THE DES-1—A REAL-TIME DIGITAL SIMULATION COMPUTER

Max Palevsky and J. V. Howell
Scientific Data Systems, Inc.
Santa Monica, California

INTRODUCTION

Historically, there have been two lines of development in the attempt to apply digital techniques to the solution of those types of differential equations that have traditionally been solved by analog methods. The first started at Northrup Aircraft in approximately 1948 and led to the development of the digital differential analyzer. The original digital differential analyzers were slow but, in context of the state of digital technology at the time, provided a computing method not otherwise available; in 1948, high-speed internally-programmed general purpose computers were not in general use. As digital techniques developed, digital differential analyzers were increased in speed and in complexity. The most recent of these, the TRICE, is a very high-speed parallel digital differential analyzer that operates at approximately ten thousand times the speed of early digital differential analyzers. The second line of development was started at the Moore School in the investigation of general purpose digital techniques for operational flight trainers. This line of development has been carried on into the present at M.I.T. The emphasis has been on stability considerations and algorithms which would permit computers with a given speed to simulate high performance aircraft.

The DES-1 is an attempt to apply present day technology to the problem of differential equation solving in view of both of the lines of development. The need for this type of computer is shown in a qualitative way by the following graph.

Analog computers—high-precision DC analog equipment—have, for most problems, sufficient frequency response but insufficient accuracy, particularly (and this is not shown by the graph) if large systems of non-linear equations are involved. Digital techniques, on the other hand, are slow. If a third axis—cost—were shown, digital techniques with reasonable frequency responses would be prohibitively expensive. The area above and to the right of the two lines represents performance that is more and more in demand in view of the increasingly difficult problems posed by our expanding technology, and these demands are not met either by digital differential analyzers or by any other available techniques. It can be shown that high-speed digital differential analyzers, in fact, will not yield the performance of presently
available medium-sized general purpose computers for highly non-linear sets of equations.

With this starting point, then, the DES-1 has been evolved . . . . "evolved" since the design has passed through a considerable number of stages starting from a large number of computing elements operating simultaneously, to the present solution which is a general purpose, digital computer with a structure suited to the classical analog problems. For example, the M.I.T. studies have shown that, in using a digital computer for flight simulation, something approaching half the time is employed for the generation of arbitrary functions. The DES-1, therefore, has extensive Search commands that permit functions to be generated with a minimum number of instructions and at a very high speed. A 15-segment function of one variable, using unevenly-spaced increments, can be generated in 77 microseconds. In order to meet the frequency response requirements for simulation, the computer is extremely fast: ADD requires 1.75 microseconds, MULTIPLY requires 7 microseconds. Both of these times include all accesses and indexing.

The DES-1, then, is a general purpose computer with a specially oriented instruction set. It also has a special set of software and a control and display console. These three elements—computer, software, console—are integrated in such a way that the DES-1 "acts" like an analog computer. The central concept that unifies the design is the notion of an "operator." An "operator" is a set of instructions that performs an operation which is analogous to the operation performed by a computing element in an analog computer. The "operators" are processed serially so that the comparison between the DES-1 and an analog computer takes the form of a given amount of time for the DES-1 as compared to a given amount of equipment for the analog computer.

1. Comparison with Analog Computers

A breakdown of such a comparison is shown in Table 1. These numbers and types of operators are processed in approximately 11 milliseconds (which corresponds to about 90 integration cycles per second). Problems involving full-scale resonant frequencies of up to 6 cycles per second can be handled at 75 integration cycles per second with an error of less than one part in $10^4$. The operators listed below require approximately four thousand words of memory.

2. Operators

The notion of an "operator" will now be described in detail. Note that an "operator" is effectively a subroutine and so need not be fixed. The "operator" concept merely permits the machine to be programmed using an analog language as opposed to the standard digital language of the general purpose computer. An "operator" is characterized by an output which is functionally related to an input. For example, if

$$x = \cos \theta,$$

$x$ is the output, $\theta$ is the input, and the “operator” is COSINE. If $\theta$ is assumed as the output of another operator, for example—operator 550, and this cosine operator is assigned an identifying number, for example—430, the operator in the example is specified as

<table>
<thead>
<tr>
<th>Operator</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>COS</td>
<td>550</td>
<td>430</td>
</tr>
</tbody>
</table>

