DEVELOPMENT OF AN EXPERIMENTAL
GENERALIZED ANALOG INTEGRATOR (EGI-2)

By John W. Robertson

Prepared for
George C. Marshall Space Flight Center
Huntsville, Alabama

Contract No. NAS8-2473

(Development of New Methods and
Applications of Analog Computation)

15 March 1967

Engineering Experiment Station
GEORGIA INSTITUTE OF TECHNOLOGY
Atlanta, Georgia
DEVELOPMENT OF AN EXPERIMENTAL GENERALIZED ANALOG INTEGRATOR (EGI-2)

by

John W. Robertson

TECHNICAL NOTE NO. 14

on

Contract No. NAS8-2473

(Development of New Methods and Applications of Analog Computation)

For

GEORGE C. MARSHALL SPACE FLIGHT CENTER
Huntsville, Alabama
CONTENTS

Chapter | Page
--- | ---
I. INTRODUCTION | 1
II. BASIC CONCEPTS OF THE "EGI" | 2
III. COMPARISON OF EGI DESIGNS | 6
IV. MODIFICATIONS MADE TO EGI-2 DESIGN | 10
V. CALIBRATION AND OPERATION | 13
VI. TEST RESULTS | 15
VII. DISCUSSION AND CONCLUSIONS | 20
Appendix - EQUIPMENT DIAGRAMS | 23

List of Figures

1. Illustrating Samples in Y Taken with Uniform Changes in X | 3
2. Simplified Block Diagram of EGI System | 3
3. Simplified Block Diagram of Incremental Detector in EGI System | 4
4. Illustrating Signals and Pulses Involved in EGI Operation | 5
5. Front and Rear Views of EGI-2 | 12
6. Typical Test Runs for Evaluation of EGI-2 Performance | 16
7. EGI-2 Error Vs. X-Input Amplitude and Frequency (Constant Y-Input) | 17
8. EGI-2 Error Vs. Constant Y-Input Level (Fixed Sinewave X-Input) | 18
A-1. Analog System Diagram | 24
A-2. Reset Pulse Logic | 25
A-3. Temperature Control Amplifier | 26
A-4. Circuit Card 1 Schematic | 26
A-5. Circuit Card 2 Schematic | 27
A-6. Circuit Card 3 Schematic | 27
A-7. Microcircuit Card 1 Logic Diagram | 28
A-8. Microcircuit Card 2 Logic Diagram | 28
A-9. Microcircuit Card 3 Logic Diagram | 29
A-10. Cold Plate Schematic | 29
I. INTRODUCTION

The history of the present Electronic Generalized Integrator (EGI-2) extends back through an earlier model (EGI-1) to an experimental analog to incremental-digital converter (AID Converter). This original device was conceived as a possible solution to the requirements for low-cost analog-to-digital data conversion in a hybrid computing system. A prototype unit was constructed during 1960-61 under Contract DA-01-009 ORD-853 for the U.S. Army Ballistic Missile Agency, Redstone Arsenal, Alabama.

The AID Converter utilized an operational amplifier as a storage element, with sensors at the output of the amplifier to detect an increase of level above a threshold value. A reset pulse of constant width and amplitude was then applied to reduce the amplifier output to a level below the threshold value. Capacitive feedback was used on the amplifier with a capacitive input for the signal to be converted, and a resistive input for reset pulses. The amplifier thus functioned as an inverter to the input signal and as an integrator to the reset pulses. The effect was to sum the reset pulses and subtract this voltage from the input voltage to maintain the output near zero. The reset pulses were then translated into the proper digital code for the output to the digital computer.

The converter described above performed most of the functions necessary to provide an electronic generalized integrator. Only sampling circuits for the signal to be integrated and a summing amplifier for the samples themselves remained to be added. This was done in 1961-62 under Contract NAS8-2473 for the George C. Marshall Space Flight Center, Huntsville, Alabama.

The major shortcomings of the first model constructed under this contract were the temperature effects in the analog sampling switches. In December 1963 a program of investigation of electronic switching was started under a continuation of the NASA contract. The results of the study indicated that the "series-shunt field-effect-transistor switch" and the "six-diode bridge gate" would be good choices as replacements for the earlier switches used in EGI-1. Under a further extension of the contract, the experimental model described in this report was constructed and evaluated.
II. BASIC CONCEPTS OF THE "EGI"

In a general-purpose electronic analog computer, time is the independent variable, and all other variables are dependent variables. Problems are solved with time-varying functions which have the same mathematical form as the original problem variables, although the original variables are not necessarily functions of time. Facilities are normally supplied for integration, multiplication, and summation on such a computer. Equations or systems of equations involving differentiation are usually integrated to remove the requirements for differentiation, with its inherently high noise level. The integration provided for in computers of the type under discussion is with respect to time, the independent variable. This limitation on integration is one of the main drawbacks of analog computers as contrasted to digital differential analyzers (digital computers). The capability of integrating one dependent variable with respect to another dependent variable--i.e., generalized integration--is a necessity in some problems and may offer major program reductions in other problems, particularly in the solution of nonlinear equations.

To perform the operation of generalized integration, use is made of the approximation

\[ \int_{X_1}^{X_n} Y \, dX \approx \sum_{i=1}^{n} Y_i \Delta X_i \],

which is valid for \( \Delta X_i \) sufficiently small. If the \( \Delta X_i \) are all equal (say \( \Delta X \)), one has:

\[ \int_{X_1}^{X_n} Y \, dX \approx \Delta X \sum_{i=1}^{n} Y_i \].

In practice, constant-width samples of the variable to be integrated (\( Y \)) are taken at times determined by changes of a given amount (\( \Delta X \)) occurring in the other variable, as illustrated in Figure 1. These samples are then summed to obtain the equivalent integral, in accordance with the basic approximation.

A simplified block diagram of the EGI system for performing the above operation of generalized analog integration is shown in Figure 2. Any change in the \( X \)-input of sufficient magnitude--whether an increase (+\( \Delta X \)) or a decrease (-\( \Delta X \))--causes a sample of the \( Y \)-input of corresponding polarity to be
taken and summed with all preceding samples in the output integrator. The output integrator serves as a storage element and register for the input samples, providing the means of converting the basic samples into a continuous output.

The Incremental Detector, which furnishes the sampling command pulses to the Samplers, is a subsystem in its own right, and is shown in more detail in Figure 3. The first amplifier is used to sum the X-input signal and the reset pulses. It consists of an operational amplifier with capacitive feedback, capacitive input for X, and resistive input for reset pulses. (The capacitive input prevents the dc portion of the X-input signal from being summed into the integrator.) The system functions as a closed loop which

![Diagram](image-url)
supplies reset pulses of constant amplitude and duration and of the proper polarity to maintain the output of the first amplifier within a narrow region about zero.

Operation of the system is illustrated in Figure 4, which shows the +Y and -Y sampling commands (or ΔX reset pulses) associated with an arbitrary X-input signal, and the corresponding sampled values of an arbitrary Y-input signal producing the incrementally developed output integral.
Figure 4. Illustrating Signals and Pulses Involved in EGI Operation.
III. COMPARISON OF EGI DESIGNS

The EGI-1 and EGI-2 differ slightly in the design approach to the incremental detector section. The original design limited the duty cycle of the reset pulse to fifty percent; that is, only half of the time available for resetting the amplifier output to zero was actually used. By designing the EGI-2 reset pulse generator with a capability to provide pulses of higher duty cycle, a more rapidly varying X-input signal could be accommodated, assuming that other factors remained constant. As the reset pulse duty cycle approaches one hundred percent, the improvement approaches a factor of two. It was further noted that by utilizing a single input resistor for the reset pulse input, instead of the two resistors previously employed, a twofold reduction in gain for the noise signal present on the ground would occur. This would be equivalent to reducing the system noise level by a factor of two, if the amplifier's internal noise was low in comparison to that of ground. The improvement in signal-to-noise ratio could be used either to widen the bandwidth or increase the resolution of the EGI.

The signal comparators are the heart of the incremental detector. These circuits monitor the output of the first operational amplifier, and control the generation of reset pulses. In the EGI-1 the signal comparators are basically single transistors, biased near cutoff, and interrogated at a given time by removing a blanking pulse which drives the transistor into cutoff at all other times. An adjustable bias control is used to compensate for transistor-to-transistor changes in the cutoff point.

In EGI-2, Fairchild µL-710 integrated-circuit differential comparators were used as signal comparators. These units are differential amplifiers with high gain, used to monitor the output of Amplifier 1 on a continuous basis. The resulting signals are then operated on by the reset pulse logic to produce the requisite reset pulses.

The reset pulse generators employed in EGI-1 and EGI-2 differ considerably. A simple single-transistor switch was utilized in EGI-1. Offset and leakage were relatively constant due to the temperature control employed, and were compensated for by summing in adjustable biases to the amplifier grid and to the input of each switch. The EGI-2 uses series-shunt field-effect transistor (FET) switches to provide reset pulses by sampling an adjustable voltage. It was hoped in this manner to make the reset pulse amplitude less
dependent on precise temperature control of the switches.

The Y-input sampling switches used in the EGI-1 are basically Bright Circuit transistor switches. These are series switches consisting of two complementary transistors connected in series, emitter to emitter, with transformer-coupled base drive voltages. In EGI-2, a series-shunt FET switch was used to provide improved switching action. An additional bonus was the ability of the series-shunt switch to operate on a dc control signal basis—i.e., continuously on or off.

The reset pulse logic of EGI-1 was synchronized by a 100 kilohertz clock. In that system, a 100 kilohertz square wave is counted down to provide a 12.5 kilohertz square wave, and a 5 microsecond pulse is formed at the time of positive excursion of this signal by a monostable multivibrator. A pulse of indeterminate width is also formed at the time of the negative excursion of the 12.5 kilohertz wave. The 5 microsecond pulse is used to interrogate the signal comparators (ΔX detectors). If an output of the proper polarity and of greater than the preset level is present at the output of the first operational amplifier, a pulse is produced, setting a flip-flop, which is reset by the pulse formed from the falling edge of the 12.5 kilohertz wave, 40 microseconds after the onset of the 5 microsecond pulse. The flip-flop outputs control the reset pulse generators and the Y-input sampling switches. Timing of the samples is thus derived from the clock and is quite precise. The pulse width of the samples is 40 microseconds and the minimum separation in time is 40 microseconds.

EGI-2 utilizes a different reset pulse logic, synchronized by a 200 kilohertz clock. The signal comparators operate on a continuous, non-interrogated basis. An output from the first operational amplifier exceeding a preset reference level causes a change of state at the comparator output. The 200 kilohertz square wave is shaped and counted down to give a 12.5 kilohertz square wave. The outputs of the various stages in the counter are then combined in a gate to produce a 5 microsecond pulse at the negative excursion of the 12.5 kilohertz wave. This pulse gates the output of the signal comparators, and the resulting pulses are fed to latching circuits which are turned on and latched on by this pulse. Unlatching is provided by the same timing pulse in the absence of an output from the signal comparators. The outputs of the latching circuits are combined in gates and run through buffer amplifiers.
to control the gate drivers and provide the reset and sample pulses. The overall effect of this is to provide a reset pulse and Y-input sample when the comparators indicate that the first operational amplifier output is outside of preset limits. If, after 75 microseconds the comparators indicate that the integrator output is back within limits, the reset pulse and Y-input pulse are terminated. If the output is not back within limits, the pulses are maintained for another 75 microseconds. A 5 microsecond period is used to determine whether or not to continue the pulses. This process of deciding every 80 microseconds is continued until the output of the first operational amplifier is within limits. A continuous reset voltage can thus be applied if necessary to cause the operational amplifier’s output to return inside the preset limits. A slightly larger rate of change of the X-input can thus be accepted before the amplifier is driven outside the limits of normal operation. The use of a longer reset pulse also allows use of X inputs with larger rates of change by providing more reset current in a given length of time, if required, than can the original design.

The logics of the reset pulse generation for EGI-1 and -2 were realized somewhat differently. EGI-1 used a crystal controlled oscillator as a clock, and nonstandard discrete-component logic circuits were used to perform the necessary digital operations. EGI-2 uses a tuning-fork oscillator as a clock and Fairchild Micrologic commercial integrated circuits to form all digital functions. Although this was the first use of integrated circuits by this group, it was felt that design time would be reduced and reliability of the completed unit enhanced by this application.

The temperature controlled environments provided in EGI-1 and EGI-2 are completely different. In EGI-1, an oven, already required for the clock oscillator, was used to house the reset pulse generators, the Y-input sampling switches, and the signal comparators. A transistorized temperature control amplifier was used with a thermistor bridge to sense oven temperature. EGI-2 was designed to utilize Frigistors, solid state Peltier-effect cooling devices, to maintain the temperature of a copper bar at 10°C. The same heat-sensitive elements as above, with the exception of the crystal oscillator, were mounted on the bar and enclosed in polystyrene foam. A thermistor bridge was used with a transistor amplifier to sense and control the temperature of the bar. The use of a low temperature is somewhat advantageous as it
reduces leakage in the FETs used for reset pulse generation and Y-input sampling. Heat from the Frigistors is dissipated by finned heat sinks.

An additional difference between EGI-1 and EGI-2 designs was in omission from the latter of two extra operational amplifiers for furnishing plus and minus Y input signals. It was decided that since the system would normally be used with a general-purpose analog computer, a pair of uncommitted computer amplifiers could be used to furnish these inputs.

The EGI-1 was originally designed to operate in conjunction with a Berkeley EASE computer, which has different relay logic for the set, reset, and compute functions than the Donner 3100 system subsequently used. A separate converter box was constructed to allow the EGI-1 to operate with the Donner computer for final checkout and evaluation. On the other hand, the EGI-2 was designed from the start to operate with the Donner computer, and the relay logic reflects the necessary changes. The logic of the Donner computer utilizes independent relays, each energized separately, for the switching necessary for each mode of the computer, e.g., "Reset." The Berkeley computer uses two relays and energizes them either separately or in combination to provide the switching for the computer modes. The internal mode relays of the EGI-2 have been connected to imitate the operation of the computer relays in their own operation.
IV. MODIFICATIONS MADE TO EGI-2 DESIGN

In checkout, trouble-shooting, and preliminary operation of EGI-2, several changes from the design described above were made. For example, diodes had to be added to isolate the EGI relay logic in order to eliminate sneak paths in the initial design.

The integrated-circuit signal comparators (ΔX detectors) were found to oscillate at a frequency of about 30 megahertz when biased near the switching point. These high-frequency oscillations caused spikes to appear on the output of the operational amplifiers. Standard techniques of interference suppression were used to reduce the spikes to a level that did not interfere with operation of the unit appreciably. Inductance-capacitance filters were added to the power supply leads, the input lead was shielded, and resistance-capacitance filters were added to the output leads. Capacitors were added to shunt radio-frequency signals to ground on several of the adjustment potentiometers. The input impedance of the comparators also changed in going from one state to another, and this affected the first operational amplifier until separate filters from the amplifier output to each comparator were added.

The reset pulse switches and Y-sampling switches required additional capacitance and resistance in parallel with the diodes used in the drive lines, so as to provide fast switching and sufficient current to bias the gate of the FET to zero volts with respect to the channel. These additional components interfered with the insulation for the cold plate and in the evaluation tests the unit was operated with the cold plate at room temperature.

The Y-input sampling switches were found to have a large signal-dependent transient and were manipulated into a shunt switch configuration which seemed to reduce the transient to somewhat more manageable proportions, allowing it to be compensated by external means. The FET switches were manipulated into shunt form by (1) disconnecting the common node of the FETs from the input to the second operational amplifier, (2) reconnecting this common node to ground, (3) inserting a resistor into the Y signal line to each FET, and (4) inserting a second resistor from this same point on each FET to the input of the amplifier. These steps could all be accomplished external to the EGI, with the exception of grounding the common node of the FETs. The change required the addition of auxiliary circuitry (Card No. 3) in order to invert the
FET driving pulses for the new configuration. The reset pulse switches were found to be satisfactory, inasmuch as the constant voltage to be switched induced no transient variations and the difference in amplitude could be compensated for by adjustment of the switched voltage.

The logic of the reset pulse generation in EGI-2 was changed by gating the previous logic output with the 5 microsecond pulse generated earlier. This was done when it was found that during this period the logic would occasionally pick up the high-frequency oscillations from the signal comparator and produce an undesired reset pulse. Noise reduction techniques were unsuccessfully tried before settling on this method of reducing the undesired reset pulses. The result of this change is to remove the dc reset current capability originally designed in. The reset pulse is now 75 microseconds long and one can be delivered every 80 microseconds. This still provides improvement over EGI-1 by a factor of almost two.

The original design of EGI-2 incorporated JK flip-flops in the reset pulse logic. Unfortunately, the Fairchild μL-92329 units obtained for this application failed to operate with set and clear inputs, although they operated well with toggle inputs. Accordingly, their use was continued in the clock count-down circuit, but a dual three-input gate connected to give an RS flip-flop equivalent replaced them in the reset pulse logic.

It was decided that EGI-2 equipment modifications would be halted at this point, because of the pressure of the terminating contract and the fact that any further possibilities for improvement would involve considerably more effort.

Photographic views of the EGI-2 systems as last modified are shown in Figure 5, with the various controls identified. Schematics for the various circuit cards and the cold plate are included in the Appendix, as are an overall system diagram and a reset pulse logic diagram. These drawings also include the last modifications made on the unit.
Figure 5. Front and Rear Views of EJI-2.
V. CALIBRATION AND OPERATION

Calibration of the EGI-2 is performed in the following sequence of steps.

1. Adjust reset pulse thresholds.
2. Adjust Operational Amplifier No. 1 for zero drift.
3. Adjust reset pulse amplitude.
4. Adjust Operational Amplifier No. 2 for zero drift.
5. Adjust compensation for error in Y sampling switches.

The reset thresholds may be set by monitoring the output of Amplifier No. 1 at J102 and the reset pulses at J101 and adjusting the comparator reference voltages with rear-panel controls R132 and R153. An adjustable output from Amplifier No. 1 is provided through the use of front-panel control R122. The EGI is placed in the reset mode and the output of Amplifier No. 1 adjusted to the desired threshold voltage by this means. The comparator reference for that polarity is then varied until reset pulses appear.

Control of the operation of the EGI-2 is usually accomplished by slaving the unit to an analog computer. Provision has been made, however, to place the EGI in the operating mode for purposes of calibration. With the \( \Delta X \) Operate/Zero switch (S103) thrown to zero position, the EGI-2 is left in the operating mode but has the X-input grounded internally. This allows zero adjustment of the first operational amplifier while in the compute mode (using the pot marked Amp. 1 Zero at upper right on the front panel). Control R122 is then set to give the same voltage at the output of Amp. 1 in the reset mode as occurs when the amplifier is first switched to the operate mode. These voltages may be observed at J102.

The above procedure is intended to minimize the transient that occurs when the unit is switched from reset to operate mode. If any measurable voltage drop exists between the EGI and the computer ground system, Amplifier No. 1 can be adjusted using the computer ground as reference by throwing the X Operate/Cal. switch (S104) to the calibrate position.

Reset pulse amplitudes may be adjusted by observing the reset pulses at J101 and varying the rear-panel controls R114 and R117 until they give equal pulses of the desired amplitude. The Reset Pulses On switch (S107) should be thrown to the off position for this adjustment.
Zero adjustment of the second operational amplifier can be performed next, by varying the pot marked Amp. 2 Zero at upper right on the front panel while monitoring the output at J105. (Again, the X-input should have been grounded by means of S103 and S104.) If additional zero adjustment is needed, a coarse control, R146, is available at the rear panel.

The error in the Y-input sampling switches is compensated for by summing into the positive and negative Y-inputs, at the computer, separately adjustable voltages. Using a cosine wave as X-input and with the Y-inputs grounded, these voltages are adjusted to give as small an EGI output voltage as possible over a desired computing period.
VI. TEST RESULTS

Evaluation tests performed on EGI-2 were of the simplest types. The error encountered was sufficiently large to be observed without resorting to special measuring procedures. Curves were run for various X- and Y-input signals in order to establish that the EGI-2 was actually integrating and to determine roughly the contribution of each of several possible sources of error. Most of the tests were made using a cosine wave for the X-input and a constant for the Y-input. In view of the fact that performance errors with EGI-2 were found to be consistently larger than had been obtained with EGI-1, it was decided that further efforts at evaluation of the new equipment in its present form would not be justified.

Figure 6 shows a typical test run on the EGI-2. Results of many such runs were reduced to tabular form, and curves of error in millivolts per second of computing time (\(e/T\)) were plotted as a function of amplitude (A) and radian frequency (\(w\)) of the cosinusoidal X-input signal; these are shown in Figure 7. From other measurements the error as a function of Y-input voltage level (B) was found to have the form shown in Figure 8.

The curves presented in Figure 7 show that EGI-2 error is relatively constant over a range of the X-input amplitude-frequency product (Aw) from 0 to 25 volts per second. The error then increases rapidly. The exact location of the zero on these curves is dependent on the precise voltages summed into the positive and negative Y-inputs to compensate for this error. The error appears to be due to the switching transients of the Y-input sampling switches which are dependent on the amplitude of the Y-inputs. At X-input amplitude-frequency products of approximately 150 volts per second, the reset pulses can no longer supply the necessary charge to maintain the output of Amplifier No. 1 within the normal operating region and insufficient Y-input samples are taken to maintain correct operation.

The plot of error as a function of Y-input amplitude, displayed in Figure 8, reveals that the error is essentially linear but with a discontinuity of slope at zero. The linearity may be attributed to the leakage of the shunt switches used as Y-input sampling switches. The shunt switch forms a voltage divider with the FET resistance to ground and the series resistance to the signal source. The sudden change of slope may be attributed to the compensating voltages summed into the positive and negative Y-input channels. The error
Note: EGI-2 performance error (ε) represented by departure of curves from horizontal baseline.

Figure 6. Typical Test Runs for Evaluation of EGI-2 Performance.
Figure 7. EGI-2 Error Vs. X-Input Amplitude and Frequency (Constant Y-Input).

Note: Compensation performed at $X = 5 \cos 5T$, $Y = 0 V$.
Figure 8. EGI-2 Error Vs. Constant Y-Input Level (Fixed Sinewave X-Input).
in this case is actually the difference in the error contributed by the positive and negative sampling switches. The compensating voltage is a constant and always adds to or subtracts from the Y-input voltage as the Y-input voltage is varied.
VII. DISCUSSION AND CONCLUSIONS

The FET gates used for sampling the Y-input signal in the present EGI system seem to be the major source of trouble remaining. The preceding study of electronic switches treated the series-shunt FET gate but the operating transients were of relatively low amplitude and the gate was considered to be feeding into an inverting amplifier, not an integrator, in that application. There will always be some switching transient, and it will be summed into a following integrator to produce an increasing error. The transient appears to be due to the junction capacitance of the FET gate-to-channel junction. Charging currents for this capacitor appear in the channel. The drive network also produces error in the form of pulses into the gate.

The FET gate-to-channel junction capacitance is variable both with the applied voltage and the temperature. Direct capacitive compensation was tried with little improvement. Even if this approach were to be properly executed, however, only the constant portion of the capacitance could be compensated for. The temperature variations would be minimized by the low thermal resistance connection to the cold plate, but the voltage sensitivity would be uncompensated for under the present scheme of operating the FET switches.

An alternative scheme that appears promising would utilize an amplifier with positive unity gain to provide an isolated voltage equal to the FET channel voltage. The driving circuit would then deliver a voltage clamped at this level, or at a constant voltage above or below this level, to provide turn-on and turn-off capacity. This would always apply the same voltage across the gate-to-channel junction, and hence reduce the effects of the voltage-sensitive capacitance. An additional benefit would be to raise the permissible input levels to the limits of the amplifier range or the breakdown voltage of the FET, less the bias voltage used. With the 2N4091s presently used, the bias voltage is 12 volts, and the junction breakdown voltage is 40 volts. This gives an input range of ±28 volts, with no safety margin, assuming the amplifier not to be the limiting factor, instead of the ±14 volts with no safety margin now achieved. Further investigation of this method for driving FET switches may be performed as part of a proposed academic program.

The possibility of introducing more sophisticated compensations for the errors encountered in the reset and sampling switches should also be con-
sidered. The well defined form of the error associated with the Y-input amplitude should allow for compensation by introducing various constant and signal-dependent voltages, for example.

The incorporation of integrated circuits into the unit was not an unmixed blessing, for several circuits failed to work as expected due to loading problems and failure of received devices to perform as specified. Unfamiliarity with the operation of the integrated circuits used accounted for some logic errors as well. Flexibility of the circuits used was very good, although inferior to that of discrete circuitry. Buffers were required on all outputs to external circuitry. Reliability of the circuits used was excellent, once the logic had been debugged. These or similar integrated circuits would be chosen again, if a new EGI were to be built.

The integrated-circuit differential comparators were not so well conceived and executed. One of the units oscillated at about 30 megahertz when biased near the switching point and demonstrated a noticeable hysteresis effect in switching point. The oscillation caused major troubles in other parts of the EGI. The hysteresis caused one reset pulse generator to deliver several pulses consecutively, when only a single pulse was required. These units caused considerably more trouble than those of EGI-1 and a return to the original design, or other improved designs, would seem desirable.

The Frigistor cooling method worked well and gave no trouble, although not used in the final evaluation tests because of insulation problems. (The modifications necessary to the circuitry mounted on the cold plate caused the insulation space to be occupied by additional circuit components.) This method of temperature control should be a desirable addition to the EGI, as should the practice of mounting temperature-sensitive components on a cold plate. Although the temperature control as such was not used in the evaluation tests, the low thermal resistance between components mounted on the cold plate, in conjunction with the considerable thermal inertia of the cold plate, undoubtedly reduced considerably the effects of ambient temperature variations.

The mechanical design of the cold plate left much to be desired from the standpoint of maintainability, however. It was a major operation to change the circuitry mounted there. A future design should make provision for removal of the cold plate, or sections of it, after wiring. The circuitry mounted here forms part of three separate circuits. Low thermal resistivity
is needed among components in the same circuit, and low thermal resistivity is needed from each circuit to the Frigistors. Bolt-on sections with connectors might offer considerable improvement in this area.
Appendix
EQUIPMENT DIAGRAMS

The following diagrams pertaining to the EGI-2 system are presented in this appendix:

A-2. Reset Pulse Logic.
A-4. Circuit Card 1 Schematic.
A-10. Cold Plate Schematic.
Figure A-1. Analog System Diagram.
Figure A-2. Reset Pulse Logic.
Figure A-3. Temperature Control Amplifier.

A = TO COLD PLATE
B = TO HEAT SINK

Figure A-4. Circuit Card 1 Schematic.
Figure A-5. Circuit Card 2 Schematic.

Figure A-6. Circuit Card 3 Schematic.
Figure A-7. Microcircuit Card 1 Logic Diagram.

Figure A-8. Microcircuit Card 2 Logic Diagram.
Figure A-9. Microcircuit Card 3 Logic Diagram.

Figure A-10. Cold Plate Schematic.


TECHNICAL NOTE NO. 15
Research Project A-588

ANALOG SIMULATION OF WIND TURBULENCE

By David L. Finn

Prepared for
George C. Marshall Space Flight Center
Huntsville, Alabama

Contract No. NAS8-2473

(Development of New Methods and Applications of Analog Computation)

1 June 1967

School of Electrical Engineering
GEORGIA INSTITUTE OF TECHNOLOGY
Atlanta, Georgia
ANALOG SIMULATION OF WIND TURBULENCE

By

David L. Finn

TECHNICAL NOTE NO. 15

on

Contract No. NAS8-2473

(Development of New Methods and Applications of Analog Computation

For

GEORGE C. MARSHALL SPACE FLIGHT CENTER

Huntsville, Alabama
ABSTRACT

A mathematical model of random wind velocity is presented for use in the synthesis of an analog computer network to simulate wind turbulence. A synthesis technique, called the covariance-expansion method, is applied to the mechanization of the model. The output of the analog computer network simulates the effect of wind turbulence on a vehicle as it moves on an arbitrary path in space. The inputs to the analog computer network are (a) a Gaussian white-noise random process and (b) appropriate functions of time characterizing the variable position of the moving vehicle.
# CONTENTS

<table>
<thead>
<tr>
<th>Title Page</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>ii</td>
</tr>
<tr>
<td>LIST OF ILLUSTRATIONS</td>
<td>iv</td>
</tr>
<tr>
<td>I. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>II. A MATHEMATICAL MODEL FOR RANDOM WIND VELOCITY</td>
<td>3</td>
</tr>
<tr>
<td>2-1. Definition of Coordinate System</td>
<td>3</td>
</tr>
<tr>
<td>2-2. Assumption 1--Magnitude and Direction</td>
<td>5</td>
</tr>
<tr>
<td>2-3. Assumption 2--Variance</td>
<td>5</td>
</tr>
<tr>
<td>2-4. Assumption 3--Correlation Coefficient</td>
<td>6</td>
</tr>
<tr>
<td>2-5. Assumption 4--Mean Value</td>
<td>7</td>
</tr>
<tr>
<td>2-6. Assumption 5--Factorization of Correlation Coefficient</td>
<td>7</td>
</tr>
<tr>
<td>III. EXPERIMENTAL DETERMINATION OF THE PARAMETERS OF THE MATHEMATICAL MODEL FOR RANDOM WIND VELOCITY</td>
<td>9</td>
</tr>
<tr>
<td>3-1. Introduction</td>
<td>9</td>
</tr>
<tr>
<td>3-2. Determination of Mean Value</td>
<td>9</td>
</tr>
<tr>
<td>3-3. Determination of Variance</td>
<td>10</td>
</tr>
<tr>
<td>3-4. Determination of the Correlation Coefficient</td>
<td>10</td>
</tr>
<tr>
<td>IV. MECHANIZATION OF THE MATHEMATICAL MODEL</td>
<td>13</td>
</tr>
<tr>
<td>4-1. Introduction</td>
<td>13</td>
</tr>
<tr>
<td>4-2. The Covariance-Expansion Synthesis Method</td>
<td>14</td>
</tr>
<tr>
<td>4-3. The Mechanization System</td>
<td>21</td>
</tr>
<tr>
<td>V. MECHANIZATION OF A SIMPLE MODEL</td>
<td>23</td>
</tr>
<tr>
<td>5-1. Introduction</td>
<td>23</td>
</tr>
<tr>
<td>5-2. Application of the Synthesis Procedure</td>
<td>23</td>
</tr>
<tr>
<td>5-3. Comparison with Experimental Wind Profiles</td>
<td>27</td>
</tr>
<tr>
<td>VI. CONCLUSIONS</td>
<td>35</td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td>36</td>
</tr>
</tbody>
</table>
# LIST OF ILLUSTRATIONS

<table>
<thead>
<tr>
<th>Figure</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1. Coordinates Describing Position and Direction of Wind Velocity $w$.</td>
<td>4</td>
</tr>
<tr>
<td>4.1. Mechanization System Derived by the Covariance-Expansion Synthesis Procedure.</td>
<td>22</td>
</tr>
<tr>
<td>5.1. Analog Computer Network for the Generation of the Random Wind Velocity Modeled in Chapter V.</td>
<td>28</td>
</tr>
<tr>
<td>5.2. Mean Value of the Horizontal Component $w_x$ of Wind Velocity.</td>
<td>29</td>
</tr>
<tr>
<td>5.3. Standard Deviation of the Horizontal Component $w_x$ of Wind Velocity.</td>
<td>30</td>
</tr>
<tr>
<td>5.4. Correlation Coefficient of the Horizontal Component $w_x$ of Wind Velocity.</td>
<td>31</td>
</tr>
<tr>
<td>5.5. Wind Velocity Profiles Generated by the Analog Computer Network of Figure 5.1.</td>
<td>33</td>
</tr>
<tr>
<td>5.6. Experimental Wind Velocity Profiles.</td>
<td>34</td>
</tr>
</tbody>
</table>
I. INTRODUCTION

This technical note summarizes the most important results of the work on nonstationary random processes that has been carried out under Georgia Tech Research Project No. A-588 for the Flight Simulation Branch, Computation Division, of George C. Marshall Space Flight Center. A mathematical model of random wind velocity is presented, and the previously developed "covariance-expansion" synthesis procedure is applied to the mechanization of this mathematical model with an analog computer network. The structure of the mathematical model has been selected to minimize the amount of experimental wind data necessary to determine the parameters of the model and to avoid excessive complexity of the analog computer network that is used to mechanize the model.

The mathematical model characterizes the three components of the vector wind velocity as Gaussian random processes that are dependent on time and on spatial position. A procedure is shown for the synthesis of an analog computer network having as outputs three variables that approximate the components of the vector wind velocity at any specified time and position in space. These outputs simulate the effect of random wind turbulence on a rocket or other aerospace vehicle in flight. The analog computer network has as inputs a Gaussian white-noise random process, a function of time representing the instantaneous altitude of the moving vehicle, and a function of time representing the scalar velocity of the vehicle. A single analog computer network is used in the simulation. It is not necessary to synthesize a different network for each different flight path.

The covariance-expansion synthesis method has been discussed in detail in Project A-588 Technical Notes Nos. 3 and 11 (see Bibliography at end of this report). Details of proof of validity for the method are therefore not included in the present summary. Only those steps are included that are necessary for the implementation of the procedure.

Chapter V of this technical note discusses the mechanization of a simple mathematical model of one horizontal component of wind velocity. This mechanization permits the simulation of the wind disturbance that affects a vehicle moving on an arbitrary path in three-dimensional space. The parameters of this model have been determined by use of experimental wind data. Wind profiles generated by the analog computer network are shown.
for comparison with actual wind profiles determined by radar observation of an ascending Jimsphere balloon.
II. A MATHEMATICAL MODEL FOR RANDOM WIND VELOCITY

In this chapter a mathematical model of random wind velocity is presented. Wind velocity is characterized as a vector-valued Gaussian random process depending on time and spatial position. The structure of the mathematical model has been selected to minimize the amount of experimental wind data necessary to determine the parameters of the model and to avoid excessive complexity of the analog computer network that is used to mechanize the model.

2-1. Definition of Coordinate System

Wind velocity \( \mathbf{w} \), expressed in meters per second, will be characterized as a vector-valued random process having three components--\( w_x \), \( w_y \), and \( w_z \). The magnitude of the wind velocity will be denoted as \( w \). The random process \( \mathbf{w} \) is assumed to depend on time and on three coordinates describing spatial position. Time \( t \) is expressed in seconds. Position is described by a rectangular coordinate system, with the axes chosen in a somewhat unorthodox fashion for reasons of convenience which will be apparent later. The positive \( y \) axis, indicating altitude, is directed vertically upward from the surface of the earth. The \( x \) axis is chosen to be positive toward the east, while the \( z \) axis is taken positive toward the north. The spatial coordinates \( x, y, \) and \( z \) are expressed in kilometers.

In this technical note, two variations of a mathematical model for characterizing vector wind velocity are presented. In the first of these variations it is advantageous to describe the direction of the vector wind velocity \( \mathbf{w} \) by two angles, \( \theta \) and \( \gamma \). As is shown in Figure 2.1, \( \theta \) is the angle between the positive \( x \) axis and a radius vector in the horizontal \( x-z \) plane. This angle is measured in radians and is taken to be positive in the direction of rotation from the positive \( x \) axis to the positive \( z \) axis. The radius vector is directed either in the same direction as the projection of \( \mathbf{w} \) in the horizontal \( x-z \) plane or in the direction opposite to this projection. As is shown in Figure 2.1, \( \gamma \) is the angle between the horizontal radius vector just defined and the vector wind velocity \( \mathbf{w} \). This angle is measured in radians and is taken to be positive in the direction of rotation from the radius vector toward the positive \( y \) axis.
The angle $\phi$ describes the vertical orientation of the vector $w$, while the angle $\theta$ is primarily related to the horizontal orientation of $w$. It should be noted that the horizontal orientation of $w$ is not uniquely determined unless both $\phi$ and $\theta$ are specified. For implementation of the mathematical model to be presented it is necessary that no constraints be imposed on the range of values that may be assumed by either directional angle.

The wind velocity may be expressed in general as

$$w = w_x u_x + w_y u_y + w_z u_z \quad ,$$

where $u_x$, $u_y$, and $u_z$ are unit vectors in the positive $x$, $y$, and $z$ directions. An inspection of Figure 2.1 shows that at any position $(x,y,z)$ the components of the vector wind velocity are given by the relationships

$$w_x = w \cos \phi \cos \theta; \quad w_y = w \sin \phi; \quad w_z = w \cos \phi \sin \theta \quad .$$

Figure 2.1. Coordinates Describing Position and Direction of Wind Velocity $w$. 
In order to further define the mathematical model for wind velocity considered as a vector-valued random process in time, assumptions are made concerning various statistical properties of its magnitude and direction characteristics, as noted in the next five sections.

