A DIGITAL SYSTEM FOR ON-LINE STUDIES OF DYNAMICAL SYSTEMS

T. C. Bartee
Harvard University,
Cambridge, Massachusetts
and
J. B. Lewis
Lincoln Laboratory,* Massachusetts Institute of Technology
Lexington, Massachusetts

INTRODUCTION

The study of dynamical systems with the aid of analog and digital computers has developed rapidly in the past two decades. Increased interest in systems described by differential equations which are nonlinear or have time-varying coefficients, has resulted in more reliance on techniques requiring on-line computation. Usually less is known a priori of how the solutions will develop or what parameter values or initial conditions should be used. The recent trend to using hybrid computers (combinations of digital and analog equipment) has been motivated by the desire to study complex dynamical systems with computer configurations which are designed with particular classes of problems in mind. Extensive use of display, plotting, and printing equipment as well as elaborate consoles attest the on-line capability of such computers.

Designing a computer for the on-line study of dynamical systems involves many factors, but among others, speed, accuracy, cost, and user convenience are particularly important. The system described here has emphasized user convenience so that experimental or “trial and error” computational methods are encouraged. An accuracy of 0.1% to 0.01% was considered adequate, and the speed (as measured by ability to solve problems in real time) is roughly equal to the fastest commercial digital computers. Although many modern analog computers are considerably faster in solving systems of ordinary differential equations, and much greater accuracy can be obtained on digital computers, this compromise still allows the study of many problems of great interest. The system is a laboratory experimental model rather than a production prototype, and the cost was kept low by using many components which were on hand. The added cost to the large time-shared system, of which it is a part, was relatively small.

The basic computer configuration is similar to many hybrid computers in that it includes a general-purpose digital computer and a special-purpose computer which is largely a collection of integrators. Most frequently, the special-purpose computer in a hybrid is a high-speed analog computer that meets the need for real-time simulation. In the system discussed here, the special-purpose computer is a high-speed digital differential analyzer (DDA)—a collection of digital summers which approximate

*Operated with support from the U.S. Air Force.
integration. (See Ref. 7 also.) One great convenience to the user results from the fact that the interconnection of the integrators is specified as part of the 86-bit words that describe the integrators. Thus, a patch board is not needed, and as a result, it is possible to write programs for a general-purpose computer to set up the DDA. It is the combination of hardware, which allows rapid, program-controlled changes in interconnection, and software, which translates a problem statement into interconnection information, that makes this system quite attractive. High-speed operation is important in reducing reaction time, and flexible controls that allow start/stop, display, sampling, and repetition are other notable features.

The interconnection of the small general-purpose digital computer, the LINe, and the DDA is described in the next section. The setup of the DDA is done by transferring information from the LINe core memory to the DDA core memory. The information in the LINe core memory is obtained by transfer from a large time-shared computer in which a mapping and scaling program operates. At present, interconnection to the Project MAC computer over a teletype line has been made, and connection to the Lincoln Laboratory IBM 360 System will be completed soon. The mapping and scaling can be done manually for simple problems and the results inserted directly into the LINe core memory. The operation of the system is described in a later section; an important feature is the special combination of LINe and DDA which the user may operate in an experimental fashion with less concern for the usual high charges for time on a large central processor. The large processor is used only when it is necessary to map or scale.

A HIGH-SPEED DIGITAL DIFFERENTIAL ANALYZER

Basic DDA Algorithms and Other Features

The design of early DDA’s was centered about the use of magnetic drums, these being memory devices of reasonable cost with a serial operation which was especially attractive in the processing of a set of DDA integrators. The development of core memories of modest price with cycle times in the 1-microsecond region provides the designer with the possibility of making a DDA with a much higher speed and at quite reasonable costs.