Specification of operators in this form is also equivalent to a multiple address instruction in a digital computer. The operation portion of the operator, COSINE, may be considered an Operation Code, 550 may be considered the address of the input, and 430 the address of the output. Since the DES-1 is a digital device, the latter interpretation is more nearly correct. However, use of the "operator" programming structure lends itself to easy transition regardless of whether the user is analog-minded or general purpose computer-minded. The general purpose computer features of the DES-1 are provided to retain the inherent flexibility of digital computer programming which is particularly suitable for representation of discrete processes and problems involving logical decision and control. The DES-1 possesses a full complement of general purpose computer single address instructions.
Table 1. Direct Comparisons

<table>
<thead>
<tr>
<th>Analog Computing Elements Type</th>
<th>Qty</th>
<th>DES-1 Equivalents Type</th>
<th>Qty</th>
<th>Time per Operator μsec</th>
<th>Total Time msec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operational Amplifiers</td>
<td>100</td>
<td>Sum (7 input)</td>
<td>130</td>
<td>14</td>
<td>1.82</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Integrator</td>
<td>30</td>
<td>80.5</td>
<td>2.415</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(30 summer-integrators, 45 summers, 25 inverters)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Potentiometers</td>
<td>150</td>
<td>Multiplier</td>
<td>150</td>
<td>10.5</td>
<td>1.575</td>
</tr>
<tr>
<td>Multipliers</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time division (10 independent)</td>
<td>20</td>
<td>Multiplier</td>
<td>20</td>
<td>10.5</td>
<td>0.210</td>
</tr>
<tr>
<td>Quarter square (10 independent)</td>
<td>30</td>
<td></td>
<td>30</td>
<td>10.5</td>
<td>0.315</td>
</tr>
<tr>
<td>Servo (10 independent)</td>
<td>50</td>
<td></td>
<td>50</td>
<td>10.5</td>
<td>0.525</td>
</tr>
<tr>
<td>Resolvers</td>
<td>5</td>
<td>Sine</td>
<td>5</td>
<td>80.5</td>
<td>0.4025</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cosine</td>
<td>5</td>
<td>82.25</td>
<td>0.41125</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Arctan</td>
<td>5</td>
<td>171.5</td>
<td>0.8575</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Square Root</td>
<td>5</td>
<td>57.75</td>
<td>0.28875</td>
</tr>
<tr>
<td>Comparators including</td>
<td>10</td>
<td>Comparator</td>
<td>10</td>
<td>26.25</td>
<td>0.2625</td>
</tr>
<tr>
<td>amplifiers and double-pole,</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>double-throw switches</td>
<td></td>
<td>(2 switched inputs)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Diode Limiters</td>
<td>15</td>
<td>Comparator</td>
<td>15</td>
<td>26.25</td>
<td>0.39375</td>
</tr>
<tr>
<td>Diode Function Generators</td>
<td>20</td>
<td>Function 1 variable</td>
<td>20</td>
<td>77</td>
<td>1.54</td>
</tr>
<tr>
<td>(10 segment)</td>
<td></td>
<td>(15 segment)</td>
<td></td>
<td></td>
<td>11.00625</td>
</tr>
</tbody>
</table>

The specification of operators in this form is equivalent to a wire list giving the connections of computing elements in an analog computer as indicated below:

```
Input from Computing Element No. 550

Cosine Element No. 430

Output of Computing Element No. 430
```

The basic operators are:
- Sum
- Scale
- Multiply
- Divide
- Integrate
- Sine
- Cosine
- Arctangent
- Square Root
- Function of One Variable
- Function of Two Variables
- Compare

The operators described here are based on fixed point, single precision (24 bits) number manipulation. However, the methods used can be extended to fixed point, double precision (48 bits) and floating point (39-bit mantissa and 9-bits exponent) at a reduction in speed. Methods used to implement these operators are based on a compromise between maximum speed and maximum generality and accuracy consistent with 24-bit representation of fixed point numerical information. It should be noted that all operators can, and should, exhibit flexibility in their mechanization. For special applications, it may be appropriate, for example, to use integration formulae other than the ones chosen here, or, if greater speed is essential, the number of terms in the sine series may be decreased to trade accuracy for greater speed.
All operators can be modified to suit special applications by general purpose programming techniques.

2.1 Sum Operator

The sum operator is specified as follows:

\[ \text{SUM, } \pm x_1, \pm x_2, \ldots, \pm x_n, y. \]

Contents of the memory locations specified by \( x_1, x_2, \ldots, x_n \) are summed in accordance with the sign preceding each \( x_i \). The result is stored in the location specified by \( y \). The sum operator requires \( 1.75(n + 1) \) microseconds, where \( n \) is the number of inputs.