2-2. Assumption 1—Magnitude and Direction

First Variation: The initial simplifying assumption to be made is that the magnitude of the vector wind velocity is statistically independent of its direction, and furthermore that the directional angle \( \theta \) is statistically independent of the directional angle \( \phi \). (Stated in another way, it is assumed that how hard the wind is blowing is not affected by the direction in which it is blowing.) It is assumed, in fine, that the three variables \( w, \theta, \) and \( \phi \) may be characterized as statistically independent, Gaussian random processes.

Second Variation: For this variation, it is assumed that the three rectangular components \( w_x, w_y, \) and \( w_z \) of the vector wind velocity may be characterized as statistically independent, Gaussian random processes.

Time limitations have not permitted an investigation of possible relationships and differences between the above two variations of Assumption 1. The second variation provides a less complex mechanization system when used with the synthesis procedure presented in this technical note.

2-3. Assumption 2—Variance

According to Assumption 1, the scalar wind-velocity magnitude \( w \) is characterized as a Gaussian random process \( w(x,y,z,t) \) depending on position and time. Let a random variable \( w_1 \) be defined by specification of \( w \) for a single position \( (x_1,y_1,z_1) \) and time \( t_1 \) as

\[
w_1 = w(x_1,y_1,z_1,t_1)
\]  

(2.3)

The variance of \( w_1 \) is defined as

\[
\sigma_{w_1}^2 = \mathbb{E} \left( (w_1 - \mathbb{E}[w_1])^2 \right)
\]  

(2.4)
where $E$ is the expected value operator.

It is assumed that the variance of the wind velocity is a function $\sigma_{w_1}^2(y_1)$ only of the altitude $y_1$ and does not vary with $x_1$, $z_1$, and $t_1$.

Stated in another way, it is assumed that the magnitude of the random variations of the wind velocity does not change with time or with alterations of position in a fixed horizontal plane.

An identical assumption is to be made for the five other random processes $\theta$, $\psi$, $w_x$, $w_y$, and $w_z$ listed in the two variations of Assumption 1.

2.4. Assumption 3--Correlation Coefficient

According to Assumption 1, the scalar wind-velocity magnitude $w$ is characterized as a Gaussian random process $w(x,y,z,t)$ depending on position and time. Let two random variables $w_1$ and $w_2$ be defined by specification of $w$ for any two positions and times as

$$w_1 = w(x_1, y_1, z_1, t_1) \quad \text{and} \quad w_2 = w(x_2, y_2, z_2, t_2)$$

(2.5)

The distance $s$ between the two point locations $(x_1, y_1, z_1)$ and $(x_2, y_2, z_2)$ will in general be given by

$$s = \left[ (x_2-x_1)^2 + (y_2-y_1)^2 + (z_2-z_1)^2 \right]^{\frac{1}{2}}$$

(2.6)

Similarly, the time interval $\tau$ between the two instants $t_1$ and $t_2$ may be expressed in general as

$$\tau = |t_2 - t_1|$$

(2.7)

The correlation coefficient $\rho$ of the random variables $w_1$ and $w_2$ is defined as

$$\rho = \frac{E[(w_1 - E[w_1])(w_2 - E[w_2])]}{\sigma_{w_1} \sigma_{w_2}}$$

(2.8)
Here, $\sigma_w$ is the standard deviation of the random variable $w_1$—i.e., the square root of the variance of $w_1$ as was defined by Equation (2.4). The correlation coefficient is a measure of the correlation or linear dependence between the random variables $w_1$ and $w_2$.

It is assumed that the correlation coefficient $\rho$ as defined above is a function $\rho(s,\tau)$ only of the two variables $s$ and $\tau$. Stated in another way, the correlation of the wind velocities at two different positions and times depends only on the distance between the two positions and on the time interval between the two time instants.

An identical assumption is made for the five other random processes $\theta, \varphi, w_x, w_y,$ and $w_z$ listed in the two variations of Assumption 1.

2-5. Assumption 4—Mean Value

The mean value $m_w$ of the random process $w(x,y,z,t)$ characterizing the scalar wind-velocity magnitude is defined to be

$$m_w = \mathbb{E}[w(x,y,z,t)].$$

It is assumed that the mean value of the wind velocity is a function $m_w(y)$ only of the altitude $y$ and does not vary with $x$, $z$, and $t$. Stated in another way, it is assumed that the average value of the wind velocity does not change with time or with alterations of position in a fixed horizontal plane.

An identical assumption is made for the five other random processes $\theta, \varphi, w_x, w_y,$ and $w_z$ listed in the two variations of Assumption 1.

2-6. Assumption 5—Factorization of Correlation Coefficient

It is assumed that the correlation coefficient $\rho(s,\tau)$ as defined in Assumption 3 above may be expressed as the product of a function depending only on $s$ multiplied by a function depending only on $\tau$. That is,

$$\rho(s,\tau) = \rho_1(s) \rho_2(\tau).$$

This assumption is made for the correlation coefficients of all six random processes $w, \varphi, \theta, w_x, w_y,$ and $w_z$ listed in the two variations of Assumption 1.
This fifth assumption appears to be considerably more restrictive than the first four. It is postulated because its use permits a simplification both in the experimental determination of the parameters of the mathematical model of the wind velocity and in the analog computer mechanization of the model.
3-1. Introduction

In Chapter II, scalar wind-velocity magnitude \( w \) has been characterized as a random process \( w(x,y,z,t) \) depending on both space and time coordinates. As was mentioned previously, this model has been constructed using assumptions selected to simplify the experimental determination of parameters of the model and also to simplify the physical mechanization of the model by use of an analog computer network. The basic assumption is that the wind velocity is represented by a Gaussian random process. Thus, the process is completely specified by a determination of its first and second order moments. The experimental determination of these moments—as represented by the mean, variance, and correlation coefficient—is discussed in this chapter.

A general investigation of the extent of validity of the mathematical model would be of very substantial magnitude and has not been undertaken as part of the present research. However, a calculation of parameters using experimental data has been carried out using a small number of wind profiles. These results are presented in Chapter V. Also, in that chapter some simulated wind profiles generated by an analog computer network are shown for comparison with experimentally determined wind profiles.

3-2. Determination of Mean Value

According to Assumption 4 of Chapter II, the mean value \( m_w(y) \) of the scalar wind velocity is a function only of the altitude \( y \) and does not vary with \( x, z, \) and \( t \). The mean value at a fixed altitude may be estimated by the use of \( N \) measured samples \( W_k, k = 1,2,...,N \), of the instantaneous wind velocity taken at the specified altitude. The estimate of the mean value \( m_w(y) \) may be taken to be the sample mean

\[
M(y) = \frac{1}{N} \sum_{k=1}^{N} W_k \quad .
\] (3.1)
3-3. Determination of Variance

According to Assumption 2 of Chapter II, the variance $\sigma_w^2(y)$ of the wind velocity is a function only of the altitude $y$ and does not vary with $x$, $z$, and $t$. The variance at a fixed altitude may be estimated by the use of $N$ measured samples $W_k$, $k = 1, 2, \ldots, N$, of the instantaneous wind velocity taken at the specified altitude. The estimate of the variance $\sigma_w^2(y)$ may be taken to be the sample variance

$$V^2(y) = \frac{1}{N} \sum_{k=1}^{N} \left( W_k - M(y) \right)^2.$$  \hspace{1cm} (3.2)

Here $M(y)$ is the sample mean defined by Equation (3.1). This is calculated using the same $N$ samples $W_k$, $k = 1, 2, \ldots, N$, that appear explicitly in Equation (3.2).

3-4. Determination of the Correlation Function

Two random variables $w_1$ and $w_2$, representing instantaneous wind velocity at two positions and times, are defined in accordance with Equations (2.5) as

$$w_1 = w(x_1, y_1, z_1, t_1)$$

$$w_2 = w(x_2, y_2, z_2, t_2).$$  \hspace{1cm} (3.3)

The distance between the two positions $(x_1, y_1, z_1)$ and $(x_2, y_2, z_2)$ is given by the variable $s$ defined in Equation (2.6), and the interval between the two time instants $t_1$ and $t_2$ is given by the variable $\tau$ defined in Equation (2.7).

The correlation coefficient $\rho$ of the random variables $w_1$ and $w_2$ is defined by Equation (2.8). According to Assumption 3 in Chapter II, the correlation coefficient is a function $\rho(s, \tau)$ dependent only on the two variables $s$ and $\tau$. The correlation coefficient for the two fixed positions and times may be estimated by the use of two sets of $N$ each experimental samples of the instantaneous wind velocity--viz., $W_k(x_1, y_1, z_1, t_1)$ and $W_k(x_2, y_2, z_2, t_2)$, $k = 1, 2, \ldots, N$. The estimate of the correlation coefficient $\rho(s, \tau)$ may be taken to be
Here again, $M(y_1)$ and $M(y_2)$ are sample means as defined by Equation (3.1), while $V(y_1)$ and $V(y_2)$ are the square roots of variance estimates as defined by Equation (3.2). All of these are to be calculated using the same two sets of $N$ each samples that appear explicitly in Equation (3.4).

The use of Assumption 5 in Chapter II greatly facilitates the estimation of the function $\rho(s,\tau)$. A collection of profiles (sample functions) showing the time variation of the wind velocity experienced by a sensor at a fixed position in space can, by use of Equation (2.10), be used to approximate the function

$$
\rho(s=0,\tau) = \rho_1(s=0) \rho_2(\tau) .
$$

Here, in the calculations determining $\tau = |t_2 - t_1|$, the time $t_1$ may be fixed at any convenient value and $t_2$ may be allowed to vary. Without loss of generality $\rho_1(s=0)$ may be assumed to equal unity. When calculations are made in the manner just described by using samples taken through use of a stationary sensor, the estimate of the function $\rho_2(\tau)$ may be expressed by the estimator defined in Equation (3.4):

$$
\rho_2(\tau) = \rho(s=0,\tau) \approx P(s=0,\tau) .
$$

Next, a collection of profiles (sample functions) showing the time variation of the wind velocity experienced by a sensor moving along any specified path may be used in making an estimate of the function $\rho(s,\tau)$ for the related values of $s$ and $\tau$ that characterize the constrained path. For example, the wind sample profiles may be determined by observations on a vertically ascending balloon. Here, the coordinates $x$ and $z$ remain fixed, and the variable $s$ is calculated as the difference of two altitudes $y_1$ and
It is necessary to make observations for $N$ independent balloon ascents to accumulate the required number of samples, and it is assumed that the rate at which the balloon rises is the same for each different ascent.

Once the function $\rho(s, \tau)$ is determined for any arbitrary sensor path, the function $\rho_1(s)$ is easily determined by use of Equations (2.10) and (3.6):

$$\rho_1(s) = \frac{\rho(s, \tau)}{\rho_2(\tau)}$$  

$$= \frac{\rho(s, \tau)}{\rho(s=0, \tau)} . \quad (3.7)$$

The determination of the function $\rho_1(s)$ as expressed in Equation (3.7) and the determination of the function $\rho_2(\tau)$ as expressed in Equation (3.6) permit the calculation of the function $\rho(s, \tau)$ defining the correlation coefficient by use of Equation (2.10):

$$\rho(s, \tau) = \rho_1(s) \rho_2(\tau) . \quad (3.8)$$

The assumption that the wind velocity does not vary with time at any fixed position may be incorporated into the mathematical model by assuming the function $\rho_2(\tau)$ in Equation (2.10) to equal the constant unity. In this case, it is not necessary to perform the set of measurements that have been described for the stationary sensor. Here, samples taken in the manner described by use of an ascending balloon are adequate to provide estimates for the mean value, the variance, and the correlation function. For this assumption, the random process characterizing wind velocity depends upon position parameters alone.

In this chapter, the formulas related to the determination of parameters for the wind velocity model have been expressed in terms of the magnitude $w$ of the vector wind velocity. The formulas are equally applicable to the determination of the parameters characterizing the five other random processes $\theta, \psi, w_x, w_y,$ and $w_z$ introduced in the mathematical model.
IV. MECHANIZATION OF THE MATHEMATICAL MODEL

4-1. Introduction

In Chapter II the wind velocity of any position in space and at any time was characterized as a random process \( w(x, y, z, t) \). The coordinates of a vehicle or sensing element moving along an arbitrary path through space may be described by three time functions \( x(t) \), \( y(t) \), and \( z(t) \). In this case, the wind velocity experienced by the sensing element is described by the composite, time-parameter random process

\[
 f(t) = w(x(t), y(t), z(t), t) .
\] (4.1)

In this chapter the covariance-expansion method of synthesis that was presented in Project A-588 Technical Notes Nos. 3 and 11 is applied to the approximation of the random process \( f(t) \). This method has been discussed in detail in these previous technical notes; hence, no details of proof of validity for the procedure are presented here. Only those steps that are necessary for implementation of the method are included.

The analog computer network synthesized by the procedure discussed in this chapter has as outputs three variables that approximate the components of the vector wind velocity at any specified time and position in space. These outputs may be used to simulate the effect of wind turbulence on a rocket or other aerospace vehicle in flight. The analog computer network has as inputs a Gaussian white-noise waveform, a function of time representing the instantaneous altitude of the moving vehicle, and a function of time representing the scalar velocity of the vehicle.

It should be noted that the mechanization that is presented here allows for a three-dimensional movement of the vehicle, or sensing element, with an analog computer network that is identical to the one that is required for a vertical, one-dimensional motion of the sensing element. This simplified mechanization is accomplished by use of certain approximations for the distance \( s \) (see Equations (4.5) and (4.6)).

The synthesis method is presented in this chapter in terms of simulation of the wind-velocity magnitude \( w \). However, the synthesis method is equally applicable to the simulation of the five other random processes \( \Theta, \Psi, \alpha, \beta, \gamma \).
\( w_x, w_y, \) and \( w_z \) listed under Assumption I for the mathematical model in Section 2-2 of Chapter II.

4-2. The Covariance-Expansion Synthesis Method

The mean value of the wind velocity, as expressed by Equation (2.9), is assumed to be a function \( m_w(y) \) of altitude \( y \) only. This is a determinate function—not a random process. Accordingly, the mean value may be generated by a function generator having as input the variable \( y \). This being accomplished, without loss of generality the covariance-expansion method is applied to the generation of a random process \( f(t) \) having a mean value equal to zero. The total wind velocity is obtained by adding the random process \( f(t) \) to the determinate function \( m_w(y) \).

The covariance function \( r(t', t) \) of the random process \( f(t) \) is defined as

\[
r(t', t) = E\left[f(t_1) f(t_2)\right]
\]

where:
- \( t' \) = larger of \( t_1 \) and \( t_2 \)
- \( t \) = smaller of \( t_1 \) and \( t_2 \).

Implementation of the covariance-expansion method requires that the covariance function be expressed as a finite expansion in the form

\[
r(t', t) = \sum_{i=1}^{n} \phi_i(t') \sigma_i(t)
\]

The covariance function for the random wind velocity \( w \), with mean value assumed equal to zero, may be found by rewriting Equation (2.8):

\[
E[w_1 w_2] = \rho(s, \tau) \sigma_w(y_1) \sigma_w(y_2)
\]

As before, \( w_1 = w(x_1, y_1, z_1, t_1) \) and \( w_2 = w(x_2, y_2, z_2, t_2) \) represent the wind velocity at any two positions and times. An inspection of Equation (4.4) shows that this covariance function depends only on altitude, on the distance \( s \) between the two space positions \((x_1, y_1, z_1)\) and \((x_2, y_2, z_2)\), and on
the time interval \( \tau = |t_2 - t_1| = t' - t \). The covariance function, expressed in Equation (4.2), of the random process \( f(t) \) is found from Equation (4.4) by allowing the position coordinates \((x, y, z)\) to become functions of time \((x(t), y(t), z(t))\) describing the varying position of the sensing element.

In order that the covariance function of \( f(t) \) may be expressed in the required form of Equation (4.3) it is expedient to approximate the distance \( s \) between any two positions of the moving sensing element in terms of the scalar velocity \( v(t) \) of the element. This provides the incidental benefit of allowing a mechanization designed for one-dimensional motion of the sensing element to be used without modification for three-dimensional motion.

Two methods of approximation of the distance \( s \) will be presented. First \( s \) may be approximated by the expression

\[
s \approx \int_{t}^{t'} v(\lambda) \, d\lambda - \int_{0}^{t} v(\lambda) \, d\lambda .
\]

Here, \( v(t) \) is the scalar velocity of the sensing element. It is noted that the first term of the expression is a function of \( t' \) only, and the second term is a function of \( t \) only.

The right-hand side of (4.5) exactly represents total distance traveled along the path of motion of the sensing element during the time interval \( \tau = t' - t \). This is obviously a good approximation for the distance between the positions at times \( t' \) and \( t \) provided motion is almost along a straight line.

The correlation coefficient \( \rho(s, \tau) \) tends to zero as distance \( s \) and time interval \( \tau \) increase. For validity, the approximation of (4.5) requires that the movement of the sensing element be approximately along a straight line either for all distances \( s \) that are small enough that \( \rho(s, \tau) \) is appreciably different from zero or for all time intervals \( \tau \) small enough that \( \rho(s, \tau) \) is appreciably different from zero. For larger values of \( s \) or \( \tau \) an accurate calculation of \( s \) is not needed because all calculations of the correlation coefficient provide a value very nearly equal to zero.

A second possible approximation for the distance \( s \) is obtained by slightly modifying the relationship that expresses distance as the product of velocity multiplied by time. Thus,
It is noted that one of the two product terms in (4.6) involves only $t'$, and the other term involves only $t$.

The approximation of (4.6) is more restrictive than that of (4.5), but in some cases is expected to lead to a less complex mechanization. For validity, the approximation of (4.6) requires that the movement of the sensing element be approximately at constant velocity and along a straight line either for all distances $s$ that are small enough that $p(s,\tau)$ is appreciably different from zero or for all time intervals $\tau$ small enough that $p(s,\tau)$ is appreciably different from zero.

It will be assumed that the covariance function for the composite random process $f(t)$, as found by use of (4.4), can be expressed in the form shown in Equation (4.3) when one of the two approximations discussed above is utilized for the distance variable $s$. Representation in this form, either exactly or as an approximation, is necessary for the implementation of the covariance-expansion synthesis method.

It is to be emphasized that the functions $\varphi_1(t)$ and $\gamma_1(t)$ in Equation (4.3) depend explicitly on the altitude $y(t)$ and on the scalar velocity $v(t)$ of the sensing element because of the use of (4.4) and either (4.5) or (4.6) in the evaluation of $r(t',t)$.

An analog computer network is to be synthesized having three inputs: a Gaussian white-noise waveform, a function of time $y(t)$ representing altitude, and a function of time $v(t)$ representing the instantaneous scalar velocity of the sensing element. The output of the network is to be the composite random process $f(t)$ representing the wind velocity observed by the moving sensing element.

The analog computer network to be synthesized is characterized by the nth-order differential equation

$$f^{(n)} + p_{n-1}(t)f^{(n-1)} + \ldots + p_1(t)f^{(1)} + p_0(t)f = q_{n-1}(t)g^{(n-1)} + \ldots + q_1(t)g^{(1)} + q_0(t)g.$$ (4.7)

Here, $f^{(k)}$ denotes the kth derivative of the function $f$ with respect to
time. The function $g$ represents the Gaussian white-noise input to the analog computer network.

In order to avoid differentiation of the noise input $g(t)$, the $n$th-order differential Equation (4.7) may be converted into a set of $n$ first-order differential equations. To make this conversion the following definitions are used:

\[
\begin{align*}
  f(t) &= f_1(t) \\
  f_1(t) &= f_2 - a_{n-1}(t)f_1 + b_{n-1}(t)g \\
  f_2(t) &= f_3 - a_{n-2}(t)f_1 + b_{n-2}(t)g \\
  &\vdots \\
  f_{n-1}(t) &= f_n - a_1(t)f_1 + b_1(t)g \\
  f_n(t) &= -a_0(t)f_1 + b_0(t)g
\end{align*}
\]

This set can be written more concisely in matrix notation as

\[
F(t) = A(t) F + B(t) g
\]  
\[
[f] = [f_1] = H F
\]

where

\[
F = \begin{bmatrix}
  f_1 \\
  f_2 \\
  \vdots \\
  f_{n-1} \\
  f_n
\end{bmatrix} \quad B(t) = \begin{bmatrix}
  b_{n-1}(t) \\
  b_{n-2}(t) \\
  \vdots \\
  b_1(t) \\
  b_0(t)
\end{bmatrix}
\]
\[ H = \begin{bmatrix} 1 & 0 & \cdot & \cdot & \cdot & 0 & 0 \end{bmatrix} \]

\[
A(t) = \begin{bmatrix}
- a_{n-1}(t) & 1 & 0 & \cdot & \cdot & 0 & 0 \\
- a_{n-2}(t) & 0 & 1 & \cdot & \cdot & 0 & 0 \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\
- a_1(t) & 0 & 0 & 0 & \cdot & \cdot & 1 \\
- a_0(t) & 0 & 0 & 0 & \cdot & \cdot & 0 \\
\end{bmatrix}
\]

The elements \(a_k\) and \(b_k\) in (4.8) are related to the coefficients \(p_k\) and \(q_k\) in (4.7) as follows:

\[
p_k = \sum_{j=0}^{n-l-k} \frac{(n-l-j)!}{k! (n-l-j-k)!} a_{n-l-j-k} \quad (4.10)
\]

\[
q_k = \sum_{j=0}^{n-l-k} \frac{(n-l-j)!}{k! (n-l-j-k)!} b_{n-l-j-k} \cdot \quad (4.11)
\]

If the \(p_k\) and \(q_k\) are known, then (4.10) and (4.11) can be solved sequentially for the \(a_k\) and \(b_k\).

A determinant \(L\) may be defined as

\[
L = \begin{vmatrix}
f(t) & \phi_1(t) & \cdot & \cdot & \phi_n(t) \\
f'(1)(t) & \phi_1(1)(t) & \cdot & \cdot & \phi_n(1)(t) \\
\cdot & \cdot & \cdot & \cdot & \cdot \\
f^{(n)}(t) & \phi_1^{(n)}(t) & \cdot & \cdot & \phi_n^{(n)}(t) \\
\end{vmatrix} \quad (4.12)
\]

If this determinant is equated to zero it can be shown that the coefficient
of $f(k)(t)$ in the resulting expression is equal to the corresponding coefficient $p_k(t)$ in Equation (4.7). The elements $a_k(t)$ that appear in the equations of (4.8) and the matrix $A(t)$ of (4.9) can be obtained directly by using (4.10). It is noted that the elements $a_k(t)$ depend explicitly on the altitude $y(t)$ and the velocity $v(t)$ of the sensing element because of the dependence of $\phi_i(t)$ on these functions.

To complete the synthesis procedure, the elements $b_k(t)$ of (4.8) and (4.9) must be determined.

The $\phi_i(t)$ of the covariance function of (4.3) may be used in the construction of a fundamental matrix $\Phi(t)$:

\[
\Phi(t) = \begin{bmatrix}
\phi_{11}(t) & \phi_{21}(t) & \cdots & \phi_{n1}(t) \\
\phi_{12}(t) & \phi_{22}(t) & \cdots & \phi_{n2}(t) \\
\cdots & \cdots & \cdots & \cdots \\
\phi_{1n}(t) & \phi_{2n}(t) & \cdots & \phi_{nn}(t)
\end{bmatrix}
\]  

(4.13)

where

\[
\phi_{ij} = \phi_{i(j-1)} + a_{n-j+1} \phi_{il}
\]

(4.14)

Here, the $\phi_k(t)$ of Equation (4.3) are assumed to be linearly independent. However, if one of the $\phi_i(t)$ is not linearly independent, it may be expressed as a combination of the remainder and the index $n$ can be reduced by one.

It must again be emphasized that the $\phi_k(t)$ depend on the functions $v(t)$ and $y(t)$. Thus, the chain rule must be used when performing the differentiation required in (4.14). Both $\phi_k$ and its derivative depend explicitly on the functions $v(t)$ and $y(t)$.

The elements of the fundamental matrix $\Phi(t)$ can be determined by application of (4.14) once the $a_k(t)$ have been found by the procedure that has just been explained.
A matrix $R(t',t)$ may now be defined as

$$R(t',t) = \Phi(t') \Phi^T(t') \quad . \quad (4.15)$$

Here, the elements $d_{ij}(t)$ of the matrix $D(t)$ are defined in terms of the $\psi_i(t)$ and $\gamma_i(t)$ of (4.3) as

$$d_{ij}(t) = \frac{\gamma_i(t)}{\psi_i(t)} \quad \text{for} \quad i = j$$

$$= 0 \quad \text{for} \quad i \neq j \quad . \quad (4.16)$$

Also, $\Phi^T$ denotes the transpose of the matrix $\Phi$.

A matrix $R^*(t',t)$ is now defined as

$$R^*(t',t) = R^T(t,t')$$

$$= \Phi(t) \Phi^T(t') \quad . \quad (4.17)$$

Let $\Delta(t',t)$ denote the difference

$$\Delta(t',t) = R(t',t) - R^*(t',t) \quad . \quad (4.18)$$

Finally, let $\delta_{ii}(t)$ be the diagonal elements of the matrix

$$\frac{\partial}{\partial t'} \Delta(t',t) \quad . \quad \begin{vmatrix} \Delta(t',t) \\ t' = t \end{vmatrix} \quad (4.19)$$

It can be shown that

$$b_{n-1}(t) = \sqrt{-\delta_{ii}(t)} \quad , \quad i = 1,2,..n \quad . \quad (4.20)$$

The matrix $B(t)$ is now specified, thus completing the synthesis
4-3. The Mechanization System

The mechanization system for generating a nonstationary random process that approximates wind turbulence is obtained by constructing an analog computer network that realizes the differential equations of (4.8) or (4.9). A block diagram of the mechanization system is shown in Figure 4.1. It is noted that differentiation of the input white-noise waveform is not required in this system.

It is important to achieve the proper integrator initial conditions at the beginning of a computation cycle (t=0) if the output of the mechanization system is to realize the correct covariance function. Different initial conditions applied to the same mechanization system with the same position-function input may give rise to widely divergent covariance functions. The initial conditions may be altered by manipulation of the white-noise input and the position-function inputs during resetting of the analog computer just prior to t = 0. In general, it is required that the steady-state operation obtained during the reset cycle correspond to the desired initial values of the position-function inputs at the beginning of the computation interval.

The mechanization system shown in Figure 4.1 generates a single component of the vector wind velocity. Two additional networks of the same type are required if all three components are to be generated. In case the second variation of Assumption 1 of the mathematical model is used, each of the three velocity components \( w_x, w_y, \) and \( w_z \) is generated by a separate white-noise generator associated with an analog computer network of the type shown in Figure 4.1.

In case the first variation of Assumption 1 of the mathematical model is used, each of the three variables \( w, \theta, \) and \( \bar{y} \) is generated by a separate white-noise generator associated with an analog computer network of the type shown in Figure 4.1. In turn, the three velocity components \( w_x, w_y, \) and \( w_z \) are obtained from the three variables \( w, \theta, \) and \( \bar{y} \) by a mechanization of the relationships given as Equation (2.2).
Note: Double lines indicate multivariable signal flow.

Figure 4.1. Mechanization System Derived by the Covariance-Expansion Synthesis Procedure.
V. MECHANIZATION OF A SIMPLE MODEL

5-1. Introduction

The covariance-expansion synthesis method is applied in this chapter to the mechanization of a simple mathematical model intended to simulate the behavior of random wind. Wind velocity profiles generated by an analog computer network used to implement the mechanization are presented for comparison with experimentally determined wind profiles.

In the example of this chapter a single component $w_x$ of wind velocity is generated. The output $f(t)$ of the analog computer network approximates the component $w_x$ of random wind velocity that is experienced by a moving vehicle or sensing element. The sensing element may follow an arbitrary path in three-dimensional space. The three required inputs to the simulation network are a Gaussian white-noise waveform, a function $y(t)$ representing the instantaneous altitude of the sensing element, and a function $v(t)$ representing the scalar velocity of the sensing element.

5-2. Application of the Synthesis Procedure

It will be assumed that the standard deviation of the wind velocity $w$ is a constant not varying with time or position:

$$\sigma_w(y) = \sigma$$

(5.1)

It will be assumed that the correlation coefficient of the wind velocity is given by the expression

$$\rho_w(s, \tau) = e^{-\alpha s} e^{-\beta \tau}$$

(5.2)

The correlation function of $w$ as expressed by (4.4) becomes

$$E[w_1 w_2] = \rho_w(s, \tau) \sigma_w(y_1) \sigma_w(y_2)$$

$$= \sigma^2 e^{-\alpha s} e^{-\beta \tau}$$

(5.3)

By using the approximation for distance $s$ given in (4.5), the defini-
tion of \( \tau \) from (2.7) and (4.2), and the covariance function of (5.3), the covariance function of \( f(t) \) as expressed by (4.2) or (4.4) becomes

\[
r(t',t) = \text{E}[w_1 w_2] = \sigma^2 \exp \left[ -\beta t' - \alpha \int_0^{t'} v(\lambda) \, d\lambda \right] \exp \left[ \beta t + \alpha \int_0^t v(\lambda) \, d\lambda \right] .
\]  

(5.4)

This expression is seen to be in the required form of (4.3) with \( n = 1 \). The coefficients \( \phi_1(t') \) and \( \gamma_1(t) \) of (4.3) are:

\[
\phi_1(t') = \sigma \exp \left[ -\beta t' - \alpha \int_0^{t'} v(\lambda) \, d\lambda \right] .
\]  

(5.5)

\[
\gamma_1(t) = \sigma \exp \left[ \beta t - \alpha \int_0^t v(\lambda) \, d\lambda \right] .
\]  

(5.6)

The \( p_k \) coefficients of (4.7) are determined by use of the determinant \( L \) defined in (4.12):

\[
L = \begin{vmatrix}
    f(t) & \phi_1(t) \\
    f(1)(t) & \phi_1(1)(t)
\end{vmatrix} .
\]  

(5.7)

Expanding this determinant by the use of (5.5) and equating the determinant to zero, there results:

\[
f(1) + (\beta + \alpha v(t)) f = 0 .
\]  

(5.8)

A comparison of (5.8) with (4.7) shows that

\[
p_0(t) = \beta + \alpha v(t) .
\]  

(5.9)

Use of (4.10) provides the single coefficient of the \( A(t) \) matrix of (4.9):

\[
a_0(t) = p_0(t) = \beta + \alpha v(t) .
\]  

(5.10)
The single coefficient of the $\Phi(t)$ matrix of (4.13) is found by use of (4.14) and (5.5):

\[
\Phi_{11}(t) = \Phi_\perp(t) = \sigma \exp \left[ -\beta t - \alpha \int_0^t v(\lambda) \, d\lambda \right].
\]

The single coefficient of the $D(t)$ matrix of (4.15) is found by the use of (4.16), (5.5), and (5.6):

\[
D_{11}(t) = \frac{V_\perp(t)}{\Phi_\perp(t)} = \exp \left[ 2 \beta t + 2 \alpha \int_0^t v(\lambda) \, d\lambda \right].
\]

The single element matrix $R(t',t)$ of (4.15) is found by use of (5.11) and (5.12):

\[
R(t',t) = \Phi(t') D(t) \Phi^T(t) = \sigma^2 \exp \left[ -\beta t' - \alpha \int_0^t v(\lambda) \, d\lambda \right] \exp \left[ \beta t + \alpha \int_0^t v(\lambda) \, d\lambda \right].
\]

The matrix $R^*(t',t)$ of (4.17) is

\[
R^*(t',t) = R^T(t',t) = \sigma^2 \exp \left[ \beta t' + \alpha \int_0^{t'} v(\lambda) \, d\lambda \right] \exp \left[ -\beta t - \alpha \int_0^t v(\lambda) \, d\lambda \right].
\]

The difference matrix of (4.18) is found by use of (5.13) and (5.14).
\[ \Delta(t',t) = R(t',t) - R^*(t',t) \]
\[ = \sigma^2 \exp \left[ -\beta t' - \alpha \int_0^{t'} v(\lambda) \, d\lambda \right] \exp \left[ \beta t + \alpha \int_0^t v(\lambda) \, d\lambda \right] \]
\[ - \sigma^2 \exp \left[ \beta t' + \alpha \int_0^{t'} v(\lambda) \, d\lambda \right] \exp \left[ -\beta t - \alpha \int_0^t v(\lambda) \, d\lambda \right]. \] (5.15)

The matrix of (4.19) is found by use of (5.15):
\[ \frac{\partial}{\partial t'} \Delta(t',t) \bigg|_{t' = t} = \begin{bmatrix} \delta_{11}(t) \end{bmatrix} \]
\[ = \left[ -2 \sigma^2 \left( \beta + \alpha v(t) \right) \right]. \] (5.16)

Finally, the single coefficient in the B(t) matrix of (4.9) is found by use of (4.20) and (5.16).
\[ b_o(t) = \sqrt{-\delta_{11}(t)} \]
\[ = \sigma \sqrt{2 \left( \beta + \alpha v(t) \right)}. \] (5.17)

Specification of this coefficient completes the synthesis procedure.

The differential equation characterizing the mechanization system for the realization of the random wind velocity \( f(t) \) is obtained by substituting the coefficients of (5.10) and (5.17) in the equation of (5.8):
\[ \frac{d}{dt} f(t) = -a_o(t) f(t) + b_o(t) g(t) \]
\[ = -\left[ \beta + \alpha v(t) \right] f(t) + \sigma g(t) \sqrt{2[\beta + \alpha v(t)]}. \] (5.18)

-26-
The solution of this differential equation realizes the composite random process \( f(t) \). As was explained in Section 4-2, the output of a function generator may be added to this solution to produce a random process having nonzero mean value \( m_w(y) \). The block diagram of an analog computer network whose output approximates the composite random process \( f(t) \) is shown in Figure 5.1.

5-3. Comparison with Experimental Wind Profiles

Experimental wind data taken by radar observation of an ascending Jimsphere balloon was used to determine parameters for the mechanization presented in Section 5-2. Thirty experimental wind profiles were used to calculate the mean value \( m(y) \) and standard deviation \( \sigma(y) \) of the \( x \) component \( w_x \) of the wind velocity. The estimators of Equations (3.1) and (3.2) were used for these calculations. These experimental values are shown in Figures 5.2 and 5.3. The same profiles were used to calculate the correlation coefficient as a function of the vertical distance interval \( s = |y_2 - y_1| \). The estimator of Equation (3.4) was used for this calculation. The experimental values are shown in Figure 5.4.

In order to fit the mathematical model of Section 5-2 to the experimental data just discussed, the following parameter values were selected:

\[
\begin{align*}
\frac{m_w(y)}{w_x} &= \begin{cases} 
40 & \text{if } 0 \leq y \leq 13 \\
3586/7 - 6/7y & \text{if } y > 13 
\end{cases} \\
\sigma_w(y) &= 12.5 \\
\rho_w(s, \tau) &= e^{-0.1s} 
\end{align*}
\]

(5.19)

These parameter values for the mathematical model are shown as solid lines in Figures 5.2, 5.3, and 5.4.

* See Reference 7 in the Bibliography.
Figure 5.1 Analog Computer Network for the Generation of the Random Wind Velocity Modeled in Chapter V.
Figure 5.2 Mean Value of the Horizontal Component $w_x$ of Wind Velocity.
Figure 5.3 Standard Deviation of the Horizontal Component \( w_x \) of Wind Velocity.
Figure 5.4 Correlation Coefficient of the Horizontal Component $w_x$ of Wind Velocity.
An inspection of (5.19) shows that the parameter values of $\alpha = 0.1$ and $\beta = 0$ have been selected for the correlation coefficient of (5.3).

An analog computer network was built to mechanize the mathematical model for the example presented in this chapter. The parameter values given in (5.19) were used for the mathematical model.

To provide an indication of the effectiveness of the simulation, wind velocity profiles as a function of altitude $y$ were generated by the computer network. To accomplish this, the sensing element was assigned a constant vertical velocity $v(t)$ as it moved from $y = 0$ to $y = 20$ kilometers. Several typical profiles generated by the analog computer network are shown in Figure 5.5. For comparison, Figure 5.6 presents several experimentally measured wind velocity profiles as given in Reference 7 of the Bibliography.
Figure 5.5. Wind Velocity Profiles Generated by the Analog Computer Network of Figure 5.1.
Figure 5.6. Experimental Wind Velocity Profiles.
VI. CONCLUSIONS

This report presents a summary of the results obtained over the past several years in research on the analog generation of nonstationary random processes. The basic synthesis procedure, called the covariance-expansion method, developed during the research was presented in Technical Note No. 3. This basic procedure permitted the synthesis of a network to generate an arbitrary, nonstationary, Gaussian random process. However, it requires the specification of the statistical moments of the process as a function of time. Thus, in applying the method of Technical Note No. 3 to the simulation of wind disturbances affecting a moving vehicle, it is necessary to construct a different network for each different flight path. In subsequent research the synthesis method was adapted to permit a single network to be used for the approximation of random disturbances affecting a vehicle following an arbitrary path of motion. Here, time functions describing the motion of the vehicle are used as inputs to the single network.