A DDA in which a core memory is used requires a structure different from the usual one. In addition, when one desires to connect a general-purpose machine in a reasonable way, and in particular, to organize the DDA so that the general-purpose machine easily can load (or change) the interconnections of the integrators, the starting and intermediate values in the integrators, the scale factors, and so forth, a different organization becomes attractive. Each word in the memory stores the information concerning a single integrator, and all operations for updating an integrator are performed using parallel arithmetic.

Before proceeding with the organization of the machine, let us briefly examine the integrator algorithm and number system selected. These date back to MADDIDA, and (to the best of our knowledge) were the work of I. S. Reed. Each integrator in the system consists of a single word in the core memory. Variables in the system are represented by 24 binary bits including sign. The number system used is a 2's complement system with sign bit complemented. For 4-bit numbers, the number representation is as shown below:

<table>
<thead>
<tr>
<th>Binary</th>
<th>Decimal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1111</td>
<td>+7</td>
</tr>
<tr>
<td>1110</td>
<td>+6</td>
</tr>
<tr>
<td>1101</td>
<td>+5</td>
</tr>
<tr>
<td>1100</td>
<td>+4</td>
</tr>
<tr>
<td>1011</td>
<td>+3</td>
</tr>
<tr>
<td>1010</td>
<td>+2</td>
</tr>
<tr>
<td>1001</td>
<td>+1</td>
</tr>
<tr>
<td>1000</td>
<td>0</td>
</tr>
<tr>
<td>0111</td>
<td>-1</td>
</tr>
<tr>
<td>0110</td>
<td>-2</td>
</tr>
<tr>
<td>0101</td>
<td>-3</td>
</tr>
<tr>
<td>0100</td>
<td>-4</td>
</tr>
<tr>
<td>0011</td>
<td>-5</td>
</tr>
<tr>
<td>0010</td>
<td>-6</td>
</tr>
<tr>
<td>0001</td>
<td>-7</td>
</tr>
<tr>
<td>0000</td>
<td>-8</td>
</tr>
</tbody>
</table>

Each integrator realizes the relation

$$dz = Cy\, dx$$

where $C$ is a constant, $dz$ an "output increment," and $dx$ an "input increment," and $y$ the integrand.

The definite integral of the above relation is

$$z(x) = z(x_0) + C \int_{x_0}^{x} y(\gamma)\, d\gamma$$

This definite integral is approximated by a sum. The interval $(\gamma_0, \gamma)$ is divided into $n$ subintervals of length $\Delta\gamma$ so that $\gamma_k = \gamma_0 + k\Delta\gamma$, and we choose $x_0 = x(\gamma_0)$ and $x = x(\gamma)$. Therefore,

$$z(x) = z(x_0) + C \sum_{k=1}^{n} y(\gamma_k) \Delta x(\gamma_k) + \epsilon_*$$

where

$$\Delta x(\gamma_k) = x(\gamma_k) - x(\gamma_{k-1})$$

and

$$y(\gamma_{k-1}) + \Delta y(\gamma_k) = y(\gamma_k)$$
Finally, letting $\gamma = (\gamma_1 \Delta x_1) = \Delta z_1$, the result is

$$z(x) = z(x_0) + C \sum_{k=1}^{n} \Delta z_k + \epsilon_n$$

where $\epsilon_n$ is again an error term.

This mathematical representation shows that the DDA performs incremental computations rather than full-word operations, and the approximation to integration is usually called rectangular integration. As shown below, the computations are fixed-point, so that scaling is very important. Finally, it should be noted that the DDA solves initial-value problems.

Let us now consider a part of an integrator register. An integrator register can be looked upon as a "black box" with two inputs, $dx$ and $dy$, an output $dz$, and a stored value $y$, such that relation (1) above is approximated.

In this machine there are 256 integrators which are sequentially updated. Each integrator is in reality a word in the core memory consisting of 24-bit $R$-register and a 24-bit $Y$-register, plus information as to which outputs from the other integrators comprise the $dy$ and $dx$ inputs to the integrator. There is also other information in each word, and this is discussed in the following section. Figure 1 is a block diagram of an integrator.