2.2 Scale Operator

The scale operator is specified as follows:

\[ \text{SCALE, } x, n, y. \]

Contents of the memory location specified by \( x \) are binary-scaled and the result,

\[ (y) = 2^n x(x) \]

is stored in the location specified by \( y \). The scale operator requires 5.25 microseconds for execution.

2.3 Multiply Operator

The multiply operator is specified as follows:

\[ \text{MUL, } x, y, z. \]

Contents of the memory location specified by \( x \) are multiplied by the contents of the memory location specified by \( y \) and the result stored in the memory location specified by \( z \). The multiply operator requires 10.5 microseconds for execution.

2.4 Divide Operator

The divide operator is specified as follows:

\[ \text{DIV, } x, y, z. \]

Contents of the memory location specified by \( x \) are divided by the contents of the memory location specified by \( y \) and the result stored in the memory location specified by \( z \). The divide operator requires 22.5 microseconds for execution.

2.5 Integration Operator

The integration operator is specified as follows:

\[ \text{INT, } \dot{Y}, Y \]

Contents of the memory location specified by \( \dot{Y} \) is the input to the selected integration formula. The integral is computed and stored in the memory location specified by \( Y \). All integration operators are predictors; that is, inputs evaluated at time \( t = nh \) (where \( h \) is the step size) are used to produce outputs at time \( t = (n + 1)h \). The evaluation of all derivatives at \( t = nh \) is insured by placing integration operators at the end of the integration cycle. Inputs to the integrator should not be taken from other integrator outputs. A sum operator or constant operator can be used to achieve separation. The integration sub-program can be modified by general purpose programming methods to obtain any numerical integration process. Integration formulae used in the standard integration sub-program are chosen for general application. Five integration methods are provided and selectable from the DES-1 console. These integration formulae are indicated in Table 2. Truncation errors are indicated for all formulae, except RK4, which has a truncation error of the order of \( h^6 \) but cannot be expressed in the same form as others. All formulae except RK4 make use of past information to achieve a form suitable for high-speed computation. However, the use of past information introduces extraneous roots in the integration. In the first order loop indicated below,

\[ \text{Input} \rightarrow \text{SUM} \rightarrow \text{Int} \rightarrow \text{Output} \]

these extraneous roots can cause instability if the product \( Ah \) is too large. In practice, this means that difficulty is encountered if the step size \( h \) is made too large. Stability regions for each predictor formula is indicated below:
A REAL-TIME DIGITAL SIMULATION COMPUTER

Table 2. Integration Formula

<table>
<thead>
<tr>
<th>Method</th>
<th>Formula</th>
<th>Stable for</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2</td>
<td>( y_{n+1} = y_n + \frac{h}{2} \left[ 3\dot{y}<em>n - \dot{y}</em>{n-1} \right] + \frac{5}{12} y''(\theta) h^3 )</td>
<td>( Ah &lt; 1 )</td>
</tr>
<tr>
<td>P3</td>
<td>( y_{n+1} = y_n + \frac{h}{12} \left[ 23\dot{y}<em>n - 16\dot{y}</em>{n-1} + 5\dot{y}_{n-2} \right] + \frac{3}{8} y''(\theta) h^4 )</td>
<td>( Ah &lt; \frac{6}{11} )</td>
</tr>
<tr>
<td>P4</td>
<td>( y_{n+1} = y_n + \frac{h}{24} \left[ 55\dot{y}<em>n - 59\dot{y}</em>{n-1} + 37\dot{y}<em>{n-2} - 9\dot{y}</em>{n-3} \right] + \frac{251}{720} y''(\theta) h^5 )</td>
<td>( Ah &lt; \frac{3}{10} )</td>
</tr>
</tbody>
</table>

P4, C4 Adams-Bashford Predictor Corrector

\[
y^{(0)}_{n+1} = y_{n+1} + \frac{h}{24} \left[ 55\dot{y}_n - 59\dot{y}_{n-1} + 37\dot{y}_{n-2} - 9\dot{y}_{n-3} \right] + \frac{251}{720} y''(\theta) h^5
\]

Truncation error in corrector

\[
\varepsilon^{(c)}_T = \frac{19}{270} (y^{(0)}_{n+1} - y^{(c)}_{n+1})
\]

RK4 4th order Runge Kutta

\[
y^{(1)}_{n+1/2} = y_n + \frac{h}{2} \dot{y}_n
\]

\[
y^{(2)}_{n+1/2} = y_n + \frac{h}{2} \dot{y}^{(1)}_{n+1/2}
\]

\[
y^{(3)}_{n+1} = y_n + h\dot{y}^{(2)}_{n+1/2}
\]

\[
y_{n+1} = y_n + \frac{h}{6} \dot{y}_n + 2\dot{y}^{(1)}_{n+1/2} + 2\dot{y}^{(2)}_{n+1/2} + \dot{y}^{(3)}_{n+1}
\]