An outline of the covariance-expansion synthesis method is included in this technical note. Also, a mathematical model for random wind turbulence is included. This model has been constructed using assumptions selected to simplify the experimental determination of parameters of the model and also to simplify the physical mechanization of the model.

Chapter V of this technical note discusses the mechanization of a simple mathematical model of one horizontal component of random wind velocity. The mechanization permits the simulation of the wind disturbance that affects a vehicle moving on an arbitrary path in three-dimensional space.

Consistent with the restricted amount of experimental wind data that has been studied in the present project, the research that has been summarized indicates that the covariance-expansion synthesis method can be used to construct an analog computer network that approximates the salient statistical characteristics of wind turbulence.
BIBLIOGRAPHY


BIBLIOGRAPHY ON TECHNIQUES FOR SOLVING
PARTIAL DIFFERENTIAL EQUATIONS
BY HYBRID COMPUTATION AND OTHER METHODS

By David L. Finn

Prepared for
George C. Marshall Space Flight Center
Huntsville, Alabama

Contract No. NAS8-2473

(Development of New Methods and
Applications of Analog Computation)

22 April 1968

School of Electrical Engineering
GEORGIA INSTITUTE OF TECHNOLOGY
Atlanta, Georgia
BIBLIOGRAPHY ON TECHNIQUES FOR SOLVING
PARTIAL DIFFERENTIAL EQUATIONS
BY HYBRID COMPUTATION AND OTHER METHODS

by

DAVID L. FINN

TECHNICAL NOTE NO. 16

on

Contract No. NAS8-2473

(Development of New Methods and
Applications of Analog Computation)

For

GEORGE C. MARSHALL SPACE FLIGHT CENTER
Huntsville, Alabama
References in this bibliography are related to the numerical solution of partial differential equations. The references are listed under three headings: Hybrid Computer Methods, Analog Computer Methods, and General Methods. Brief comments are given for most of the references on Hybrid Computer Methods.
# TABLE OF CONTENTS

I. INTRODUCTION .................................................. 1

II. HYBRID COMPUTER REFERENCES .............................. 3

III. ANALOG COMPUTER REFERENCES ............................ 11

IV. GENERAL REFERENCES ......................................... 19
I. INTRODUCTION

This bibliography lists references related to the numerical solution of partial differential equations. References are listed under three headings: Hybrid Computer Methods, Analog Computer Methods, and General Methods. The primary objective of the literature search on which this bibliography is based was the finding of papers related to the solution of partial differential equations by hybrid computation techniques. References on analog computer methods and general methods are included because hybrid techniques necessarily represent an integration of analog and digital methods utilized within the framework of general numerical analysis.

Brief comments are included for most of the references listed under the Hybrid Computation Methods heading.

Most of the references listed in this bibliography are concerned with the solution of partial differential equations of the types classified by mathematicians as elliptic, parabolic, and hyperbolic. These three classifications include most of the partial differential equations that are of importance in scientific and engineering problems.

The elliptic equation classification includes as its most important members the equations of Laplace and Poisson. In general, these equations describe equilibrium processes. Laplace's equation describes systems that do not contain internal sources. Poisson's equation describes systems in which sources are present.

Laplace's equation characterizes free space potential fields in electrostatics, magnetostatics, and gravitation. It also characterizes steady-state heat transfer in a homogeneous, isotropic medium, and the stationary flow of homogeneous, incompressible fluids.
Poisson's equation characterizes the same type phenomena as Laplace's equation except that, as was mentioned previously, it is applicable in regions containing sources.

The class of parabolic partial differential equations contains the diffusion equation. The diffusion equation characterizes equilization processes. This equation describes non-equilibrium heat-transfer fields. It also characterizes diffusion processes that involve the equilization of material particles or electric charges.

The most widely occurring hyperbolic partial differential equation is the wave equation. This equation describes wave motion in lossless systems and, in general, characterizes oscillating processes. It governs the behavior of lossless transmission lines, the adiabatic flow of a compressible fluid, and the undamped vibration of strings and thin elastic membranes.
II. HYBRID COMPUTER REFERENCES

This chapter gives an alphabetical listing of references pertaining to the solution of partial differential equations using hybrid computers. A brief summary is included for each reference.


   This paper considers the analysis of the thermal system of a layer-type transformer winding. The analysis, using a resistance network analogue with nodal currents calculated by a digital computer, is described.


   A solution of Laplace's equation in two dimensions is discussed. In the computation system, a technique is effected in which the analog integrators are utilized in two distinct groups. When one group of integrators is operating the second group is placed in "hold" and vice versa.


   This paper discusses the solution by hybrid computation of the boundary value problems associated with the analysis of a co-current laminar flow double-pipe heat exchanger.


   This paper looks into the possibility of the use of a hybrid computer for the solution of a particular type of initial-value problem involving partial differential equations: namely, reactor fault studies on a single channel in the flattened zone of a nuclear reactor. Two methods of solution that could
be used on a hybrid computer are assessed and compared with corresponding methods that would be used on a digital computer.


   This report describes the transient simulation of a tubular chemical reactor, its product separator, and its control system.


   This is a collection of six technical reports on the subject of applications of analog and hybrid computers in the solution of partial differential equations.


   This paper considers the use of methods of successive approximations in the solution of boundary value problems of ordinary differential equations and partial differential equations.


   Many two-point boundary value problems are included in the class of problems for which trial and error techniques must be used in obtaining solutions.
Searching operations for solution of these problems can be performed automatically on a hybrid computer. This paper discusses the use of a hybrid computer for the determination of the first N normal modes of a one-dimensional heat flow problem.


In this paper an iterative procedure is devised for the solution of a two-point boundary value problem involving one-dimensional flow of heat in a slab. A computer for implementing the procedure is described. Numerical results obtained in the solution of this problem are presented.


A discussion is given of the feasibility of using hybrid computation in three different classes of problems. These include initial and boundary-value problems involving ordinary differential equations and initial-value problems involving partial differential equations. These problems are all nonlinear and are of interest to chemical engineers. The problems include the analysis of two lumped-parameter systems and one distributed parameter system.


This thesis discusses a hybrid computer implementation of the Monte Carlo Technique of solution of elliptic and parabolic partial differential equations. The hybrid computer is capable of taking statistics over 1,000 two or three-dimensional random walks each second. The Monte Carlo method has been extended.
to a wide class of boundary conditions especially applicable to heat-conduction, diffusion problems.


The application of a hybrid computer to the solution of elliptic and parabolic partial differential equations is discussed. The exceptional computing speed and flexible digital control of the computing system permit direct plotting of partial differential equation solutions.


A hybrid simulation of an ablating nose cone is discussed. This results in the provision of a time history of the ablating body shape.


This report summarizes the results of a preliminary investigation of the use of a hybrid computer technique that involves a continuous-time, discrete-space method for the solution of partial differential equations. A combined parallel-serial computation approach is used in which the digital computer is employed primarily for function storage and playback.


This paper describes application of a system that includes a small digital computer connected in a closed loop with a network of analog elements. The analog network consists of an array of inexpensive node modules, one for each grid point in space. The analog unit serves as a subroutine that solves a
system of difference equations whenever this solution is required by the digital program. This technique has been developed to take advantage of the fact that an analog computer, by virtue of parallel mode operation, generates solutions of simultaneous algebraic equations almost instantaneously regardless of the number of equations involved.


A brief description is given of a hybrid computer system in which the analog devices act as major subroutines for the digital computer and carry out all the mathematical steps required in a specific computational task. These analog subroutines are particularly effective when a time-consuming loop in the digital flow chart can be replaced by a single "pass" through an analog system. A brief discussion is given on the application of the hybrid system to the solution of partial differential equations.


This item in the correspondence section of the Annales briefly summarizes the main ideas of a talk by Dr. Karplus scheduled for presentation at a panel session of the Fall Joint Computer Conference in 1966.


Several typical hybrid computer applications are discussed briefly including the hybrid solution of a set of partial differential equations used in the
simulation of a tubular reactor. The solution employs the method of characteristics whereby the original partial differential equations are reduced to ordinary differential equations.


A discussion is given on the application of a hybrid computer to the solution of partial differential equations by the Monte Carlo method. The random walks required in the solution were implemented on a conventional analog computer requiring about five minutes for a thousand random walks. A digital computer controlled the analog computer operation.


In the Monte Carlo solution of boundary value problems associated with partial differential equations, both the boundaries and the boundary conditions must be generated and detected. This note discusses techniques for the generation and detection of these boundaries and boundary conditions.


This report examines some aspects of the mathematical stability of solutions for linear transport and diffusion equations. Also considered is the effect of finite analog accuracy on time differences together with conditions for iterative satisfaction of boundary value problems.


This paper outlines the preliminary design for a hybrid computer utilizing both digital and analog techniques. In essence, it consists of a small stored program digital computer that controls and interacts with a large network of analog elements. The basic network element, corresponding to a single node, consists of a capacitor and two resistors. However, other elements, including active ones, may be used. This computer was conceived of primarily with complex heat flow problems in mind, but it is expected that it will be useful for many other kinds of problems.


This presentation considers the solution of partial differential equations by the use of a hybrid system in which random walks are accomplished and boundary conditions established on the analog computer and the walk averaging and control is done with the digital computer.


The utilization of hybrid computation is discussed in the application of the method of characteristics to the solution of first order partial differential equations.


A discussion is given of the conceptual details of the hybrid solution of equations arising in the study of the transient behavior of a chemical tubular reactor under various conditions when an optimizing controller is installed. Solutions are based on a function iteration scheme that utilizes digital function storage to reduce the amount of analog non-linear equipment required through the time-sharing of analog operations.


III. ANALOG COMPUTER REFERENCES

This chapter gives an alphabetical listing of references pertaining to the solution of partial differential equations using analog computers.


IV. GENERAL REFERENCES

This chapter gives an alphabetical listing of references pertaining to the numerical solution of partial differential equations. While most of the references listed below do not emphasize the implementation of the calculation, use of a digital rather than an analog or hybrid computer is implied.


63. Ehrlich, L. W., "Monte Carlo Solutions of Boundary Value Problems Involving the Difference Analogue of
\[ \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + k \frac{\partial u}{\partial y} = 0 \]


70. Evans, G. W., R. Brusseau, and R. Kierstead, Instability Considerations for Various Difference Equations Derived from the Diffusion Equation, Stanford Research Institute, Stanford, California, 1954.


131. Kellog, R. B., "Difference Equations on a Mesh Arising From a General

132. Kelly, Louis G., Handbook of Numerical Methods and Applications, Addison-

133. Klerer, Melvin and Granino Korn, Digital Computer User's Handbook, McGraw-

134. Kosolev, A. I., "Convergence of the Method of Successive Approximations

135. Kreiss, H. O., "On Difference Approximations of the Dissipative Type for

136. Kreiss, H. O., "Über die Differenax Approximation hoher Genauigkeit bei

137. Kreiss, H. O., "Über die Lösung von Anfangswertaufgaben für partielle

138. Kreiss, H. O., "Über die Stabilitätsdefinition für Differenzengleichungen
die partielle Differentialgleichungen approximieren", Nordisk Tidskr. Informations-

139. Kreiss, H. O., "Über implizite Differenzenmethoden für partielle Differential-

140. Ladyzhenskaya, O. A., "Solution of Cauchy's Problem for Hyperbolic Systems
by the Method of Finite Differences", Leningrad Gos. Univ. Uch. Zap., Vol. 144,


TECHNICAL NOTE NO. 17

Research Project A-588

SAMPLING ERRORS IN CLOSED LOOP HYBRID COMPUTER PROGRAMS

By Joseph L. Hammond, Jr.

Prepared for
George C. Marshall Space Flight Center
Huntsville, Alabama

Contract No. NAS8-2473

(Development of New Methods and Application of Analog Computation)

15 May 1968

School of Electrical Engineering
GEORGIA INSTITUTE OF TECHNOLOGY
Atlanta, Georgia
SAMPLING ERRORS IN CLOSED LOOP HYBRID COMPUTER PROGRAMS

By
Joseph L. Hammond, Jr.

TECHNICAL NOTE NO. 17

on
Contract No. NAS8-2473
(Development of New Methods and Applications of Analog Computation)

For
GEORGE C. MARSHALL SPACE FLIGHT CENTER
Huntsville, Alabama
ABSTRACT

A vector differential equation for the error due to sampling in closed loop hybrid computer programs is developed using several approximations derived with Taylor series expansions. The equation is applicable to general hybrid programs with no restriction on the manner in which the computing operations are allocated between the analog and digital computers. The major restrictions necessary for the error equation to apply are:

1. The problem equation must be expressible as a set of first order (linear, nonlinear or time varying) equations,
2. The digital-to-analog converters must be zero-order hold,
3. All converters and numerical methods of the digital computer must have the same sampling period,
4. All digitally generated functions must be computed during the same time period and converted D-to-A at the same time and finally,
5. The sampling period must be small.

The error equation is linear. Its homogeneous part is independent of the allocation of operations between the analog and digital computers, but its forcing function depends on the details of such allocation and certain constants, namely: the sampling period, the digital execution time and the order of numerical methods used in the digital computer. Both parts of the error equation depend on the problem solution variables, but these can be either the true solutions or the actual hybrid computer solutions.

Solution of the error equation typically requires machine computation, but several properties of sampling error are apparent from the form of the equation. For example, since the forcing function on the error equation is proportional to the first power of the sampling period, it follows that the hybrid computer is
a first order computational method. It is also apparent from the error equation that the execution time of digitally generated functions has the same general effect as a non-zero sampling period but weighted twice as heavily.

The error equation is expected to be useful in studying existing hybrid computer programs, in allocating computing operations between the analog and digital computers and in compensating against sampling error.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>ii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>v</td>
</tr>
<tr>
<td>GLOSSARY OF SYMBOLS</td>
<td>vi</td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>DEVELOPMENT OF HYBRID COMPUTER EQUATION</td>
<td>3</td>
</tr>
<tr>
<td>DEVELOPMENT OF SAMPLING ERROR EQUATION</td>
<td>10</td>
</tr>
<tr>
<td>THE ERROR EQUATION FOR CERTAIN SPECIAL CASES</td>
<td>15</td>
</tr>
<tr>
<td>Hybrid implementation with zero sampling error</td>
<td>15</td>
</tr>
<tr>
<td>Hybrid implementation with negligible error in the numerical method.</td>
<td>15</td>
</tr>
<tr>
<td>Hybrid implementations which are predominately analog or digital</td>
<td>16</td>
</tr>
<tr>
<td>Equations with additive forcing functions</td>
<td>17</td>
</tr>
<tr>
<td>Linear constant coefficient equations</td>
<td>18</td>
</tr>
<tr>
<td>EXAMPLE</td>
<td>19</td>
</tr>
<tr>
<td>CONCLUSIONS</td>
<td>22</td>
</tr>
<tr>
<td>APPENDIX</td>
<td>25</td>
</tr>
<tr>
<td>BIBLIOGRAPHY</td>
<td>28</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 1.</td>
<td>General Hybrid Computer Implementation of Equation (1)</td>
<td>4</td>
</tr>
<tr>
<td>Figure 2.</td>
<td>Various Errors and the True Solution Variables for a Hybrid Computer Solution of Duffin's Equation.</td>
<td>20</td>
</tr>
</tbody>
</table>
GLOSSARY OF PRINCIPAL SYMBOLS

\[ u(t) \] - ideal vector solution of problem equation
\[ x(t) \] - ideal vector solution of equation programmed on the analog computer
\[ z(t) \] - ideal vector solution of equation solved digitally
\[ h \] - vector of functions equated to \( \dot{u}(t) \)
\[ f \] - vector of functions equated to \( \dot{x}(t) \)
\[ f_a \] - vector of functions generated and used in the analog computer
\[ f_d \] - vector of functions generated digitally for use in the analog computer
\[ g \] - vector of functions equated to \( \dot{z}(t) \)
\[ t \] - time
\[ n \] - order of vectors
\[ p \] - order of numerical method
\[ \varphi \] - principal error function of numerical method
\[ \delta \] - sampling period
\[ e \] - execution time of digital computer
\[ k \] - index of discrete time
\[ \varepsilon \] - discretization error in numerical method
\[ \Gamma(t) \] - total hybrid computer sampling error vector
\[ \gamma(t) \] - sampling error vector for analog computer
\[ \sigma(t) \] - sampling error vector for digital computer
\[ F_d \] - output of D/A converter with \( f_d \) as input
\[ Z(t) \] - output of D/A converter with \( z(t) \) as input
\[ u^*(t) \] - vector solution of problem equations which accounts for hybrid computer sampling error (similar notation for other variables)
GLOSSARY OF PRINCIPAL SYMBOLS (CONT'D.)

\( \frac{\partial h}{\partial u} \) - a matrix defined in the Appendix
(a similar notation for other variables)

\( O[g(\delta)] \) - a symbol denoting that some variable (which is \( O[g(\delta)] \)) goes to zero with \( \delta \) at least as rapidly as \( g(\delta) \)

\[
\begin{align*}
u(t) &= \begin{bmatrix} x(t) \\ z(t) \end{bmatrix} \\
h &= \begin{bmatrix} f_a + f_d \\ g \end{bmatrix} \\
f &= f_a + f_d \\
\Gamma(t) &= \begin{bmatrix} \gamma(t) \\ \alpha(t) \end{bmatrix}
\end{align*}
\]
INTRODUCTION

The work reported herein is a part of a general study of sampling errors in hybrid computation being carried out as one of several tasks under the current research program. Earlier work on the same problem has been reported in Technical Notes Nos. 8, 12, and 13. [1],[2],[3]

In these reports a differential equation for sampling error in certain hybrid computer applications is derived and evaluated in test examples.

The objective of the present technical note is to document further study which has unified and generalized earlier work. Specifically the present document derives a differential equation for the sampling error in a general class of hybrid computer applications. The results apply to any allocation of computing tasks between the analog and digital equipment and take into account errors in the numerical techniques used in the digital computer. The restrictions on the present analysis are listed on pages 5 and 6 in the body of the report. The most important restrictions are:

(a) The implicit requirement that the problem equation can be expressed in "state variable" form (given as (1) on page 3).

(b) The assumption that the D/A converters are zero-order hold.

(c) The assumption that all converters and the numerical method have the same sampling period.

(d) The assumption that all digitally generated functions are converted D-to-A at the same time after a common execution time.

†Numbers refer to references listed in the Bibliography.
These assumptions are much less restrictive than those used in Technical Notes Nos. 8, 12, and 13 since the earlier work applies only in applications for which the digital computer serves as a digital function generator. It can be noted, however, that the general sampling error equations derived in this technical note reduce to those of the earlier notes in the special cases for which the earlier work applies.

In order to place clearly in evidence the underlying assumptions leading to the sampling error equation, the approach in this report is to develop equations for the hybrid computer system including the effect of sampling error. Using this result an equation for the sampling error, defined as the difference between an ideal solution of the problem equation and a hybrid solution accounting for sampling error, is derived. The technical note then examines the form assumed by the error equation in several special cases and gives a numerical example illustrating a general hybrid computer implementation.
DEVELOPMENT OF HYBRID COMPUTER EQUATIONS

In order to obtain tractable results it will be assumed that the hybrid computer is programmed to solve the ideal equation

\[ \dot{u}(t) = h[u(t);t] . \]  

(1)

The \( u(t) \) is an \( n \)-vector\(^\dagger \) of component outputs.

Equation (1) is implemented on the hybrid computer as shown in Figure 1, where the double lines represent the flow of vector quantities. The ideal equation (1) can be partitioned to place in evidence the ideal equation solved by the analog computer and the ideal equation solved by the digital computer. The result is

\[ \dot{x}(t) = f_a[x(t),z(t);t] + f_d[x(t),z(t);t] \]  

(2)

and

\[ \dot{z}(t) = g[x(t),z(t);t] , \]  

(3)

where (2) is integrated by the analog computer to yield \( x(t) \), (3) is integrated by the digital computer to yield \( z(t) \), \( f_a \) is a vector function generated in the analog computer, \( f_d \) is a vector function generated in the digital computer,

\[ u = \begin{bmatrix} x \\ z \end{bmatrix} \text{ and } h = \begin{bmatrix} f_a + f_d \\ g \end{bmatrix} \]

\(^\dagger\)Throughout the report variables without numerical subscripts are vector quantities, whereas variables with numerical subscripts are scalar quantities.
Figure 1. General Hybrid Computer Implementation of Equation (1).
Note that the hybrid computer is used in a general way with the exception that no functions are generated with the analog equipment for use in the digital computer. Such a possibility is not included for two reasons, namely, it seldom affords any practical advantage and the sampling errors occurring when the analog equipment generates functions for use in the digital computer are no different than when such functions are produced in the digital computer.

Since (2) and (3) are ideal equations, the next step is to formulate equations which account for the sampling errors present in actual hybrid equipment. The following assumptions concerning the operation of the equipment are made:

(a) The only sources of error are the non-zero sampling period of the A/D converters, and the discretization error of the numerical method used by the digital computer.

(b) The numerical method used by the digital computer to solve (3) is a method of order p with principal error function \( \varphi \).

(c) The D/A converters are zero-order hold so that their outputs, \( F_d \) or \( Z \), are stair-step functions of time.

(d) The vector of variables \( z(k\delta) \) is computed digitally prior to the time \( t = k\delta \) when its components are sampled by the D/A converters. This assumption requires that the execution time, necessary in computing \( z(k\delta) \) from values of \( z[(k-1)\delta] \) and \( x[(k-1)\delta] \), be less than \( \delta \).

(e) The sampling periods of all of the converters and the step size of the numerical methods are all equal and denoted \( \delta \).
Finally all components of the function \( f_d[x, z; k\delta] \) are computed in a time interval \( \epsilon \) from values of \( x \) and \( z \) at \( k\delta \) and all are converted D-to-A at \( t = k\delta + \epsilon \).

While the assumptions are made to make the problem tractable, they are all reasonable. Assumption (a) is made to isolate one type of error for consideration. Assumption (b) limits the numerical method of the digital computer to one for which "order" and "principal error function" are useful properties. Assumption (c) and (d) are satisfied in almost all hybrid computer applications. Assumption (e), that all converters and the numerical method have a period \( \delta \), is restrictive but is a condition frequently used in practice. Finally, assumption (f), while restrictive, corresponds to one fairly common mode of operation.

Non-ideal equations corresponding to (2) and (3) and to the interconnection diagram of Figure 1 can now be obtained by expressing the assumptions listed above mathematically. Since any error will cause the solution of the non-ideal equations to differ from that of the ideal equations, the non-ideal hybrid variables will be denoted \( x^*(t) \) and \( z^*(t) \).

Assumption (c) results in the expressions

\[
Z^*(t) = z^*(i\delta), \ i\delta \leq t < (i+1)\delta \tag{4}
\]

and

\[
F_d(t) = f_d \{ x^*(i\delta), z^*(i\delta); i\delta \}, \ i\delta + \epsilon \leq t < (i+1)\delta + \epsilon \tag{5}
\]

for \( Z^*(t) \) and \( F_d(t) \), the outputs of the D/A converters. Note that execution time \( \epsilon \) appears in (5) in accordance with assumption (f). Equations (4) and (5) along with Figure 1 can be used to express the equation solved by the analog computer as
The ideal equation to be solved by the numerical method in the digital computer is (2) with \( x(t) \) replaced by \( x^*(t) \). Because of this change the solution of the ideal equation is no longer \( z(t) \), and (2) is written as

\[
\dot{z}'(t) = g[x^*(t), z'(t); t]
\]

(7)

A relation between \( z'(t) \), the true solution of (7), and \( z^*(i\delta) \), the numerical solution of (7), must now be obtained. In numerical analysis, sampling error is typically referred to as "discretization" error, and in keeping with the general assumptions other errors introduced in the numerical solution of (7) will be neglected. Many different algorithms for the numerical solution of (7) exist. For most of these, the so-called accumulated discretization error given by

\[
\epsilon(i\delta) = z^*(i\delta) - z'(i\delta), \quad i = 0, 1, 2 \ldots
\]

(8)

can be approximated asymptotically as the solution to an equation of the form

\[
\dot{\epsilon}(t) = \frac{3R}{\partial z}, \quad \epsilon(t) + \delta P \varphi(z'; t); \quad \epsilon(0) = 0
\]

(9)

Such equations are discussed for example by Henrici \[4\]. For (9) to apply for Runge-Kutta type algorithms the function \( g \) must be evaluated at several points internal to the basic \([i\delta, (i+1)\delta]\) interval. This requires that a forcing function implicit in \( g \) must be sampled at a rate higher than \( 1/\delta \). This does not pose a problem if digital execution time rather than characteristics of the D/A and A/D converters are assumed to determine the minimum sampling period.

For predictor-corrector type algorithms (9) is more approximate than for one-step or Runge-Kutta type algorithms since it neglects error in starting predictor-corrector procedures.
where \( g \) is the function on the right-hand side of (7), \( p \) is the order of the numerical method and the accumulated discretization error at \( t = i\delta \) is \( \varepsilon(t = i\delta) \). The function \( \varphi \) is termed the principal error function, and in all cases for which (9) applies expressions for it are available. The symbol \( \frac{\partial g}{\partial z'} \) with \( g \) and \( z' \) both vectors is defined in the Appendix. Note that (9) is a linear equation whose solution can be expressed as

\[
\varepsilon(t) = \delta^p \varepsilon_o(t)
\]

where \( \varepsilon_o(t) \) is the solution of (9) with \( p = 0 \). Hence if \( \varepsilon_o(t) \) is bounded as will be assumed, for small \( \delta \) \( \varepsilon(t) \) is \( O(\delta^p) \).

It is useful to introduce a continuous variable \( z^*(t) \) by the definition

\[
z^*(t) = z'(t) + \varepsilon(t).
\]

(10)

Here \( z'(t) \) is the true solution of (7) and \( \varepsilon(t) \) is obtained as the solution of (9). An inspection of (8) and (10) shows that \( z^*(t) \) is the digital computer output at the discrete time instants \( t = i\delta \), \( i = 0, 1, 2, \ldots \). A differential equation for \( z^*(t) \), namely

\[
\dot{z}^*(t) = g[z^*(t), z'(t); t] + \frac{\partial g}{\partial z'} \varepsilon(t) + \delta^p \varphi(z'; t),
\]

(11)

results from differentiating (10) and using (7) and (9).

Equation (11), which involves the variables \( z^*(t) \), \( z'(t) \), and \( \varepsilon(t) \) can be expressed in terms of only the first of these variables through use of the general approximations developed in the Appendix. Consider the first

\[
\text{The symbol } O(\delta^p) \text{ has the following significance. If } f(\delta) \text{ is } O[g(\delta)] \text{ as } \delta \to 0, \text{ then there exists a positive constant } c \text{ such that } |f(\delta)| \leq c|g(\delta)| \text{ for } \delta \text{ sufficiently close to zero.}
\]
two terms on the right-hand side of (11), namely \( g[x^*(t), z'(t); t] + \frac{\partial g}{\partial z}, \varepsilon(t) \).

Use of (A-2)\(^\dagger\) shows that for small \( \varepsilon(t) \) this sum is approximately equal to \( g[x^*(t), z'(t) + \varepsilon(t); t] \), or using (10), to \( g[x^*(t), z^*(t); t] \). Similar reasoning results in the expression

\[
\varphi(z'; t) = \varphi[z^*(t) - \varepsilon(t); t] = \varphi[z^*(t); t] - \frac{\partial \varphi}{\partial z^*} \varepsilon(t).
\]  

(12)

Using these expressions for \( g \) and \( \varphi \) in (11) then yields

\[
\dot{z}^*(t) = g[x^*(t), z^*(t); t] + \delta^P \varphi[z^*(t); t] \tag{13}
\]

after neglecting the term \( \delta^P \frac{\partial \varphi}{\partial z^*} \varepsilon(t) \) which is \( O(\delta^{2p}) \).

Equations (13) and (6) are the equations effectively solved by the hybrid computer system accounting for sampling error but assuming that \( \delta \) is small.

\[^\dagger\text{Equations numbered (A-\( ) \) appear in the Appendix.}\]
DEVELOPMENT OF SAMPLING ERROR EQUATIONS

A sampling error vector $\Gamma(t)$ can be defined as the difference between the solution of the ideal equations and the hybrid computer equations which account for sampling error. Thus

$$\Gamma(t) = \begin{bmatrix} \gamma(t) \\ \alpha(t) \end{bmatrix} = \begin{bmatrix} x(t) - x^*(t) \\ z(t) - z^*(t) \end{bmatrix} \tag{14}$$

where $\Gamma(t)$ is the total hybrid computer error vector and $\gamma(t)$ and $\alpha(t)$ are, respectively, the analog and digital computer error vectors. Differentiation of (14) yields

$$\dot{\Gamma}(t) = \begin{bmatrix} \dot{\gamma}(t) \\ \dot{\alpha}(t) \end{bmatrix} = \begin{bmatrix} \dot{x}(t) - \dot{x}^*(t) \\ \dot{z}(t) - \dot{z}^*(t) \end{bmatrix} \tag{15}$$

A differential equation for error is obtained by subtracting (6) from (2) and (13) from (3). The result is

$$\dot{\gamma}(t) = f_a[x(t), z(t); t] - f_a[x^*(t), z^*(t); t] + f_d[x(t), z(t); t] - f_d(t) \tag{16}$$

$$\dot{\alpha}(t) = g[x(t), z(t); t] - g[x^*(t), z^*(t); t] - \delta^P \varphi[z^*(t); t] \tag{17}$$

Use is now made of the general approximations developed in the Appendix to reduce (16) and (17) to a more tractable form by neglecting higher order terms in the variables $\delta$, $\alpha$ and $\gamma$ which are assumed to be "small." Specifically, terms $O(\delta^2)$, $O(\gamma^2)$, $O(\alpha^2)$ and $O(\alpha \delta)$ are neglected.
First consider the staircase variables $F_d(t)$ and $Z^*(t)$. Using (A-6) and (A-8) these can be approximated as

$$Z^*(t) = z^*(t) - [\delta/2 + S(t)]z^*(t)$$

(18)

$$F_d(t) = f_d[x^*(t),z^*(t);t] - \left[\frac{\delta+2e}{2} + S(t-e)\right]f_d[x^*(t),z^*(t);t].$$

(19)

Using (18) and (19) in (16) then yields

$$\dot{\gamma}(t) = f_a[x(t),z(t);t] - f_a[x^*(t),z^*(t)] - [\delta/2 + S(t)]z^*(t);t] \quad (20)$$

$$+ f_d[x(t),z(t);t] - f_d[x^*(t),z^*(t);t] + \left[\frac{\delta+2e}{2} + S(t-e)\right]f_d[x^*(t),z^*(t);t].$$

Equation (A-2) is then used to modify (20) and (17) to the extent that all of the functions on the right-hand side of these equations have as argument the non-ideal hybrid variables $x^*(t)$ and $z^*(t)$. For example,

$$f_a[x(t),z(t);t] = f_a[x^*(t),z^*(t);t] + \frac{\partial f_a}{\partial x^*} \gamma(t) + \frac{\partial f_a}{\partial z^*} \alpha(t).$$

The result of this step after canceling certain terms is

$$\dot{\gamma}(t) = \frac{\partial f_a}{\partial x^*} \gamma(t) + \frac{\partial f_a}{\partial z^*} \alpha(t) + \frac{\partial f_d}{\partial x^*} \gamma(t) + \frac{\partial f_d}{\partial z^*} \alpha(t) + S(t) \frac{\partial f}{\partial z^*} z^*(t) + \frac{\delta}{2} \frac{\partial f_a}{\partial z^*} z^*(t)$$

$$+ \left(\frac{\delta+2e}{2}\right) f_d[x^*(t),z^*(t);t] + S(t-e) f_d[x^*(t),z^*(t);t].$$

(21)

(22)

In these equations and the error equations below $\alpha(t=0) = \gamma(t=0) = 0$. 

11
In (21) note that the two terms involving \( S(t) \) are forcing functions for the linear differential equation. Thus, assuming the conditions of approximation 3 in the Appendix are met, the contributions of these terms to \( y(t) \) will be \( O(\delta^2) \) and hence they can be neglected.

Each term on the right-hand side of (21) and (22), after neglecting the terms involving \( S \), is multiplied by one of the small quantities \( \gamma, \alpha, \) or \( \delta \). Thus an application of (A-2) to change the arguments of the functions on the right-hand side of (21) and (22) to the ideal variables \( x(t) \) and \( z(t) \) will result in added terms which are all of higher order in the small quantities.

For example, the term \( \frac{\delta f_d}{\delta x^*} y(t) \) can be expressed as

\[
\gamma(t) \frac{\delta f_d}{\delta x^*} \{x(t), z(t); t\} = \gamma(t) \frac{\delta f_d}{\delta x} \{x(t), z(t); t\} - \frac{\delta f_d}{\delta x} \gamma(t) - \frac{\delta f_d}{\delta z} \alpha y
\]

and the last two terms are \( O(\gamma^2) \) and \( O(\alpha y) \), respectively. Thus (21) and (22) can be written as

\[
\dot{\gamma}(t) = \frac{\partial f_a}{\partial x} \gamma(t) + \frac{\partial f_a}{\partial \gamma} \alpha(t) + \frac{\partial f_d}{\partial x} \gamma(t) + \frac{\partial f_d}{\partial \gamma} \alpha(t) + \frac{\delta}{2} \frac{\partial f}{\partial z} z + \frac{(\delta + 2\epsilon)}{2} f_d \tag{23}
\]

and

\[
\dot{\alpha}(t) = \frac{\partial g}{\partial x} \gamma(t) + \frac{\partial g}{\partial \gamma} \alpha(t) - \delta \psi \tag{24}
\]

\( ^\dagger \) All practical numerical methods have \( p \geq 1 \) since this is necessary for convergence of the numerical solution to the ideal solution as \( \delta \) approaches zero.
where to the accuracy being considered the arguments of the functions can be either the ideal or the non-ideal hybrid solution variables and time. In obtaining (23), (3) is used to replace \( \dot{z}(t) \) with \( g[x(t),z(t);t] \).

Equations (23) and (24) represent the most general form of the equations for sampling error to be considered in this report. Note that the equations are linear but can be time varying if the partial derivatives of \( f_a, f_d \) or \( g \) depend on time.

Using the previously defined quantities \( h \) and \( \Gamma \), (23) and (24) can be written more compactly as

\[
\dot{\Gamma}(t) = \frac{\partial h}{\partial u} \Gamma(t) + \begin{bmatrix}
\frac{\partial f_a}{\partial z} g + \frac{\delta + 2e}{2} f_d \\
- \delta^2 \varphi
\end{bmatrix}.
\]

Note that the homogeneous part of the sampling error equation depends only on the total function \( h \), which is independent of how the equations are divided between the analog and digital computers in the hybrid implementation. The function \( h \) in turn depends on either the ideal solutions or the hybrid solutions and on time. The forcing function for (25), on the other hand, depends on how the equations are divided between the analog and the digital computers (i.e., on \( f_d \) and \( g \)). It also depends on the principal error function \( \varphi \) of the numerical method, either the ideal or hybrid solution variables and on the constants \( \delta, e \) and \( p \).

\[\text{\dag}\] Such arguments will be understood below unless otherwise stated. As a further change in notation the approximate equality sign \( \approx \) will be replaced by the standard equality sign.
For purposes of solving the error equation numerically, it may be expedient to use the chain rule to express \( f_d \) in (25) as

\[
\frac{df_d}{dt} = \frac{\partial f_d}{\partial z} g + \frac{\partial f_d}{\partial x} f + \frac{\partial f_d}{\partial t}
\]  

(26)

where \( \frac{\partial f_d}{\partial t} \) accounts for the explicit dependence of \( f_d \) on \( t \). With the possible use of the result of (26), (25) can be programmed for a general purpose computer. The required vector operations can be performed by the computer whose only inputs can be a specification of the functions \( f_a, f_d, g \) and \( \varphi \); either the ideal or hybrid solutions of the given equation (i.e. \( x(t), z(t) \) or \( x^*(t) \) and \( z^*(t) \)) and the constants \( \delta, e \) and \( p \). It is, of course, possible to closely approximate \( x(t) \) and \( z(t) \) by also solving (2) and (3) when using a digital computer and thus remove the necessity for these functions as inputs. Another possibility would be to solve (25) on the hybrid computer along with (2) and (3). In this case \( x^*(t) \) and \( z^*(t) \) would be used in (25).

\[\text{---}
\]

\[\dagger\text{The work of Moore [5] and Reiter [6] may provide an efficient and accurate method for machine generation of the required partial derivatives.}\]
THE ERROR EQUATION FOR CERTAIN SPECIAL CASES

Certain more specific assumptions concerning the given equation or the method of hybrid implementation result in simplifications of the general error equation.

