Methods of Simulating a Differential Analyzer on a Digital Computer*

F. Lesh

California Institute of Technology, Pasadena, Calif.

Introduction

One of the problems facing large computer installations today is the increasingly large ratio between problem preparation time and problem solution time. Sometimes weeks or even months may elapse between the time a problem is first presented to a programmer and the time the first production occurs. Many methods have been tried for simplifying this presentation. Compilers, formula translators, and interpretive routines have been written and tested on a wide range of problems and some of them are quite effective.

In the field of differential equations, the preparation of a problem for solution on an electronic analog computer is frequently much easier than the preparation of the same problem for solution on a digital computer. For this reason there has been some work done to simulate analog computers on digital computers. Selfridge has done much of the introductory work along these lines [1]. The system described here is an extension of Selfridge's work in that it uses a more powerful integration scheme (Runge-Kutta) and a different method of problem presentation.

The routine described in this paper called DEPI, for Differential Equations Pseudo-code Interpreter, was written to simulate a general-purpose analog computer on a digital computer in the hopes of combining the programming ease and program flexibility of the analog computer with the accuracy and reliability of the digital computer. In general, the plan was to simulate the operation of actual analog computers unless considerable simplification would result from the introduction of non-similarity.

DEPI's Component Structure

Perhaps the most obvious thing about an analog computer is that it is built of modules, each performing a distinct mathematical function. DEPI's component structure, illustrated in figure 1, corresponds generally to these respective modules.

Notice that no multipliers are included as components. This is because multiplication is very simple on a digital computer, and it was much easier to include multiplication as a part of the input function of amplifiers and integrators

* This paper presents one phase of research carried out at the Jet Propulsion Laboratory, California Institute of Technology, under Contract No. DA-04-495-Ord 18, sponsored by the Department of the Army, Ordnance Corps. It is an expanded version of a joint paper by the author and W. R. Hoover presented at the meeting of the Association, June 19–21, 1957.

Component Type	Input	Output
1. Amplifier	$\theta_1, \cdots, \theta_n$ $\phi_1, \cdots, \phi_n$	$\theta_1\phi_1 + \cdots + \theta_n\phi_n$
2. Integrator	$\theta_1, \cdots, \theta_n$ $\phi_1, \cdots, \phi_n$	$\int(\theta_1\phi_1 + \cdots + \theta_n\phi_n)\, dt$
3. Divider	θ, ϕ	θ/ϕ
4. Square Root Generator	θ	$\phi^{1/2}$
5. Resolver	θ	$\begin{cases} \sin \\ \cos \end{cases} \theta$
6. Relay	θ, ϕ	$\theta, \ \gamma < \gamma_s$ $\phi, \ \gamma \geq \gamma_s$
7. Function Generator	θ	$f(\theta)$
8. Potentiometer	none	K
9. Output	$\theta_1, \cdots, \theta_n$	Listing of the values of $\theta_1, \cdots, \theta_n$

Fig. 1. Component Structure

as shown. The φ and θ here refer to any of the variables or constants in the system. As with an analog computer, integrators are simply amplifiers whose output is the time integral of the input.

Divider and square root circuitry in an analog computer involves several amplifiers and a multiplier and would be very difficult to simulate digitally. Since these operations are so simple on a digital computer, dividers and square root generators have been included as basic components.

Resolvers, whose output is the sine or cosine of the input, are included as modules in some analog computers and they are included in DEPI also.

Relays on an analog computer are simply switches which serve to introduce program changes in the middle of a computation, and DEPI relays serve the same purpose.

Function generators on an analog computer are usually some sort of electromechanical servo device whose output is an arbitrary function of the input. Function generators are also among the components of DEPI.

Analog potentiometers multiply a variable by a constant which must be less than one. In DEPI, multiplication is handled by the amplifiers and integrators so that all that is needed is a set of constants stored in memory. Since the decimal point is considered to be after the fourth digit in DEPI operation, these constants will all be less than ten thousand.

For each type of component except potentiometers there is a special subroutine which does nothing but calculate and store the outputs for the components of that type. The output subroutine does nothing but list the desired output. All of the component subroutines operate in the interpretive mode, that is to say, they read and interpret each control word on each pass. Each individual component has a three-digit component number between 000 and 199, and a single memory cell in the computer where its output is stored.