The $Y$-register part of each word contains values of the variable $y$. Provision is made so that the $Y$-register can be added to or subtracted from the $R$-register, and this is controlled by the $dx$ input, which is a one-bit input that we call $\Delta X$.

The overflow or carry from the most significant bit of $R$ (each time $Y$ is added to $R$) is called $\Delta Z$, and it has value $+1$ if an overflow of $R$ occurs and $-1$ if no overflow occurs.

The $dy$ input is actually a sum of the outputs (or $\Delta Z$'s) from up to 16 other integrators. We call this sum $\Delta Y$ and the particular $\Delta Z$'s which are added together to form $\Delta Y$ are selected in a manner which is described later.

The $\Delta Z$'s from each integrator are stored in a circular shift register of 256 flip-flops. Each position in the shift register corresponds to an address in the memory. The integrators are processed or updated starting with the integrator at address 1 in the memory proceeding through the integrators until all those that are being used have been updated; then the integrator at location 1 is again updated followed by the others. The core memory is a split-cycle memory requiring 0.75 microsec for each half-cycle so that it takes 1.5 microsec to update a single integrator. A problem with 16 integrators would then require 24 microsec for a single iteration or updating of all integrators.

During each updating of an integrator the following operations occur (see Figs. 1 and 2):

1. The $\Delta Z$'s in the shift register, the sum of which comprise the $\Delta Y$ for the inte-
grator being updated, are selected and added together and stored in five flip-flops.

2. The $\Delta X$ input is selected from the $\Delta Z$ shift register and stored in a flip-flop.

3. The value of $\Delta Y$ is added to the current value of $Y$.

4. The new value of $Y$ is multiplied by $\Delta X$ and this is added to $R$. The value of the overflow from $R$ is then stored in the appropriate location in the $\Delta Z$ shift register.

Proof that this algorithm will yield a system in which the integrators approximate the relation given earlier may be found in Ref. 1. Other DDA's are described in Refs. 2, 3, 4, and 8.

A servo mode of operation for the integrators also is included, in which, instead of using the overflow from $R$ as the value to be placed in the $\Delta Z$ shift register, the sign of $y$ is placed in $\Delta Z$ each time. This makes it possible to obtain an increment which approximates a sum of increments. A bit in each integrator word tells the control whether servo mode or conventional mode is to be used with a particular integrator.

It is also possible to invert or complement the $\Delta Z$ output from a given integrator. A bit in each integrator word tells whether or not the overflow from $R$ (or sign of $Y$ in the servo mode) is to be complemented before being stored in the $\Delta Z$ shift register.

Description of Machine Organization

The basic difference between this DDA and others which have been constructed to date is that each integrator carries with it information as to which of the other integrators comprise the $\Delta Y$ and $\Delta X$ inputs. Also, complete information is provided as to whether it is to be used as an integrator or in the servo mode, whether or not it should be sampled, and so forth. As a result, each word in the core memory contains 86 bits.

Figure 3 shows a simplified block diagram of the DDA. Each time an integrator is to be updated, it is selected and read into the memory buffer register, which is a part of the memory. The integrator word is then transferred into the $W$ register. The parts of each 86-bit word are as follows:

1. $R$ is the remainder, or running sum, as explained previously; 24 bits are used for $R$.

2. $Y$ contains the current value of the variable $y$ as explained in the previous section; 24 bits are used for $Y$.

3. $\Delta X$, contains 9 binary bits, 8 of these are used to select the value of $\Delta X$ from the 256-bit $\Delta Z$ register, or an independent increment, $\Delta T$, is selected by the 9th bit.