Hybrid Implementation with Zero Sampling Error

Examination of (25) shows that in certain cases (which are trivial from a practical point of view) the general error equation reduces to a homogeneous equation with zero initial conditions so that the approximate error vectors \( \alpha(t) \) and \( \gamma(t) \) are zero for all time. This occurs for two cases, namely: (1) the hybrid computer has \( \delta \) identically zero or (2) the following conditions apply - \( f_a \) is independent of \( z \) (hence \( \frac{\partial f_a}{\partial z} = 0 \)), \( f_d \) is zero (hence \( \frac{\partial f_d}{\partial z} = 0 \)), and \( p \) is large (hence \( \delta^p \) is negligibly small).

That error is zero in both of these cases is intuitively obvious. Case 1 requires no further comment. Case 2 is just the condition for the equations for \( \gamma(t) \) and \( \alpha(t) \) to be independent and the error in the numerical method negligible.

Hybrid Implementations with Negligible Error in the Numerical Method

Consider the term \( \delta^p \varphi \) in (25). This term which accounts for error in the numerical method is \( O(\delta^p) \) and hence will be negligible for \( p \geq 2 \) if \( \varphi \) is bounded. For such cases the error equation becomes

\[
\dot{\Gamma}(t) = \frac{\partial h}{\partial u} \Gamma(t) + \begin{bmatrix}
\delta/2 \frac{\partial f_a}{\partial z} g + \frac{\delta+2e}{2} f_d
\end{bmatrix}.
\]

This equation applies in almost any case for which sophisticated numerical methods are used in the digital computer.
Hybrid Implementations Which are Predominately Analog or Digital

There are hybrid implementations for which either the analog or the digital computations are trivial in the sense that only algebraic rather than differential equations are solved. In such cases, the given equations are integrated exclusively on either the analog or the digital computer. Equation (25) then reduces to either

\[ \dot{y}(t) = \frac{\partial f}{\partial x} y(t) + \frac{\delta + 2\epsilon}{2} f_d \]

or

\[ \dot{\alpha}(t) = \frac{\partial g}{\partial z} \alpha(t) - \delta P \phi \]

where \( f = f_a + f_d \).

Equation (27) applies in the relatively important case of a hybrid implementation which uses the digital subsystem entirely as a function generator.\(^\dagger\)

Equation (28) applies, when the analog computer is used exclusively as an "algebraic function generator" for the digital computer. Such implementations are not in common use.

Hybrid Implementations with Uncoupled Analog and Digital Computations

Examination of (23) and (24) indicates that the equation for \( y(t) \) is coupled to the equation for \( \alpha(t) \) by terms multiplied by \( \frac{\partial f}{\partial z} \). Similarly, the equation for \( \alpha(t) \) is coupled to the equation for \( y(t) \) by terms multiplied by \( \frac{\partial g}{\partial x} \). Thus for systems with \( \frac{\partial f}{\partial z} = 0 \) (23) is independent of (24) and for systems with \( \frac{\partial g}{\partial x} = 0 \) (24) is independent of (23).

\(^\dagger\) This special case has been examined in detail in earlier work. See [1], [2], [3].
Note that when the latter condition applies \( \alpha(t) \) is negligibly small if \( p \geq 2 \). Note also that (23) and (24) being uncoupled does not necessarily make either \( \gamma(t) \) or \( \alpha(t) \) equal to zero.

**Equations with Additive Forcing Functions**

In the equations (2) and (3) specified for solution on the hybrid computer both \( g \) and \( f \) can contain implicitly the effect of forcing functions. For equations arising in many practical applications, the forcing functions have an additive property such that \( f \) and \( g \) can be expressed as

\[
\begin{align*}
  f &= \hat{f}(x,z) + v_1(t) \\
  g &= \hat{g}(x,z) + v_2(t)
\end{align*}
\]

where \( v_1(t) \) and \( v_2(t) \) are vector forcing functions which account for all explicit variations of \( f \) and \( g \) with \( t \).

Equations (29) and (30) can be used with any of the forms of the error equation. A tractable equation, for example, results from the use of (29) and (30) in (25), namely

\[
\dot{\Gamma}(t) = \frac{\partial h}{\partial u} \Gamma(t) + \left[ \frac{\delta}{2} \frac{\partial f}{\partial z} (\hat{g} + v_2) + \frac{\delta}{2} \frac{\partial f}{\partial x} (\hat{g} + v_2) + \frac{\partial f}{\partial x} (\hat{f} + v_1) + V_1(t) \right] + V_1(t) \cdot P \cdot \dot{v}_1
\]

where

\[
V_1(t) = \begin{cases} 
  \dot{v}_1(t), & v_1 \text{ introduced in the digital computer} \\
  0, & v_1 \text{ introduced in the analog computer}
\end{cases}
\]
Linear Constant Coefficient Equations

For the hybrid solution of linear constant coefficient equations the error equation reduces to a very tractable form which will be discussed in detail in a later report.
EXAMPLE

To illustrate the use of the error equations in a general but tractable context, consider using a hybrid computer to solve Duffin's equations in the form

\[ \dot{u}_1(t) = u_2(t) \quad (32) \]
\[ \dot{u}_2(t) = -u_1(t) - 0.06 u_1^3(t) - 2u_2(t) \quad (33) \]

\[ u_1(t=0) = 4, \quad u_2(t=0) = 0 . \]

Assume that the digital computer uses Euler's method to solve (32), and that (33) is solved on the analog computer with the term \(-0.06 u_1^3(t)\) generated digitally. The general hybrid computer equations then become

\[ \dot{x}_1(t) = f_{a1} + f_{d1} = [-2 x_1(t) - z_1(t)] + [-0.06 z_1^3(t)] \quad (34) \]

\[ \dot{z}_1(t) = x_1(t) \quad (35) \]

\[ z_1(t=0) = 4, \quad x_1(t=0) = 0 . \]

Use of (23) and (24) yield for the approximate error equations

\[ \gamma_1(t) = -2\gamma_1(t) - [1 + 0.18 z_1^2(t)] x_1(t) - \delta/2 x_1(t) - 0.18 \left(\frac{\delta+2\epsilon}{2}\right) z_1^2(t) \times_1(t) \quad (36) \]

\[ \dot{\gamma}_1(t) = \gamma_1(t) - \delta/2[z_1(t) + 0.06 z_1^3(t) + 2 x_1(t)] \quad (37) \]

Several examples illustrating the application of the specialized error equation (27) have been reported earlier. See [3].
SOLUTION OF APPROXIMATE
ERROR EQUATION FOR ANY \( \delta \).
ERROR FROM SIMULATION
OF HYBRID SYSTEM

\( \delta = 0.01 \)
\( \delta = 0.05 \)
\( \delta = 0.1 \)

NORMALIZED ERROR

TIME (Sec) →

\( \alpha_1/\delta \)
\( \gamma_1/\delta \)

TRUE SOLUTION VARIABLES

TIME (Sec) →

\( \chi_1 \)

Figure 2. Various Errors and the True Solution Variables for a
Hybrid Computer Solution of Duffin's Equation.
In obtaining (37), \( \varphi(t) \) (which equals \(-\dot{z}(t)/2\) for Euler's method\(^\dagger\)) is expressed by the quantity in square brackets using (35) and (34).

The results of solving (36) and (37) on a digital computer assuming \( e = \delta/2 \) are shown in Figure 2. The variables \( \alpha/\delta \) and \( \gamma/\delta \) are plotted versus time in the figure and since (36) and (37) are linear the curves apply for any \( \delta \). For comparison the errors were also determined by simulating the hybrid computer on a digital computer and subtracting the simulated hybrid variables from the ideal solution variables. Curves of \( \alpha/\delta \) and \( \gamma/\delta \) determined in this manner are shown in the figure for three values of \( \delta \). The ideal solution variables \( x_1(t) \) and \( z_1(t) \) are also given.

Note that the agreement between error computed from (23) and (24) and that determined from the simulated hybrid system is almost exact for \( \delta = .01 \), becoming less exact but still reasonably good as \( \delta \) is increased ten fold to .1.

\(^\dagger\)See for example Henrici.\([4]\)
CONCLUSIONS

Equations (23) and (24) or the more compact (25) are differential equations for the sampling error in a general hybrid computer system programmed to solve a vector equation in the form of (1). The equations have evolved in the course of several years of study on the subject contract and include all error equations derived earlier as special cases.

The allocation of operations between the analog and digital computers can be arbitrary but operation of the hybrid computer must conform to the assumptions listed on pages 5 and 6. The error equations hold for a small sampling period and thus could be termed "asymptotic" error equations.

The complexity of the equation is comparable to that of the given problem equation. However, the equation is an explicit expression for sampling error in terms of computable quantities and thus machine solution is quite feasible. The assumptions as to the form of the problem equation and as to the details of the hybrid implementation, while somewhat restrictive, do not prevent a range of practical applications.

The sampling error equation is expected to have application in three areas, namely:

The evaluation of sampling error for a given hybrid computer implementation.

The study of various allocations of operations between the analog and digital equipment with a view to minimizing sampling error, and

In compensating for sampling error.

Although the major use of (25) is expected to occur in specific applications for which an explicit numerical solution can be obtained, there are several facts of interest which result from examining the forcing function of this equation in its general form. First it should be noted that since (25) is linear each part
of the forcing function produces a distinct effect on the solution of the equation, i.e. on sampling error. There are three parts to the forcing function, namely:

(1) \( \frac{\delta f}{\delta z} g \), which arises because of the discrete nature of \( z \) as produced in the digital computer,

(2) \( \frac{\delta + 2e}{2} f_d \), which arises because the vector function \( f_d \) is generated digitally for use in the analog computer, and

(3) \( \delta^p \varphi \), which is caused by the discretization error of the numerical method in the digital computer.

Since the initial conditions on the error equation are zero, its solution, the sampling error, has three parts corresponding to the three parts of the forcing function. Parts (1) and (2) of the forcing function depend on \( \delta \) to the first power and thus the hybrid computer, (with zero order sample and hold devices), must be classified as a first order computational method.

The fact that part (3) of the forcing function depends on \( \delta \) to the \( p \)th power brings up an interesting design point. If \( p \), the order of the numerical method, is 2 or greater, the term \( \delta^p \) is negligible with respect to the other two. Thus since the execution time of the digital computer, (and hence \( \delta \)) increases with the order of the numerical method, it would seem inefficient to use a numerical method of order greater than 2. Further, it would seem desirable and possible to allot some computer time to compensating for sampling error and increasing the order of the overall hybrid system. This could be done by concurrently solving the error equation, expressed in terms of the non-ideal variables, and using \( x^*(t) + \delta(t) \) and \( z^*(t) + \alpha(t) \) as compensated variables.
Finally it should be noted from examining part (2) of the forcing function that execution time, \( e \), has the same effect on error as the sampling period but weighted twice as heavily. Thus every effort should be made to keep execution time at a minimum.
**APPENDIX**

**Differentiation of a Vector with Respect to a Vector:** Let \( x \) be an \( n \) vector, \( y \) a \( q \) vector and \( g(x,y) \) an \( m \) vector. Then the symbol \( \frac{\partial g(x,y)}{\partial x} \) is defined as

\[
\frac{\partial g(x,y)}{\partial x} = \begin{vmatrix}
\frac{\partial g_1(x,y)}{\partial x_1} & \cdots & \frac{\partial g_1(x,y)}{\partial x_n} \\
\vdots & \ddots & \vdots \\
\frac{\partial g_m(x,y)}{\partial x_1} & \cdots & \frac{\partial g_m(x,y)}{\partial x_m}
\end{vmatrix}. \tag{A-1}
\]

**Approximation (1):** Consider the vector function \( g(x_1,x_2+\Delta) \) where \( \Delta \) is a vector of increments in a portion, \( x_2 \), of the vector argument of \( g \). If \( g \) is analytic, use of its Taylor series expansion yields the approximation

\[
g(x_1,x_2+\Delta) = g(x_1,x_2) + \frac{\partial g(x_1,x_2)}{\partial x_2} \Delta \tag{A-2}
\]

which neglects terms which are \( O(\Delta_1^2) \).

**Approximation (2):** Consider an analytic function \( g(t) \) depending on the scalar argument \( t \). Denote by \( I_k \) the intervals \([k\delta,(k+1)\delta]\), \( k = 0,1,2, \ldots \). On any interval \( I_k \) the Taylor series for \( g(t) \) can be used to obtain the approximation

\[
g(k\delta) = g(t) - (t-k\delta) g(t), \quad t \in I_k \tag{A-3}
\]

which neglects terms \( O(\delta^2) \) since on \( I_k \), \( |t-k\delta| < \delta \).

25
The term \((t-k\delta), k = 0, 1, 2 \ldots\) in (A-3) is a sawtooth function with period \(\delta\) and thus has a Fourier series representation of the form

\[
(t-k\delta), t \in I_k = a_0 + S(t)
\]

where

\[
S(t) = \sum_{i=1}^{\infty} (a_i \cos \omega_i t + b_i \sin \omega_i t)
\]

and \(a_0 = \delta/2, \omega_i = 2\pi i/\delta, a_i = 0, b_i = -\delta/\pi i\). Using (A-5) in (A-3) yields

\[
g(k\delta) = g(t) - [\delta/2 + S(t)] \dot{g}(t), t \in I_k
\]

An approximation of the form (A-6) can also be obtained for the function \(G(t)\) defined by

\[
G(t) = g(k\delta); t \in I_{k+e}
\]

where \(I_{k+e}\) is the interval \([k\delta+e, (k+1)\delta + e)\). The result is

\[
G(t) \approx g(t) = \frac{\delta+2e}{2} - S(t-e) \dot{g}(t)
\]

and the neglected terms are still \(O(\delta^2)\).

**Approximation (3):** Consider a linear differential equation with the term \(S(t)\dot{g}(t)\) from (A-6) as a forcing function. Such an equation can be expressed as

\[
\dot{x}(t) = a(t)x(t) + S(t)\dot{g}(t), \quad (A-9)
\]
with zero initial conditions. The solution of (A-9) is given by

\[ x(t) = \int_0^t \varphi(t;\lambda) S(\lambda) \dot{g}(\lambda) \, d\lambda. \]  

(A-10)

Under the assumption that the functions \( \varphi(t,\lambda) \) and \( \dot{g}(\lambda) \) in (A-10) and also their derivatives are bounded and continuous, it can be shown that the solution of the differential equation, \( x(t) \), is \( O(\delta^2) \).


SAMPLING ERRORS IN THE HYBRID SOLUTION OF LINEAR CONSTANT COEFFICIENT EQUATIONS

By Joseph L. Hammond, Jr.
George C. Caldwell

Prepared for
George C. Marshall Space Flight Center
Huntsville, Alabama

Contract No. NAS8-2473

(Development of New Methods and Applications of Analog Computation)

30 August 1968
SAMPLING ERRORS IN THE HYBRID SOLUTION OF LINEAR CONSTANT COEFFICIENT EQUATIONS

by

Joseph L. Hammond, Jr., and George C. Caldwell

TECHNICAL NOTE NO. 18

on

Contract No. NAS8-2473

(Development of New Methods and Applications of Analog Computation)

For

GEORGE C. MARSHALL SPACE FLIGHT CENTER
Huntsville, Alabama
FOREWORD

Dr. George C. Caldwell is Associate Director of the School of Mathematics at Georgia Tech. He has participated in the research reported in this technical note as part of his academic activity and not as an employee of Contract NAS8-2473.
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOREWORD</td>
<td>iii</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>v</td>
</tr>
<tr>
<td>GLOSSARY OF PRINCIPAL SYMBOLS</td>
<td>vi</td>
</tr>
<tr>
<td>INTRODUCTION AND STATEMENT OF PROBLEM</td>
<td>1</td>
</tr>
<tr>
<td>ERROR EQUATION</td>
<td>3</td>
</tr>
<tr>
<td>HYBRID EQUATION</td>
<td>5</td>
</tr>
<tr>
<td>ROOT SHIFT DUE TO SAMPLING</td>
<td>8</td>
</tr>
<tr>
<td>HYBRID PROGRAMMING CONSIDERATIONS TO CONTROL</td>
<td></td>
</tr>
<tr>
<td>ROOT SHIFT DUE TO SAMPLING</td>
<td>12</td>
</tr>
<tr>
<td>EXAMPLES DEALING WITH ROOT SHIFT DUE TO SAMPLING</td>
<td>15</td>
</tr>
<tr>
<td>Example 1</td>
<td>15</td>
</tr>
<tr>
<td>Example 2</td>
<td>15</td>
</tr>
<tr>
<td>Example 3</td>
<td>16</td>
</tr>
<tr>
<td>APPENDIX</td>
<td>21</td>
</tr>
</tbody>
</table>
ABSTRACT

This technical note documents a continuation of the study of sampling error in hybrid computer programs based on the general error equation given in Technical Note No. 17 of the subject contract. Specifically attention is given sampling error for linear constant coefficient problem equations. Such equations could arise as the given problem equation or as an approximation to general nonlinear equations.

The work shows that the hybrid equation accounting for sampling error is a linear constant coefficient equation differing from the ideal equation because of perturbations in both is homogeneous and nonhomogeneous parts. Explicit equations for the solution to both the ideal equation and the hybrid equation accounting for sampling error are given in terms of allocation matrices, the sampling period and the execution time of digital operations.

Two types of errors are identified, namely: that which arises in the nonhomogeneous part of the hybrid equation and which depends primarily on the forcing function of the ideal equation and errors in the transition matrix caused by perturbations in the homogeneous part of the hybrid equation. The latter is examined in detail through a study of root shift due to sampling.

A tractable equation for computing root shift is derived along with useful bounds for this quantity. The equations are used in discussing the problem of programming a hybrid computer to minimize root shift. Three examples are presented to illustrate the determination of root shift.
GLOSSARY OF PRINCIPAL SYMBOLS

- u(t) - ideal vector solution of problem equation
- x(t) - ideal vector solution of equation programmed on the analog computer
- z(t) - ideal vector solution of equation solved digitally
- h - vector of functions equated to u(t)
- f - vector of functions equated to x(t)
- f_a - vector of functions generated and used in the analog computer
- f_d - vector of functions generated digitally for use in the analog computer
- g - vector of functions equated to z(t)
- t - time
- n - order of vectors
- p - order of numerical method
- \( \phi \) - principal error function of numerical method
- \( \delta \) - sampling period
- e - execution time of digital computer
- k - index of discrete time
- \( \Gamma(t) \) - total hybrid computer sampling error vector
- \( u^*(t) \) - vector solution of problem equations which accounts for hybrid computer sampling error (similar notation for other variables)
- \( \partial h/\partial u \) - a matrix defined in the Appendix of Technical Note No. 17
- \( O[g(\delta)] \) - a symbol denoting that some variable (which is \( O[g(\delta)] \)) goes to zero with \( \delta \) at least as rapidly as \( g(\delta) \)
- \( S_1, S_2 \) - sets defined on page 13
- B - matrix of constants
GLOSSARY OF PRINCIPAL SYMBOLS (CONT'D.)

\[ y(t) \] - vector forcing function

\[ x^T \] - transpose of a matrix \( x \)

\[ \|x\|_s \] - spectral norm of \( x \)

\[ C = [c_{ij}] \] - matrix of constants

\( C_{\text{al}1}, C_{\text{al}2}, C_{\text{a}2} \) - matrices in partitions of \( C \)

\( C_{\text{dl}1}, C_{\text{dl}2} \)

\( \lambda_i, \lambda_i^* \) - eigenvalues of \( C \) and \( C^* \)

\( w_i \) - right eigenvector of \( C \)

\( v_i \) - left eigenvector of \( C \)

\[ C^* = C - \frac{\delta}{2} \delta C \]

\[ B^* = B - \frac{\delta}{2} \delta B \]

\( \delta = \theta_1 + b \theta_2 \) - allocation matrix

\( b = \frac{\delta + 2e}{\delta} \) - measure of execution time

\[ u(t) = \begin{bmatrix} x(t) \\ z(t) \end{bmatrix} \]

\[ h = \begin{bmatrix} f_a + f_d \\ g \end{bmatrix} \]

\[ \Gamma(t) = \begin{bmatrix} \gamma(t) \\ \alpha(t) \end{bmatrix} \]

\[ f = f_a + f_d \]
INTRODUCTION AND STATEMENT OF PROBLEM

Technical Note No. 17 of the subject contract presents the derivation of a general differential equation for the sampling error caused by a nonzero period between discrete samples of the analog signals in closed-loop hybrid computer programs. The result can be stated concisely as follows: Given the vector equation

\[ \dot{u}(t) = h(u; t), \]  

which is programmed on a hybrid computer through use of the partitioned equations

\[ \begin{align*}
\dot{x}(t) &= f_a(x, z; t) + f_d(x, z; t) \\
\dot{z}(t) &= g(x, z; t),
\end{align*} \]  

the sampling error \( \Gamma(t) \) is shown to satisfy

\[ \Gamma(t) = \frac{\partial h}{\partial u} \Gamma(t) + \begin{bmatrix}
\frac{\partial f_a}{\partial x} g + \frac{\partial f_d}{\partial z} \frac{p}{\delta} \\
\frac{\partial f_d}{\partial x} \frac{p}{\delta} \end{bmatrix}. \]  

Equation (4) neglects terms of second and higher order in the small quantities and to this order of accuracy the arguments of \( g, f_a, f_d \) and \( \phi \) are time and either the solutions, \( u(t) \), of the ideal equation (1) or the solutions \( u^*(t) \) computed from (1) using the hybrid computer.


2 For convenience in this report \( b = \frac{e^2 + 2e}{\delta} \) has been introduced in (25) of Technical Note No. 17 to obtain (4) above. Since \( e > 0 \) and since for reasonable hybrid operation \( e \leq \delta \), it follows that \( 1 \leq b \leq 3 \) with the particular value of \( b \) depending on the execution time \( e \).
Note that (2) is the vector equation integrated on the analog computer, (3) is the vector equation integrated digitally, \( f_a \) is the vector of functions generated in the analog computer and \( f_d \) is the vector of functions computed digitally. The three terms in the forcing function of the error equation arise as follows: 

- \( \varepsilon/2 \frac{\partial f_a}{\partial z} \) is caused by the fact that \( z \) is generated as a discrete digital variable.
- \( \varepsilon/2 b \cdot f_d \) is caused by generating \( f_d \) digitally with unavoidable execution time and
- \( \delta P \) results from truncation errors in the numerical method used in the digital computer.

As might be expected, the error equation reduces to a tractable form if (1) can be written as

\[
\dot{u}(t) = C \ u(t) + B \ y(t) \tag{5}
\]

where \( C \) and \( B \) are constant matrices with elements \( [c_{ij}] \), \( [b_{ij}] \) and \( y(t) \) is a forcing function representing, for example, inputs to a physical system described by (5). Equation (5) could be the form of the equation to be solved by the hybrid computer. It could also be an approximation to the "local" behavior of a more complicated nonlinear or time varying vector equation such as given by (1). In the latter case, the variables in (5) would represent changes in the variables of the general equation.

The purpose of this report is to examine the error equation for hybrid programs for (5). Tractable equations and bounds for the sampling error are obtained. The results are useful in a quantitative sense when (5) is the equation to be solved by the hybrid computer. In more common cases where the hybrid computer is programmed for a much more general equation than (5), the results will be useful in a qualitative sense as a guide to sampling error.
ERROR EQUATION

To express (5) in the form of (2) and (3), the matrices \( C \) and \( B \) are partitioned to obtain

\[
\begin{bmatrix}
(a) & (d) \\
\begin{bmatrix}
C_{a1} + C_{a1} \\
C_{a2} + C_{a2} \\
C_{d1} \\
C_{d2}
\end{bmatrix}
\end{bmatrix}
\begin{bmatrix}
(a) & (d) \\
\begin{bmatrix}
B_{a1} + B_{a1} \\
B_{a2} + B_{a2} \\
B_{d1} \\
B_{d2}
\end{bmatrix}
\end{bmatrix}.
\]

In some cases it will be convenient to use the additional notation

\[
C_{a1} + C_{a1} = C_{a1}; C_{a2} + C_{a2} = C_{a2}; B_{a1} + B_{a2} = B_{a1}; B_{a2} + B_{a2} = B_{a2}.
\]

The partitioned ideal equations then become

\[
x(t) = \left[ \begin{bmatrix}
C_{a1}x + C_{a2}z + B_{a1}y_1 + B_{a2}y_2 \\
C_{d1}x + C_{d2}z + B_{d1}y_1 + B_{d2}y_2
\end{bmatrix} \right] + \left[ \begin{bmatrix}
C_{a1}x + C_{a2}z + B_{a1}y_1 + B_{a2}y_2 \\
C_{d1}x + C_{d2}z + B_{d1}y_1 + B_{d2}y_2
\end{bmatrix}\right] (6)
\]

\[
z(t) = \left[ \begin{bmatrix}
C_{a1}x + C_{a2}z + B_{a1}y_1 + B_{a2}y_2 \\
C_{d1}x + C_{d2}z + B_{d1}y_1 + B_{d2}y_2
\end{bmatrix} \right] (7)
\]

where the first bracketed term in (6) is \( f_a \), the second is \( f_d \), the bracketed term in (7) is \( g \) and \( y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} \).

Using (6) and (7) the terms required in (4) can be determined. After some algebraic manipulation the results are

\[
\frac{\partial h}{\partial u} = C
\]

\[
\frac{\partial f_a}{\partial z} = C_{a2}
\]

\[
f_d = C_{a1}(C_{a1}x + C_{a2}z + B_{a1}y_1 + B_{a2}y_2) + C_{a2}(C_{d1}x + C_{d2}z + B_{d1}y_1 + B_{d2}y_2) + B_{a1}y_1 + B_{a2}y_2
\]

The principal error function, \( \phi \), of the numerical method depends on the particular method chosen. In general it will depend on \( z \) and in many
cases, on derivatives of $z$ which can be expressed in terms of $z$ itself and $x$. For example for Euler's method

$$
\phi(x, z, y; t) = -\frac{1}{2} \left[ C_{d1}(C_{a1}x + C_{a2}z + B_{a1}y_1 + B_{a2}y_2) + C_{d2}(C_{d1}x + C_{d2}z + B_{d1}y_1 + B_{d2}y_2) + B_{d1}y_1 + B_{d2}y_2 \right].
$$

Considerable simplification of the error equation results from use of the definitions

$$
\begin{align*}
\theta_1 &= \begin{bmatrix} (d) & (d) \\ C_{a1} & C_{a2} \\ 0 & 0 \end{bmatrix} \\
\theta_2 &= \begin{bmatrix} (a) \\ 0 & C_{a2} \\ 0 & 0 \end{bmatrix} \\
\beta &= \begin{bmatrix} (d) & (d) \\ B_{a1} & B_{a2} \\ 0 & 0 \end{bmatrix}
\end{align*}
$$

(8), (9), (10)

$$
\phi(x, z, y; t) = \begin{bmatrix} 0 \\ -\delta P \phi(x, z, y; t) \end{bmatrix}
$$

(11)

Using (8) - (11), (4) then becomes

$$
\dot{\hat{r}}(t) = C\Gamma(t) + \frac{\delta}{2}(b\theta_1 + \theta_2)C\ u(t) + \frac{\delta}{2}(b\theta_1 + \theta_2)B\ y(t) + \frac{\delta}{2}b\beta\ y(t) + \phi(u, y; t).
$$

(12)

In (12) the hybrid variable $u^*$ could be used instead of the ideal variable within the order of accuracy of the equation.
Equation (12), while being an explicit expression for sampling error, has the undesirable property that its solution depends on either \( u(t) \) or \( u^*(t) \). Since this is the case 2n equations, for example (6), (7), and (12), must be solved to obtain the error. However, because of special properties of linear constant coefficient equations, it is possible to derive from these equations a set of n first order equations which completely specify the hybrid program within the accuracy being considered (i.e., neglecting terms of higher than first order in the small quantities).

The desired result is obtained by noting that the derivative of the hybrid solution vector can be expressed as

\[
\dot{u}^*(t) = \dot{u}(t) - \dot{\Gamma}(t),
\]

and then using the right-hand sides of (5) and (12) to substitute for \( \dot{u}(t) \) and \( \dot{\Gamma}(t) \). In the expression for \( \dot{\Gamma}(t) \) the hybrid variable \( u^*(t) \) is used. The result is

\[
\dot{u}^*(t) = C\dot{u}(t) + B\dot{y}(t) - C[u(t) - u^*(t)] - \varepsilon/2[\theta_1 + \theta_2]C u^*(t) - \varepsilon/2[\theta_1 + \theta_2]B y(t) - \varepsilon/2 b\dot{y}(t) - \Phi(u^*, y; t).
\]

Note that the two terms involving \( u(t) \) cancel\(^3\) so that (14) is a set of n first-order equations for \( u^*(t) \).

To obtain tractable results it will be assumed that the numerical method used in the digital computer falls into one of two categories, either: case (a) the method is of order two or larger so that the term \( \Phi \)

\(^3\)This cancellation is peculiar to linear constant coefficient equations and does not occur in general.
in (14) is negligible or case (b) the method is of first order with the additional property that \( \phi \) can be expressed as

\[
\phi(u^*, y; t) = \begin{bmatrix}
0 \\
\varepsilon/2[\theta Cu^*(t) + \theta By(t) + \beta'y(t)]
\end{bmatrix}
\]

(15)

where \( \theta \) and \( \beta' \) are matrices (similar to \( \theta_1, \theta_2, \) and \( \beta \)) with zeros in the first row and submatrices in the second row determined by the particular numerical method used. It will be assumed that \( \theta \) can be obtained from \( C \) by setting certain of its elements equal to zero. Note that Euler's method results in a \( \phi \) satisfying (15).

For case (b),

\[
\begin{align*}
\dot{u}^*(t) &= [C - \varepsilon/2(\theta + b\theta_1 + \theta_2)C]u^*(t) + [B - \varepsilon/2(\theta + b\theta_1 + \theta_2)B]y(t) \\
&\quad - \varepsilon/2(b\beta + \beta')y(t) .
\end{align*}
\]

(16)

For case (a) an accurate approximation for \( u^*(t) \) results from use of (16) with \( \theta \) and \( \beta' = 0 \). Thus since the hybrid computer solution of (5) is the true solution of (16), (with appropriate values for \( \theta \) and \( \beta' \)), the effects of sampling are placed in evidence by a comparison of (5) and (16). To aid in the comparison the definitions

\[
C^* = C - \varepsilon/2\theta C \quad (17)
\]

\[
B^* = B - \varepsilon/2\theta B \quad (18)
\]

\[
\theta = \theta + b\theta_1 + \theta_2 , \quad (19)
\]

can be introduced so that (16) becomes

\[\text{Note that} \quad \theta, \text{ as defined in (19), could be termed an "allocation matrix" for a particular hybrid program.}\]
\[ u^*(t) = C^* u^*(t) + b^* y(t) - \frac{\delta}{2}(b\beta + \beta')y(t). \]  

Equations (20) and (5) are both linear constant coefficient equations which, when compared, show the effects of sampling to be: (a) a perturbation of \( C \) to \( C^* \) as given by (17); (b) a perturbation of \( B \) to \( B^* \) as given by (18) and (c) the addition of a forcing function \(-\frac{\delta}{2}(b\beta + \beta')y(t)\). Note that (20) reduces to (5) for \( \delta = 0 \) as would be expected.

Since (5) and (20) are linear constant coefficient equations, expressions for their solutions, \( u(t) \) and \( u^*(t) \), can be expressed in closed form using standard techniques. In obtaining the solution of (20) it is straightforward to express the result in terms of \( y(t) \) and remove the dependence on \( y(t) \). The resulting equations are

\[ u(t) = \eta(t) u(t = 0) + \int_0^t \eta(t - \tau) B y(\tau) d\tau \]  
\[ u^*(t) = \eta^*(t) u^*(t = 0) + \int_0^t \eta^*(t - \tau) B y(\tau) d\tau \]

\[ + \frac{\delta}{2} \left[ \eta^*(t)(b\beta + \beta') y(t = 0) - (b\beta + \beta') y(t) \right] \]

\[ - \int_0^t \eta^*(t - \tau) \left[ b\beta + C(b\beta + \beta') \right] y(\tau) d\tau \]

where \( \eta(t) \) is the transition matrix determined by the solution of the homogeneous part of (5) and \( \eta^*(t) \) is the transition matrix determined by the solution of the homogeneous part of (20). In (22) a term \( O(\delta^2) \) has been neglected.

In comparing (21) and (22), the error can be classified in two parts:

(a) error caused by the presence in (22) of \( \eta^*(t) \) replacing \( \eta(t) \) and

---

(b) error caused by the presence of $\delta/2$ times the term in the brackets \[[\] in (22). Note that the former is determined by the extent of the perturbations in $C$ and that the latter is determined primarily by the nature of $y(t)$. In many cases it is reasonable to assume that the difference in $\eta(t)$ and $\eta^*(t)$ is $O(\delta)$ and in such cases by neglecting terms $O(\delta^2)$ error (b) can be computed from the bracketed term in (22) with $\eta^*(t)$ replaced by $\eta(t)$. Little more of general nature can be said regarding this type of error for arbitrary $y(t)$.

Attention will be directed in the next section to a more detailed discussion of error (a) using the approach of examining the roots of $C$ and $C^*$ which of course determine $\eta(t)$ and $\eta^*(t)$.

ROOT SHIFT DUE TO SAMPLING

The homogeneous solutions and the transition matrices of (5) and (20) are determined by the eigenvalues of $C$ and $C^*$. The purpose of this section is to relate the eigenvalues of these two matrices and thus place in evidence the root shift caused by sampling error. In many cases a study of sampling error can be restricted to a consideration of this root shift.

Changes in the eigenvalues of a matrix due to small changes in the elements of the matrix have been studied in conjunction with numerical techniques for determining the eigenvalues of a matrix. An excellent account of this problem is given by Wilkinson. 6

Attention is restricted to those \( C \) for which the eigenvalues are simple (but not necessarily real) and isolated. With the definitions

- eigenvalues of \( C = \lambda_i, \ i = 1, 2, \ldots, n \)
- eigenvalues of \( C^* = \lambda_i^*, \ i = 1, 2, \ldots, n \)
- right eigenvectors of \( C = w_i, \ i = 1, 2, \ldots, n \)
- left eigenvectors of \( C = v_i, \ i = 1, 2, \ldots, n \)

Wilkinson shows that for \( \varepsilon \to 0 \)

\[
\lambda_i^* \approx \lambda_i - \frac{\varepsilon}{2} \frac{v_i^T \delta C w_i}{v_i^T w_i}
\]  

(23)

Using the relation

\[
C w_i = \lambda_i w_i
\]  

(24)

which holds for the eigenvalues and eigenvectors of \( C \), (23) can be reduced to

\[
\Delta \lambda_i = \frac{\lambda_i^* - \lambda_i}{\lambda_i} \approx -\frac{\varepsilon}{2} \frac{v_i^T \delta w_i}{v_i^T w_i}
\]  

(25)

Note that \( \Delta \lambda_i \), the fractional root shift for the \( i \)th root, can be approximated by (25) if \( \varepsilon \) is not "too large." Unfortunately the exact range of \( \varepsilon \) for (25) to be reasonable approximation cannot be determined without evaluating the eigenvalues of both \( C \) and \( C^* \).

Note further that the normalization of the eigenvalues of \( C \) in (25) is arbitrary and thus the choice
\[ v_i^T w_i = 1, \text{ all } i \]  

(26)

can be made. Thus using (26) in (25)

\[ \Delta \lambda_i \simeq -\frac{\delta}{2} v_i^T \Theta w_i; \quad v_i^T v_i = 1 \]  

(27)

For some purposes a more tractable expression for \( \Delta \lambda_i \) results from expressing the matrix operations in (27) as equivalent sums to obtain

\[ \Delta \lambda_i \simeq -\frac{\delta}{2} \sum_{r=1}^{n} \sum_{s=1}^{n} \Theta_{rs} w_i^s v_i^r; \quad v_i^T v_i = 1 \]  

(28)

\( (s) \quad (r) \)

where \( w_i^s \) and \( v_i^r \) are the components of \( w_i \) and \( v_i \) respectively and \( \Theta_{rs} \) are the elements of the matrix \( \Theta \).

By using the alternative normalization

\[ ||w_i||_s = ||v_i||_s = 1, \text{ all } i \]

and the Schwartz inequality a bound on \( |\Delta \lambda_i| \) can be obtained from (25).