The Runge Kutta Integration methods possess the property of requiring no past information. Runge Kutta methods are thus very useful in determining starting values for the other four integration methods. Starting values are computed by use of RK4 when the DES-1 is put into the RESET mode and during computation when step size changes are called for by P4C4. The RK4 method is also useful in obtaining an alternate problem solution for checking against any of the other integration methods. The integration cycle is divided into four subcycles during which new values of all derivatives are computed. In effect, this implies that a problem will run about four times slower using RK4 than when using any of the predictors. The use of RK4 is in general restricted to obtaining check solutions.

Execution times for integration methods are shown in Table 3.

<table>
<thead>
<tr>
<th>Method</th>
<th>Time in ( \mu \text{sec} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>P2</td>
<td>42</td>
</tr>
<tr>
<td>P3</td>
<td>61.25</td>
</tr>
<tr>
<td>P4</td>
<td>80.5</td>
</tr>
</tbody>
</table>

The Predictor Corrector Integration Method, P4C4, is very useful for problems in which a linear relation between problem time and real time is not required. The integration cycle is divided into two subcycles. During the first subcycle, P4 is used in all integration operators to move ahead in time by \( h \). During the second subcycle, C4 is used to obtain a second value of the integrated quantities. The truncation error in C4 is estimated as indicated in Table 2. If the truncation error in any integration operator exceeds a preassigned amount \( \varepsilon_T \) max, the step size is halved. If the truncation error in all integration operators is less than a second pre-assigned quantity \( \varepsilon_T \) min, the step size is doubled. The truncation error is controlled by use of the P4C4 integration technique to remain approximately within the range \( \varepsilon_T \) min to \( \varepsilon_T \) max by slowing the problem down (taking smaller steps when rates of change are large) and by speeding the problem up (taking larger steps when rates of change are small). For example, the Predictor Corrector technique can be applied to missile trajectory calculations, where rates of change are high during powered flight and re-entry but small during free flight, to achieve a significant net saving in run time.
The following table is an approximation to the accuracy band-width performance of the DES-1 with various integration schemes. It assumes that the step size is 10 milliseconds and that the step error is $10^{-4}$:

<table>
<thead>
<tr>
<th>Order of Integration</th>
<th>Maximum Resonant Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>8.5 rad ~ 1.35 cps</td>
</tr>
<tr>
<td>3</td>
<td>22 rad ~ 3.5 cps</td>
</tr>
<tr>
<td>4</td>
<td>41 rad ~ 6.53 cps</td>
</tr>
</tbody>
</table>

2.6 Sine Operator

The sine operator is specified as follows:

\[ \text{SIN, } \theta, Y. \]

Contents of the memory location specified by the sine of \( \theta \) is computed and stored in the memory location specified by the contents of \( Y \). The operator input \( (\theta) \) is in revolutions, \(-\frac{1}{2} < (\theta) < \frac{1}{2} \). Second and third quadrant angles are reduced to the equivalent first or fourth quadrant angle \( \varphi \). The sine is computed from the power series,

\[
\sin 2\pi \varphi = C_1 \varphi + C_3 \varphi^3 + C_5 \varphi^5 + C_7 \varphi^7 + C_9 \varphi^9 \\
\text{where } \frac{1}{2} \leq \varphi < \frac{1}{2} \\
\text{and } C_1 = 1.5707963 \\
C_3 = 0.6459637 \\
C_5 = 0.0796897 \\
C_7 = 0.0046738 \\
C_9 = 0.0001515
\]

with a maximum error of approximately three parts in \( 10^7 \). Execution time for the entire process is 80.5 microseconds.

2.7 Cosine Operator

The cosine operator is specified as follows:

\[ \text{COS, } \theta, X. \]

The cosine of the contents of the memory location specified by \( \theta \) is computed and stored in the memory location specified by the contents of \( X \). The operator input is in revolutions, \(-\frac{1}{2} \leq (\theta) < \frac{1}{2} \). The cosine of \( (\theta) \) is obtained by computing \( \sin 2\pi ((\theta) + \frac{1}{4}) \) as described under sine operator. The execution time for the cosine operator is 82.25 microseconds.