Control Word Structure

Programming for DEPI is done directly from a flow diagram, and consists of writing a sequence of control words on a coding sheet. The structure of the control words is shown in figure 2. Notice that each control word has a *number* which appears in its first three digits. The φ and θ in the integrator and amplifier control words stand for the numbers of two components whose outputs are multiplied together to form a single input. Each integrator and amplifier can have as many control words as desired. The total input is the sum of all the designated product pairs. The φ and θ of the divider control word indicate the component numbers of the numerator and denominator respectively. The θ's in the square-root generator, resolver, and function generator control words indicate the arguments of the operations and the U indicates whether a sine or cosine is desired.

The θ in the function generator control word indicates the argument of the desired function. The Y is the address of a potentiometer which contains the relay switching constant, and the θ is the address of the input variable. The s in the output listing routine control word designates the number of digits to be ignored at the left end of a number to be listed. The b gives the number of digits to be printed before the decimal point and a the number after. The θ indicates the component number of the variable to be listed. This control word can be either positive or negative. If it is positive the sign of the variable is listed, but if it is negative, the sign is suppressed.

The flexibility of the program is obtained largely by means of the relays, as shown by figure 3. Each relay consists of a set of ten memory cells in which control words may be stored. The function of the relay subroutine is to exchange the control words stored in the relay for the ones with the same control word number in the problem code. In the case shown here, only seven of the ten

Component	Structure of Control Word				
	control word number	ϕ		θ	
1. Amplifier	+ (xxx)	(xxx)	0	(xxx)	
		ϕ		θ	
2. Integrator	+ (xxx)	(xxx)	0	(xxx)	
		ϕ		θ	
3. Divider	+ (xxx)	(xxx)	0	(xxx)	
				θ	
4. Square Root Generator	+ (xxx)	0000		(xxx)	
			u	θ	
5. Resolver	+ (xxx)	000	x	(xxx)	u = 0 cosine; u = 3 sine
				θ	
6. Function Generator	+ (xxx)	0000		(xxx)	
		y_s		θ	
7. Relay	+ (xxx)	(xxx)	0	(xxx)	
		s b	a	θ	
8. Output Listing	$\pm$ (xxx)	(x)	(xx)	(x)	(xxx)

Fig. 2. Control Word Structure

possible storage positions for the relay are being used. Notice that the memory location for any control word in the code can be found by adding 1400 to the control word number.

Integration

One of the most important choices affecting the outcome of a simulation program is whether to use a simple integration scheme and a very fine mesh or a more powerful integration scheme and a larger mesh. There are several advantages to the use of a simple integration formula. The most obvious of these is that the coding is simpler. There is also the advantage that frequently the mesh size may be taken small enough that all relays can operate at a mesh point instead of having to calculate new time increments and integrate back-