The result is

\[ |\Delta \lambda_i| \leq \frac{\delta}{2} \frac{||\Theta||_s}{||v_i^T v_i||} ; \quad ||w_i||_s = ||v_i||_s = 1 \]  

(29)

where the spectral norm \( || \cdot ||_s \) is defined as the square root of the largest eigenvalue of \( \Theta^T \Theta \). Wilkinson, for example, shows that \( ||\Theta||_s \) can be bounded by
\[
\| \Theta \|_s \leq \left( \sum_{r=1}^{n} \sum_{s=1}^{n} |\theta_{rs}|^2 \right)^{\frac{1}{2}} \tag{30}
\]

or by
\[
\| \Theta \|_s \leq \left[ \max_s \sum_{r=1}^{n} |\theta_{rs}| \max_r \sum_{s=1}^{n} |\theta_{rs}| \right]^{\frac{1}{2}} \tag{31}
\]

Note that in (31) the first term on the right-hand side is the maximum row sum of the elements of \( \Theta \) and that the second term is the maximum column sum.

Either (30) or (31) can be used with (29) and the resulting bounds for fractional root shift are
\[
|\Delta \lambda_1| \leq \frac{\varepsilon}{2} \frac{1}{|v_i^T w_i|} \left[ \sum_{r=1}^{n} \sum_{s=1}^{n} |\theta_{rs}|^2 \right]^{\frac{1}{2}} \leq \frac{\varepsilon}{2} \frac{n \max_{r,s} |\theta_{rs}|}{|v_i^T w_i|} \tag{32}
\]

and
\[
|\Delta \lambda_1| \leq \frac{\varepsilon}{2} \frac{1}{|v_i^T w_i|} \left[ \max_s \sum_{r=1}^{n} |\theta_{rs}| \max_r \sum_{s=1}^{n} |\theta_{rs}| \right]^{\frac{1}{2}} \tag{33}
\]

where in both equations \( \| w_i \|_s = \| v_i \|_s = 1 \).

The results of this section, principally (27), (28), (32), and (33), will be used in the next section to discuss the problem of programming a hybrid computer to control sampling error as measured by shifts in the roots of the homogeneous equation.
HYBRID PROGRAMMING CONSIDERATIONS TO
CONTROL ROOT SHIFT DUE TO SAMPLING

Specification of a particular equation in the form of (5) for hybrid solution fixes a C matrix. A particular hybrid program is then determined by specifying in addition: (a) the allocation of operations between the analog and digital computers, (b) the numerical method to be used by the digital computer, (c) the execution time of the digital operations and (d) the sampling period. The effect of a particular C and each of the choices (a), (b), (c) and (d) on root shift is placed in evidence by the equations derived in the previous section.

Specification of C determines the eigenvalues \( \lambda_i \), the corresponding eigenvectors \( w_i \) and \( v_i \), and the elements which can potentially be contained in \( \Theta \). The latter comment follows from the fact that \( \Theta \), \( \Theta_1 \) and \( \Theta_2 \), the constituent parts of \( \Theta \), are all obtained from C by setting certain of its elements equal to zero. This fact makes it possible to modify the last inequality of (32) to read

\[
|\Delta \lambda_i| \leq \frac{\delta/2}{\max_r \max_s |c_{rs}| \frac{r_s}{v_i^T w_i}} ; ||w_i||_s = ||v_i||_s = 1
\]  

Note that this bound depends entirely on properties of C and not on how the computing operations are allocated between the analog and digital computers. The number

\[
\frac{n \max_{r,s} |c_{rs}|}{\frac{r_s}{v_i^T w_i}} ; ||w_i|| = ||v_i||_s = 1
\]
which is completely determined by \( C \), can thus be used as a measure of the suitability of a given vector equation for hybrid solution.

Assuming that \( C \) is fixed the next consideration is the choice of a particular hybrid program as described by the choices (a) - (d) listed above. To begin this consideration it is useful to express (27) in the form

\[
\Delta \lambda_1 = -\frac{\delta}{2} (v_i^T \theta w_i + b v_i^T \theta_1 w_i + v_i^T \theta_2 w_i); \quad v_i^T w_i = 1
\]  \hspace{1cm} (35)

This equation shows explicitly the effects of sampling period, \( \delta \), and the measure of execution time, \( b \), on root shift. Furthermore, (35) shows that the effect of the numerical method, which determines \( \theta \), can be considered independently of the allocation of operations between the analog and digital computers. The latter will now be considered in more detail independently of the former by the expedient of setting \( \delta = 0 \).

Consider (28) with \( \delta = 0 \). As has been noted previously, \( w_i \) and \( v_i \) (and hence all of their components \( w_i \) and \( v_i \)) are determined by a given \( C \). What remains is an interpretation of the double sum over the elements of \( \theta \). Reference to (8) and (9) defining \( \theta_1 \) and \( \theta_2 \) shows that these matrices are formed from \( C \) by setting certain of its elements equal to zero. From the definition of \( \theta_1 \) and \( \theta_2 \) (28) can be expressed as

\[
\Delta \lambda_1 \approx -\frac{\delta}{2} \sum_{r, s \in S_1} c_{rs} w_i v_i + \sum_{r, s \in S_2} c_{rs} w_i v_i \]  \hspace{1cm} (36)

where, with \( m \) the number of equations integrated with the analog computer, \( S_1 \) and \( S_2 \) are defined as
\[ S_1: \ r, s \ \text{such that} \ 1 \leq r \leq m, \ c_{rs} \ \text{multiplied digitally} \]

\[ S_2: \ r, s \ \text{such that} \ 1 \leq r \leq m, \ m+1 \leq s \leq n, \ c_{rs} \ \text{multiplied in the analog computer.} \]

The definition of the allocation matrix \( \Theta \) and hence all the work to this point has assumed that the equations to be integrated on the analog computer are the first \( m \) equations and hence correspond to the first \( m \) rows of \( C \). As is demonstrated in the Appendix, the results are independent of this requirement if the sets \( S_1 \) and \( S_2 \) or an allocation matrix \( \Theta \) are defined by the following rules:

1. Choose the equations to be solved on the analog computer. (The others will be solved on the digital computer.)
2. The choice (1) defines a collection of analog and a collection of digital variables.
3. Define \( S_1 \) as all \( r, s \), unrestricted as to position in the \( C \) matrix, such that \( c_{rs} \) is in one of the equations integrated on the analog computer and \( c_{rs} \) is multiplied by the appropriate variable in the digital computer.
4. Define \( S_2 \) as all \( r, s \), unrestricted as to position in the \( C \) matrix, such that \( c_{rs} \) is in one of the equations integrated on the analog computer and \( c_{rs} \) is multiplied by a digital variable in the analog computer.

Note that having selected the equations to be integrated on the analog computer, there is no restriction on which elements are multiplied by their variables in the digital computer. Both choices however, (i.e., analog equations and digital multiplications), uniquely define \( S_1 \) and \( S_2 \), an allocation matrix \( \Theta \) and hence \( \Delta_{1} \).

Equation (36) shows concisely the effect of various allocations of the computing operations as determined by the sets \( S_1 \) and \( S_2 \). The bounds
expressed by (32) and (33) also apply and show in general that to keep root shift small, \( \Theta \) should be allowed to contain only the smallest elements of \( C \).

**EXAMPLES DEALING WITH ROOT SHIFT DUE TO SAMPLING**

**Example 1:** The special case resulting if all equations are integrated on the analog computer and all of the multiplications are carried out on the digital computer yields very tractable results. For this case, assuming \( \Theta = 0 \),

\[
C = C_{al} = \Theta_1 = \Theta
\]

and (25) yields

\[
\Delta \lambda_i \approx -\frac{\delta}{2} b \lambda_i.
\]  

Equation (37) shows that for the case being considered \( b \delta \lambda_i \) must be kept small for \( i = 1, \ldots, n \) to avoid excessive fractional root shift and that the largest eigenvalue is the determining factor in overall error. To insure a fractional root shift of less than 0.01, for example, requires \( b \delta \lambda_i \leq .02 \) or

\[
\text{samples/cycle of } \lambda_i \leq 100b; \ i = 1, \ldots, n.
\]  

**Example 2:** If

\[
C = \begin{bmatrix}
\lambda_1 & 0 \\
0 & \lambda_n
\end{bmatrix}
\]

so that the given equations are uncoupled, then
\( w_i = v_i = \begin{bmatrix} 0 & \ldots & 1 & \ldots & 0 \end{bmatrix}^T \)

and

\[
\begin{bmatrix}
(\text{d}) \\
\alpha_1 \\
0 \\
0
\end{bmatrix}
\begin{bmatrix}
b \\
a_1 \\
0 \\
0
\end{bmatrix}
= 
\begin{bmatrix}
\alpha_1 & \ldots & \alpha_m \\
0 & \ldots & 0
\end{bmatrix}
\]

where \( \alpha_i \) is zero if \( \lambda_i \) is multiplied with the analog computer and \( \lambda_i \) if the multiplication is done digitally. Using (27) a straightforward calculation gives

\[
\Delta \lambda_i = -\frac{5}{2} b \alpha_i.
\] (40)

Thus

\[
\Delta \lambda_i \leq -\frac{5}{2} b \lambda_{\text{max}}, \text{ all } i
\] (41)

where \( \lambda_{\text{max}} \) is the largest \( \alpha_i \).

Note that in this case the fractional root shift, for all roots, is bounded by \( b \delta / 2 \) times the largest \( \lambda_i \) for which multiplication is done digitally.

Example 3: This example illustrates the use of (36) in finding \( \Delta \lambda_i \) for a matrix without special properties. Let the following be given

\[
\begin{align*}
\cdot & 
\dot{u}_1 = 2u_1 + u_2 + u_3 \quad (42-1) \\
\cdot & 
\dot{u}_2 = 3u_2 \quad (42-2) \\
\cdot & 
\dot{u}_3 = 3u_1 + 4u_3 \quad . (42-3)
\end{align*}
\]
Then

\[ C = \begin{bmatrix} 2 & 1 & 1 \\ 0 & 3 & 0 \\ 3 & 0 & 4 \end{bmatrix}, \quad \mathbf{w}_1 = \begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}, \quad \mathbf{w}_2 = \begin{bmatrix} 4 \\ -3 \\ 3 \end{bmatrix}, \quad \mathbf{w}_3 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \]

\[ \lambda_1 = 1, \quad \lambda_2 = 3, \quad \lambda_3 = 5, \quad \mathbf{v}_1 = \begin{bmatrix} 3/4 \\ -3/8 \\ -1/4 \end{bmatrix}, \quad \mathbf{v}_2 = \begin{bmatrix} 0 \\ 1/4 \\ 0 \end{bmatrix}, \quad \mathbf{v}_3 = \begin{bmatrix} 1/4 \\ 1/8 \\ 1/4 \end{bmatrix} \]

Note that the \( \mathbf{w}_i \) and \( \mathbf{v}_i \) are normalized so that \( \mathbf{v}_i^T \mathbf{w}_i = 1 \).

Consider the problem of allocating the computing operations between the analog and digital computers. This is the problem of choosing a particular allocation matrix, \( \Theta \), or particular sets \( S_1 \) and \( S_2 \) associated with (35). The choice of all analog or all digital computation, which results in zero sampling error, will not be considered further. Thus there remains 6 different allocations of the equations, namely:

\begin{align*}
(36-1) & \quad A & D & D & A & D & A \\
(36-2) & \quad D & A & D & A & A & D \\
(36-3) & \quad D & D & A & D & A & A \\
\end{align*}

where A indicates analog integration and D digital integration. Furthermore, any of the elements in the analog equations can be multiplied by their respective variables in the digital computer.

To facilitate the general study (36) is written as

\[ -\frac{2}{\delta} \Delta \lambda_i = \sum_r \sum_s c_{rs} \mathbf{w}_i r_i \quad \text{(s)} \quad \text{(r)} \]

\[ (43) \]
where the double sum is shown to include all \(r, s\). In using (43) it will be understood that for a particular allocation those \(c_{rs}'\) for \(r, s\) not in either \(S_1\) or \(S_2\), (as defined on page 13) will be zero; those \(c_{rs}'\) for \(r, s\) in \(S_1\) will be replaced by \(bc_{rs}\) and those \(c_{rs}'\) for \(r, s\) in \(S_2\) will be replaced by \(c_{rs}\). Applying (39) to the problem equations (42-1), (42-2) and (42-3) yield: for any allocation

\[
-2/5 \Delta \lambda_1 = 3/4 c_{11}' - 1/4 c_{31}' - 3/4 c_{13}' - 1/4 c_{33}' \tag{44}
\]

\[
-2/5 \Delta \lambda_2 = c_{22}' \tag{45}
\]

\[
-2/5 \Delta \lambda_3 = 1/4 c_{11}' + 1/4 c_{31}' + 3/4 c_{13}' + 3/4 c_{33}' \tag{46}
\]

Note that the \(c_{rs}'\) which are always zero are not included.

Allowing for potential digital multiplication in addition to the allocations of the equation listed above, there are 42 different possible hybrid programs for the equation of this example. For each different case there are values for the shifts in each of the three roots. The results of a complete study of the various allocations are presented in Table I which gives 2/5 times the root shift for all allocations in terms of \(b\), the measure of execution time. (Recall that \(b\) lies between 1 and 3.)

Note that zero root shift is obtained for two cases, namely: No. 2, (42-1)D, (42-2)A, (42-3)D and No. 5, (42-1)A, (42-2)D, (42-3)A with no digital multiplications. This is reasonable since (42-2) is decoupled from the other two equations.

For other cases the largest absolute value of \(2/5 \Delta \lambda_1\) for a given allocation ranges from \(3/4\) to \(29/4\) \(b\).
TABLE I

Values of $2/5$ times fractional root shift for each root, six equation allocations and all possible allocations of the multiplication operations for Example 3. Values of $c'_{rs}$ which are a multiple of $b$ indicate that the corresponding element is multiplied in the digital computer.

<table>
<thead>
<tr>
<th>Allocation</th>
<th>$2/5 \lambda_1$</th>
<th>$2/5 \lambda_2$</th>
<th>$2/5 \lambda_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. $42-1$ A, $42-2$ D, $42-3$ D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$c'<em>{11} = 0$, $c'</em>{13} = 1$</td>
<td>$3/4$</td>
<td>$0$</td>
<td>$-3/4$</td>
</tr>
<tr>
<td>$c'<em>{11} = 0$, $c'</em>{13} = b$</td>
<td>$3/4b$</td>
<td>$0$</td>
<td>$-3/4b$</td>
</tr>
<tr>
<td>$c'<em>{11} = 2b$, $c'</em>{13} = 1$</td>
<td>$-3/2b + 3/4$</td>
<td>$0$</td>
<td>$-b/2 - 3/4$</td>
</tr>
<tr>
<td>$c'<em>{11} = 2b$, $c'</em>{13} = b$</td>
<td>$3/4b$</td>
<td>$0$</td>
<td>$-5/4b$</td>
</tr>
<tr>
<td>2. $42-1$ D, $42-2$ A, $42-3$ D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$c'_{22} = 0$</td>
<td>$0$</td>
<td>$0$</td>
<td>$0$</td>
</tr>
<tr>
<td>$c'_{22} = 3b$</td>
<td>$0$</td>
<td>$-3b$</td>
<td>$0$</td>
</tr>
<tr>
<td>3. $42-1$ D, $42-2$ D, $42-3$ A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$c'<em>{31} = 3$, $c'</em>{33} = 0$</td>
<td>$3/4$</td>
<td>$0$</td>
<td>$-3/4$</td>
</tr>
<tr>
<td>$c'<em>{31} = 3$, $c'</em>{33} = 4b$</td>
<td>$3/4 + b$</td>
<td>$0$</td>
<td>$-3/4 - 3b$</td>
</tr>
<tr>
<td>$c'<em>{31} = 3b$, $c'</em>{33} = 0$</td>
<td>$3/4b$</td>
<td>$0$</td>
<td>$-3/4b$</td>
</tr>
<tr>
<td>$c'<em>{31} = 3b$, $c'</em>{33} = 4b$</td>
<td>$7/4b$</td>
<td>$0$</td>
<td>$-15/4b$</td>
</tr>
<tr>
<td>4. $42-1$ A, $42-2$ A, $42-3$ D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$c'<em>{11} = 0$, $c'</em>{13} = 2$, $c'_{22} = 0$</td>
<td>$3/2$</td>
<td>$0$</td>
<td>$-3/2$</td>
</tr>
<tr>
<td>$c'<em>{11} = 0$, $c'</em>{13} = 2$, $c'_{22} = 3b$</td>
<td>$3/2$</td>
<td>$-3b$</td>
<td>$-3/2$</td>
</tr>
<tr>
<td>$c'<em>{11} = 0$, $c'</em>{13} = 2b$, $c'_{22} = 0$</td>
<td>$3/2b$</td>
<td>$0$</td>
<td>$-3/2b$</td>
</tr>
<tr>
<td>$c'<em>{11} = 0$, $c'</em>{13} = 2b$, $c'_{22} = 3b$</td>
<td>$3/2b$</td>
<td>$-3b$</td>
<td>$-3/2b$</td>
</tr>
<tr>
<td>$c'<em>{11} = 2b$, $c'</em>{13} = 2$, $c'_{22} = 0$</td>
<td>$-3/2b + 3/2$</td>
<td>$0$</td>
<td>$-(b/2+3/2)$</td>
</tr>
<tr>
<td>$c'<em>{11} = 2b$, $c'</em>{13} = 2$, $c'_{22} = 3b$</td>
<td>$-3/2b + 3/2$</td>
<td>$-3b$</td>
<td>$-(b/2+3/2)$</td>
</tr>
<tr>
<td>$c'<em>{11} = 2b$, $c'</em>{13} = 2b$, $c'_{22} = 0$</td>
<td>$0$</td>
<td>$0$</td>
<td>$-2b$</td>
</tr>
<tr>
<td>$c'<em>{11} = 2b$, $c'</em>{13} = 2b$, $c'_{22} = 3b$</td>
<td>$0$</td>
<td>$-3b$</td>
<td>$-2b$</td>
</tr>
</tbody>
</table>

19
|   | 42-1 A, 42-2 D, 42-3 A | \( \Delta \lambda_1 \) | \( \Delta \lambda_2 \) | \( \Delta \lambda_3 \) |
|---|---|---|---|
| 5. | \( c'_{11} = 0, c'_{13} = 0, c'_{31} = 0, c'_{33} = 0 \) | 0 | 0 | 0 |
|   | \( c'_{11} = 0, c'_{13} = 0, c'_{31} = 0, c'_{33} = 4b \) | \( b \) | 0 | -3b |
|   | \( c'_{11} = 0, c'_{13} = 0, c'_{31} = 3b, c'_{33} = 0 \) | \( 3/4b \) | 0 | -3/4b |
|   | \( c'_{11} = 0, c'_{13} = 0, c'_{31} = 3b, c'_{33} = 4b \) | \( 7/4b \) | 0 | -15/4b |
|   | \( c'_{11} = 0, c'_{13} = 4b, c'_{31} = 0, c'_{33} = 0 \) | 3b | 0 | -3b |
|   | \( c'_{11} = 0, c'_{13} = 4b, c'_{31} = 0, c'_{33} = 4b \) | 4b | 0 | -6b |
|   | \( c'_{11} = 0, c'_{13} = 4b, c'_{31} = 3b, c'_{33} = 0 \) | \( 15/4b \) | 0 | -15/4b |
|   | \( c'_{11} = 0, c'_{13} = 4b, c'_{31} = 3b, c'_{33} = 4b \) | \( 19/4b \) | 0 | -27/4b |
|   | \( c'_{11} = 2b, c'_{13} = 0, c'_{31} = 0, c'_{33} = 0 \) | -3/2b | 0 | -b/2 |
|   | \( c'_{11} = 2b, c'_{13} = 0, c'_{31} = 0, c'_{33} = 4b \) | -1/2b | 0 | -7/2b |
|   | \( c'_{11} = 2b, c'_{13} = 0, c'_{31} = 3b, c'_{33} = 0 \) | -3/4b | 0 | -5/4b |
|   | \( c'_{11} = 2b, c'_{13} = 0, c'_{31} = 3b, c'_{33} = 4b \) | 1/4b | 0 | -17/4b |
|   | \( c'_{11} = 2b, c'_{13} = 4b, c'_{31} = 0, c'_{33} = 0 \) | 3/2b | 0 | -7/4b |
|   | \( c'_{11} = 2b, c'_{13} = 4b, c'_{31} = 0, c'_{33} = 4b \) | 5/2b | 0 | -13/2b |
|   | \( c'_{11} = 2b, c'_{13} = 4b, c'_{31} = 3b, c'_{33} = 0 \) | 9/4b | 0 | -17/4b |
|   | \( c'_{11} = 2b, c'_{13} = 4b, c'_{31} = 3b, c'_{33} = 4b \) | 13/4b | 0 | -29/4b |

<table>
<thead>
<tr>
<th></th>
<th>( \Delta \lambda_1 )</th>
<th>( \Delta \lambda_2 )</th>
<th>( \Delta \lambda_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.</td>
<td>( c'<em>{22} = 0, c'</em>{31} = 3, c'_{33} = 0 )</td>
<td>3/4</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>( c'<em>{22} = 0, c'</em>{31} = 3, c'_{33} = 4b )</td>
<td>( 3/4 + b )</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>( c'<em>{22} = 0, c'</em>{31} = 3b, c'_{33} = 0 )</td>
<td>( 3/4b )</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>( c'<em>{22} = 0, c'</em>{31} = 3b, c'_{33} = 4b )</td>
<td>( 7/4b )</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>( c'<em>{22} = 3b, c'</em>{31} = 3, c'_{33} = 0 )</td>
<td>( 3/4 )</td>
<td>-3b</td>
</tr>
<tr>
<td></td>
<td>( c'<em>{22} = 3b, c'</em>{31} = 3, c'_{33} = 4b )</td>
<td>( 3/4 + b )</td>
<td>-3b</td>
</tr>
<tr>
<td></td>
<td>( c'<em>{22} = 3b, c'</em>{31} = 3b, c'_{33} = 0 )</td>
<td>( 3/4b )</td>
<td>-3b</td>
</tr>
<tr>
<td></td>
<td>( c'<em>{22} = 3b, c'</em>{31} = 3b, c'_{33} = 4b )</td>
<td>( 7/4b )</td>
<td>-3b</td>
</tr>
</tbody>
</table>
APPENDIX

Proof of Invariance of the Root Shift Equations to a General Definition of the Allocation Matrix: Given a set of \( n \) linear constant coefficient differential equations. Order the equations in some way so as to construct a matrix \( \mathbf{C} \) and let \( \lambda_i, \mathbf{w}_i, \) and \( \mathbf{v}_i \) be respectively the eigenvalues and left and right eigenvectors of \( \mathbf{C} \).

In \( \mathbf{C} \) identify the elements which are in the equations to be solved on the analog computer. The allocation matrix, \( \Theta' \), is then formed from \( \mathbf{C} \) by first equating to zero those elements of \( \mathbf{C} \) which are not in the equations to be solved on the analog computer and then equating to zero those elements in these equations which are not multiplied by appropriate variables digitally or are not multiplied by a digital variable in the analog computer. Finally, the nonzero elements which are multiplied digitally are multiplied by \( b \) to give \( \Theta' \).

In standard texts on numerical methods it is shown that row \( i \) and row \( j \) and column \( i \) and column \( j \) of a matrix \( \mathbf{A} \) can be interchanged by the operation

\[
\Psi_{ij} \mathbf{A} \Psi_{ji}^{-1}
\]

where \( \Psi_{ij} \) and \( \Psi_{ji} \) are elementary matrices with the properties

\[
\Psi_{ij} = \Psi_{ji} = \Psi_{ji} \Psi_{ij} = \Psi_{ji}^{-1} = \Psi_{ij}^{-1}.
\]

It then follows that a matrix \( \mathbf{C}^* \), with the equations to be solved on the analog computer in its first \( m \) rows, can be obtained from \( \mathbf{C} \) by a

---

sequence of row and column interchanges so that

\[ C^* = (\psi_{ke} \cdots \psi_{ij}) C (T_{ij} \cdots T_{ke}) \]

Corresponding to \( C^* \) there is a matrix \( \Theta^* \) given by

\[ \Theta^* = [\psi_{ke} \cdots \psi_{ij}] \Theta \begin{bmatrix} T_{ij} & \cdots & T_{ke} \end{bmatrix} \]

Let the eigenvalues and eigenvectors corresponding to \( C^* \) be \( \lambda_i^* \), \( w_i^* \) and \( v_i^* \). Using the properties of \( \psi_{ij} \) and \( T_{ij} \) stated above it is not difficult to show the following

\[ \lambda_i^* = \lambda_i; \ i = 1, 2, \ldots, n \]

\[ w_i^* = [\psi_{ke} \cdots \psi_{ij}] w_i; \ i = 1, 2, \ldots, n \]

\[ v_i^T = v_i^T[T_{ij} \cdots T_{ke}]; \ i = 1, 2, \ldots, n \]

Now since \( C^* \) is in the required form, (25) applies and thus

\[ \Delta \lambda_i^* = \Delta \lambda_i \approx -\delta/2 \ v_i^T \Theta^* \ w_i^* \]

Using the expressions above for \( v_i^T \), \( \Theta^* \) and \( w_i^* \) the last equation becomes

\[ \Delta \lambda_i \approx -\delta/2 \ v_i^T[T_{ij} \cdots T_{ke}][\psi_{ke} \cdots \psi_{ij}] \Theta \begin{bmatrix} T_{ij} & \cdots & T_{ke} \end{bmatrix}[\psi_{ke} \cdots \psi_{ij}] w_i \]

which reduces to

22
\[ \Delta \lambda_i \approx -\gamma/2 \nu_i^{T} \Theta \nu_i \]

since

\[ T_{ij} = \psi_{ij}^{-1} \]

for any \( i, j \).
TECHNICAL NOTE NO. 19

Research Project A-588

HYBRID COMPUTER SOFTWARE FOR SOLUTION ERROR REDUCTION

By Cecil O. Alford
Joseph L. Hammond, Jr.

Prepared for
George C. Marshall Space Flight Center
Huntsville, Alabama

Contract No. NAS8-2473

(Development of New Methods and Applications of Analog Computation)

January 15, 1970

School of Electrical Engineering
GEORGIA INSTITUTE OF TECHNOLOGY
Atlanta, Georgia
HYBRID COMPUTER SOFTWARE FOR
SOLUTION ERROR REDUCTION

by
Cecil O. Alford, and Joseph L. Hammond, Jr.

TECHNICAL NOTE NO. 19

on
Contract No. NAS8-2473

(Development of New Methods and
Applications of Analog Computation)

For
GEORGE C. MARSHALL SPACE FLIGHT CENTER
Huntsville, Alabama
TABLE OF CONTENTS

ABSTRACT ............................................ ii
GLOSSARY OF PRINCIPAL SYMBOLS .................... iii
INTRODUCTION AND STATEMENT OF PROBLEM .......... 1
CHOICE OF NUMERICAL INTEGRATION METHODS ............ 4
COMPENSATION FOR ZERO ORDER HOLD ................... 24
COMPENSATION FOR EXECUTION TIME DELAY ............... 37
CONCLUSIONS ........................................ 55
REFERENCES .......................................... 57
ABSTRACT

This technical note is an outgrowth of a continuing study on the sampling errors in hybrid computer programs. In particular, Technical Note No. 17 documents the error equations for a general class of hybrid programming problems, and forms the basis for this technical note. This note presents hybrid software techniques to eliminate or decrease the various error terms derived in Technical Note No. 17.

After a statement of the problem in Chapter I, attention is given to the choice of a numerical integration method when dynamic equations are programmed on the digital part of the hybrid computer. Chapter II analyzes and compares several numerical integration methods with respect to timing considerations, errors, and execution time. The chapter concludes with considerations for the optimum choice of a particular method.

Attention is next focused on errors due to sampling and the use of zero order hold devices in hybrid computers. Several schemes for the compensation of these errors are presented in Chapter III.

Another major source of error is caused by function evaluation, or execution time delay, in the digital computer. Chapter IV presents compensation methods for hybrid computers with zero-order hold devices and with the compensated hold devices presented in Chapter III.

Concluding remarks are given in Chapter V.
GLOSSARY OF PRINCIPAL SYMBOLS

$u(t)$ - ideal vector solution of problem equation

$x(t)$ - ideal vector solution of equation programmed on the analog computer

$z(t)$ - ideal vector solution of equation solved digitally

$f$ - vector of functions equated to $x(t)$

$f_a$ - vector of functions generated and used in the analog computer

$f_d$ - vector of functions generated digitally for use in the analog computer

$g$ - vector of functions equated to $z(t)$

$t$ - time

$p$ - order of numerical method

$\phi$ - principal error function of numerical method

$\delta$ - sampling period

$e$ - execution time of digital computer

$\Gamma(t)$ - total hybrid computer sampling error vector

$\frac{\partial h}{\partial u}$ - a matrix defined in the Appendix of Technical Note No. 17

$h$ - vector of functions equated to $u(t)$ or step size of numerical method

$\tilde{F}_1$ - forcing function proportional to first power of sampling period

$\tilde{F}_2$ - forcing function caused by execution time

$\tilde{F}_3$ - forcing function caused by truncation errors of the numerical method used to integrate the $z$ equation

$F_0(t)$ - output of zero order hold in D/A unit

$F_1(t)$ - output of first order hold in D/A unit

$\hat{F}_1(t)$ - output of a first order hold approximated with a multirate sampler

$f(t)$ - input signal to hold devices in D/A unit

$\delta^*$ - sampling period of multirate sampling unit in zero order hold
\( F_d(t) \) - desired output of first order hold

\( F_d(t-e) \) - delayed output function of first order hold

\( F'_d \) - extended output function of first order hold

\( r(t) \) - input vector

\( \hat{f}_d \) - estimate of \( f_d \) at \( t = t_n \)

\( E \) - error incurred in approximating \( f(t) \) by the Lagrangian interpolation method of the digital equipment

\( \hat{z}(t_n) \) - first estimate of \( z(t_n) \) using multistep algorithm

\( z_n \) - final estimate of \( z(t_n) \) using multistep algorithm

\( \hat{z}(\tau) \) - Runge-Kutta estimate of \( z(\tau) \)

\( e_{nm} \) - execution time of numerical method for one interval, \( h \)

\( M_1 \) - ratio \( \delta/h \)

\( M_2 \) - ratio \( h/\Delta \)

\( O(\delta) \) - order of solution error due to sampling rate

\( O(h^p) \) - order of solution error due to truncation error

\( O(e) \) - order of solution error due to execution time

\( O(e_{nm}) \) - order of solution error due to execution time of the numerical method

\( \Delta \) - A/D sample period, D/A conversion period or "frame time"
CHAPTER I

INTRODUCTION AND STATEMENT OF PROBLEM

In earlier work on the subject contract reported in [1], [2], [3] and [4], a general equation for the sampling error in a closed loop hybrid program has been derived. A study of this equation indicates sources of sampling error and suggests methods which can be used to reduce sampling error. In this note the more promising of these methods for reducing error will be discussed in terms of their potential for being included in improved hybrid software.

The sampling error equation has been developed in [1] on the assumption that the hybrid computer is programmed to solve the problem equation

\[ \dot{u}(t) = h(u(t);t) \]  

where \( u(t) \) is a vector of state variables and \( h \) is a vector function. Equation (1) is programmed with the computing operations allocated between the digital and analog computers so that the ideal machine equation is given by

\[
\begin{bmatrix}
\dot{x}(t) \\
\dot{z}(t)
\end{bmatrix} = 
\begin{bmatrix}
f_a(x(t),z(t);t)+f_d(x(t),z(t);t) \\
g(x(t),z(t);t)
\end{bmatrix} = h(u(t);t) \quad (2)
\]

where the \( x \) variables are obtained by integration on the analog computer, the \( z \) variables are obtained by integration on the digital computer, the \( f_a \) are functions generated on the analog computer and the \( f_d \) are functions generated by the digital computer. The general error equation, as given by equation (25) of [1], is expressed as

\[ \text{error} = f_a(x(t),z(t);t)+f_d(x(t),z(t);t) - g(x(t),z(t);t) \varepsilon \]
P(t) = \hat{P}(t) + \frac{\hat{e}}{2} \phi(\frac{\hat{f}_d}{2} + \frac{\hat{e} + \hat{e}^2}{2} \frac{\hat{f}_d}{\phi})

\hat{P}(t=0) = 0

where \( \Gamma \) is a vector of errors defined as the difference between the true and hybrid state variables, \( \hat{e} \) is the sampling period of the A/D and D/A converters, \( e \) is the execution time of the digital equipment in generating \( f_d \) and \( \phi \) is the principal error function of the numerical method used to integrate the equation for \( z \). In (3) it is assumed that the sampling period of the A/D converters and the step size of the numerical method are equal. If the step size \( h \) of the numerical method is not equal to \( \hat{e} \), the term \(-\hat{e}^2 \phi\) in the forcing function should be replaced by \(-h^2 \phi\).

Since the error equation has zero initial conditions, any non-zero solution of the error equation must be caused by the forcing functions and since (3) is also linear, the forcing functions

\[ \tilde{F}_1 = \begin{bmatrix} \frac{\hat{e}}{2} \left( \frac{\partial f}{\partial z} + f_d \right) \\ 0 \end{bmatrix}, \quad \tilde{F}_2 = \begin{bmatrix} e f_d \\ 0 \end{bmatrix}, \quad \tilde{F}_3 = \begin{bmatrix} 0 \\ -\hat{e}^2 \phi \end{bmatrix} \]  

(4)

can be considered separately.

The function \( \tilde{F}_1 \) is caused by the fact that \( z \) and \( f_d \) are computed digitally and coupled to the analog computer through zero order hold D/A converters. The magnitude of \( \tilde{F}_1 \) is proportional to the first power of the sampling period and produces an error which is \( O(\hat{e}) \). It is this term, with its dependence on \( \hat{e} \), that makes the hybrid computer a first
order numerical method. In Chapter III of this note techniques for increasing the order of the hybrid computer method will be given.

The function $\tilde{F}_2$ is caused by the execution time of the digital equipment in evaluating $f_d$. It produces an error which is $O(e)$. Note from the expressions for $\tilde{F}_1$ and $\tilde{F}_2$ that execution time, (which satisfies $0 \leq e \leq \delta$ for feasible operation), is weighted twice as heavily as $\delta$ in determining sampling error. In Chapter IV methods for reducing the effect of execution time will be discussed.

The function $\tilde{F}_3$ is caused by truncation errors of the numerical method used to integrate the $z$ equation. It produces an error which is $O(\delta^p)$, (or $O(h^p)$ if $h$ and $\delta$ are unequal), showing that for $p \geq 2$ errors from this source are negligible compared to the $O(\delta)$ terms caused by $\tilde{F}_1$ and $\tilde{F}_2$. Chapter II of this note will examine consequences of this error term and suggest appropriate numerical methods for hybrid computation.
CHAPTER II

CHOICE OF NUMERICAL INTEGRATION METHODS

2-1. General Characteristics of Numerical Integration Methods

The literature discussing numerical integration methods for solving ordinary differential equations using all digital calculations and comparing these methods is extensive. Recent publications which are representative of a much larger volume of material are the papers by Benyon [5], Giloi [6], and Mortens [7] and the report by Lomax [8]. The paper by Benyon contains an extensive bibliography to other references (75 entries). A number of textbooks covering numerical integration techniques are also available. The book of Henrici [9] is possibly the most complete study devoted exclusively to numerical techniques for solving ordinary differential equations. Beckett [10] and Conte [11] are representative of elementary treatments of the subject, while Ralston [12] and Hamming [13] are two additional books on the level of Henrici. The book of Beckey [14] contains some material pertaining specifically to numerical methods for hybrid computers but no other current literature on this topic was found.

This section will review certain general characteristics of various numerical integration methods which are important for the use of such methods in hybrid computers. A basic knowledge of numerical integration methods such as that given in Beckett [10] will be assumed.

Several terms necessary in discussing numerical integration methods are now defined for reference (1):

(1) An extended discussion of these parameters is given in such texts as Henrici [9].
**step size** - A digital numerical method produces solutions to a differential equation at discrete values, \( t_n \), of \( t \). Step size is the difference between successive values of \( t \) at which solutions are produced, i.e. \( t_{n+1} - t_n \). In this work \( t_{n+1} - t_n \) is assumed to be a constant denoted \( h \).

**discretization or sampling error** - error caused by evaluating the dependent variables at isolated discrete times.

**truncation error** - discretization error per step of the numerical method.

**accumulated discretization error** - discretization error after several steps of the numerical method.

**round off error** - error caused by using a finite word length in representing the dependent variables (not considered in this discussion).

