impulsive perturbations applied at successive points in time has been implemented by Fogarty and Howe.[10] The primary advantages of this approach are its simplicity, ease of programming, and ease of determining effects of terminal constraints. Naturally, its computation time is longer than would be anticipated if the gradient were computed via the adjoint method.

The essence of these works is that the high-speed solution to differential equations by analog computers allows substantially faster solutions to optimization problems than is possible with purely digital techniques. This paper considers this hypothesis for the method suggested by References 8 and 9 by solving a specific example and comparing the results with those obtained by the all-digital approach.

THEORY

The problem considered here is that of unconstrained function minimization. Let E^n denote n-dimensional Euclidean space, and ϕ: $E^n \to E^1$ be a $\mathcal{C}^2$ (twice continuously differentiable) function defined on it. Find a bounded piecewise-continuous control function $u^*(\cdot)$ defined on the interval $[t_0, t_f]$, taking values in E^r such that

$$J(u^*) = \phi(x)\Big|_{t=t_f} \leq J(u) \qquad (1)$$

for all bounded piecewise-continuous control functions $u(\cdot)$ taking values in E^r, where the vector $x(t)$ takes values in E^n and is constrained by the vector differential equation

$$\dot{x}(t) = f(x,u,t); \quad t_0 \leq t \leq t_f; \quad x(t_0) = x_0 \qquad (2)$$

where $f(\cdot,\cdot,\cdot)$ is a mapping from $E^n \times E^r \times E^1$ into E^n which is $\mathcal{C}^1$ with respect to all arguments. $J(\cdot)$ will be called the cost function.

To form a simple steepest-descent algorithm for this problem,[7] first consider the variation in $J(u)$ resulting from a small change in the control. Let $u(t)$ and $\delta u(t)$ be bounded piecewise-continuous controls defined on $[t_0,t_f]$. The change in the cost function resulting from a perturbation $\delta u(\cdot)$ of the control $u(\cdot)$ can readily be shown to be (for example, see p. 47 of Reference 16)

$$J(u+\delta u) - J(u) = \int_{t_0}^{t_f} \left\langle \frac{\partial H}{\partial u}(x,u,\lambda,t), \ \delta u(t) \right\rangle dt + R_1 \qquad (3)$$

where $\langle\cdot,\cdot\rangle$ is used to denote the usual inner product on E^k for any positive integer k, and $H(x,u,\lambda,t)$ and $\lambda(t)$ result from forming the Hamiltonian and adjoint system for the problem.[16] R_1 is a remainder term and the notation

$$\frac{\partial H}{\partial u} \qquad \begin{bmatrix} \frac{\partial H}{\partial u_1} \\ \cdot \\ \cdot \\ \cdot \\ \frac{\partial H}{\partial u_2} \end{bmatrix}$$

is used. That is, $\lambda(t)$ takes values in E^n, the Hamiltonian

$$H(x,u,\lambda,t) = \langle \lambda(t), f(x,u,t) \rangle \qquad (4)$$

is formed, and $\lambda(t)$ is required to satisfy the adjoint equation

$$\frac{d}{dt}\lambda_i(t) = -\frac{\partial H(x,u,\lambda,t)}{\partial x_i}; \quad i = 1,\ldots,n \qquad (5)$$

with boundary conditions

$$\lambda_i(t_f) = \frac{\partial \phi(x)}{\partial x_i}\Big|_{t=t_f}; \quad i = 1,\ldots,n \qquad (6)$$

The remainder R_1 becomes negligible as $\delta u(t)$ becomes small. Thus, for small $\delta u(t)$, Equation 3 becomes approximately

$$J(u+\delta u) - J(u) \approx \int_{t_0}^{t_f} \left\langle \frac{\partial H(x,u,\lambda,t)}{\partial u}, \ \delta u(t) \right\rangle dt \qquad (7)$$

Clearly, then a decrease in cost can be achieved (if $\frac{\partial H(x,u,\lambda,t)}{\partial u} \neq 0$) by choosing

$$\delta u(t) = -\alpha \frac{\partial H(x,u,\lambda,t)}{\partial u} \qquad (8)$$

for α small enough. $\frac{\partial H(x,u,\lambda,t)}{\partial u}$ is called the gradient.