4. $\Delta Y$, contains 18 bits. This part of the register is used to select the $\Delta Y$-inputs to be added to $Y$. The selection can be made in three ways, and 2 bits of the register tell which way $\Delta Y$, is to be used. The possibilities are:

a) $\Delta Y$, may be used as a linear mask; in which case, the 16 selection bits of $\Delta Y$, are simply placed over the $\Delta Z$ register with the 8th bit on the $\Delta Z$ from the current integrator. If a given bit in $\Delta Y$, is a 1, the corresponding $\Delta Z$ is selected to be added into $\Delta Y$; if the bit is a 0, the corresponding $\Delta Z$ is not added into $\Delta Y$.

b) The first 8 of the selection bits of $\Delta Y$, can be used to select a single $\Delta Z$ to be added to $Y$. Any one of the 256 $\Delta Z$'s can be selected in this manner.

c) The first 8 selection bits of $\Delta Y$, can be used to select a $\Delta Z$ and the second 8 bits to select another $\Delta Z$, so that $\Delta Y$ will be the sum of these two selected $\Delta Z$'s.

5. $D$ consists of two bits and tells the display equipment whether or not the current value of $Y$ is
to be displayed on the oscilloscope; and if it is to be displayed, whether it should be used to deflect along the X or Y axis of the scope.

6. E is a 1-bit register that tells whether or not the Y-value of the integrator should be given to the LINC computer for examination. The frequency at which the integrators are to be examined by the LINC computer is loaded into the DDA as the number of iterations to be performed between each examination. Only those registers which have 1's in their E-position will have their Y-values transferred to the LINC computer. At this time, the system also can change values in the Y-registers. This enables the user to introduce step functions or other functions and also to check on current programs, record data, or display it when required.

7. S contains 5 bits and is used to give the scale factor that determines the length of the Y and R registers for the integrator. S determines which carry or overflow from the added stages is examined to form ΔZ for a given integrator.

8. M consists of two mode bits. One bit indicates whether the integrator is to operate in normal or servo mode. The second indicates that the programmer wishes to stop the DDA if C(Y) = 0.

9. ΔZ_sign is a bit which indicates whether the ΔZ output is to be complemented.

The use of ΔY, as a linear selector permits up to 16 ΔZ inputs to a single integrator i, but these must be within the interval (i - 7, i + 8). A study of the problem of interconnecting integrators with this capability revealed that while most problems could be mapped by adding "dummy" integrators, it made the actual number of integrators in use in some cases unreasonably large. The addition of an optional selection mode allowing either one or two ΔZ's from among any of the 256 integrators alleviated this problem and greatly increased the apparent capacity of the DDA.

In setting up a problem, the LINC computer must give the DDA certain information in addition to that in the integrator. For instance, the number of integrators to be used is first entered into DDA circuitry. The machine is constructed so that the number of integrators processed is determined by this number. Integrators are processed in sequence, and one can use 16, 32, 48, . . . , or 256 integrators in a problem. Thus for small problems only 16 integrators are processed each cycle of the machine, requiring only 24µsec. However, for a 100-integrator problem, 112 integrators would be processed each cycle and a single updating would require 168 µsec. Any unused integrators in these groups are processed, but no harm results except a slight loss in speed of operation. The LINC also loads a counter with the number of iterations or passes through the integrators which are to be made. The DDA will then stop after having performed the required number of iterations. Finally, the LINC loads the number of iterations between samples into the DDA. The sampling rate is determined by this. If the LINC loads the number N into the DDA, after each N iterations the DDA will pause on each integrator having a 1 in the E bit and transfer the contents (Y-register) of this integrator to the LINC. The LINC program examines each of the selected integrators in turn, and the value in each of the integrators also may be changed.

In order to use the memory efficiently, a split-cycle memory is used, and the integrators are not processed in a direct line. When the word associated with an integrator is called, and the memory delivers the word, there is not enough time to process the integrator and write the results back into the memory without a delay. We therefore process integrator i while writing the word for i - 1 into memory and reading i + 1 from memory. This is possible since an extra buffer, the Q-register, is included in addition to the W-register and the memory buffer register (MBR). Three transfers are involved: MBR to W, W to Q, and Q to MBR. Figure 2 illustrates the timing of the combined processing of i, the writing of i - 1, and the reading of i + 1 during a 1.5-µsec cycle.