**order** - a method of order \( p \) has a truncation error which is \( O(h^{p+1}) \).

(i.e. truncation error for a \( p \)th order method can be expressed as \( Ch^{p+1} \) where \( C \) is a finite constant.)

**principle error function**\(^{(2)}\) - in certain cases, truncation error can be expressed or approximated as \( h^p \phi + O(h^{p+1}) \), where \( \phi \) is the principle error function.

General methods for the numerical solution of ordinary differential equations with prescribed initial conditions are almost all based on some type of "marching" algorithm. That is to say, an algorithm which begins with the initial values of the dependent variables at \( t_o \) and computes

\(^{(2)}\)The principle error function depends on both the independent and dependent variables. In many cases of importance \( \phi \) can be expressed or approximated as a derivative of the true solution to the problem equation. For high order algorithms exact formulas for calculating \( \phi \) are not tractable but numerical approximations can be obtained. For first and second order methods explicit expressions for \( \phi \) can often be obtained.
values for the dependent variables at $t_1 = t_0 + h$. The new values of the dependent variables at $t_1$ are then used to determine values at $t_2 = t_0 + 2h$ and the process is continued indefinitely one step at a time.

Two general classes of algorithms for solving ordinary differential equations are in common use, namely the multistep methods and the one-step or Runge-Kutta methods. The characteristics and distinct properties of these two classes of algorithms can be illustrated by diagrams displaying the required sequence of steps for specific, but representative, algorithms from the two classes.

Consider the multistep algorithm described by Figure 1 as a means for solving $z(t) = g(z(t), t)$. After the completion of the $n$th cycle approximations to $z(t_{n-1})$, $z(t_n)$, $g(z(t_{n-1}), t_{n-1})$ and $g(z(t_n), t_n)$, denoted respectively as $z_{n-1}$, $z_n$, $g(z_{n-1}, t_{n-1})$ and $g(z_n, t_n)$, are stored in memory.

The first step in cycle $(n+1)$ is to estimate $z(t_{n+1})$ by $\hat{z}(t_{n+1})$ and then compute $g(\hat{z}(t_{n+1}), t_{n+1})$. In the second step an improved and final estimate, $z_{n+1}$, of $z(t_{n+1})$ is obtained and $g(z_{n+1}, t_{n+1})$ is calculated. The final estimates obtained at $t_n$ and $t_{n+1}$ are then stored for use in $(n+2)$. Note that for the multistep algorithm calculations are made only for time values corresponding to the discrete times $t_{n-1}$, $t_n$, $t_{n+1}$, etc., at which values of $z$ are available as outputs of the method. Note also that past values of $z_i$ and $g(z_i, t_i)$ must be stored for later use and that special considerations are required in the first cycle for which only one set of initial values is available, whereas values at two different times are required.

Of further interest with respect to the multistep algorithm is the fact that a crude estimate of $z(t_{n+1})$ is obtained in the first step from

---

(3) The diagrams to be used seem to have been originated by Lomax [8].
Figure 1: Solution of $\dot{z}(t) = g(z(t), t)$ using a Multistep Algorithm
data at times \( t_{n-1} \) and \( t_n \) both less than \( t_{n+1} \). This data could be output as the final estimate of \( z(t_{n+1}) \). Such a method (using only the first step in Figure 2) is also a multistep method, called an open multistep method. The complete method described in Figure 2 is termed a predictor-corrector method, the first step being referred to as the predictor step and the second as the corrector step. For predictor-corrector methods, the last step requires calculation of the final estimate of \( z(t_{n+1}) \) from data which includes \( g[z(t_{n+1}), t_{n+1}] \). This point is important with respect to use of such methods in hybrid computers.

Figure 2 shows the sequence of steps for two Runge-Kutta algorithms. For such algorithms only \( z_n \) and \( g[z_n, t_n] \) are stored from the \( n \)th cycle. The first step in cycle \((n+1)\) is to obtain \( \hat{z}(\tau) \) as an estimate of \( z(\tau) \) at some time, \( \tau \), in the interval \([t_n, t_{n+1}]\) and use \( z(\tau) \) to compute \( g \). Method (a) uses \( \tau = t_n + h/2 \) and method (B) uses \( \tau = t_{n+1} \). In general \( \tau \) can be any point in \([t_n, t_{n+1}]\). The second step then uses \( z_n, g[z_n, t_n] \) and \( g[\hat{z}(\tau), \tau] \) and to compute final estimates of \( z \) and \( g \) at \( t_{n+1} \). The values of \( z_{n+1} \) and \( g[z_{n+1}, t_n] \) are retained for the next cycle.

Note that for Runge-Kutta algorithms calculations of \( z(\tau) \) are made at values of \( \tau \) in the interval \([t_n, t_{n+1}]\) for which data is not output from the method. (Higher order Runge-Kutta formulas may compute \( z \) at several values of \( \tau \) in the interval \([t_n, t_{n+1}]\).) Note also that data at only one past point is stored for later use and that because of this no special considerations are required in the first cycle. Runge-Kutta algorithms may or may not require evaluation of \( z_{n+1} \) from data which includes \( g[z_{n+1}, t_{n+1}] \), as illustrated by the two variations shown in Figure 2.

Execution time of the calculations involved in advancing either a multistep or a Runge-Kutta method through one step is determined approximately
Figure 2: Solution of $\dot{z}(t) = g(z(t), t)$ using two Runge-Kutta Algorithms
by the number of evaluations of the function $g$ required in each cycle. In both Figures 1 and 2, one evaluation of $g$ is required for each step and a total of two evaluations is required for each cycle.

In general a Runge-Kutta method of order $p$ requires $p$ evaluations of $g$ in each cycle. A multistep method requires one evaluation of $g$ for each predictor step and one evaluation of $g$ for each corrector step, independently of the order of the method. Thus, a method with one predictor and three corrector steps per cycle would require four evaluations of $g$ per cycle.

A multistep method of order $p$ requires on the order of $p$ past values of $z$ and $g$. Generally, predictor-corrector and Runge-Kutta algorithms of the same order have comparable truncation error. Open multistep algorithms, on the other hand, typically have larger truncation error than either predictor-corrector or Runge-Kutta algorithms of the same order.

Accumulated discretization error, i.e. the affect of truncation errors after several steps of the algorithm, is given for Runge-Kutta methods by a differential equation whose forcing function is equal to $h$ times the principle error function. Multistep methods have an accumulated discretization error component determined by the principle error function in the same manner as for the Runge-Kutta algorithms, but they have an additional accumulated error due to inaccurate starting values.

Before comparing numerical methods with respect to their use in hybrid computers, it is necessary to discuss special timing problems for hybrid computers which arise because of the coupling of a digital computer to an analog computer.
2-2. **Timing of Hybrid Computations**

The typical structure of a hybrid computer from the point of view of numerical integration is shown in Figure 3. The equation integrated numerically is expressed as

\[ \dot{z}(t) = g(x(k\Delta), z(t); t) \]  \tag{5}

to show explicitly the use of data from the analog subsystem. The following symbols are employed in the figure:

- \( \Delta \) - A/D sampling period
- \( h \) - step size of numerical integration method
- \( \delta \) - D/A conversion period or "frame" time
- \( f_d \) - vector of digitally generated functions

For any type of numerical integration method, a restraint on the sequencing of A/D conversion is imposed by the fact that new values of \( x(k\Delta) \) can be obtained only from samples of \( x(t) \). Thus, for accurate operation, \( x \) must be sampled at the times required by a particular algorithm. For example, to advance a multistep algorithm from \( t_n \) to \( t_{n+1} \), \( g(x(t_n), z_n, t_n) \) must be evaluated and thus the value of \( x(t_n) \) is required. Since this is the case, the proper sequencing of the three major operations, A/D conversion, D/A conversion and output of the numerical integration is a matter of importance. The number of possible timing sequences is greatly reduced by the following practical assumptions as to the mode of operation:

1. A new evaluation of \( g \) is not undertaken until new data on \( x \) is obtained. This fixes \( \Delta \leq h \).

2. A new value of \( z \) should be available in time for each D/A conversion, i.e. at the end of each frame. This fixes \( h \leq \delta \).
Figure 3. Typical Structure of a Hybrid Computer
Within the limits imposed by the above assumptions, the numerical method can advance through any number of steps in one frame by making $\delta$ an appropriate multiple $M_1$ of $h$. Once $h$ is fixed, A/D conversions must take place in the proper sequence to provide $z$ at the times required for use in the numerical integration algorithm. This fixes

$$\Delta = h : \text{multistep algorithms}$$

$$h = M_2\Delta : \text{Runge-Kutta algorithms}^{(4)}$$

where $M_2$ is a constant determined by the particular algorithm. A diagram showing the sequencing of operations is given in Figure 2.

In pure digital work the steps in the evaluation of a differential equation are not related to real times. In hybrid computation, however, the necessity for using samples of $x(t)$ from the analog subsystem indexes the digital calculation to real time as indicated by the sequence of time values at the top of Figure 4. Thus, all digital operations must be accomplished in a time compatible with the sequence shown. If it is assumed that D/A and A/D conversions require a negligible amount of time, then two execution times are of importance; namely, the time, $e_{\text{nm}}$, to advance the numerical method through an interval $h$ and the execution time, $e$, per frame of all other digital calculations excluding numerical integration.

From the definition of these quantities it is clear that $e_{\text{nm}}$ and $e$ must satisfy

$$e_{\text{nm}} \leq h$$

To require that $x$ be evaluated at times $k\Delta$ which are submultiples of $h$ is a restriction on the type of Runge-Kutta algorithms used. However, this category would seem to include most algorithms used in practice for hybrid computation.
Figure 4. Sequencing of Hybrid Computer Operations
and

\[ e \leq \delta - M_1 e_{nm} \]

for the required type of operation.

Since the numerical integration algorithm requires real time data, the best time to advance the algorithm is at the end of each frame as shown in Figure 4. In general data is not used immediately after A/D conversion or calculation and thus storage is required. The general timing restrictions which must be satisfied in advancing the integration algorithm are indicated in Figure 4 by showing the time zones within which the various calculations must be made.

In later discussion, it will be convenient to identify two cases, namely:

Case 1 - \( M_1 e_{nm} \) is negligible compared to \( e \), so that \( \delta \approx e \)

Case 2 - \( e \) is negligible compared to \( M_1 e_{nm} \), so that \( \delta \approx M_1 e_{nm} \)

2-3. Errors in Hybrid Computation

The general sampling error equation given as (3) above characterizes the major source of error for the complete hybrid computer. More specifically (3) gives an approximate expression for the accumulated "sampling" error of the complete hybrid computer system for the conditions that:

(a) \( h \) and \( \delta \) are small with respect to the appropriate time constants of the problem equations and (b) the complete hybrid computer system is stable. Equation (3), in effect, is a measure of accuracy for small \( h \) and \( \delta \), assuming stability.

Stability of a numerical method in the digital subsystem and of the complete hybrid computer system is a function of the relation of \( \delta \) and \( h \) to the time constants of the problem equations. The choice of numerical
method and the allocation of operations between the analog and digital computers also affects stability. To make the present discussion tractable, it will be assumed that \( h \) is chosen so that the numerical integration method in the digital subsystem is stable and that \( \delta \) is chosen so that the overall hybrid computer system, for the chosen allocation of operations, is also stable.

Reference to the general error equation, (3), shows that use of numerical integration produces two of the forcing functions in the error equation namely \( \frac{\delta}{2} \frac{\partial f}{\partial z} g \) and \(-h^P \phi\). The former produces an error in the analog calculations because of the discrete nature of \( z \). This will be referred to as "analog sampling error." The term \(-h^P \phi\) produces an error due to truncation in the numerical method. This will be referred to as "numerical truncation error." It is desirable to design a system so that the net error caused by these two different effects of the numerical integration method is a minimum.

For Case 1, \( \delta \), and hence the analog sampling error is determined by \( e \) and not by the numerical method. Thus, for this case the choice of numerical method determines only the truncation error and the best that can be accomplished is to minimize this error by choosing a numerical method for which \( h^P \phi \) produces a negligible response of the error equation. Consistent with this requirement, to minimize second order effects the numerical method should also have small execution time.

For Case 2, the execution time of the numerical method determines \( \delta \) and hence the analog sampling error which is \( O(\delta) \). Thus, both analog sampling error and numerical truncation error are determined by choice.
of the numerical method and the net effect of these two errors should be minimized.

Note that typical operation may not fall exactly in either Case 1 or Case 2. However, these cases do cover the two extremes.

In addition to the effect on sampling error and stability, there are, of course, other desirable properties of the numerical method such as small storage requirements and ease of implementation. Such characteristics are assumed to be of secondary importance in this work.

2-4. Choice of Numerical Integration Methods for Hybrid Computation

(a) Considerations Involving Timing:

As noted in Section 2, a hybrid computer has restraints caused by the timing required to make the analog variables available to the digital computer. Timing requirements for each of the general types of methods will thus be discussed for hybrid computer applications.

In the discussion of a typical predictor-corrector method in Section 1, it was noted that the corrector step requires evaluation of a function which depends on \( g(\hat{z}(t_{n+1}^{});t_{n+1}^{}) \). In a hybrid computer \( g \) also depends in general on \( x \) so that a corrector step in a hybrid computer would require use of \( g(x(t_{n+1}^{}),z(t_{n+1}^{});t_{n+1}^{}) \). But \( x(t_{n+1}^{}) \) can only be obtained by sampling \( x(t) \) at \( t = t_{n+1}^{} \) and therefore, \( x(t_{n+1}^{}) \) cannot be available before \( t_{n+1}^{} \). Thus, first corrector step for predictor-corrector methods cannot be initiated before \( t = t_{n+1}^{} \) and because of execution time \( z_{n+1}^{} \) cannot be available for output until sometime after \( t_{n+1}^{} \).

Thus, if predictor-corrector methods are used, the timing of the D/A conversions must be such that \( z_{n+1}^{} \) is provided late to the analog computer. This added execution time would cause errors in addition to those accounted
for by the error equation (3). Furthermore, unless the digitally generated functions are converted D/A separately from the results of the z calculations, these functions would suffer an increased effective execution time also.

There seems to be no advantage of the predictor-corrector method over the Runge-Kutta method to offset the execution time it adds to the computed z variables because of timing difficulties. It would, therefore, not seem reasonable to use such algorithms in hybrid computation.

With respect to timing for Runge-Kutta methods, it should be noted that in advancing from \( t_n \) to \( t_{n+1} \) such methods require evaluation of \( g \) for values of time in the interval \([t_n, t_{n+1}]\). Thus, for such methods the A/D converters must run faster than the D/A converters, but at the present state-of-the-art this would seem to pose no problem.

Considerations analogous to those just discussed for the predictor-corrector methods would seem to rule out Runge-Kutta formulas which require use of \( g(x(t_{n+1}), z(t_{n+1}); t_{n+1}) \) in the last step. However, formulas which allow an appropriate execution time for the last step would seem to offer no other timing difficulties.

No timing difficulties are presented by the open multistep algorithms although they are, of course, not self-starting.

(b) **Choice of Methods for Case 1 (\( \delta \geq e \))**:

If the frame time, \( \delta \), is determined primarily by the execution time, \( e \), of operations other than numerical integration, then analog sampling error is fixed and the choice of the numerical method is dictated by numerical truncation error and stability of the digital calculation.

Numerical truncation error has negligible effect if the term \( h^p \delta \) produces a response of the error equation which is negligible compared to the
terms which are $O(\delta)$. If $h$ is chosen to be equal to $\delta$, a second or third order method would make the numerical truncation error small compared to other sources of error. The same effect could be accomplished, but possibly not as well, by using a low order method and a small value of $h$ as compared to $\delta$. This would require several steps of the numerical method per frame time.

The timing considerations discussed above have narrowed the field of possible numerical methods to either: (a) Certain Runge-Kutta methods or (b) open multistep methods. For either type of method, truncation error can be made negligible by choice of appropriate order or by choosing $h$ small compared to $\delta$. Furthermore, appropriate selection of $h$ can make either type of method stable.

It would thus seem that first order effects do not lead to a choice between the designated Runge-Kutta methods and open multistep methods for Case 1 operation. Thus, with respect to the first order effects, any particular member of these two classes would seem to be satisfactory.

If second order effects are also considered, they would favor a method with the smallest execution time, consistent with a given accuracy and stability, and one with the least difficulty with such implementation considerations as starting, providing for A/D sampling at a rate faster than D/A conversion and providing for the required storage. The most important implementation trade-off would seem to be between the multirate A/D sampling required by the Runge-Kutta methods and the starting procedure required by the open multistep algorithms.

With respect to execution time, empirical data obtained by Benyon [5] would seem to apply for Case 1 operation. Benyon compares a particular Runge-Kutta method, an open multistep formula (Adams-Bashforth) and a
predictor-corrector method (Adams-Moulton) under the following conditions: (a) the solution to a specific nontrivial problem is the basis for comparison; (b) for a particular method, step size is varied to find the largest step size for acceptable accuracy and stability; and (c) the total solution time is recorded. The portion of his results which are of interest here are summarized in Table 1 which tabulates computing time as a fraction of real time.

Under the stated conditions for fourth order methods, the Runge-Kutta method is approximately twice as fast as the Adams-Bashforth method. As a function of order, second order methods are the fastest and second order Runge-Kutta and Adams-Bashforth methods are comparable.

<table>
<thead>
<tr>
<th>Order</th>
<th>1st</th>
<th>2nd</th>
<th>3rd</th>
<th>4th</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runge-Kutta</td>
<td>60</td>
<td>3.3</td>
<td>3.9</td>
<td>5.2</td>
</tr>
<tr>
<td>Adams-Bashforth</td>
<td>60</td>
<td>3.8</td>
<td>6.7</td>
<td>13.3</td>
</tr>
</tbody>
</table>

TABLE 1

Computing Time as a Fraction of Real Time for Runge-Kutta and Adams-Bashforth Methods of Several Orders

The table also shows that use of a first order method, (and small step size) is very much inferior to other methods from the point of view of execution time.

These results are reasonably general since, as will be discussed in the next section, the Adams-Bashforth is, in a sense, an optimum open multistep formula, the results given by Benyon would be expected to be typical, at least to an order of magnitude.
Benyon's data leads to the following conclusions with respect to minimizing execution time. For Case 1, hybrid computer operation: (a) a second order method should be chosen; (b) the step size, h, should be adjusted for negligible numerical truncation error and adequate stability (h may then be equal to or less than \( \delta \)); and, (c) from the point of view of execution time the Runge-Kutta and Adams-Bashforth methods are equivalent.

Since choice of a second order method is consistent with the conclusion for first order effects, a final choice considering all factors would seem to be between a second order Runge-Kutta or a second order multistep method of which class an Adams-Bashforth is typical and, in some senses, optimum. The choice between these two methods can be made on the basis of implementation considerations which are dictated by the characteristics of a particular machine.

(c) Choice of Methods for Case 2 (\( \delta \approx M_{1e_{nm}} \)):

If the frame time, \( \delta \), is determined primarily by the execution time, \( M_{1e_{nm}} \), of the numerical method, then the choice of numerical method determines both the numerical truncation error and the analog sampling error. For this type of operation the numerical method should be chosen to minimize the combined effects of both these types of error.

The following comments summarize the restraints for Case 2 operation. Analog sampling error is \( O(\delta) \) whereas numerical truncation error is \( O(h^p) \). The parameter \( p \) satisfies, \( p \geq 1 \), and is a property of the numerical method. For Case 2 operation, \( \delta \approx M_{1e_{nm}} \) and in general \( h \leq \delta \).

Any numerical method will have \( p \) at least equal to 1. Thus, from the point of view of accuracy, errors which are \( O(\delta) \) and \( O(h^p) \) must be
minimized when \( h \leq \delta \leq M_1 e_{nm} \). Since \( \delta \) and \( h \) are independent in the sense that they are determined by different factors, \( \delta \) should clearly be made as small as possible relative to \( h \). This results in choosing \( \delta = h \) for which choice \( M_1 = 1 \) and \( \delta \leq e_{nm} \). Considering first order effects, the problem then becomes that of minimizing errors which are \( O(e_{nm}) \). This is, of course, accomplished by choosing a numerical integration method with the smallest possible execution time. Because of the presence of numerical truncation errors which are \( O(h^p) \), a further reduction in error can be made if \( p \) can be made two or greater without substantial increase in \( e_{nm} \).

Consider the two classes of numerical methods, which meet the timing requirements of Section 2, from the point of view of execution time per step of the method. For Runge-Kutta methods the execution time is approximately the time to evaluate \( g \), times the order of the method. On the other hand for open multistep formulas the execution time is essentially the time for one evaluation of \( g \) independent of the order.

On this basis an open multistep formula would be chosen to minimize execution time. Furthermore, since execution time per step is independent of order for this type of algorithm, an algorithm of order two or higher should be chosen in order to make numerical truncation error negligible compared to analog sampling error.

For multistep methods the number of past points used in the algorithm is generally closely related to the order of the method. Thus, from the point of view of the second order effects of storage and number of starting values, the order of the multistep method should be no higher than required by the first order effects. This consideration leads to the choice of a second order open multistep formula.

Henrici [9] shows that for a fixed number of past points no multistep method can have an order greater than the Adams-Bashforth. The Nyström
method, among others, also has this property but the Nyström method is less stable than the Adama-Bashforth. Thus, the Adams-Bashforth second order method would seem to be an excellent choice for Case 2 operation.

It must be assumed for Case 2 operation that the hybrid system and the numerical method operating with \( h = e_{nm} \) are stable. If this is not the case, another allocation of operations or a reduction in time scale must be used. It is unlikely that a second order numerical method can be found which is more stable than the Adams-Bashforth.

(d) Particular Algorithms for Use in Hybrid Computation:

Illustrative examples of the two types of algorithms arrived at above for use in hybrid computation are given below. The algorithms given are not unique but since the chosen methods are second order, the variations are limited. Both algorithms are expressed on the assumption that the vector equation \( z = g(x,z;t) \) is given for solution.

Runge-Kutta Second Order:

\[
z_{n+1} = z_n + h g(x(t_n + \frac{h}{2}), z_n + \frac{h}{2} g[x(t_n), z_n ; t_n]; t_n + \frac{h}{2})
\]  

Adams-Bashforth Second Order(5):

\[
z_{n+1} = z_n + h[\frac{3}{2} g_n - \frac{1}{2} g_{n-1}]
\]  

where \( g_n = g(x(t_n), z_n ; t_n) \).

(5) The following sequence of steps could be used for starting a multistep method.

(a) Scale time down by say a factor of ten for the whole system.
(b) Reduce the multistep method to Euler's method by setting the appropriate coefficients to zero and one.
(c) Run with the reduced time scale to \( n' = 10 \) which corresponds to \( t_1 = h \).
(d) Record the values of \( z_1 \) and \( g_1 \) in storage. These are all the values required for starting a second order method.
(e) Return to the original time scale and set the coefficients to the proper value for the Adams-Bashforth procedure.
CHAPTER III

COMPENSATION FOR ZERO ORDER HOLD

1. Introduction

The hybrid computer error equation, given by equation (3) and repeated below

\[
\dot{\Gamma}(t) = \frac{\partial h[u(t);t]}{\partial u(t)} \Gamma(t) + \left[ \frac{\delta}{2} \frac{\partial f}{\partial z} g + \frac{\delta + 2e}{2} \cdot \right] - \delta \rho \Phi
\]

clearly indicates that the error depends on \( \delta \). This seems to be a basic property of the zero order hold D/A converters. For a fixed value of \( \delta \), in order to reduce the error significantly it is necessary to make the error independent of \( \delta \) to the first power and dependent on higher powers of \( \delta \). This can be accomplished by replacing the zero order hold devices with first order hold devices. For first order hold devices the error equation becomes (6)

\[
\dot{\Gamma}(t) = \frac{\partial h[u(t);t]}{\partial u(t)} \Gamma(t) + \left[ \frac{5}{12} \delta^2 \frac{\partial f}{\partial z} g + \frac{\delta^2}{3} - \frac{\delta e}{6} - \frac{e^2}{2} \cdot \right] - \delta \rho \Phi
\]

and since \( e \leq \delta \), (8) shows the hybrid computer with a first order hold is equivalent to a second order numerical method provided \( p \geq 2 \). An increase in order for the hybrid computer reduces truncation error and in most cases provides a more efficient solution to a given problem for the same accuracy. It is thus important to be able to implement first order hold devices.

(6) The derivation of this result will be presented in a later Technical Note.
2. First Order Implementation Methods

The D/A units in conventional hybrid computers contain a zero order hold as the output device. The output from such a device is described by the equation

\[ F_0(t) = f(n\delta) ; \quad n\delta \leq t < (n+1)\delta \]  

where \( \delta \) is the sampling period, \( f(t) \) is the input signal and \( f(n\delta) \) is \( f(t) \) at \( t = n\delta \). This function \( F_0(t) \) is the familiar stair-step function with discontinuities at the sampling times \( n\delta \), \( n = 1, 2, \ldots \).

A first order hold is designed to implement the equation

\[ F_1(t) = f(n\delta) + \frac{f(n\delta) - f((n-1)\delta)}{\delta} (t - n\delta) , \quad n\delta \leq t < (n+1)\delta . \]  

Equation (10) will be implemented in three distinct ways, progressing from an all analog method through combined analog-digital methods to an all digital method.

(a) METHOD I - Analog Hardware Implementation

One of the most direct methods to achieve the desired result is to implement (10) using a zero order hold and basic analog computing hardware as shown in Figure 5 for \( f(t) \) an analog signal. In case the input signal, \( f(t) \) originates in the digital computer, the sampler is not present and the number \( f(n\delta) \) represents the output of the D/A hold unit. The remainder of the hardware for implementing the first-order hold is unchanged however.

(b) METHOD II - Digital-Analog Hardware Implementation

Since the signals being sampled in Method I are those transferring information between the analog and digital computers, it is possible to
Figure 5. Method 1: Analog First Order Hold
modify this method by taking advantage of the storage feature inherent in the digital computer. This modification, shown in Figure 6(a), utilizes computer storage and one D/A channel to replace the analog computer track-and-store-unit. In Figure 6 another D/A channel serves the function of the sample and zero-order hold device of Figure 5. This system has the advantage of removing the error associated with the analog track-and-store unit and the disadvantage of doubling the digital computer output storage and buffer requirements.

A logical extension of this method is to perform the subtraction and division in the digital computer, eliminating one summing amplifier and a scalar multiplication. This change is shown in Figure 6(b). The result is to increase the computational requirements of the digital computer by one addition and one multiplication.

(c) METHOD III - Digital Hardware-Software Implementation

An approximation to a first order sample-hold unit can be generated using a zero order sample-hold running at a faster rate. For reasons of timing, it is best to use synchronous multirate sampling. The approximating equation can be developed by assuming that the fundamental sampling period is $\delta$ and that a multirate sampling unit is available with sampling period $\delta^*$ where $\delta = p\delta^*$ and $p$ is an integer. Equation (10) can then be approximated by the equation

$$
\hat{F}_1(t) = f(n\delta) + \frac{f(n\delta) - f((n-1)\delta)}{\delta} (\delta^*) \\
n\delta + \delta^* \leq t < n\delta + (\delta+1)\delta^* \\
\ell = 0, 1, \ldots, p-1.
$$

(11)

A comparison of $F_1(t)$ from (10) and $\hat{F}_1(t)$ from (11) is shown in Figure 7 for
Figure 6(a)

Figure 6(b)

Figure 6. Method II: Digital-Analog First Order Hold
Figure 7. Ideal and Approximate Representations for First Order Hold Outputs
the case $p = 4$. It can be seen that as $p$ becomes large, say ten or greater, the function $\hat{F}_1(t)$ becomes almost identical to the function $F_1(t)$.

Equation (11) can be implemented as shown in Figure 8. It is assumed that the output signals to the digital computer are available at the fundamental sampling times $n\delta$. This permits the calculation of the digital function $f(n\delta)$. Using shifting and additional storage, the previous point, $f((n-1)\delta)$, is also preserved. The algorithm given by (11) is then used to compute various intermediate points which are transferred to the analog computer through a zero order hold with sampling period $\delta^*$. The computation can be done in one sequence with all values stored for the interval $[n\delta, (n+1)\delta]$, as indicated in Figure 8. These values would then be converted from binary to analog form in proper sequence through the multirate sample and zero-order hold device.

A second alternative is to compute a value only when it is needed and then convert from digital to analog. This places the timing burden on the digital computer and forces additional interruptions on other algorithms which are being performed.

It is obvious from Figure 8 that the disadvantages of this scheme are the timing and sequencing problems. However, it should be pointed out that the method can easily be extended to higher order interpolation with minor changes in storage requirements and the computational algorithm. This gives the distinct possibility of extending the hybrid computer to an equivalent $n$th order numerical method.

Each of the above methods approximates a first-order hold with varying degrees of accuracy. The particular errors associated with the generated function for each method will be considered in the following section.
\[ F_1(t) = f(n\delta) + \frac{f(n\delta) - f((n-1)\delta)}{\delta} \quad \ell \delta^* \]

\[ \ell = 0, 1, \ldots, (p-1) \]

**Figure 8.** Method III: Digital First Order Hold
3. **Errors in First Order Hold Implementation Methods**

(a) **Ideal**

The ideal first order hold output is illustrated in Figure 7 as given by (10). This function is seen to be one with constant slope on the interval \([n_5, (n+1)_5]\). Any departure from this interpolation function, \(F_1(t)\), represents an error in the desired output and indicates the implementation is only an approximation to the true first order hold device.

The following will discuss errors in the various methods due to computation time and other approximations. It should be recalled that sections 1 and 2 of this chapter have dealt with an arbitrary function \(f(t)\). The hybrid computer, however, has the specific task of computing digitally a function \(f_d(t)\) which is a function, in general, of \(x(t), z(t)\) and time. It is the evaluation of this function at the proper time which causes the problem leading to errors.

(b) **METHODS I & II**

In either of these methods the primary error is that associated with the computation delay, \(e\), in evaluating the function \(f_d(n_5)\). Since \(x(n_5)\) is not available until \(t = n_5\) the function cannot be evaluated at \(t = n_5\), but at some later time \(t = n_5 + e\). The result is that the desired output of the first order hold, \(F_d(t)\), cannot be computed over the interval \([n_5, (n+1)_5]\), but, as shown in Figure 9, can be computed in either of two ways: 1) a delayed function \(F_d(t-e)\) or, 2) an extended function \(F_d^*(t)\). The delayed function is much easier to compute and is the function on which (2) is based.

The computation of \(F_d^*(t)\) requires a knowledge of \(e\), and since this is usually not known exactly, would introduce an unknown vertical displacement between \(F_d(t)\) and \(F_d^*(t)\).
Figure 9. Effect of Execution Time on Implementation of First Order Hold by Methods 1 and 11
When several functions are generated digitally the approximation becomes particularly acute. Consider Figure 10 where four functions are considered to be evaluated in sequence with respective execution times $e_1$, $e_2$, $e_3$, and $e_4$. The desired outputs and the delayed outputs are shown in Figure 11 for functions $f_{d_1}$ and $f_{d_2}$. It is apparent from this figure that the approximation becomes much worse as the number of functions or total execution time increases.

(c) **METHOD III**

The method illustrated in Figure 12 computes a set of points which will approximate the first-order hold. It should be noted that this method closely duplicates Methods I & II when a large number of intermediate points are calculated on the interval $[n\delta, (n+1)\delta]$. Hence, the method has the delay due to execution which is exhibited by Methods I and II in Figure 9, and also adds a quantization error to this basic computation delay error. It should be emphasized that the error equation associated with this particular method has not been derived and is not given by (8). Equation (8) should hold as an asymptotic approximation for Method III, when using sampling ratios greater than 100/1. For ratios less than 10/1 as illustrated in Figure 8, it is anticipated there would be some modifications in the constants of (8), but the equation should remain second order in $\delta$ and $e$.

The previous discussion has pointed out the predominance of errors due to execution time in the first order hold implementations of Method I, II and III. Chapter IV will suggest methods for removing this error source in the implementation of first order hold methods. When this is accomplished, (8) becomes the error equation for Methods I and II, while it is only asymptotically correct for Method III.
Figure 10. Sequencing of Multi-Function Generation.

Figure 11. Effect of Execution Time of Multi-Function Generation
Figure 12. Errors in Method III Implementation of First-Order Hold
4-1. General Statement of the Problem

As pointed out in the section Introduction and Statement of Problem, there are three computational errors associated with the hybrid computer simulation solution to a physical problem. These are given by (3) and later delineated in (4) by the functions $\tilde{F}_1$, $\tilde{F}_2$, and $\tilde{F}_3$. The function $\tilde{F}_2 = [ef_d,0]^T$ is seen to contribute errors in proportion to the execution time, $e$, as a result of the digital computation of the function $f_d$. Since $f_d$ is normally a function of time and the vectors $x$, $z$ and possibly an input vector, $r$, it is impossible to compute $f_d[x(t_n),z(t_n),r(t_n);t_n]$ at $t=t_n$, but rather at $t=t_n+e$ where $e$ is the time associated with the evaluation of $f_d$. The section on compensation for zero order hold devices has also demonstrated that function evaluation delays, cause errors in these compensation methods. Thus, the construction of first order hold algorithmic methods depends on function evaluation at time $t=t_n$ from data available at $t=t_n$.

This section will examine three aspects of execution time: 1) methods to minimize errors due to execution time through a proper sequencing of the digital operations; 2) elimination of execution time by extrapolation and estimation techniques; and, 3) execution time compensation applied to first order hold algorithms.

4-2. Minimization of Execution Time Errors by Proper Sequencing of Operations

During any one frame period a given set of digital computations is to be executed in some prescribed order. It is extremely important that
priority be given to the computation of $f_d$ in this computational sequence. It is further necessary that this value of $f_d$ be converted for use by the analog computer, immediately upon the completion of the computation of the function value. Any delays in starting the computation or converting the value for use in the analog system has an additive effect on the execution time.

Within the above constraints comes the priority of evaluation of the components of the vector $f_d$. Since any error contribution is the product of the execution time and $f_d$, one strategy is to order the components of $f_d$ according to the maximum absolute magnitude of their respective time derivatives and evaluate the functions beginning with the largest absolute time derivative value and continuing in order through the smallest. There are many other strategies based on $f_d$ that may work equally well or better. The key is to make use of available information on $f_d$ in arriving at some priority for the evaluation of the components of $f_d$.

In summary the basic principles for reduction of errors due to execution time are:

1) Evaluate $f_d$ as early in the frame period as possible;
2) Convert each component evaluation of $f_d$ from digital to analog immediately after computation; and
3) Evaluate the components of $f_d$ in an order based on some estimate of the component time derivatives which orders large values first in the computational sequence.

These principles are depicted in Figure 13 for one complete frame time of the hybrid system. It should be noted that in this figure all functions are shown as being computed prior to the numerical integration steps. This can be done if a multistep algorithm is being employed, but may need to be
At Evaluation of Functions Necessary for Numerical Integration of z.

Output to Buffer for D/A Conversion
Evaluate component of \( f_d \) which has second largest absolute magnitude of time derivative
Output to Buffer for D/A Conversion
Evaluate component of \( f_d \) which has largest absolute magnitude of time derivative

Output to Buffer for D/A Conversion
Evaluate component of \( f_d \) which has smallest absolute magnitude of time derivative

Total Function Execution Time = \( e \)

Figure 13. Sequencing of Digital Computations to Minimize Execution Time Error

Evaluation of Functions Necessary for Numerical Integration of \( z \).
altered for one step methods. The alteration consists in interspersing any required intermediate digital data collection and evaluations within the function evaluation sequence. For this reason, multistep algorithms may give some advantages when ε is large with respect to ε.

One final problem concerns proper timing of the operations. Figure 14 illustrates the operational hardware for the case of a single digital output channel to a time-shared buffer. The case of multiple output channels eliminates the problem of demultiplexer sequencing. The demultiplexer is synchronized by a timing pulse at the fundamental sampling rate, ε. This pulse sets the demultiplexer to some reference state. The succession of numbers entering the buffer is then placed in an ordered array of output registers, each controlling a separate D/A converter. The operational sequence is shown in Figure 15. It is seen in Figure 15 that as soon as component j has been evaluated and the number transferred to the buffer, a pulse is generated to sequence the demultiplexer to the (D/A)j input register. When the number transfer is complete, a second pulse initiates the D/A conversion process and disconnects the demultiplexer.