The steepest-descent method[8,9] results from repeated use of Equation 8. Choose an initial control $u^0(t)$† and generate a sequence of controls $\{u^i(t)\}$ via the iteration

†As a matter of notation, subscripts will be used to denote components of vectors, and superscripts the elements of a sequence. Powers of a quantity are denoted by enclosing it in parentheses and then using superscripts.

$$u^{i+1}(t) = u^i(t) - \alpha^i g^i(t) = u^i(t) - \alpha^i \frac{\partial H(x^i,u^i,\lambda^i,t)}{\partial u} \tag{9}$$

where α^i is a step-size parameter chosen to be certain that a function decrease is obtained at each step, and where for convenience $[\partial H(x^i,u,\lambda^i,t)]/\partial u$ is denoted by $g^i(t)$. Evaluation of $g^i(t)$ is accomplished by just integrating the state equation (2) forward in time, using the control $u^i(t)$ [and storing the solution $x^i(t)$], computing the terminal boundary condition via Equation 6 and solving the adjoint equations (5) backwards in time. $g^i(t)$ may be evaluated and stored during this backward sweep.

This procedure results in an infinite sequence of controls. Practically, it must be stopped after some finite number of iterations. The choice of stopping criterion is based on a necessary condition for this problem,[16] that

$$\frac{\partial H(x,u,\lambda,t)}{\partial u}\Big|_{u=u^i} = 0.$$

An $\varepsilon>0$ will be chosen and the procedure stopped whenever the gradient norm becomes less than E_1, that is, when

$$||g^i||^2 = \int_{t_0}^{t_f} < \frac{\partial H(x^i,u^i,\lambda^i,t)}{\partial u}, \frac{\partial H(x^i,u^i,\lambda^i,t)}{\partial u} > dt < \varepsilon^2 \tag{10}$$

The notation $||\cdot||_1$ is used to represent the integral norm.

In summary, the algorithm is:

i) Select $u^0(t)$, $\varepsilon>0$, $k=0$, and replace $g^0(t)$ by 0. (This is merely to get started properly.)

ii) Update $u^k(t)$ via Equation 9, integrate (2) using $u^k(t)$, and store $x^k(t)$.

iii) Evaluate the boundary conditions of (6).

iv) Play back $u^k(t)$ and the stored solution to $x^k(t)$ and solve Equation 5 backwards in time, storing the new values of $g^k(t)$. During this same integration compute the LHS of inequality (10).

v) If inequality (10) is satisfied, stop. Otherwise choose α^k so that

$$J(u^i - \alpha^k g^k) < J(u^k)$$

Replace k by $k+1$ and go to step ii.

IMPLEMENTATION

An approximation to the steepest-descent algorithm of the previous section has been programmed on a hybrid computation system consisting of an AD/4 hybrid computer coupled to a PDP-9 digital computer. The approximation is the usual one resulting from the inability to store $x^k(t)$ and $u^k(t)$ for all $t\varepsilon[t_0,t_f]$. The functions are represented in the digital unit by a set of sample values, and the functions played back are only reconstructed approximations to the actual function. For closely spaced samples, however, a good approximation to the original problem results.

The implementation of the algorithm is a true hybrid program involving the simultaneous interaction of analog, logic, and digital parts of the system. The differential equations of (2) and (5) are wired on the analog patchboard. Playback of functions stored in the digital unit is handled for convenience by zero-order hold reconstruction of the sampled values. Double-buffered digital-to-analog converters (DAC's) are periodically updated with the proper values.