A cause for interruption of the processing of the integrators is the overflow of a Y-register. The contents of a given Y-register can exceed the capacity of the (scaled) register in either a positive or a negative direction, and each Y-register is checked before and after the ΔY is added to see if overflow has occurred. If this happens, the DDA transfers this information to the LINC.

Finally, a stop on a zero crossing, (i.e., C(Y) = 0) can be programmed so that some decision capability is included. This allows the user to stop the DDA when certain dependent variables reach selected values.

METHOD OF USING THE SYSTEM

The basic input to the system that the user must supply is very much like the usual format used in the study of ordinary differential equations. The user must have his equation in first-order normal
form. After calling the mapping program, he first enters his equations. For example, if the original equation is

\[ \dot{q} - q - \sin q = 0 \]  

let

\[ q_1 = \sin q \quad \text{and} \quad dq_1 = \cos q \, dq \]
\[ q_2 = \cos q \quad \text{and} \quad dq_2 = -\sin q \, dq \]
\[ q_3 = q \quad \text{and} \quad dq_3 = dq = \dot{q} \, dt \]
\[ q_4 = \dot{q} \quad \text{and} \quad dq_4 = dq = \dot{q} \, dt \]  

and the required form is then

\[ dq_1 = q_2 \, dq_3 \]
\[ dq_2 = -q_1 \, dq_3 \]
\[ dq_3 = q_4 \, dt \]
\[ dq_4 = q_3 \, q_4 \, dt + q_1 \, dt \]  

This information is typed as

\[ \text{DQ}_1 = Q2^* \text{DQ}_3 \]
\[ \text{DQ}_2 = -Q1^* \text{DQ}_3 \]
\[ \text{DQ}_3 = Q4^* DT \]
\[ \text{DQ}_4 = Q3^* Q4^* DT + Q1^* DT \]  

and the mapping program then generates a map. Although a number of arbitrary rules are used in the mapping, test problems and recent experience indicate that the program generates maps which are almost as efficient as those done manually in very simple problems. In complex cases, maps that would be quite time-consuming if done manually are generated in several seconds.\(^5\)

The interconnection table that is generated by the mapping program is next used as an input to the scaling program. The user also must supply an estimate of the maximum magnitude of each variable, i.e., \( |q_i|_{\max}, i = 1, 2, \ldots \). This information is then used to compute an optimal set of scale factors using a linear programming routine which maximizes the sum of the number of bits used in all the \( Y \)-registers.\(^6\) Finally, \( \epsilon_0 \), a variable in the scaling program that relates problem time to machine time, must be specified within limits set by the scaling program.

The interconnection and scaling are illustrated in Fig. 4 and Tables 1 and 2. Since the user generally does not need this information, it is saved in the large computer system and supplied only on special request. Figure 4 is a map of Eqs. (6), and Table 1 is the corresponding interconnection table. Table 2 is a table of scale factors that illustrates the interaction of scaling variables; the time scale and maximum values were chosen arbitrarily.

Upon completion of mapping and scaling, the initial contents of \( R \) (set to zero), \( \Delta X_s, \Delta Y_s, M, \Delta Z_{\max}, \) and \( S \) have been specified. The remaining inputs required of the user are the initial values \( q_i(0) \), the number of iterations to be run, the sampling period, and an indication of what variables are to be displayed and sampled. The transfer of this binary information to the LINC is initiated, and after completion, the run on the DDA is made.