2.8 Arctangent Operator

The arctangent operator is specified as follows:

\[ \text{ATAN, } Y, X, \theta. \]

The arctangent of \( (y)/(x) \) is computed and stored in the location specified by the contents of \( \theta \). The operator output is in revolutions, \(-\frac{1}{2} \leq (\theta) < \frac{1}{2} \). The principal angle \( \varphi \) is computed from the power series,

\[
\varphi = C_1 U + C_3 U^3 + C_5 U^5 + C_7 U^7 + C_9 U^9 + C_{11} U^{11} + C_{13} U^{13} \\
\text{where } U = x/y \text{ if } |x| \geq |y| \\
U = x/y \text{ if } |x| < |y|
\]

and \( C_1 = 0.15915433 \)
\( C_3 = 0.05302625 \)
\( C_5 = 0.03152625 \)
\( C_7 = 0.02160178 \)
\( C_9 = 0.01267292 \)
\( C_{11} = 0.00534860 \)
\( C_{13} = 0.000108423 \)

The output angle \( (\theta) \) is obtained from the principal angle \( \varphi \) by quadrant correction. All four quadrants are resolved and the error in \( \theta \) is less than 3 parts in \( 10^7 \). Execution time for the arctangent operator is 171.5 microseconds.

2.9 Square Root Operator

The square root operator is specified as follows:

\[ \text{SQRT, } X, Y. \]

The square root of the contents of the memory location specified by \( X \) is computed and stored in the location specified by the contents of \( Y \). The square root of \( (x) \) is obtained as follows:

The input \( (x) \) is normalized such that \( x^2 \) lies within the range \( \frac{1}{2} \) to 1. This range is divided into four equally spaced intervals
and within each interval $\sqrt{x}$ is representable by a third degree power series in $x$. The square root of $x$ is obtained by evaluation of one of the four power series and the square root of $(x)$ is obtained from

$$\sqrt{x} = 2^{n/2}\sqrt{x_n}.$$ 

Execution time for the square root operator is 57.75 microseconds and the maximum error is three parts in $10^7$.

2.10. Function of One Variable Operator

The function of one variable operator is specified as follows:

$$\text{FUN 1, X, Y.}$$

The contents of the memory location specified by $X$ is the input. The function of one variable is computed and the result stored in the memory location specified by $Y$. The method of straight line approximations is used to construct the function. Fifteen values of the independent variable and dependent variable are stored relative to the location of the operator output.

$$x_1 \quad y_1$$
$$x_2 \quad y_2$$
$$x_3 \quad y_3$$
$$\cdots$$
$$\cdots$$
$$\cdots$$
$$x_{15} \quad y_{15}$$

The function $y$ is computed from the relation

$$y = y_i + \frac{y_{i+1} - y_i}{x_{i+1} - x_i}(x - x_i)$$

where $x_i < x \leq x_{i+1}$

The independent variable points can be arbitrarily spaced. During computation, the interval within which $x$ lies is determined by the interval used during the previous integration cycle by checking this interval plus intervals on either side. If $x$ does not lie in any of these three intervals, all intervals are scanned linearly. In practice, the variable $x$ should never change so rapidly that the linear scan is required. The linear scan is used primarily when the DES-1 is put into the RESET mode. During computation, the function of one variable operator requires 77 microseconds for execution. A slightly longer period is needed if the linear scan is necessary. The accuracy in this method of function generation is dependent upon how well the function can be approximated with straight line segments.

2.11 Function of Two Variables Operator

The function of two variables operator is specified as follows:

$$\text{FUN 2, X, Y, Z.}$$

The contents of memory locations specified by $X$ and $Y$ are the inputs. The function of two variables is computed and the result stored in the memory location specified by $Z$. The method of straight line approximations is used to construct the function. Values of dependent and independent variables are stored relative to the location specified by $Z$ indicated, as follows:

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$z_{11}$</th>
<th>$z_{12}$</th>
<th>$\cdots$</th>
<th>$z_{17}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_2$</td>
<td>$z_{21}$</td>
<td>$z_{22}$</td>
<td>$\cdots$</td>
<td>$z_{27}$</td>
</tr>
<tr>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
<td>$\cdots$</td>
</tr>
<tr>
<td>$x_7$</td>
<td>$z_{71}$</td>
<td>$x_{72}$</td>
<td>$\cdots$</td>
<td>$z_{77}$</td>
</tr>
</tbody>
</table>