Since sample-and-hold amplifiers are not available with the ADC multiplexer in the hybrid system, additional general-purpose analog integrators are used to avoid time skew in converting analog signals to digital form for storage and subsequent playback. Those signals which are to be sampled are brought to the IC connection on an integrator (0.001 or 0.0001μF capacitor used as needed). Normally, these integrators are in the IC mode. Whenever samples are required, these integrators (and only these integrators) are placed in the HOLD mode. The signals can then be sequentially converted and yet have the samples all taken at effectively the same time.

Timing is handled jointly by the logic of the hybrid and the digital unit, with the hybrid counting timing pulses to determine the sample period and the digital counting sample periods to determine the total run period.

The heart of the implementation is the sample-playback subroutine which controls the solution to the differential equations. Figure 1 shows a simplified flow diagram of the digital part of the subprogram. After placing the analog section of the hybrid in the OPERATE mode to obtain a solution to one of the sets of differential equations, the program goes into a loop which is executed once for each sample period. The loop begins when the hybrid logic places a sense line in the logic 1 state, indicating the beginning of a sample period. This same logic signal also triggers the update of the DAC's from the buffer values and places the integrators being used for track-store in the HOLD mode. The next task performed is that of A-D conversion of the signals being sampled. As each A-D conversion takes approximately 40 μs on the converter used, this time is utilized, when appropriate, to update one control sample via Equation 9. At the end of a run all control samples have been updated. After the conversion, the DAC buffers are loaded in preparation for the next sample period. The program then enters a WAIT state for the next sample period.

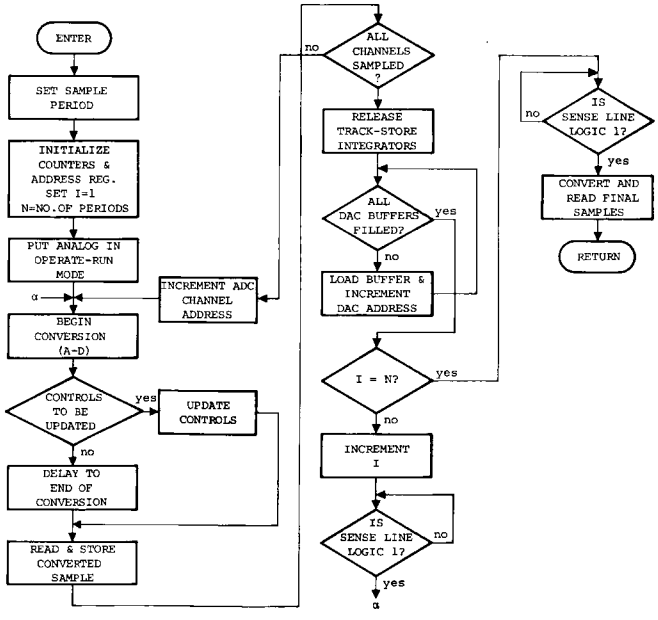


Figure 1 - Sample-playback flow diagram

The time required by this loop is critical to the time required for the hybrid solution, and is related to the dimension of both the state and control vectors. The most significant factor is the number of analog signals to be multiplexed through the A-D converter. The loop time is approximately 160 μs for sampling and playing back a single signal. Each additional signal sampled adds about 45 μs, and each function played back, about 15 μs.

The remainder of the programming for the algorithm is relatively straightforward and is carried out strictly on the digital unit. The step-size parameter, α_i, is chosen according to a modification of the Goldstein Constant step rule.[15] After every N trials an acceleration step consisting of a one-dimensional minimization search is carried out. For the example run, N was chosen empirically to be 10. The digital computer, then, controls the overall flow through the algorithm and merely calls the storage-playback program whenever a solution to either the state or adjoint variables is required.