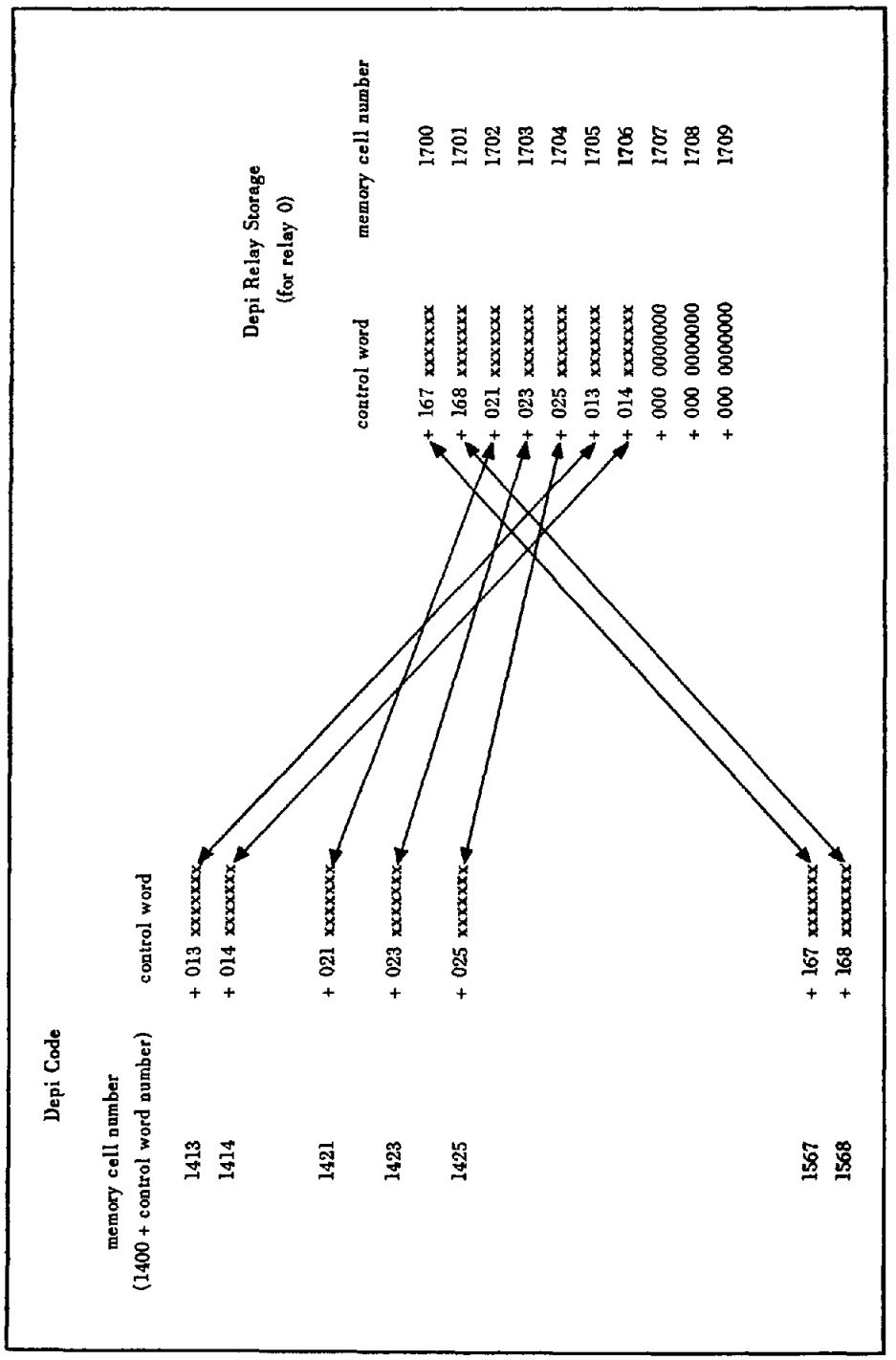

Fig. 3. Relay Operation

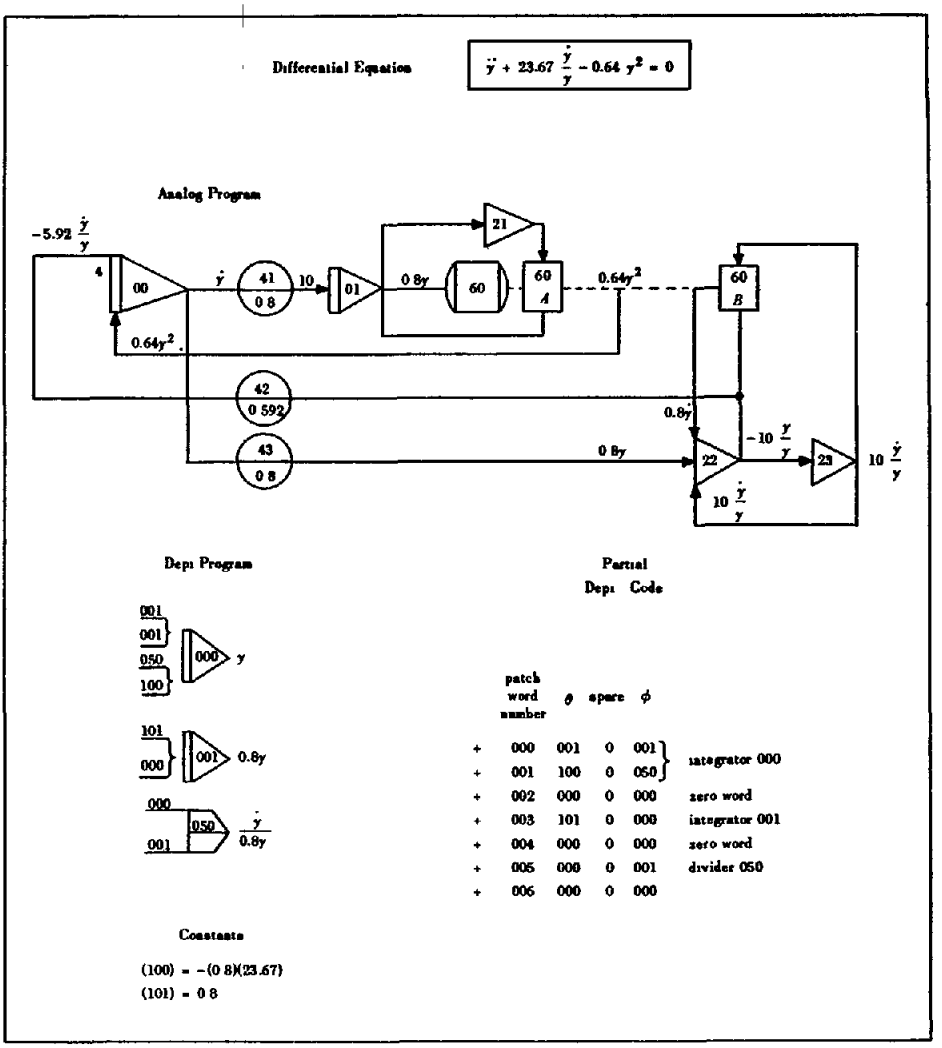

Fig. 4. Programming-Coding Comparison

ward or forward to some intermediate point. On the other hand, the truncation error for a scheme like Euler's, say, is so large that frequently the mesh size must be kept quite small to preserve accuracy. This can increase calculate time so much that the more accurate methods are actually faster. The method chosen for DEPI was fourth-order Runge-Kutta [2]. This method has high accuracy, requires no starting procedure, the interval of integration can be changed at will, and discontinuities present no special problems.

Programming and Coding for DEPI

A simple comparison between analog and DEPI programming is shown on figure 4. Notice that because of the ease with which multiplication and division

can be performed on a digital computer, the DEPI program contains only half as many components as the corresponding analog program. The input pairs to integrator 000 in the DEPI program specify that its input will be the square of the output of integrator 001 plus the output of divider 050 times constant number 100. The translation of the program into a code is quite simple. Integrator 000 has two control words since it has two input pairs. The first of these expresses the product of the output of integrator 001 by itself and the next specifies the product of the outputs of divider 050 and constant number 100. The zero word indicates the end of the control words for integrator 000. The next control word specifies the input to integrator 001 as the output of integrator 000 times constant number 101. This is the only input to integrator 000 and the next zero word indicates this fact. In a specific location in the computer memory the programmer must place a 2 so that the integrator routine can compare this number with the number of zero words it comes to and transfer control to the amplifier subroutine when they are equal. This code is not complete, of course, since it does not even specify the output, but there is perhaps enough shown to give some idea of the simplicity of the problem formulation.

Conclusions