The function $z$ is computed from the relation,

$$z = z_{ij} + \frac{z_{i+1,j} - z_{ij}}{x_{i+1} - x_i}(x - x_i) + \frac{z_{i,j+1} - z_{ij}}{y_{i+1} - y_i}(y - y_i)$$

$$+ \frac{(z_{i+1,j+1} - z_{i+1,j}) - (z_{i,j+1} - z_{ij})}{(x_{i+1} - x_i)(y_{j+1} - y_j)}(x - x_i)(y - y_j)$$
where \( x_i < x \leq x_{i+1} \)
\[ y_i < y \leq y_{j+1} \]
The above relation is exactly satisfied on all four corners of the square
\[ x_0, y_j \]
\[ x_{i+1}, y_j \]
\[ x_0, y_{j+1} \]
\[ x_{i+1}, y_{j+1} \]

The independent variable points can be arbitrarily spaced so long as a square mesh is maintained. The region within which \( x \) and \( y \) lie, is determined by the region used during the previous integration cycle by an extension of the method used for a function of one variable. During computation, the function of two variables operator requires 231 microseconds for execution.

2.12 Comparator Operator

The comparator operator is specified as follows:

\[ \text{COMP } X, Y, X_1, Y_1, Z_1, X_2, Y_2, Z_2, \ldots, X_k, Y_k, Z_k \]

The comparator operator simulates a \( K \) pole double-throw switch. The comparator operator functions as follows:

\[
\begin{align*}
\text{if } x \leq y \\
z_1 &= x_1 \\
z_2 &= x_2 \\
&\quad \vdots \\
z_k &= x_k \\
\text{if } x < y \\
z_1 &= y_1 \\
z_2 &= y_2 \\
&\quad \vdots \\
z_k &= y_k 
\end{align*}
\]

where \( x \) represents the contents of the memory location specified by \( X \)

\( y \) represents the contents of the memory location specified by \( Y \)

\( x \), represents the contents of the memory location specified by \( X_1 \)

Execution time for the comparator operator is \( 12.25 + 7k \) microseconds where \( k \) is the number of switched inputs.

2.13 Other Useful Operators

The operators previously described are considered to be most universal in application. However, there are many others which are useful on occasion. For a particular application, a requirement may exist for Bessel Functions, or Legendre Polynomials, or exponentials or logarithms.

In addition to operators which are useful for computation, it is necessary to have operators which provide a convenient means of input/output. A number of these are provided for analog input/output, for typewriter output, etc.

3. Programming Example

An example will illustrate the programming of the DES-1. This example is complex in order to illustrate, first of all, the method in which the analog language of the DES-1 is used to program the machine and, secondly, to illustrate the homogeneity of the analog language with the standard DES-1 general purpose language.

The Pulsed Jet Control System problem is an example of this type of problem because a portion of the system (the rigid body dynamics) is continuous, and a portion of the system (the pulsed jet control) is digital in nature. Differential equations expressing the angular orientation of the vehicle are handled by operators. Jet commands and jet characteristics are digital in nature and are handled by the general purpose computer instructions.

The equations to be mechanized for solving this problem are given in Table 4. Problem mechanization is shown in Figures 1 and 2. A listing of operator connections is given in Table 5. It should be noted that Table 5 consists of a mixture of operators and general purpose computer instructions. The general purpose instructions are used to mechanize the Sampler indicated in Figure 2. Operators and
Figure 1. Angular Dynamics.

Figure 2. Jet Control.
general purpose instructions can be mixed in any fashion in the DES-I, thereby retaining decision and logical capabilities of general purpose computers.

Table 4. Equations

<table>
<thead>
<tr>
<th>Equation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$1_{x} \dot{\theta} = M_{x}$</td>
<td>Pitch angular motion</td>
</tr>
<tr>
<td>$\epsilon_{s} = \theta_{s} - \theta - K_{s} \dot{\theta}$</td>
<td>Pitch error</td>
</tr>
<tr>
<td>$\epsilon_{s}^{*} = \epsilon_{s} \text{ at } t = nt$</td>
<td>Sample pitch error</td>
</tr>
<tr>
<td>$t_{s} = \frac{\epsilon_{s}^{*}}{\epsilon_{s_{\text{max}}}} T$</td>
<td>Jet pulse duration</td>
</tr>
</tbody>
</table>

$C_{s}$

$$C_{s} = \begin{cases} 1 \text{ if } \epsilon_{s} > \epsilon_{s_{\text{min}}} \text{ and } t - nt < t_{s} \\ 0 \text{ if } \epsilon_{s} \leq \epsilon_{s_{\text{min}}} \end{cases}$$

Moment commands

$$M_{s} = M_{x}C_{s} - M_{x-C_{s}}$$ Control moment

The meaning of Table 5 should be emphasized. The initial instructions are in the analog language and are analogous to a list prepared for plugboard programming in an analog machine. Starting with instruction “LDA”, the instructions are in digital language since the operations to be performed are appropriate to a digital computer rather than analog elements. The analog language starts again with the instruction “SUM”. Thus, this example illustrates first that the computer is programmed like an analog machine and, secondly, the homogeneity of the analog and digital languages. Of course, in addition to the program interconnections, initial condition data would also have to be entered into the computer. This is analogous to setting up Potentiometers or feeding tapes into an automatic analog initial condition set-up system.