COMPARISON OF DIGITAL AND HYBRID SOLUTIONS

A comparison of digital and hybrid solutions is a lot like comparing apples and oranges — nearly impossible. Both types of equipment can take many different configurations, and there are many different algorithms one could choose. Each combination could produce different results. Moreover, without a large number of examples, it is difficult to reach any general conclusions. Yet the claim of a speed advantage for the hybrid over the digital makes such an attempt desirable. The comparison given below is by no means exhaustive. Nevertheless, if viewed in the context of the specific equipment and algorithms used and the likely effects of variation of these, it may be of some value.

The computer used for the all-digital portion of the study was the university's 360/67 system, and the hybrid system consisted of the AD/4 and PDP-9 mentioned earlier. Choice of the algorithm to be used was not so simple. The use of the same algorithm for both systems would almost certainly mean that one or the other would not be operating efficiently. With the simple steepest-descent procedure, the digital computer is operating very inefficiently. On the other hand, many of the more efficient algorithms used on digital computers require one-dimensional searches to greater accuracies than can be obtained on a hybrid. It was decided to restrict both machines to first-order direct methods, but to allow each machine to use a method chosen to its advantage.

Recently there has been a trend toward the use of conjugate direction methods for parameter optimization. Some of these techniques have been extended to function minimization.[11,12,13] These methods generally approach the efficiency of second-order methods. Furthermore, they have the property of being stable. That is, they will not diverge as Newton's method sometimes does. The function-space version of the Davidon-Fletcher-Powell algorithm suggested by Tripathi and Narendra[12] was selected and programmed on the University of Michigan's IBM 360/67. Algorithms for use on hybrid systems are less well developed owing to equipment limitations, scaling difficulties, and the longer programming times required. Consequently, the simple steepest-descent method described above was chosen for the hybrid system.

As an example to be used in this comparison a second-order system based upon the Vanderpol equation

$$\dot{x}_1 = (1 - x_1^2 - x_2^2)x_1 - x_2 + u; \quad x_1(0) = 1 \tag{11a}$$

$$\dot{x}_2 = x_1; \quad x_2(0) = 0 \tag{11b}$$

was used. With $u(t) = 0$, $x_1(t)$ and $x_2(t)$ exhibit a stable limit cycle of amplitude 1 about the origin. The cost function selected for this problem was

$$J(u) = (1-K)x_3(t_f) + Kx_4(t_f) \tag{12}$$

where

$$\dot{x}_3 = u^2; \quad x_3(0) = 0 \tag{13a}$$

$$\dot{x}_4 = x_1^2 + x_2^2; \quad x_4(0) = 0 \tag{13b}$$

A final time of $t_f = 6$ seconds was chosen because this corresponds to roughly one cycle of the unforced oscillation. For nonzero K, the cost function used will tend to force the state towards the origin. K is a relative weighting on the importance of achieving this objective and the control energy required to do so.

As the analog signals do have some random noise present, consecutive solutions for the same value of K will neither yield precisely the same answer, nor take precisely the same time. The A-D converter used produced 15 bits plus sign, scaled so that 1 reference unit was converted to the number 10,000. The values sampled, then, were resolved to 1/10,000 of reference value. The analog solutions to the differential equations were quite repeatable for a single control. Typically, one would see only a variation in the cost function of at most 1 in the last figure read. In obtaining a solution to the problem, however, one sees not the variation in a single solution, but an accumulation of the deviations of the solution at each step of the iteration. In addition, if the path taken by the sequence $\{u^i(t)\}$ comes near a ridge in the cost surface, a small accumulated variation could result in different sequences of trials for the same K falling on different sides of a ridge for some iteration I_1. For $i>I_1$, the paths followed would be *different*, and, even though solutions were performed accurately, differing amounts of time would be required. Since the iteration is stopped after a finite length of time, the solutions would be different, though within specified tolerance. There is some evidence to suggest that this latter effect was the major contributor to the variation observed since consecutive trials for the same K often took approximately the same number of iterations, with occasional trials taking a substantially different number of iterations. To present more clearly this situation, each case was run eight or more times, and the averages and standard deviations of the results calculated. An initial control of $u(t) = 0$ was used.