4-3. Extrapolation and Estimation Techniques

The central problem in removing execution time completely involves the evaluation of a function, \( f_d(x, z, r; t) \) at \( t = t_{n+1} \) to obtain \( f_d[x(t_{n+1}), z(t_{n+1}), r(t_{n+1}); t_{n+1}] \). Since any computation will take a finite amount of time and the values \( x(t_{n+1}) \) and \( r(t_{n+1}) \) are not available until \( t = t_{n+1} \), it is impossible to do this exactly. However, by computing some estimate of \( x(t_{n+1}) \) and \( r(t_{n+1}) \), designated \( x_{n+1} \) and \( r_{n+1} \), and using one of the numerical integration techniques, discussed in Chapter II to produce an estimate of \( z(t_{n+1}) \), designated \( z_{n+1} \), it is possible to compute \( \hat{f}_d = f_d(x_{n+1}, z_{n+1}, r_{n+1}; t_{n+1}) \) at \( t = t_{n+1} \). If this value can be made acceptably close to \( f_d \) then execution
Figure 14. Synchronism of Operations
Figure 15. Hardware Timing Sequence and Operation
time can be set to zero in all error calculations.

Chapter II has discussed the use of various numerical integration techniques in the computation of $z_{n+1}$, from data accumulated prior to the time $t=t_{n+1}$. The two remaining problems are the computation of $x_{n+1}$ and $r_{n+1}$ from data samples prior to the time $t=t_{n+1}$.

The general problem formulation given in Chapter I requires the analog equations to be in the form

$$\dot{x} = f_a(x,z;t) + f_d(x,z;t)$$

where $x$ represents a set of state variables in the analog system. As has been explained for the numerical integration of the $z(t)$ equation, values are obtained for the equation at $t=t_{n+1}$ from computation at some time prior to $t=t_{n+1}$. It is possible, then, to use a numerical integration technique for the $\dot{x}(t)$ equation and obtain a similar value for $x_{n+1}$ prior to the time $t=t_{n+1}$. It is theoretically possible to use any method (multistep or one-step) of any order. This does have the disadvantage, however, of duplicating the analog solution of the differential equation on the digital computer and placing an extra computational burden on the digital machine. It is desirable to achieve this same result without the corresponding disadvantage of solving the differential equation in $\dot{x}(t)$. Two particularly useful techniques which meet these conditions; Taylor Series Extrapolation, and Polynomial Extrapolation, will be presented in some detail.

4-4. Taylor Series Extrapolation

In many cases the analog computer will be programmed such that the components of the state vector $x(t)$, appear at the outputs of integrators with their explicit time derivatives, $\dot{x}(t)$ available at the inputs. Expanding the analog computer state vector in a Taylor Series as
\[ x(t) = x(t_n) + (t-t_n)x(t_n) + \ldots \quad (12) \]

yields an algorithm for computing an estimate of \( x(t_{n+1}) \),

\[ x(t_{n+1}) \approx x(t_n) + (t_{n+1}-t_n)x(t_n) = x_{n+1}. \quad (13) \]

The implementation of such an algorithm is shown in Figure 16. It will be noted that the only penalty in this method is an increase in the number of A/D channels (possibly twice as many) and computation time for one multiplication and one addition per function. The resulting hybrid computer program would appear as in Figure 17 with the extrapolation algorithm included. This programming technique will completely eliminate execution time from the error equations in those cases where the analog state variable and its derivative are available and depend only on time and the vectors \( x \) and \( z \).

It should be noted that although this is a first order numerical technique, the errors are not comparable to such methods. General integration techniques of this type have a per step error and an accumulated error. The error for the method being used in (12) is similar to the per step error since \( x(t_{n+1}) \) will be calculated and used in calculations after \( t=t_{n+1} \). Hence, this method should prove considerably more accurate than a first order numerical method.

4-5. Polynomial Extrapolation

In case the simulation contains an input, \( r(t) \) such that \( f_d = f_d(x,z,r;t) \), it becomes necessary to obtain \( r(t_{n+1}) \), as well as the other variables, prior to the time \( t_{n+1} \) in order to compute \( f_d \) prior to \( t_{n+1} \). If \( r(t) \) is being generated on the analog computer as the solution to a differential equation (such as \( \sin t \) or \( e^{at} \)) the computation of \( r_{n+1} \) can proceed
Taylor Series Extrapolation Algorithm

\[ x_{n+1} = x(t_n) + (t_{n+1} - t_n) \ddot{x}(t_n) \]

Figure 16. Implementation of Taylor Series Extrapolation Algorithm
Figure 17. Hybrid Solution Program to Eliminate Execution Time
using the Taylor Series Extrapolation Method as discussed in Section 4-4.

If \( r(t) \) is not the solution to a differential equation, or is not known a priori, it then becomes necessary to use some type of extrapolation algorithm to predict \( r_{n+1} \) from data available on \( r(t) \) for \( t < t_{n+1} \). The literature contains a large number of references on interpolation and extrapolation techniques, see for example [15]. Most of these methods are polynomial approximation techniques either derived from or related to the Lagrangian interpolation formula.

The Lagrangian interpolation formula is given by

\[
f(t) = \sum_{j=1}^{n} \ell_j(t)f(t_j) + E(t)
\]

where \( f(t_j) \) are the function values at \( t = t_j; j=1, \ldots, n \),

\[
\ell_j(t) = \frac{p_n(t)}{(t-t_j)p'_n(t_j)},
\]

\[
p_n(t) = \prod_{i=1}^{n}(t-t_i),
\]

\[
p'_n(t_j) = \frac{d}{dt} p_n(t) \bigg|_{t=t_j}
\]

and

\[ E(t) \] is the error in the approximation.

For the particular case of interest, \( t = t_{n+1} \) and \( t_{j+1} - t_j = \delta, j=1, \ldots, n-1 \). With these substitutions equations (4) and (5) become

\[
f(t_{n+1}) = \sum_{j=1}^{n} \frac{(-1)^{n-j}n!}{(j-1)!(n-j)!(n+1-j)} f(t_j) + E(\zeta)
\]

where the error in the approximation is given by \( E(\zeta) = \delta^n f^{(n)}(\zeta) \) with \( \delta \) is an unknown function of \( t \) on the interval \([t_1, t_{n+1}]\). The equation for
error indicates this extrapolation procedure to be $0(c^n)$ so that it is a procedure of order $(n-1)$. Sufficient accuracy for the present application can be obtained by using $n=3$ which is actually a quadratic interpolation-extrapolation procedure. When $E(\xi)$ is set to zero in (16) the result is an approximation to $f(t_{n+1})$, given as $f_{n+1}$.

The Lagrangian Extrapolation procedure can be implemented as shown in Figure 18 with an A/D converter for each signal component of the input vector $r(t)$. The total storage for each component is $n$ data points, $n$ constants and $n$ products or a total of $3n$ words. The computational operations required are $n$ multiplications, $(n-1)$ additions and $(n+1)$ data transfers. It can be seen that when $n=3$, this method is relatively simple and may be preferred to Taylor Series Extrapolation, even when the function and its time derivative are available. It should be emphasized that the Lagrangian Extrapolation Algorithm is a general procedure applicable to the analog state vector, $x(t)$, as well as any input vector $r(t)$. The algorithm can also be used in conjunction with the Taylor Series Extrapolation method, in which it is used to compute $x_{n+1}$ for those components of $x(t)$ which cannot be treated by the Taylor Series Extrapolation Algorithm.

4-6. Execution Time Compensation Applied to First Order Hold Algorithms

It is appropriate to distinguish between the elimination of execution time errors in zero-order hold systems as discussed in the preceding two sections and first order hold systems to be discussed here. In the zero-order hold system, elimination of execution time error does not change the fact that the hybrid system is still equivalent to a first order numerical method with an error term which is $O(c)$. On the contrary, if a first order hold is implemented, and a second order (or higher) numerical method is used, the system is equivalent to a second order numerical method with
Figure 18. Implementation of the Lagrangian Extrapolation Algorithm
errors which are \(O(\varepsilon^2)\). As in the case of zero order systems, some of the error terms can be eliminated if execution time is zero to zero. The methods for doing this are exactly the same as those discussed in the first two parts of this section. The purpose of this section is to illustrate the application of these methods for eliminating execution time errors in the implementation of first-order hold algorithms. Chapter III has shown the principal errors in zero order hold compensation algorithms to be those due to execution time.

Three such methods have been discussed in the section on first-order hold implementation schemes. The following discussion will treat two cases; 1) Methods I and II, which are computationally identical and, 2) Method III.

(a) **Execution Time Elimination in Methods I and II**

In order to implement (10), where the arbitrary function \(f\) is replaced by the specific function \(f_d\), it is necessary to have available at \(t=t_n\) the values \(z(t_{n-1}), z(t_n), f_d(t_{n-1})\) and \(f_d(t_n)\). Given these values it is a straightforward problem to form the desired functions according to Figures 5 or 6. Further, it has been shown that the numerical integration method produces \(z_n\) prior to the time \(t=t_n\). Hence, a storage of the prior point \(z_{n-1}\), by either the analog computer (Figure 5) or digital computer (Figure 6) is all that is needed to solve half the problem.

The computation of \(f_d[x(t_n),z(t_n),r(t_n);t_n]\) prior to the time \(t=t_n\) has been discussed in sections 4-3, 4-4 and 4-5. By application of one or a combination of these methods, the approximation \(f_d(x_n,z_n,r_n;t_n)\) can be calculated while \(f_d(t_{n-1})\) is stored from the previous calculation. The remaining steps are then the same as those for the function \(f(t)\) and are illustrated in Figures 5 and 6.
Method III requires the same techniques described in the previous section to obtain $z(t_{n-1}), z(t_n), f_d(t_{n-1})$ and $f_d(t_n)$. In addition, it is necessary to have these values at the proper time, to evaluate the function, $\hat{F}_1(t)$, given by (11), prior to the time $t=t_n$. The values obtained must be transferred to the appropriate storage registers for demultiplexing, D/A conversion and storage by a zero-order hold device. The end result is an approximation to a first order hold which becomes increasingly more accurate as $\delta^* \to 0$.

The required computational procedures, methods to be used and their respective relations to the various first-order hold schemes is summarized in Figure 19. The order of computation is not rigid, but some ordering must be observed. If a one step numerical method is being used, the evaluation of $z(t_n)$ must be delayed until late in the computation period. Further, if $f_d(t_n)$ is being evaluated as a function of $z(t_n)$ it must follow this computation, and all algorithms based on $f_d(t_n)$ must come last.

One final comment should be made concerning the use of the Lagrangian Extrapolation procedure to find $f_d(t_n)$. Since this procedure is based on prior values of $f_d$, it is not necessary to compute $x_n, z_n$ and $r_n$ before evaluating $f_d(t_n)$. It is necessary, however, to eventually obtain $f_d(t_n)$ from samples of $x(t_n)$ and $r(t_n)$ and the computed value of $z(t_n)$, but this can be delayed in the computational procedure. The computational sequence and methods to be used when following this procedure are given in Figure 20.

Comparing Figure 19 with Figure 20 shows the two procedures to be similar. Figure 20 differs in requiring a Lagrangian Extrapolation of the vector $f_d(t)$ while the method depicted in Figure 19 requires a
<table>
<thead>
<tr>
<th>Used In First Order-Hold Method</th>
<th>Method</th>
<th>Computation</th>
</tr>
</thead>
<tbody>
<tr>
<td>I, II, III</td>
<td>Lagrangian Extrapolation</td>
<td>Computation of $r(t_{n+1})$</td>
</tr>
<tr>
<td>I, II, III</td>
<td>Taylor Series and/or Lagrangian Ext.</td>
<td>Computation of $x(t_{n+1})$</td>
</tr>
<tr>
<td>I, II, III</td>
<td>Mult-Step or one step</td>
<td>Computation of $z(t_{n+1})$</td>
</tr>
<tr>
<td>I, II, III</td>
<td>Function Evaluation</td>
<td>Computation of $f_d(t_{n+1})$</td>
</tr>
<tr>
<td>III</td>
<td>Function Evaluation</td>
<td>Computation of Intermediate Data for Fast Sampler or</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Computation of Constant $\frac{1}{T}[f_d(t_{n+1}) - f_d(t_n)]$</td>
</tr>
<tr>
<td>II(b)</td>
<td></td>
<td>Numerical Integration</td>
</tr>
</tbody>
</table>

**Figure 19. First Order Hold Computation Requirements and Methods**
<table>
<thead>
<tr>
<th>Used In First Order Hold Method</th>
<th>Method</th>
<th>Computation</th>
</tr>
</thead>
<tbody>
<tr>
<td>I, II, III</td>
<td>A/D Conversion, etc.</td>
<td>Input ( r(t_n), z(t_n), x(t_n) )</td>
</tr>
<tr>
<td>I, II, III</td>
<td>Function Evaluation</td>
<td>Compute ( f_d(t_n) )</td>
</tr>
<tr>
<td>I, II, III</td>
<td>Lagrangian Extrapolation</td>
<td>Compute ( f_d(t_{n+1}) )</td>
</tr>
<tr>
<td>I, II, III</td>
<td>Multistep or One Step</td>
<td>Compute ( z(t_{n+1}) )</td>
</tr>
<tr>
<td>II(b)</td>
<td>Function Evaluation</td>
<td>Compute Constant</td>
</tr>
<tr>
<td>III</td>
<td>Function Evaluation</td>
<td>Compute Intermediate Data</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Numerical Integration</td>
</tr>
</tbody>
</table>

Figure 20. First Order Hold Computation Requirements and Methods
Lagrangian Extrapolation of the vector \( r(t) \) and a Taylor Series Extrapolation (or Lagrangian Extrapolation) of the vector \( x(t) \). Since the vector \( f_d(t) \) has a dimension at most equal to that of \( x(t) \), the computational operations associated with Figure 20 should place a smaller burden on the digital computer and make possible a shorter frame time.
CHAPTER V

SUMMARY

This Technical Note has presented in Chapter I a statement of the problem and the resulting error equation as derived in Technical Note No. 17. The object of the remaining chapters is to focus on various aspects of this equation and indicate methods for reducing or eliminating particular error components. This is accomplished for three major terms; (1) numerical integration, (2) sampling with zero order hold, and (3) execution time delay.

Chapter II presents a review of general characteristics of various numerical integration methods important for application in hybrid computer software. Special consideration is given to the problems of timing of hybrid computation and errors in hybrid computations. The chapter concludes with a section on considerations for the choice of a numerical integration method for hybrid computation.

Chapter III presents a comparison of errors due to sampling with zero order hold and sampling with first order hold. It is shown that increasing the order of the hold effectively increases the numerical order of the hybrid computational technique. Three methods for implementing a first order hold are discussed including the errors associated with each method.

Chapter IV discusses the error component due to execution time delay. Several methods are discussed for reducing this error including proper sequencing, extrapolation and estimation, Taylor Series Extrapolation and Polynomial Extrapolation. Execution time error associated with first
order hold methods is also considered. Methods for compensation of the first order hold implementations presented in Chapter III are discussed.

Finally, it should be noted that the hybrid software methods suggested here are rather preliminary. These techniques have been devised by observing the error equation and asking what can be done to improve the computational accuracy. The second step of implementing the ideas in actual hybrid programs is yet to be done. Also, remaining is the critical step of evaluating the effectiveness of the ideas with respect to cost, speed, accuracy, and overall usefulness.
REFERENCES


PRELIMINARY EVALUATION OF A TECHNIQUE FOR COMPENSATING AGAINST SAMPLING ERRORS IN HYBRID COMPUTERS

By Shing Ted Li
Joseph L. Hammond, Jr.

Prepared for
George C. Marshall Space Flight Center
Huntsville, Alabama

Contract No. NAS8-2473

(Development of New Methods and Applications of Analog Computation)

September 15, 1970

School of Electrical Engineering
GEORGIA INSTITUTE OF TECHNOLOGY
Atlanta, Georgia
PRELIMINARY EVALUATION OF A TECHNIQUE FOR COMPENSATING AGAINST SAMPLING ERRORS IN HYBRID COMPUTERS

by

Shing Ted Li and Joseph L. Hammond, Jr.

TECHNICAL NOTE NO. 20

on

Contract No. NAS8-2473

(Development of New Methods and Applications of Analog Computation)

For

GEORGE C. MARSHALL SPACE FLIGHT CENTER
Huntsville, Alabama
ABSTRACT

The results of preliminary studies of the digital hardware-software method, described in Technical Note No. 19, for compensating against sampling errors in hybrid computers are reported. Three examples in which the method is applied are studied with a digital simulation of a hybrid computer. In all three cases significant reduction in error is observed.

INTRODUCTION

The hybrid computer error equation developed on the subject contract indicates that the error depends on the first power of the sampling period, δ, if zero order hold D/A converters are used. Three methods for compensation which effectively convert a zero order hold into a first order hold have been suggested in Technical Note No. 19 [1]. These methods, identified as analog hardware implementation, digital-analog hardware implementation and digital hardware-software implementation, all reduce sampling error so that it depends on δ² rather than δ.

This report gives the results of a preliminary study of compensation using the digital hardware-software implementation. The hybrid system is simulated digitally using the DIHYSYS digital hybrid simulation program developed recently [2].
Table I. Input Data for DHYSYS Program

<table>
<thead>
<tr>
<th>QUANTITY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type of Hold</td>
<td>zero</td>
</tr>
<tr>
<td>Sampling Period A/D</td>
<td>0.1</td>
</tr>
<tr>
<td>Sampling Period D/A</td>
<td>variable</td>
</tr>
<tr>
<td>&quot;Analog&quot; equation error</td>
<td>$10^{-4}$</td>
</tr>
<tr>
<td>Number of subdivisions in sampling interval</td>
<td>10</td>
</tr>
<tr>
<td>Step size for solving digital equations</td>
<td>0.1</td>
</tr>
<tr>
<td>Maximum permissible number in analog system</td>
<td>100</td>
</tr>
<tr>
<td>Maximum permissible number of digital system</td>
<td>1000</td>
</tr>
<tr>
<td>Number of bits for analog and digital system and hold device</td>
<td>38</td>
</tr>
<tr>
<td>Print time</td>
<td>0.1</td>
</tr>
<tr>
<td>Execution time</td>
<td>0</td>
</tr>
</tbody>
</table>
DY(3) = Y(4)
DY(4) = -4Y(3)

Procedure DIGEQ

B(1) = -4Y(1)

Procedure OUTEQ

H(1) = Y(3)
H(2) = Y(1)
H(3) = Y(3) - Y(1)

The program produces Y(3) and Y(4) as true solutions to the given equations and Y(1) and Y(2) as "hybrid" solutions to the equations using an uncompensated zero-order hold with -4Y(1) being produced digitally and other operations carried out in the "analog" computer. The variables Y(1), Y(3) and Y(3) - Y(1) are printed out.

IMPLEMENTATION OF COMPENSATION METHOD

The method of compensation investigated in this study is described in detail in [1] pp. 27-30. The method replaces the output, F(t), of a standard zero-order hold by a compensated variable, $F^*(t)$. If $\delta$ is the fundamental sampling period a fast D/A unit with sampling period $\delta^*$ is required so that $\delta^*$, F(t) and $F^*(t)$ satisfy the equations

$$\delta^* = \delta/p, \ p \text{ an integer}$$
\[ F(t) = f(n\delta), \quad n\delta \leq t < (n+1)\delta \]

\[ F^*(t) = f(n\delta) + \frac{f(n\delta) - f((n-1)\delta)}{\delta} \quad (m\delta*) \]

\[ n\delta + m\delta* \leq t < n\delta + (m+1)\delta* \]

\[ m = 0, 1, \ldots, p-1 \]

These equations can be implemented in a straightforward fashion with the DIHYSY program. The equations for example one, for illustration, are modified as follows. The conversation period of the D/A converters is changed from \( \delta \) to \( \delta* = \delta/p \), (in the example used \( \delta = 0.1 \) and \( p = 10 \)), while the period of the A/D converter is effectively maintained at \( \delta \). The DIGEQ procedure is changed to

**Procedure DIGEQ**

\[ B(1) = -4Y(1) + \frac{-4Y(1) - [-4Y^*(1)]}{10\delta*} \quad m\delta* \]

while the other procedures remain the same. Note that provisions are made to store \( Y(1) \) for one iteration to obtain \( Y^*(1) \), that \( Y(1) \) is updated with a period \( \delta \) and that \( m \) is a counter variable corresponding to a period \( \delta* \).

**EXAMPLES**

**Example 1:**

The "hybrid" solution of the equations

\[ \dot{x}_1(t) = x_2(t), \quad x_1(0) = 1, \quad x_2(0) = 0 \]
\[ \dot{x}_2(t) = -4x_1(t) \]

is studied with DIHYSYS under the following condition: 

\(-4x_1(t)\) is generated digitally, \(\delta = 0.1\) and \(\delta^* = 0.01\). Figure 1a gives curves of the true solution for \(x_1(t)\) and the hybrid solution for \(x_1(t)\) both compensated and uncompensated. Figure 1b gives the corresponding error curves.

**Example 2:** This example deals with Duffing's equation in the form

\[
\begin{align*}
\dot{x}_1(t) &= x_2(t), \quad x_1(0) = 4, x_2(0) = 0 \\
\dot{x}_2(t) &= -x_1(t) - 0.06 x_1^3(t) - 2x_2(t)
\end{align*}
\]

In solving these equations \(-0.06 x_1^3(t)\) and \(-2x_2(t)\) are generated digitally, \(\delta = 0.1\) and \(\delta^* = 0.01\). A plot of the true solutions to Duffing's equation is given in Figure 2. The errors of \(x_1(t)\) and \(x_2(t)\) with and without compensation are given in Figure 3 and the corresponding per cent error in Figure 4.

**Example 3:** This example deals with Duffing's equation with zero initial conditions and a random forcing function in the form

\[ \dot{x}_1(t) = x_2(t) + n(t) \]

\[ \dot{x}_2(t) = -x_1(t) - 0.06 x_1^3(t) - 2x_2(t). \]

Allocations of operations and the values for \(\delta\) and \(\delta^*\) are the same as for example 2. The variable \(n(t)\) is a sample function from a Gaussian mean zero unit variance random process. This variable is applied to the analog part of the system.
Figure la. Study of Solutions for Hybrid System of Example 1
Figure 1b. Study of Errors for Hybrid System of Example 1
Figure 2. True Solution of Duffing's Equation
Figure 3. Errors of the Solutions of Duffing's Equation for Example 2.

(a) Errors of State Variable $x_1(t)$

(b) Errors of State Variable $x_2(t)$
Figure 4. Percentage Errors of the Solution of Diffing's Equation for Example 2

(a) Percentage Errors of State Variable $x_1(t)$

(b) Percentage Errors of State Variable $x_2(t)$
The error of $x_1(t)$ with and without compensation is given in Figure 5.

**COMMENTS ON THE RESULTS**

In all three examples considerable reduction in error is brought about by the compensation technique. For example in Figure 4 the per cent error is reduced by compensation from 103% to 11% during the first six seconds.

The effect of changing the ratio of $\delta$ to $\delta^*$ was investigated by changing $p$ from 10 to 20. This produced no significant change in the error for the compensated case.

It is interesting to note in comparing the results of examples two and three that sampling errors accumulate to a larger extent with deterministic excitation (with initial conditions) than with random excitation.

**REFERENCES**


Figure 5. Errors of $x_1(t)$ for Example 3
TECHNICAL NOTE NO. 21
Research Project A-588

A PRELIMINARY STUDY OF METHODS FOR SOLVING PARTIAL DIFFERENTIAL
EQUATIONS WITH A HYBRID COMPUTER

By William M. O'Dowd
Joseph L. Hammond, Jr.

Prepared for
George C. Marshall Space Flight Center
Huntsville, Alabama

Contract No. NAS8-2473

(Development of New Methods and
Applications of Analog Computation)

17 September 1970

School of Electrical Engineering
GEORGIA INSTITUTE OF TECHNOLOGY
Atlanta, Georgia
A PRELIMINARY STUDY OF METHODS FOR SOLVING PARTIAL DIFFERENTIAL EQUATIONS WITH A HYBRID COMPUTER

by

William M. O'Dowd and Joseph L. Hammond, Jr.

TECHNICAL NOTE NO. 21

on

Contract No. NAS8-2473

(Development of New Methods and Applications of Analog Computation)

For

GEORGE C. MARSHALL SPACE FLIGHT CENTER
Huntsville, Alabama
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>iv</td>
</tr>
<tr>
<td>I. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>II. PROBLEM STATEMENT AND SETUP</td>
<td>3</td>
</tr>
<tr>
<td>III. ZERO ORDER HOLD RESULTS</td>
<td>6</td>
</tr>
<tr>
<td>IV. POLYNOMIAL APPROXIMATION</td>
<td>18</td>
</tr>
<tr>
<td>V. CONCLUSIONS</td>
<td>19</td>
</tr>
<tr>
<td>VI. BIBLIOGRAPHY</td>
<td>19</td>
</tr>
</tbody>
</table>
ABSTRACT

This technical note gives the results of a preliminary study of methods for solving partial differential equations with a hybrid computer. The study deals with one form of serial solution and investigates two methods of treating stored data.

The one dimensional heat equation is used as a test case for the study. The partial differential equation is converted to a set of ordinary differential equations in the time variable using seven discrete space "stations." The ordinary differential equations are solved using a serial, iterative technique which requires storage of the complete time solution at each space point. An allocation of hybrid operations using analog integration of the differential equations and digital storage is simulated on an all digital computer.

Results are obtained under a variety of conditions for a zero order hold approximation to stored data with and without a time shift in the data. A limited amount of data is obtained for a polynomial approximation to the stored data.

The polynomial approximation gives the best results of the methods tried. For the zero order hold, it seems necessary to shift the data in time a precise amount for the results of the computation to converge to the correct answer.
I. INTRODUCTION

The solution to partial differential equations (p.d.e.s.) presents a very formidable problem to mathematicians and engineers. Solutions of any kind (analytical, analog, or digital) are in general very difficult to obtain. Analytical solutions are known for only a small class of problems. Since a p.d.e. has more than one independent variable, it cannot be solved directly with an analog or digital computer. The efficient and well developed computer methods (both analog and digital) for solving ordinary differential equations (o.d.e.s.) cannot be applied directly to a p.d.e.

A great variety of ways and methods have been tried in order to solve p.d.e.s. A complete bibliography of methods for solving p.d.e.s. was prepared by D. L. Finn [1] in a previous A-588 technical note.

The purpose of this technical note is to examine several hybrid computer techniques using the one dimensional heat flow equation.

\[ \frac{\partial^2 u(x,t)}{\partial x^2} = \frac{\partial u(x,t)}{\partial t} \]  

as a test problem. This equation is a second order parabolic p.d.e. The basic method of attack is to convert the p.d.e. to a coupled set of o.d.e.s. by making the space variable a discrete variable. This same problem was studied by Howe and Hsu [2]. The second partial of \( u(x,t) \) with respect to \( x \) is replaced by its finite difference approximation.
\[
\frac{\partial^2 u(x,t)}{\partial x^2} \bigg|_{x = i \Delta x} = \frac{u((i+1)\Delta x,t) - 2u(i\Delta x,t) + u((i-1)\Delta x,t)}{\Delta x^2} - 2\frac{u(x,t)}{\Delta x} + \frac{u(x,t)}{\Delta x^2}
\]  

(2)

Putting (2) into (1) yields a set of o.d.e.s. of the general form

\[
\frac{d u_i(t)}{dt} + 2\frac{u_i(t)}{\Delta x} = \frac{u_{i+1}(t) + u_{i-1}(t)}{\Delta x^2}
\]

(3)

\[i = 1, 2, \ldots, n\]

Note that use of (3) replaces the p.d.e. by \(\frac{x_{\text{max}}}{\Delta x}\) coupled o.d.e.s. Conceptually these coupled equations can be solved quite easily on a hybrid computer. However, equipment limitations make their solution difficult.

One straightforward solution, the parallel solution, is to simulate the entire set of equations on the analog computer at once. Since the number of o.d.e.s. is typically large, this method requires excessive equipment in most cases.

The solution herein examined is the serial solution. Each o.d.e. is solved individually, and in succession. Since the equation for \(u_i(t)\) has \(u_{i+1}(t)\) and \(u_{i-1}(t)\) as forcing functions, it is necessary to have these functions stored (the details of this method are described in the next section). The analog section of the hybrid computer is used to solve each o.d.e., and the digital section is used for the necessary storage. Since each \(u_i(t)\) will have to be stored, the required storage can become very large. To reduce the storage requires
that fewer than the desired number of points be stored. Since it will be necessary to furnish an analog signal corresponding to this stored data, the problem arises as to how can the discrete data best be converted to a continuous signal for further solutions.

In section two of this note the difference equations for the problem are set up and the solution procedure explained. In section three the results are presented for a computed solution using a zero order hold to get an analog signal. Since a zero order hold often makes the information look like it is delayed in time, a forward shift in the zero order hold is also examined. In section four a method is studied which uses a polynomial approximation to obtain points between the stored values. This permits a zero order hold with a much shorter hold time to be used.

Since a hybrid computer was not available the methods studied were simulated on a large scale general purpose digital computer (Univac 1108). A fourth order Range-Kutta method was used to solve the o.d.e.s. For the methods studied the digital simulation provides an accurate approximation to the true hybrid behavior. However, the efficiency and cost of solving nontrivial problems are higher for the digital computer.

II. PROBLEM STATEMENT AND SETUP

In this section a set of o.d.e.s. in the form of equation (3) is obtained to replace equation (1) by making the space variable discrete.
The specific boundary conditions

\[ u(0,t) = 0 \]
\[ \frac{\partial u(i,t)}{\partial x} = 0 \]
\[ u(x,0) = 1 \] (4)

are assumed. The problem is solved for the range \([0,1]\) in both the space and time dimensions. The space variable is divided into seven discrete stations \((u_0(t)\) through \(u_6(t)\)) as shown in Figure 1.

Figure 1.
Because of the initial and boundary conditions,

\[ u_0(t) = u(0,t) \]

and

\[ u_{n+1}(t) = u_n(t), \quad n = \frac{x}{\Delta x} \]  \hspace{1cm} (5)

Examination of equation (3), (which defines an o.d.e. for each station variable), shows that each equation involves \( u_i, u_{i-1}, \) and \( u_{i+1}. \) At the first station, \( i = 1, u_0(t) \) is known from the boundary condition. The variable \( u_2(t) \) is, however, unknown and thus a straightforward integration of the equation is not possible. A similar situation exists at the other stations.

An iterative procedure for solving equation (3) is studied in this work. The procedure assumes values, \( u_2^{(0)}(t), \) for \( u_2(t) \) and solves the o.d.e. for \( u_1(t), \) (which is denoted \( u_1^{(1)}(t) \)). The set of values for \( u_1^{(1)}(t) \) is then used to solve for \( u_2^{(1)}(t) \) etc. until \( u_2^{(1)}(t), u_3^{(1)}(t), \ldots, u_6^{(1)}(t) \) are all obtained. The quantity \( u_2^{(0)}(t) \) is then replaced by \( u_2^{(1)}(t) \) and the process is repeated using the previously computed \( u_1(t) \)'s as inputs.

The iterative procedure is defined for the case being studied by the equations

\[
\frac{diu(t)}{dt} + \frac{2}{\Delta x^2} u_1^{j}(t) = \frac{u_2^{j-1}(t) - u_0(t)}{\Delta x^2} \]  \hspace{1cm} (6)

\[
\frac{du_2^{j}(t)}{ct} + \frac{2}{\Delta x^2} u_2^{j}(t) = \frac{u_3^{j-1}(t) - u_1^{j}(t)}{\Delta x^2} \]  \hspace{1cm} (7)
\[
\frac{du_j^3(t)}{dt} + \frac{2}{\Delta x^2} u_j^1(t) = \frac{u_j^{j-1}(t) - u_j^j(t)}{\Delta x^2} \tag{8}
\]
\[
\frac{du_j^4(t)}{dt} + \frac{2}{\Delta x^2} u_j^3(t) = \frac{u_j^{j-1}(t) - u_j^j(t)}{\Delta x^2} \tag{9}
\]
\[
\frac{du_j^5(t)}{dt} + \frac{2}{\Delta x^2} u_j^4(t) = \frac{u_j^{j-1}(t) - u_j^j(t)}{\Delta x^2} \tag{10}
\]
\[
\frac{du_j^6(t)}{dt} + \frac{1}{\Delta x^2} u_j^5(t) = -\frac{u_j^j(t)}{\Delta x^2} \tag{11}
\]

Solutions to this set of equations are obtained for \( j = 1, 2, \ldots \) until convergence is obtained. Final convergence and rate of convergence are of central interest in the study.

In the computer study, a fifth order Runge-Kutta method with a step size of \( \Delta t = 0.005 \) is used to solve the o.d.e.s. In order to economize on storage the dependent variable is quantized. That is, only values of \( u_j^1(t) \) corresponding to \( t = p(\Delta t) \) \( p = (1, \ldots, m) \) are kept. Therefore, only every \( n^{th} \) value of \( u_j^1(t) \) is stored.

A value for the input is needed for every \( i\Delta t \), but a value is available at only every \( i(n\Delta t) \) position. It is, therefore, necessary to approximate the missing data.

III. ZERO ORDER HOLD RESULTS

In this section computed results on equation (3) are given
for the case of a zero order hold used to approximate the missing data. If the o.d.e.s. possess any low pass filtering properties, the zero order hold tends to delay the input in time. Therefore, forward shifts in data are also examined in an attempt to reduce any error due to this apparent delay in inputs.

Figures 2, 3, 4, 5, 6, and 7 show the solution for stations one through six respectively with the storage quantized by sixteen (i.e. every sixteenth point is stored) and the inputs shifted six places forward. Solutions were obtained (not shown in this report) for 1, 2, 5, 10, 20, 30, and 50 iterations (10 iterations means that equations (6) through (11) were solved 10 times) with storage quantization and shifted inputs as shown in Table I.

Table I.

<table>
<thead>
<tr>
<th>Quantization</th>
<th>Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>16</td>
<td>4</td>
</tr>
<tr>
<td>16</td>
<td>6</td>
</tr>
<tr>
<td>16</td>
<td>7</td>
</tr>
<tr>
<td>16</td>
<td>9</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
</tr>
</tbody>
</table>

Figures 8, 9, and 10 show the error after fifty iterations when the problem is solved using storage quantization by 16 and inputs shifted by 4, 5, and 6 places respectively. The problem error was also obtained (not shown in this report) for solutions using the quantization and shift shown in Table II.
Figure 2. Computer Solution for Station 1
Figure 3. Computer Solution for Station 2
Figure 4. Computer Solution for Station 3
Figure 5. Computer Solution for Station 4
Figure 6. Computer Solution for Station 5
Figure 7. Computer Solution for Station 6
Station 1 has least error.
Station 6 has most error.

Figure 8. Error Results for Quantization by 16, Shifted 4 Positions
Figure 9. Error Results for Quantization by 16, Shifted 5 Positions

Station 1 has least error.

Station 6 has most error.
Station 1 has least error.
Station 6 has most error.

Figure 10. Error Results for Quantization by 16, Shifted 6 Positions
The error plots clearly show that when quantization is used with no shift that the solutions converge to the same wrong answer regardless of the quantization (4, 8, 16, 32) used. When the inputs are shifted the error is improved, especially for large $t$. For small $t$ there is some improvement, but the error is very sensitive to the amount of shift. Figures 7, 8, and 9 show this clearly. The best results are obtained for a shift of just less than half the quantization size. The best results in shift for each quantization level are listed in Table III.

The results indicate that for storage quantization utilizing a zero order hold, the error can be reduced by an order of magnitude by properly shifting the stored functions forward in time. However,
the error reduction is very sensitive to the amount of shift.

Table III.

<table>
<thead>
<tr>
<th>Quantization</th>
<th>Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
</tr>
<tr>
<td>16</td>
<td>5</td>
</tr>
</tbody>
</table>

IV. POLYNOMIAL APPROXIMATION

This section discusses qualitative results of a preliminary study of a technique which uses a curve fitting scheme to approximate data at points not stored. In an effort to get a better approximation than that given by the zero order hold, a polynomial is fitted to the stored points. The needed missing data is taken as the value of the polynomial at the particular point of interest.

Two cases are studied. For the first case, data quantized by four is fitted to a first order polynomial (linear interpolation). For the other case data quantized by four is fitted to a second order polynomial (quadratic interpolation).

Comparing these results with those of section three for the zero order hold, both curve fitted cases seem to be far superior to the zero order hold with any amount of shift. As could be expected, the quadratic interpolation is better than the linear interpolation.
V. CONCLUSIONS

If in solving a p.d.e. the equation is changed to a set of coupled o.d.e.s. by making all but one of the independent variables discrete, a large amount of storage is required in most problems. One way to circumvent this difficulty to store less than the desired amount of information. This necessitates approximating the data at points not stored when this data is required in the solution of the problem.

When a zero order hold is used to approximate this data, the solution converges to the wrong answer. The error can be reduced significantly, however, if the data is shifted forward in time by a proper amount. A study of such shifts indicates that the error is very sensitive to the amount of shift and for poor choices of shift, the results may not be improved at all.