Table 5. Operator Connections

<table>
<thead>
<tr>
<th>Operator</th>
<th>Connection</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUM,</td>
<td>+40, 32</td>
</tr>
<tr>
<td>MUL,</td>
<td>40, 4</td>
</tr>
<tr>
<td>SUM,</td>
<td>+5, -50, -50, -23, 33</td>
</tr>
<tr>
<td>SUM,</td>
<td>+60, -15</td>
</tr>
<tr>
<td>LDA</td>
<td>34</td>
</tr>
<tr>
<td>SKG</td>
<td>14</td>
</tr>
<tr>
<td>TRU</td>
<td>14</td>
</tr>
<tr>
<td>LDA</td>
<td>15</td>
</tr>
<tr>
<td>ADM</td>
<td>33</td>
</tr>
<tr>
<td>LDA</td>
<td>34</td>
</tr>
<tr>
<td>STA</td>
<td>16</td>
</tr>
<tr>
<td>SUM,</td>
<td>-16, 34</td>
</tr>
<tr>
<td>COMP,</td>
<td>16, 11</td>
</tr>
<tr>
<td>COMP,</td>
<td>16, 7</td>
</tr>
<tr>
<td>COMP,</td>
<td>16, 10</td>
</tr>
<tr>
<td>MUL,</td>
<td>71, 9</td>
</tr>
<tr>
<td>COMP,</td>
<td>24, 34</td>
</tr>
<tr>
<td>MUL,</td>
<td>1, 74</td>
</tr>
<tr>
<td>MUL,</td>
<td>2, 75</td>
</tr>
<tr>
<td>COMP,</td>
<td>21, 22</td>
</tr>
<tr>
<td>MUL,</td>
<td>31, 31</td>
</tr>
<tr>
<td>MUL,</td>
<td>25, 40</td>
</tr>
<tr>
<td>MUL,</td>
<td>32, 60</td>
</tr>
<tr>
<td>MUL,</td>
<td>6, 60</td>
</tr>
</tbody>
</table>

Load accumulator with $t - nt$

skip if $t - nt > T$

transfer (do not sample)

load accumulator with $T$

add to memory $nT + T \rightarrow nT$

load accumulator with $\epsilon_{\phi}$

store $\epsilon_{\phi}$ in $\epsilon_{s}^{*}$

16, 34, 71. 12, 11, 72. 11, 12, 72. 24. 72, 11, 74, 73, 11, 75. 21. 22. 21. 31. 25.
4. The Control and Display Console

The two elements of the DES-1 thus far described—the computer proper, and the operators are tied together and operated from the control and display console. This console aids the operator in solving and debugging problems in conjunction with the SDS 9300 Computer console and uses the typewriter, photo-reader, and paper tape punch belonging to that console. A drawing of the DES-1 console is shown in Figure 3.

**Mode Selector**

Three operation modes are provided by the mode selector: COMPUTE, IDLE, and RESET. The problem solution is generated in the COMPUTE mode. IDLE mode is provided for stopping the problem solution at any time to inspect various parts of the solution. RESET mode is used to automatically insert initial conditions for re-run of the problem. In addition, single cycle and single iteration control are provided to facilitate problem checkout.

**Control Selector**

Three sources of control are provided by the CONTROL SELECTOR: internal, manual and external. INT refers to program control of the operation modes. MAN gives problem control to the mode selector. EXT gives problem control to an external device allowing the DES-1 to be slaved to the external device.

**Integration Selector**

A means of selecting one of five types of integration is provided by INTEGRATION SELECTOR. These integration types are normally:
1) Predictor—2nd order
2) Predictor—3rd order
3) Predictor—4th order
4) Predictor-corrector—4th order
5) Runge Kutta—4th order

**Integration Period Selector**

A means of synchronizing machine iterations to real time is provided by the INTEGRATION PERIOD selector. The minimum integration is dependent upon problem size, but can be extended to any convenient period up to 10 seconds in increments of 10 microseconds. An on-off switch is provided for switching out the integration period selector in problems requiring no correlation to real time.