The number of iterations required, of course, depends on the accuracy requested of the algorithm. It was found that the best accuracy obtainable with the hybrid system was between 0.5% and 1.0% for K in the range 0.2 to 0.7. As expected with simple gradient procedures, 30% to 50% of the time required was for the last 1% improvement in accuracy. For engineering purposes 1% - 3% would normally be sufficient. This was roughly achieved when a stopping criterion $||g||_1^2 < 0.04$ was used. For purposes of comparison both algorithms were given this same stopping criterion. The numerical integration for the digital algorithm was performed by a fourth-order Runge-Kutta procedure. The results are summarized in Table 1.

Table 1

	DIGITAL COMPUTATION			HYBRID COMPUTATION					
K	J	$\|\|g\|\|^2$	CPU time	J_{ave}	Standard dev. of J	$\|\|g\|\|_{ave}$	Average CPU time	Stand. dev. CPU time	% diff.
.20	1.03154	.0091	3.9 sec	1.0380	.0001	.0154	.5 sec	.00 sec	.69
.25	1.09731	.0166	17.5 sec	1.1391	.0421	.0252	2.3 sec	1.46 sec	4.28
.30	1.09137	.0310	22.3 sec	1.1073	.0036	.0352	2.8 sec	.33 sec	1.48
.35	1.08011	.0190	17.0 sec	1.0919	.0007	.0348	3.5 sec	.07 sec	1.40
.40	1.05430	.0066	26.9 sec	1.0674	.0051	.0316	4.3 sec	.55 sec	1.30
.45	1.02430	.0125	27.4 sec	1.0437	.0052	.0376	4.7 sec	.8 sec	1.96
.50	.98700	.0004	30.2 sec	1.0000	.0055	.0282	4.9 sec	.85 sec	1.33
.55	.94392	.0010	31.3 sec	.9657	.0011	.0384	7.6 sec	2.58 sec	2.31
.60	.89459	.0013	30.4 sec	.9159	.0024	.0360	11.4 sec	3.5 sec	2.40
.65	.83860	.0107	32.1 sec	.8608	.0040	.0392	10.3 sec	4.2 sec	2.67
.70	.77860	.0300	34.2 sec	.7987	.0046	.0464	15.4 sec	5.9 sec	3.04
.75	.70850	.0135	30.4 sec	.7177	.0052	.0492	20.1 sec	7.1 sec	1.97
.80	.62405	.0122	35.8 sec	.6406	.0041	.0836	28.2 sec	5.0 sec	2.85

The digital solutions appear to be more accurate in some cases only because the last iteration before the stopping criterion was met yielded a significant improvement. If a more stringent stopping criterion had been used, say $||g||^2 \leq 0.012$, which was found empirically to be as good as one can do with the hybrid procedure, both the digital and hybrid times would increase, but the hybrid the more significantly. Even in this case, though, the hybrid shows a significant advantage. Thus, for the example considered the simple steepest-descent hybrid approach is significantly faster than a *sophisticated complex algorithm* on the digital computer.

To put this in perspective, however, there are a number of salient points which must be considered. First, the hybrid program is actually more difficult to implement than the digital owing to the interaction taking place between the various parts of the hybrid system and the lack of adequate hybrid programming systems for the hybrid used. Secondly, there are available digital computers significantly faster than the IBM 360/67. On the other hand the limiting factors in the hybrid speed were the conversion rate of the A/D converter (only 27,000 samples per second) and the limited speed of the PDP-9. The analog solution could easily be run an order of magnitude faster. Equipped with a better converter and a higher-speed digital computer, the hybrid solution could also be several times faster. Moreover, the use of linear interpolation in reconstructing the sampled function would allow fewer samples to be used for the same accuracy. Since the speed at which the hybrid solutions are run is determined by the speed of the sample-playback loop and the number of sample periods, linear interpolation would allow a further increase in hybrid speed.