A DEPI program has been written for a Datatron 204 at the Jet Propulsion Laboratory in Pasadena. Two problems were run on it for evaluation purposes. The first of these was a set of two non-linear, first-order differential equations describing flame temperatures which could not conveniently be run on the analog installation because of the range of the variables involved. The second problem was a flat earth trajectory, many of which have been run both on the analog computer and the Datatron 204. This is a set of four first-order differential equations with several arbitrary functions, a square root, and several discontinuities. The preparation time for this problem was two to three days as compared to two to three weeks for a standard digital program or about a week for an analog program. The resulting code was far more flexible than the standard digital one and at least as flexible as an analog code. The ratio of computing times, however, is high—a single DEPI integration step taking, for this problem, four times as long as the same step in the standard flat-earth trajectory program. Since the standard digital solution to the trajectory takes about two and one-half times as long as the analog solution at comparable accuracies, the ratio of DEPI time to analog time is approximately ten to one. Datatron 204 is a medium-speed machine, however, and this ratio would reverse on some high speed computers, so that the DEPI time would only be one-tenth of the analog time.

Several basic improvements could be made to the methods used in DEPI. For one thing, the attempt to simulate the component structure of an analog computer seems artificial and a freer approach would probably lead to a great improvement in the simplicity and coding ease of the resulting scheme. The necessity for serial servicing in DEPI is highly undesirable from the point of

view of the programmer and could probably be eliminated altogether. The fact that DEPI operates in the interpretive mode causes its computing speed to be about a third of what it could be. Compiling techniques could probably overcome this defect, though a routine using this technique would be much harder to write and to keep flexible. The Runge-Kutta type integration is probably one of the best features of DEPI. It allows the attainment of high accuracy without a prohibitive expenditure of computing time.

REFERENCES

1. R. G. SELFRIDGE, Coding a General-Purpose Digital Computer to Operate as a Differential Analyzer, *Proceedings of the Western Computer Conference* (Joint IRE-AIEE-ACM Computer Conference, Los Angeles, California, March 1-3, 1955), pp. 82-84 (New York, Institute of Radio Engineers).
2. F. B. HILDEBRAND, *Introduction to Numerical Analysis* (New York, McGraw-Hill Book Company, Inc., 1956).