Table 1. Interconnection Table

<table>
<thead>
<tr>
<th>Int. No.</th>
<th>DX-Input</th>
<th>DY-Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>3.5</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

**Note:** 0 indicates a DT input.

Table 2. Scaling Table

<table>
<thead>
<tr>
<th>Int. No.</th>
<th>( \eta )</th>
<th>( \gamma )</th>
<th>( \alpha )</th>
<th>( \epsilon )</th>
<th>( \delta )</th>
<th>( \nu )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1*</td>
<td>-12</td>
<td>1</td>
<td>-11</td>
<td>-11</td>
<td>12</td>
</tr>
<tr>
<td>2</td>
<td>1*</td>
<td>-12</td>
<td>1</td>
<td>-11</td>
<td>-11</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>1*</td>
<td>-15†</td>
<td>6</td>
<td>-9†</td>
<td>-11</td>
<td>17</td>
</tr>
<tr>
<td>4</td>
<td>3*</td>
<td>-15†</td>
<td>3</td>
<td>-12</td>
<td>-9</td>
<td>12</td>
</tr>
<tr>
<td>5</td>
<td>3*</td>
<td>-12</td>
<td>3</td>
<td>-9†</td>
<td>-12</td>
<td>15</td>
</tr>
</tbody>
</table>

*Obtained from user estimates as in b) below.
†Determine DT scale relative to problem time.
\( \dagger \) These must be equal.

**Notes:**

a) \( \gamma_i, \delta_i, \epsilon_i \) are scales on \( dx_i, dy_i, dz_i \), e.g., \( dx_i = 2^\gamma_i dx_i \).

b) \( \gamma_i \geq \eta_i \) where \( 2 | y_i |_{\max} \geq 2^\gamma_i > | y_i |_{\max} \), and \( q_i \) is \( y_i \) in the example.

c) \( v_i = \alpha_i - \delta_i \) and \( 23 \geq v_i \geq 4 \).

d) \( \alpha_i + \gamma_i = \epsilon_i \).

e) \( S_i = v_i + 1 \), i.e., the \( S \)-bits of Integrator \( i \).
Repeated runs may be made and new variables may be sampled or displayed by making changes at the LINC keyboard. If parameters in the differential equations are changed, rescaling may be required, and the scaling program is then called again on the time-shared computer. The same is necessary if overflow occurs because of wrong estimates on the $|q_j|$ max.

It is of interest to estimate some typical operating times. After input typing is completed, the reaction times are between several minutes and several seconds depending on the accessibility of the time-shared system programs. The loading and running time on the DDA is at most about 20 seconds for $2^{15}$ iterations of 256 integrators. The time for repeated runs depends largely on the time required for the user to enter new information at the LINC keyboard.

SUMMARY

The design of a system for on-line studies of dynamical systems has been described, and the details of operation of the high-speed DDA have been given. Two applications of the system which are showing its usefulness are

1. Spectral analyses of radar data in which a number of frequencies, spectral windows, smoothing times, and range gates were examined. On-line techniques enable the user to search for combinations of interest rather quickly. The LINC/DDA combination can generate 30 spectral lines, compute a periodogram, and display the results in about 20 seconds.

2. Trajectory generation in simulation studies where the effect of changing parameters in an estimation algorithm are of interest. A set of 3-degree-of-freedom equations that includes atmospheric drag variations and gravitational variations requires 55 integrators. One run (i.e., one trajectory) requires 5 seconds in this case.

A subject requiring further study on the system is error analysis. Error prediction for equations that are integrated by incremental arithmetic operations is extremely difficult, but it is believed that more experience will provide some insight. Very little theoretical work has been done, and the combination of nonlinear equations, incremental methods, and quantization makes the prospect of estimating useful error bounds discouraging. It also may be noted that the sequential processing of the integrators introduces an ordering problem in the mapping program, and the effect of a given ordering scheme on computational errors is again difficult to predict.

Nevertheless, the system is proving very useful in problems where on-line searching and experimentation lead to more complete understanding of certain physical problems. The combination of general and special-purpose digital computers is of great value for studying complex dynamical systems.

ACKNOWLEDGMENTS

The authors wish to thank Dr. H. K. Knudsen for his valuable work on the mapping and scaling programs, and Mr. R. A. Carroll for his work in the planning and construction of the DDA.

REFERENCES