**Readout**

The output display and ADDRESS SELECTOR provide a convenient means of monitoring operators and intermediate results, especially during program debugging. An operator output or memory location is selected by the ADDRESS SELECTOR. A binary exponent is selected by the BINARY SCALE to allow the information readout to be presented in a properly scaled form. The DISPLAY pushbutton causes the contents of the selected address to be scaled by the binary exponent, converted to decimal, and presented on the output display in the form of a signed fraction and decimal exponent. The information presented on the display can be typed out on the typewriter by operation of the TYPE pushbutton. A READ-OUT MODE control allows all combinations of DECIMAL or OCTAL and SINGLE or DOUBLE fixed point or FLOATING point number readout.

**Sense Switches**

A group of eight sense switches located on the DES-1 console are for program-determined use.
**Arithmetic Overflow**

Arithmetic overflow is indicated by the OVERFLOW light. The DES-I can be set to HALT on overflow or CONTINUE on overflow.

**Program Input/Output**

The photo-reader or typewriter is used for program input. Normally, the photo-reader is used to load the program and changes or corrections are made from the typewriter. The corrected program can be punched and/or typed out for future use.

**Data Input/Output**

Normally the photo-reader is used to load initial conditions, constants and other data into the computer. Changes or new values can be loaded for either the photo-reader or typewriter.

**Digital Potentiometer**

Continuously variable 16-turn digital potentiometers can be supplied to permit continuous variation of a problem variable during actual solution.

5. **Input/Output**

A typical input/output configuration which can be used to integrate the DES-I into a system is shown in Figure 4. Five types of input/output are used:

A. **Analog Input**

This type of input accepts analog signals, converts them to digital form and enters the results into the DES-I. In this example, a multiplexer provides for 96 inputs; these may be selected either by channel number or the channels can be sampled sequentially. The former case requires additional computer time, since in the latter case the sequencing can be performed independently of the computer. Note that both SET and ADVANCE controls are provided.

A sample-and-hold amplifier is provided for signals with higher frequency contents. In some systems the time lag caused by the sequen-

---

Figure 4. Typical Input/Output Configuration.
tial nature of the conversion process may generate intolerable errors. A simultaneous sample-and-hold, which samples all channels simultaneously, can be used in these cases. Note that the analog and digital grounds are isolated by transformer coupling.

The total computer time required to process 96 channels into the computer is less than 180 microseconds.

B. Boolean Input

This type of input accepts on-off signals and enters them into the computer. These signals may be generated by various types of switch closures or by combinations of lines which represent other control information. Again, individual groups of 24 can be sampled or the sampling can be sequential. The total time required for 96 inputs is less than 10 microseconds.

C. Analog Output

This type of output accepts digital information from the computer and converts it to analog form. A total of 96 registers are provided to drive converters. Some of the registers are shown without converters to indicate that special devices, such as digital servos, can be employed. Alternately, a single register can be used sequentially with a single digital-to-analog and a number of holding circuits to provide slower, less accurate but more economical conversion. Again, SET and ADVANCE lines are provided and transformer coupling isolates the grounds.

The total time required to process 96 channels is less than 180 microseconds.

D. Boolean Output

This type of output generates on-off signals for driving relays, indicators, etc. Individual groups of 24 can be generated or sampling can be sequential. The total time required for 96 inputs is less than 10 microseconds.

E. Clock

The solution of differential equations by numerical methods proceeds by steps or cycles. A clock is provided to signal the start of each cycle. The cycle time is set into the clock by the computer to the nearest microsecond. The clock accuracy exceeds one part in $10^5$. At the appropriate times, the clock activates an interrupt which sends the program to the start of the next cycle.

The total time for the entire input/output operation as shown in Figure 4 is less than 400 microseconds. If solutions are generated 25 times per second, this represents less than 1% of the available computing time.

Since the DES-1 is oriented towards classical simulation problems, and since in many simulations actual hardware must be tied into the computing system, the input/output system is a critical area.

6. Summary

This brief talk has attempted to indicate how the DES-1 can solve some of the classical problems of simulation, digitally, and to the greater accuracies that are required by today's technology. Further, the DES-1 is, in many cases, less expensive than analog equipment. Several other points should be made that are less technical but perhaps as important. Being a digital system, the maintenance problem of the DES-1 is considerably less severe than that of an analog machine. Further, set-up time is relatively insignificant since all information is stored on paper tape or punched cards which can be read rapidly and accurately. No initial "tuning" is required. It is certainly the case that in some problems the frequency response of the DES-1 is insufficient; however, the DES-1 represents the first step in what may be a change in the technology which will replace analog computers to a large measure by digital technique.

REFERENCES