Also important to the speed of solution are the dimensions of the problem. There were only four state variables in the example given, of which two were used only to represent the terms of the cost functional. As the dimensionality of the state vector is increased, one would normally expect the hybrid to exhibit a much greater advantage because of the parallel nature of the analog computation used for solving the differential equations. To examine this hypothesis a little more closely, divide the question into two parts: (a) efficiency of the algorithm in terms of the number of iterations required and (b) the time required for each iteration, which is mostly spent in solving the differential equations. From the discussion of the implementation of the sample-playback loop, it is evident that the time per iteration for the hybrid solution to the state equation is of the form $k_1+k_2+k_3n+k_4r$, where k_1 represents the digital overhead for management of the algorithm, k_2 is the fixed-time portion of the sample-playback loop, and k_3 and k_4 represent the per-channel times for data sampling and playback. A similar expression for the time to solve the adjoint equation exists. Thus, both the dimensions of the state and of the control vector enter the per-iteration time as linear terms.

The per-iteration time required by the digital may be expected to increase more rapidly than this. For many digital integrators the time required is roughly proportional to the time required to evaluate $f(x,u,t)$ in (2). For a simple linear system $\dot{x} = Ax$, this is dominated by the number of multiplications in forming the product Ax. If A is full, the number is n^2 and the time would increase as the square of the dimension of the state vector. For most practical systems A would not be full, but for many the coupling between elements of the state vector would produce an increase in time rising more rapidly than linearly in n. Similarly, for terms linear in the control u (e.g., Bx), the time would rise as the product $n \cdot r$ if B is full. For fixed state dimension, this would be linear in r. The introduction of nonlinearities merely slows down the digital solutions more, while not affecting the per-iteration time of the hybrid at all.

It thus appears that the per-iteration time advantage for the hybrid will increase as state dimensions are increased (through not as much as might at first have been expected), while it is not clear that there would be much relative change, one way or the other, from increasing the dimension of the control vector.

If the same algorithm had been used for both of the comparisons, this would be a reasonable conclusion for the algorithm as a whole. However, this was not the case. While, except for isolated special cases, both algorithms can be expected to require an increasing number of iterations as the problem dimensions are increased, the nature of this increase is uncertain. It is highly dependent upon the structure of the particular problems being considered. It is thus difficult to speculate on the total overall effect of the dimension of the state and control vectors, except to note that unless the digital algorithm becomes relatively more efficient in terms of the number of iterations required as the state dimensions increase, the hybrid solutions are likely to show increasing advantage.

The choice of algorithms also has strong bearing on the results. The conjugate-direction algorithm used by the digital computer is more efficient in the sense that it requires far fewer iterations (evaluation of $g^1(t)$) to satisfy the stopping criterion. The hybrid advantage arises strictly from the fact that the higher integration speed outweighs this. Clearly, if there were a hybrid algorithm as efficient as the digital algorithm, the hybrid solutions would show a much improved advantage. The difficulty in this area, of course, is that accurate one-dimensional searches are required in most of the more efficient procedures. Recently, however, several rapidly convergent methods which do not require accurate one-dimensional searches for parameter optimization have been introduced, e.g., Fletcher.[14] It appears that these methods can be adapted to infinite-dimensional problems in the same manner as the Davidon-Fletcher-Powell algorithm.[12,13] If this is developed and implemented, it should produce major improvements in the hybrid performance, particularly for problems with high state dimensions.

Still another pertinent factor is cost. The hybrid system used is far less expensive than the digital system. This compounds the hybrid advantage when viewed on a cost-per-solution basis. On the other hand, program preparation and documentation are much more expensive for the hybrid solutions. Problems which only require a single solution are thus probably best done on a digital computer, while problems for which many solutions are sought (say under different operating conditions) or which must be treated in real time may best be handled by a hybrid system.

SUMMARY

This paper has illustrated the use of an oft-suggested steepest descent for function optimization on a hybrid computation system and compared the results with those obtained on a purely digital system using an efficient conjugate-direction algorithm. Perhaps the most surprising thing about the results was that the hybrid did not show a bigger speed advantage. There are a variety of factors influencing this: the necessity of interaction between the digital and analog units during solution of the differential equations, which requires the analog speed to be slowed down to match the digital speed; the consideration of only one problem with low state dimension; the speed of the analog-to-digital converter used; and the vast difference in efficiency between the algorithms used. Though faster converters and digital computers are available, it is clear that they are not yet fast enough to take full advantage of the high-speed analog solutions for function-minimization problems using storage-playback schemes.

It will depend on a given application whether the limited speed advantage is worth the reduced accuracy of the hybrid. It seems evident, however, that the development of much more efficient hybrid algorithms, such as extension of Fletcher's method, should be possible. These remain an area for future investigation and should eventually provide further advantages for the hybrid.

ACKNOWLEDGEMENT

This investigation was supported by a faculty research grant of the University of Michigan. The author is also grateful to the Simulation Center for the use of their facilities.

REFERENCES

1 HALBERT P W LANDAUER J P WITSENHAUSEN H S
Hybrid Simulation of Adaptive Path Control
SIMULATION June 1964 p R-24

2 STEINMETZ H S
Using Pontryagin's Maximum Principle
SIMULATION June 1965 pp 382-389

3 ANDERSON M D GUPTA S C
Backward Time Analog Computer Solutions of Optimum Control Problem
Proceedings AFIPS Spring Joint Computer Conference
vol 30 1967 pp 133-139

4 MAYBECK R L
Solution of Optimal Control Problems on a High-Speed Hybrid Computer
SIMULATION November 1965 pp 238-245

5 MIURA T TSUDA J IWATA J
Hybrid Computer Solution of Optimal Control Problems by the Maximum Principle
IEEE Transactions on Electronic Computers
vol EC-16 October 1967 pp 666-670

6 GILBERT E G
The Application of Hybrid Computers to the Iterative Solution of Optimal Control Problems
In *Computer Methods in Optimization Problems*
Balakrishnan and Neustadt, eds.
Academic Press New York 1969 pp 261-284

7 BRYSON A E DENHAM W F
A Steepest-Ascent Method of Solving Optimum Programming Problems
ASME Transactions ser E vol 29 1962
pp 247-257

8 BEKEY G A KARPLUS W J
Hybrid Computation
Wiley New York 1968

9 HOWE R M
Hybrid Computer Solution of Optimization Problems
Applied Dynamics report Ann Arbor Michigan
September 1969

10 FOGARTY L E HOWE R M
Trajectory Optimization by a Direct Descent Process
International Astronautical Congress Belgrade
1967

11 LASDON L S MITTER S K WARREN A D
The Conjugate Gradient Method for Optimal Control Problems
IEEE Transactions on Automatic Control vol AC-12
April 1967 pp 132-138

12 TRIPATHI S S NARENDRA K S
Optimization Using Conjugate Gradient Methods
IEEE Transactions on Automatic Control vol AC-15
no 2 April 1970 pp 268-270

13 LASDON L S
Conjugate Direction Methods for Optimal Control
IEEE Transactions on Automatic Control vol AC-15
no 2 April 1970 pp 267-268

14 FLETCHER R
A New Approach to Variable Metric Algorithms
Report no HL 69/4734 UKAEA Research Group
Atomic Energy Research Establishment Harwell

15 GOLDSTEIN A A
Constructive Real Analysis
Harper and Row New York 1967 p 28

16 BRYSON A E Jr HO Y-C
Applied Optimal Control
Ginn and Blaisdell Waltkin Massachusetts 1969