The method with most promise seems to be that of obtaining the missing data from a polynomial which is fitted to the stored values. From the very limited study reported in Section IV, it would seem that this method offers significant improvement over the zero order hold method.

VI. BIBLIOGRAPHY

DIFFERENCE EQUATIONS TO APPROXIMATE ORDINARY DIFFERENTIAL EQUATIONS

By Joseph L. Hammond, Jr.

Prepared for
George C. Marshall Space Flight Center
Huntsville, Alabama

Contract No. NAS8-2473

(Development of New Methods and Applications of Analog Computation)

24 September 1970

Engineering Experiment Station
GEORGIA INSTITUTE OF TECHNOLOGY
Atlanta, Georgia
DIFFERENCE EQUATIONS TO APPROXIMATE
ORDINARY DIFFERENTIAL EQUATIONS

by
Joseph L. Hammond, Jr.

TECHNICAL NOTE NO. 22

on

Contract No. NAS8-2473

(Development of New Methods and
Applications of Analog Computation)

For
GEORGE C. MARSHALL SPACE FLIGHT CENTER
Huntsville, Alabama
FOREWORD

The work reported in this technical note has been supported partially by Contract No. NAS8-2473 and partially by the Georgia Tech School of Electrical Engineering. The results are related to and motivated by the subject contract and also by the graduate course sequence EE 641, 642, 643 - Computer Simulation, taught by Dr. Hammond.

The author wishes to thank Dr. C. O. Alford for several discussions of certain topics covered by the technical note, and for reading a portion of the text.
## TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOREWORD</td>
<td>iii</td>
</tr>
<tr>
<td>ABSTRACT</td>
<td>v</td>
</tr>
<tr>
<td>I. INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>2. METHODS FOR OBTAINING DIFFERENCE EQUATIONS TO APPROXIMATE DIFFERENTIAL EQUATIONS</td>
<td>2</td>
</tr>
<tr>
<td>2-1 One Step Methods</td>
<td>4</td>
</tr>
<tr>
<td>2-2 Multistep Methods</td>
<td>6</td>
</tr>
<tr>
<td>2-3 A General Difference Formula</td>
<td>9</td>
</tr>
<tr>
<td>3. MEASURES OF QUALITY OF APPROXIMATION</td>
<td>14</td>
</tr>
<tr>
<td>4. POINTWISE ERROR</td>
<td>18</td>
</tr>
<tr>
<td>5. EVALUATION OF APPROXIMATIONS WITH TEST PROBLEMS</td>
<td>23</td>
</tr>
<tr>
<td>5-1 Stability</td>
<td>24</td>
</tr>
<tr>
<td>5-2 Accuracy</td>
<td>25</td>
</tr>
<tr>
<td>5-2.1 Root Shift</td>
<td>27</td>
</tr>
<tr>
<td>5-2.2 Forced Error</td>
<td>28</td>
</tr>
<tr>
<td>6. EXAMPLES</td>
<td>30</td>
</tr>
<tr>
<td>6-1 General Example</td>
<td>30</td>
</tr>
<tr>
<td>6-2 Digital Filter</td>
<td>33</td>
</tr>
<tr>
<td>6-3 Stiff Equations</td>
<td>38</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>40</td>
</tr>
</tbody>
</table>
ABSTRACT

This technical note reviews and consolidates material pertaining to one-step and multistep numerical methods with the objective of developing a tractable method for evaluating the error resulting from the use of difference equations to approximate ordinary differential equations.

A tractable approximate relation is developed between the error due to finite sampling rates and parameters describing numerical methods. The results are valid for small values of the product of the sampling period, $h$, and natural frequencies, $\lambda$, of the analog system.

The approach used approximates the analog system by isolated modes and the digital algorithm for each mode by its order, $p$, and certain constants $C, A_0, \ldots, A_p$. The results show that approximating an analog system by a digital algorithm results in a shift of each root and an error corresponding to each forcing function. The root shift is given by $-Ch^p\lambda^p$, while the error caused by each forcing function, $\gamma e^{ct}$, is given by $-Ch^p\sum_{i=0}^{p}A_i\lambda_i^{p-i}$.

The results of the study are applied to a nontrivial numerical analysis problem, a digital filter problem and to an elucidation of the problem of "stiff" equations.
DIFFERENCE EQUATIONS TO APPROXIMATE ORDINARY
DIFFERENTIAL EQUATIONS

I. INTRODUCTION

The interest in digital simulation using large scale computers and the
development of digital filtering techniques has led in recent years to exten-
sive use of digital systems described by difference equations as a means of
either solving differential equations or replacing analog systems described
by differential equations.

The underlying mathematical problem of approximating the solution of a
differential equation by the solution of a difference equation has received
considerable attention over a number of years in the area of numerical analysis
and more recently, for linear equations, in the area of digital filter theory.
Results in the former area are typified by the detailed text of Henrici [1]
and the summary articles of Benyon [2] and Giloi [3]. The latter area is dis-
cussed, for example, in the book by Gold and Rader [4]. References [1], [2],
and [4] have extensive bibliographies.

Applications in general simulation and digital filters are not such as to
make possible a single unique answer to the question of what difference equation
to use in approximating a given differential equation. Thus, the literature
contains a number of methods for obtaining difference equations which in some
sense approximate differential equations. The plurality of methods leads to
the necessity of defining measures of the quality of approximation and also to
an analysis problem in evaluating particular methods.

The purpose of the present paper is to present a new and tractable method
for studying the accuracy of stable difference equation approximations to the
solution of differential equations. Background for the new method is developed through a tutorial summary giving: (1) a unified discussion of the common methods used in numerical analysis for approximating differential equations with difference equations and (2) a discussion of the important parameters and central theorems necessary in evaluating the approximations.

The work is intended to be useful in the design and analysis of digital filters and digital simulations. For such applications errors arise from two sources, namely: (1) in approximating a differential equation by a difference equation and (2) in solving the resulting difference equation. This paper gives a detailed analysis of the former source of error but does not consider the latter. Thus, the results of this paper give a complete answer for accuracy if errors (primarily due to round off) in solving the required difference equation are negligible. In general, errors due to round off and related sources must be determined through additional analysis.

2. METHODS FOR OBTAINING DIFFERENCE EQUATIONS TO APPROXIMATE DIFFERENTIAL EQUATIONS

In this section some of the more common classes of methods for obtaining difference equations to approximate differential equations will be developed using direct procedures for approximating continuous variables with discrete variables. With this motivation, a much more general class of difference equations is then presented for analysis in later sections of the paper.

Attention will be restricted to the class of differential equations which can be expressed as

\[ y(t) = f(y,t) \]  

(1)
where \( y \) and \( f \) are vectors and \( f \) is suitably restricted so that a unique solution exists. Commonly assumed restrictions on \( f \) are given, for example, by Henrici [1]. If (1) models a physical system any input is included implicitly in \( f \). This work is concerned with obtaining difference equations whose solutions in some sense approximate the solution, \( y(t) \), of (1).

Equation (1) can be integrated to yield

\[
y(T+t) - y(T) = \int_T^{T+t} f[y(\tau), \tau] \, d\tau,
\]

which can be used as a basis for obtaining many of the desired difference equations. Let \( T \) and \( t \) be integer multiples of a basic step size \( h \) so that \( T = nh \) and \( t = kh \). Using the notation \( y(mh) = y_m \), (2) can be written as

\[
y_{n+k} - y_n = \int_{nh}^{(n+k)h} f[y(\tau), \tau] \, d\tau.
\]

A difference equation approximating (1) results when any one of several discrete approximations are used for the right-hand side of (3).

In the special case of a linear equation with constant coefficients, (1) becomes

\[
\dot{y}(t) = Ay(t) + Bu(t)
\]

where the inputs, \( u(t) \), are now shown explicitly. The equation corresponding to (3) for the linear constant coefficient case is

\[
y_{n+k} - y_n = \int_{nh}^{(n+k)h} [Ay(\tau) + Bu(\tau)] \, d\tau.
\]

In this case, however, use can be made of the closed form solution of (4) to obtain the equation
\[
(y_{n+k}) = \xi(T) y_{nh}
\]

where \( \xi(t) \) is the transition matrix of \((4)\). Note in \((6)\) that for \( u(t) = 0 \) the equation

\[
y_{n+k} = \xi(kh) y_{nh}
\]

gives \( y \) for \( t = (n+k)h \) exactly for any value of \( k \). This fact makes \((6)\) much more accurate than \((5)\) in most applications where \((6)\) is valid. If \( u(t) \) in \((6)\) is not zero, however, it is necessary in obtaining a difference equation from \((6)\) to use a discrete approximation for the integral in the same manner as in the general case.

Common methods for obtaining a discrete approximation for the right-hand side of \((3)\) are classified as one step methods and multistep methods. Direct intuitive procedures resulting in subsets of these two classes will be presented below followed by a general formulation.

2-1 One Step Methods

For one step methods, \( k \) is set equal to one and \((3)\) becomes

\[
y_{n+1} - y_n = \int_{nh}^{(n+1)h} f[y(\tau), \tau] \, d\tau.
\]

The integrand can be expanded in a Taylor series about \( t = nh \) to obtain

\[
f[y(\tau), \tau] = f_n + (\tau - nh) f'_n + \frac{(\tau - nh)^2}{2} f''_n + \ldots.
\]

where

\[
f_n = f[y(nh), nh] \quad \text{and} \quad f_n = \frac{d^i f[y(\tau), \tau]}{dt^i} \bigg|_{t=nh}
\]

Introducing \((8)\) into \((7)\) then yields
\[ y_{n+1} - y_n = hf_n + \frac{h^2}{2} f_n' + \frac{h^3}{6} f_n'' + \cdots + \frac{h^p}{p!} f_n^{(p-1)} + \cdots \]  

(9)

Truncation of the infinite series of (9) after \( p \) terms yields

\[ y_{n+1} - y_n = h[f_n + \frac{h}{2} f_n' + \cdots + \frac{h^{p-1}}{p!} f_n^{(p-1)}], \]  

(10)

a class of difference equations referred to as the Taylor expansion algorithms. The special case of \( p = 1 \) is called Euler's method. Note that \( p \) is the only parameter of this method.

In practical work, it is seldom desirable to generate derivatives of \( f \) because of the noise producing nature of the derivative operation. Thus, the algorithms of (10) with \( p > 1 \) have limited use.

The class of Runge-Kutta formulas make use of the fact that derivatives of \( f \) can be approximated in terms of \( f \) itself. The particular approximation identified with this name is given by

\[ \int_{nh}^{(n+1)h} f[y(\tau), \tau] \, d\tau \approx h \sum_{i=0}^{N} w_i f[\mu_i, \eta_i] \]  

(11)

where

\[ \mu_0 = nh, \quad \mu_i = \mu_{i-1} + \alpha_i h, \quad \alpha_i \leq 1; \ i = 1, 2, \ldots, N \]

\[ \eta_0 = y_n, \quad \eta_i = \eta_{i-1} + h \sum_{k=0}^{i-1} \beta_{ik} f(\mu_k, \eta_k); \ i = 1, 2, \ldots, N. \]

As can be noted in (11), the Runge-Kutta class of difference equations has the following parameters: \( N, N+1 \) values \( \omega_i \), \( N \) values of \( \alpha_i \) and \( \frac{N}{2}(N+1) \) values of \( \beta_{ik} \). These parameters are evaluated by expanding.
both in Taylor series and choosing the parameters so that the coefficients of \( h^r \) are equal for \( r = 1, 2, \ldots, M \). The tedious expansion operation is discussed, for example, by Henrici [1] and Ralston [5].

The results of a study of Runge-Kutta methods can be summarized as follows. A matching of coefficients can be carried only as far as \( M = N + 1 \). Practical Runge-Kutta methods employ \( N = 1, 2, 3, \) and sometimes \( 4 \). For any value of \( N \), the matching of coefficients of \( h^r \) will fix most of the parameters but at least one always remains free so that a class of algorithms exists for each value of \( N \). For example, for \( N = 1 \) there is one independent degree of freedom and for \( N = 3 \) there are two independent degrees of freedom.

The Taylor series and Runge-Kutta methods are the more important one step methods, and attention is now given to multistep methods.

2-2 Multistep Methods

Many of the common multistep methods can be developed from (3) in the following manner. The function \( f[y(t), t] \) in (3) is approximated by an interpolating polynomial and the necessary integration is then carried out to obtain a difference equation.

Given a function \( z(t) \), an interpolating polynomial \( P(t) \) of degree less than or equal to \( p \) can be found such that

\[
P(t_i) = z(t_i) ; \ i = 0, 1, \ldots, p .
\]

This polynomial can be expressed in terms of backward differences as:\(^1\)

\(^1\)See, for example, Henrici [1].
\[ P(t) = \sum_{m=0}^{p} (-1)^m \binom{n-t}{m} \nabla^m z_n \]  

(12)

where \( \nabla^m z_n \) is the \( m \)th backward difference of \( z(t) \) at \( t_n \) given by

\[ \nabla^0 z_n = z_n \]
\[ \nabla^1 z_n = z_n - z_{n-1} \]
\[ \vdots \]
\[ \nabla^j z_n = \sum_{i=0}^{j} (-1)^i \binom{j}{i} z_{n-i} , \]  

(13)

and \( \binom{u}{v} \) are the binomial coefficients given by

\[ \binom{u}{0} = 1, \quad \binom{u}{v} = \frac{u(u-1) \ldots (u-v+1)}{1 \cdot 2 \cdot \ldots \cdot v} . \]  

(14)

Note that, as in the case of (12), \( u \) in the binomial coefficient does not have to be a natural number. It is also useful to observe in (13) that \( \nabla^j z_n \) depends exclusively on the set of discrete values of \( z \) given by \( \{ z_n, z_{n-1}, \ldots, z_{n-j} \} \).

The polynomial \( P(t) \) can be introduced in (3) to approximate \( f[y(t), t] \).

Equation (3) then becomes

\[ y_{n+k} - y_n = \int_{nh}^{(n+k)h} \sum_{m=0}^{p} (-1)^m \binom{\frac{t_n+r-t}{h}}{m} \nabla^m r_{n+r} \, dr . \]  

(15)
Note that since the left-hand side of (15) involves discrete values of $y$ at the times $nh$ and $(n+k)h$, backward differences of $f$ can reasonably be taken for any time between $nh$ and $(n+k)h$ so that $r$ can range between 0 and $k$. Since only the binomial coefficients depend on $\tau$, (15) can be expressed as

$$y_{n+k} - y_n = h \sum_{m=0}^{p} \gamma_{m,k,r} \gamma^{m}_{f_{n+r}}$$

(16)

where

$$\gamma_{m,k,r} = \left(\frac{-1}{h}\right)^m \int_{nh}^{(n+k)h} \left(\frac{t_{n+r} - \tau}{h}\right)^m d\tau$$

(17)

It can be shown that since $\gamma^{m}_{f_{n+r}}$ depends exclusively on $f_{n+r}$, $\ldots$, $f_{n+r-m}$, (16) can also be expressed as

$$y_{n+k} - y_n = h \sum_{m=0}^{p} \beta_{m,k,r,p} \gamma^{m}_{f_{n+r-m}}$$

(18)

Either (16) or (18) defines a general class of difference equations depending on the parameters $p$, $k$, and $r$ so that particular values of $p$, $k$, and $r$ result in fixed values of the $\gamma_{m,k,r}$ and $\beta_{m,k,r,p}$. The $\gamma_{m,k,r}$ are given explicitly by (17) and similar relations can be derived for the $\beta_{m,k,r,p}$.

The general class of difference equations given by (18) has two major subdivisions determined by whether or not $r$ is chosen equal to $k$ so that values of $f_{n+k} = f[y_{n+k}, t_{n+k}]$ are required in the algorithm. If $r = k$, (16), or (18) then has $y_{n+k}$ on the left-hand side and also on the right-hand side in $f_{n+k}$. Such formulas are called implicit or closed formulas since in general they cannot be solved directly for $y_{n+k}$. If $r \neq k$, the resulting formulas can be solved directly and are called open or explicit formulas.
Closed formulas can be solved iteratively by the following procedure:

(0) (1)
(a) estimate \( y_{n+k} \) for use in \( f_{n+k} \), (b) solve (16) or (18) for \( y_{n+k} \), (c) use
(1) (2)
\( y_{n+k} \) in \( f_{n+k} \) to obtain \( y_{n+k} \) from (16) or (18), and (d) repeating the process.

Typically, the first estimation is accomplished with an open formula called a predictor. The closed formula is then referred to as a corrector and the complete calculation as a predictor-corrector algorithm.

Examination of (18) shows that to compute \( y_{n+k} \) requires \( y_n \) and values of \( f_i \) from \( i = n+r \) to \( i = n+r-p \). Since \( p \) is in general greater than \( r \), values of \( f \) prior to \( t_n \) are typically required. This causes a problem in starting multistep methods since (18) cannot be applied unless all required past values of \( f_i \) are given. The starting problem can be solved by computing the required values with some type of one step method since these methods are self-starting.

2-3 A General Difference Formula

A general difference formula which includes all of the formulas above as special cases is given by

\[
\sum_{i=0}^{k} \alpha_i y_{n+i} = h F\{h, n, y_{n-p}, \ldots, y_{n+k}; f\} \quad (19)
\]

where \( k \) and \( p \) are fixed and \( n = p, p+1, \ldots \). Equation (19) has parameters \( k \), \( p \), the \( \alpha_i \) and the function \( F \).

For the common multistep algorithms \( F \) is given by

\[
F = \sum_{i=0}^{k} \beta_i f_{n+i} \quad (20)
\]

so that (19) becomes
Note that (21) has parameters $k$, the $\alpha_i$, and the $\beta_i$. Table I gives the parameter values for a number of common open multistep methods. For such methods $\beta_k$ is always zero. Table II gives the parameters for common closed multistep methods. Both tables give truncation error for the algorithms, a parameter which will be defined and discussed below.

For one step methods, $\alpha_0 = -1, \alpha_1 = 1$ and all other $\alpha_k$ in (19) are zero. For the Runge-Kutta algorithms, which are the most practical one step methods, $F$ is the function defined in (11). Thus, (19) reduces to

$$
\sum_{i=0}^{k} \alpha_i y_{n+i} = h \sum_{i=0}^{k} \beta_i f_{n+i}.
$$

(21)

Typical values for the parameters $N, w_i, \theta_i$, and $\beta_{ik}$ of the Runge-Kutta algorithm are given in Table III.

The formulas preceding (19) were derived so as to insure that their solutions approximate that of (1) in some sense. Equation (19), on the other hand, is much more general and in order for it to have any utility in approximating the solutions of (1), restrictions on its parameters must be developed. The next section discusses such restrictions through defining measures of quality for the required approximations.
# Table I: Open Multistep Methods

<table>
<thead>
<tr>
<th>Method</th>
<th>$k$</th>
<th>$\alpha_0$</th>
<th>$\alpha_1$</th>
<th>$\alpha_2$</th>
<th>$\alpha_3$</th>
<th>$\alpha_4$</th>
<th>$\beta_0$</th>
<th>$\beta_1$</th>
<th>$\beta_2$</th>
<th>$\beta_3$</th>
<th>Truncation Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>Euler</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>$\frac{1}{2h^2}$</td>
</tr>
<tr>
<td>Nystrom</td>
<td>2</td>
<td>-1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>$\frac{1}{2}$</td>
<td>3</td>
<td></td>
<td></td>
<td>$\frac{1}{4}$</td>
</tr>
<tr>
<td>A-B 2 Step</td>
<td>2</td>
<td>-1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>$\frac{1}{2}$</td>
<td>$\frac{3}{2}$</td>
<td>$\frac{1}{12}$</td>
<td></td>
<td>$\frac{5}{12}$</td>
</tr>
<tr>
<td>A-B 3 Step</td>
<td>3</td>
<td>-1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>$\frac{5}{12}$</td>
<td>$\frac{16}{12}$</td>
<td>$\frac{23}{12}$</td>
<td>$\frac{25}{24}$</td>
<td>$\frac{3}{8}$</td>
</tr>
<tr>
<td>A-B 4 Step</td>
<td>4</td>
<td>-1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>$\frac{9}{24}$</td>
<td>$\frac{37}{24}$</td>
<td>$\frac{59}{24}$</td>
<td>$\frac{55}{24}$</td>
<td>$\frac{25}{72}$</td>
</tr>
<tr>
<td>Milne-Hamming</td>
<td>4</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\frac{1}{3}$</td>
<td>$\frac{4}{3}$</td>
<td>$\frac{8}{3}$</td>
<td></td>
<td>$\frac{14}{45}$</td>
</tr>
<tr>
<td>Hamming</td>
<td>5</td>
<td>1</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
<td>$\frac{8}{3}$</td>
<td>4</td>
<td>-4</td>
<td></td>
<td>$\frac{11}{72}$</td>
</tr>
<tr>
<td>Method</td>
<td>$k$</td>
<td>$\alpha_0$</td>
<td>$\alpha_1$</td>
<td>$\alpha_2$</td>
<td>$\alpha_3$</td>
<td>$\alpha_4$</td>
<td>$\alpha_5$</td>
<td>$\beta_0$</td>
<td>$\beta_1$</td>
<td>$\beta_2$</td>
<td>$\beta_3$</td>
</tr>
<tr>
<td>--------------</td>
<td>-----</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
<td>------------</td>
</tr>
<tr>
<td>Modified Euler</td>
<td>1</td>
<td>1</td>
<td>-1</td>
<td>1</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A-M 2 Step</td>
<td>2</td>
<td>-1</td>
<td>1</td>
<td>-1</td>
<td>$\frac{1}{12}$</td>
<td>$\frac{1}{24}$</td>
<td>$\frac{5}{12}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A-M 3 Step</td>
<td>3</td>
<td>-1</td>
<td>1</td>
<td>-1</td>
<td>$\frac{1}{24}$</td>
<td>$\frac{5}{24}$</td>
<td>$\frac{19}{24}$</td>
<td>$\frac{9}{24}$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Milne</td>
<td>3</td>
<td>-1</td>
<td>1</td>
<td>-1</td>
<td>$\frac{1}{3}$</td>
<td>$\frac{4}{3}$</td>
<td>$\frac{1}{3}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hamming</td>
<td>3</td>
<td>$\frac{1}{6}$</td>
<td>-1</td>
<td>1</td>
<td>$\frac{3}{8}$</td>
<td>$\frac{6}{8}$</td>
<td>$\frac{3}{8}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A-M 4 Step</td>
<td>4</td>
<td>-1</td>
<td>1</td>
<td>-1</td>
<td>$\frac{19}{720}$</td>
<td>$\frac{106}{720}$</td>
<td>$\frac{264}{720}$</td>
<td>$\frac{646}{720}$</td>
<td>$\frac{251}{720}$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hamming</td>
<td>4</td>
<td>$\frac{9}{121}$</td>
<td>$\frac{14}{121}$</td>
<td>$\frac{126}{121}$</td>
<td>$\frac{24}{121}$</td>
<td>$\frac{54}{121}$</td>
<td>$\frac{108}{121}$</td>
<td>$\frac{42}{121}$</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
**TABLE III: RUNGE-KUTTA ONE STEP METHODS**

<table>
<thead>
<tr>
<th>N</th>
<th>$\omega_0$</th>
<th>$\omega_1$</th>
<th>$\omega_2$</th>
<th>$\omega_3$</th>
<th>$\theta_1$</th>
<th>$\theta_2$</th>
<th>$\theta_3$</th>
<th>$\beta_{10}$</th>
<th>$\beta_{20}$</th>
<th>$\beta_{21}$</th>
<th>$\beta_{30}$</th>
<th>$\beta_{31}$</th>
<th>$\beta_{32}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>$\frac{1}{6}$</td>
<td>$\frac{1}{3}$</td>
<td>$\frac{1}{3}$</td>
<td>$\frac{1}{6}$</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>0</td>
<td>$\frac{1}{2}$</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Principle Error Function</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$\phi(t, y) = \frac{1}{2880} f - \frac{1}{576} f_y f + \frac{1}{288} (f_y^2 - f_{ty} - f_{yy} f) f$

$+ \frac{1}{192} (2f_{ty} f_y + 3f_{yy} f_y f - 2f_y^3 + f_{yy} f_t) f$

<table>
<thead>
<tr>
<th>N</th>
<th>$\omega_0$</th>
<th>$\omega_1$</th>
<th>$\omega_2$</th>
<th>$\omega_3$</th>
<th>$\theta_1$</th>
<th>$\theta_2$</th>
<th>$\theta_3$</th>
<th>$\beta_{10}$</th>
<th>$\beta_{20}$</th>
<th>$\beta_{21}$</th>
<th>$\beta_{30}$</th>
<th>$\beta_{31}$</th>
<th>$\beta_{32}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>$\frac{2}{9}$</td>
<td>$\frac{1}{3}$</td>
<td>$\frac{4}{9}$</td>
<td>$\frac{1}{6}$</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{3}{4}$</td>
<td>$\frac{1}{2}$</td>
<td>0</td>
<td>$\frac{3}{4}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>$\frac{1}{2}$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

$\phi(t, y) = (\frac{1}{8\alpha} - \frac{1}{6}) f - \frac{1}{8\alpha} f_y f$
3. MEASURES OF QUALITY OF APPROXIMATION

Attention is given in this section to the problem of restricting the parameters of (19) in such a way as to insure that its solution, \( y_n \), for \( n = p, p+1, \ldots \), approximates in some sense the solution, \( y(t) \), of (1) at \( t = nh \) for the same values of \( n \). To avoid confusion, in the remainder of the paper the solution of (1) will be denoted as \( Y(t) \).

Two a priori restrictions are placed on \( F \), namely: (1) \( F \) is restricted so that small changes in its parameters result in small changes in \( F \), a condition similar to that required of \( f \), and (2) \( F \) is restricted so that \( F[h,n, y_{n-p}, \ldots, y_{n+k}; 0] = 0 \).

Equation (19) is an algorithm for determining \( y_{n+1} \) given \( y_n \) and other required data. Thus, much of the evaluation of (19) can be based on a consideration of the typical step in the iterative calculation of \( y_n \). To consider the effect of a single step in the calculation, \( y_n \) and other required data are assumed to be exact at the beginning of the typical step and (19) is used to calculate \( y_{n+1} \). The quantity \( y_{n+1} \). Assuming the errors occurring in only one step, is denoted\(^2 \hat{y}_{n+1} \).

A quantity of central importance in studying (19) is local truncation error, \( T_{n+1} \), defined by

\[
T_{n+1} = \hat{y}_{n+1} - y[(n+1)h] \tag{23}
\]

\(^2\)Note that in computing \( y_{n+1} \), true values are assumed for all \( y_n, i=0,1, \ldots, k-1 \) and also for any \( y_{n+k} \) required as an argument of \( F \).
Equations (19) and (23) can be used to express $T_{n+k}$ as

$$T_{n+k} = hF[h,n,Y[(n-p)h], \ldots, Y[(n+k)h];f] - \sum_{i=0}^{k} \alpha_i Y[(n+i)h]$$

(24)

where the subscript $n+k$ is used because the general algorithm evaluates $y_{n+k}$ using past data.

For a given algorithm $T_{n+k}$ is conveniently computed by expanding

$$h \sum_{i=0}^{k} \alpha_i Y(t)$$

and $hF[h,n,Y[(n-p)h], \ldots, Y[(n+k)h];f]$ in Taylor series about a convenient point such as $t = nh$ and subtracting the series term by term to obtain

$$T_{n+k} = \phi_0 + h \phi_1 + \frac{h^2}{2} \phi_2 + \frac{h^3}{6} \phi_3 + \ldots$$

(25)

where the $\phi_i$ depend on both $F$ and $Y(t)$. In any particular case the first $p+1$ functions $\phi_i$, $i = 0, 1, 2, \ldots, p$ will be zero and the method is termed to be of order $p$.

It will be useful below to note that for many methods $T_{n+k}$ can be expressed as

$$T_{n+k} = h^{p+1} \phi(u,Y) + O(h^{p+2})$$

(26)

where $u$ is a point "near" $(n+k-1)h$ and $O(h^{p+2})$ indicates a group of terms, which approach zero as $h$ approaches zero at least as rapidly as $h^{p+2}$.

The function $\phi(u,Y)$ is called the principle error function by Henrici [1] in discussions of one step methods. For many such methods $\phi(u,Y)$ for a scalar problem equation is given by
\( \phi(u,Y) = a_{p+1}(u,Y) Y(u) + a_p(u,Y) Y(u) + \ldots + a_2(u,Y) Y(u) \) \tag{27}

where \( p \) is the order of the method and the \( a_i \) are in general functions of \( u \) and \( Y \) which depended on the given problem equation. The principle error functions for several Runge-Kutta Algorithms applied to scalar problem equations are given in Table III.

For most multistep methods

\[
\phi(u,Y) = C Y^{(p+1)}(u)
\] \tag{28}

Thus, for such methods \( \phi(u,Y) \) can be obtained by dividing the truncation error (as given for several methods in Tables I and II) by \( h^{p+1} \).

If a method is of at least first order, it is said to be consistent. Note that for a consistent method the local truncation error is at least \( O(h^2) \).

The cumulative effect of truncation error in all of the steps from \( n = p \) to \( n = j \) is a total pointwise error, \( e_j \), at \( t = jh \). This quantity, given by

\[
e_j = y_j - Y(jh),
\] \tag{29}

differs from \( e_j \) in that \( e_j \) includes the total effect of errors in many steps, rather than in a single step.

Using pointwise error it is possible to define convergence of \( y_n \) to \( Y(nh) \) for a given method as follows.

If for any \( f(y,t) \)

\[
\max_n |e_n| \to 0 \text{ as } h \to 0,
\]
then the method is convergent.

Using (19) to express \( y_n \) and (23) and (24) to express \( Y(nh) \), a difference equation can be obtained for \( e_n \) from (29) as

\[
\sum_{j=0}^{k} \alpha_{n+j} = h[F[h,n,y_{n-p},...,y_{n+k};f] - F[h,n,Y((n-p)h,..., Y[(n+k)h];f] + T_{n+k}.
\]

Several types of stability can be discussed for algorithms in the class (19). A type which is tractable for the general case can be termed limiting stability in the sense that it is applied for either \( h \) or \( f \) assumed to be identically zero. With this assumption, (19) becomes

\[
\sum_{j=0}^{k} \alpha_j y_{n-j} = 0.
\]

The solution to this difference equations has the form \( \sum_{i=1}^{k} \alpha_i \beta_i^n \) where the \( \beta_i \) are the roots of the polynomial

\[
P(\beta) = \sum_{j=0}^{k} \alpha_j \beta^j = 0.
\]

Thus, (31) is stable if and only if the \( \beta_i \) satisfy the conditions

\[
|\beta_i| \leq 1
\]

with the additional requirement that if \( |\beta_j| = 1 \) then \( \beta_j \) must be a simple root. This condition is referred to as the root condition. The root condition, which guarantees limiting stability is necessary but not sufficient for more general types of stability.
The following theorem which relates the root condition to convergence of \( y_n \) to \( Y(nh) \) can be proved: \(^3\) If (19) is consistent, then it is convergent if and only if the root condition is satisfied. Thus, if the root condition is satisfied and (19) is consistent, some small enough \( h \) can always be found so that \( y_n \) is a good approximation to \( Y(nh) \). The stability and pointwise error of (19) for fixed nonzero values of \( h \) will be investigated below but much more restrictive conditions will have to be assumed.

4. POINTWISE ERROR

The purpose of this section is to derive a tractable differential equation for pointwise error starting from the difference equation (30) assuming that the method is stable and that \( T_{n+k} \) can be approximated by \( h^{p+1} f(u,Y) \), (see (26)). Consider the function \( F[h,n,Y[(n-p)h], \ldots, Y[(n+k)h]; f] \).

Using (29), the true value \( Y(nh) \) can be expressed as

\[
Y(nh) = y_n - e_n,
\]

and \( F \) can then be expanded in a Taylor series about the points \( y_n \). Assuming that \( e_n \) is small it is reasonable to truncate the Taylor series to two terms and the result is

\[
F[h,n,Y[(n-p)h], \ldots, Y[(n+k)h]; f] = \ 
\]

\[
F[h,n,Y_{n-p}, \ldots, Y_{n+k}; f] - \left[ \frac{\partial F}{\partial y_{n-p}} e_{n-p} + \ldots + \frac{\partial F}{\partial y_{n+k}} e_{n+k} \right].
\]

Using (35) in (30) then gives the difference equation

\[
\sum_{j=0}^{k} \alpha_j e_{n+j} = h \left[ \frac{\partial F}{\partial y_{n-p}} e_{n-p} + \frac{\partial F}{\partial y_{n+k}} e_{n+k} \right] + T_{n+k},
\]

\(^3\) See for example Isaacson and Keller [6].
which applies for a general $F$.

The approach in approximating the solution $e_n$ of (36) by the solution of a differential equation is to find, to a given order of accuracy, a differential equation which results in the difference equation (36) through application of the algorithm of the method being investigated. The procedure will be carried out for the two special cases of one-step and multistep methods.

For one-step methods (19) becomes

$$y_{n+1} - y_n = hF\{h,n,y_n;f\} \tag{37}$$

and the equation analogous to (36) for the specific $F$ of (37) becomes

$$e_{n+1} - e_n = h \left[ \frac{\partial F}{\partial y_n} e_n + h^p \phi(\mu,Y) \right] \tag{38}$$

with $nh \leq \mu \leq (n+1)h$.

Now by defining

$$\bar{e}_n = e_n h^{-p} \tag{39}$$

(39) becomes

$$\bar{e}_{n+1} - \bar{e}_n = h \left[ \frac{\partial F}{\partial y_n} \bar{e}_n + \phi(\mu,Y) \right], \tag{40}$$

But Euler's method applied to $z = g(z;t)$ yields the difference equation

$$z_{n+1} - z_n = h g(z_n,nh) \tag{41}$$

Thus, it can be asserted by choosing

$$g(z_n,nh) = \frac{\partial F}{\partial y_n} \bar{e}_n + \phi(\mu,Y)$$
that the differential equation

\[
\dot{e}(t) = \frac{\partial f}{\partial Y} e(t) + \phi(\mu, Y)
\]  

(42)

has a solution \(e(t)\) which approximates \(\tilde{e}_n\) to within an error \(O(h)\). In obtaining (42) use is made of the fact that \(F\) approximates \(f\) within errors which are \(O(h)\). Equation (42) written in terms of \(e(t)\) rather \(e(t)\) becomes

\[
\dot{e}(t) = \frac{\partial f}{\partial Y} e(t) + h^p \phi(\mu, Y)
\]  

(43)

and the errors are now \(O(h^{p+1})\).

A similar line of reasoning is used to give the result for multistep methods. For such methods

\[
F = \frac{1}{2} \sum_{j=0}^{k-1} \beta_j f_{n+j}
\]  

(44)

and

\[
\frac{\partial F}{\partial Y_{n+j}} = \beta_j \frac{\partial f_{n+j}}{\partial Y_{n+j}}
\]

Thus, the equation corresponding to (36) becomes

\[
\frac{h}{\sum_{j=0}^{k} \alpha_j} e_{n+j} = h \left[ \sum_{j=0}^{k} \beta_j \frac{\partial f_{n+j}}{\partial Y_{n+j}} e_{n+j} + h^p \phi(\mu, Y) \right].
\]  

(45)

A multistep algorithm using (44) can be expressed as

\[
z_{n+1} - z_n = h \sum_{j=0}^{k} \beta_j z_{n+j}.
\]  

(46)
where $z = f(z,t)$ has been assumed. Equation (46) shows that within an error which is $O(h)$ the following equation holds

$$
\frac{z_{n+1} - z_n}{h} \sim z_n = \sum_{j=0}^{k} \beta_j z_{n+j}.
$$

If $\phi(\mu,Y)$ is identified with $z(t)$ in (47), it follows that within an error which is $O(h)$, $\phi(\mu,Y)$ can be expressed as

$$
\phi(\mu,Y) = \sum_{j=0}^{k} \beta_j \phi(\mu + jh,Y).
$$

Hence (49) can be written as

$$
\sum_{j=0}^{k} \beta_j \bar{e}_{n+j} = h \sum_{j=0}^{k} \beta_j \left[ \frac{\partial f}{\partial Y} \bar{e}_{n+j} + \phi(\mu + jh,Y) \right].
$$

where $\bar{e}_n$ is defined as in (39). But the differential equation

$$
\bar{e}(t) = \frac{\partial f}{\partial Y} \bar{e}(t) + \phi(\mu,Y)
$$

results in the difference equation (49) using the multistep methods. Thus, when (50) is transformed to use $e(t)$ instead of $\bar{e}(t)$, an equation identical to (43) results.

If a method with pointwise error $e_n$ has an order $p$, then $e(nh)$ given by (43) satisfies

$$
e_n = e(nh) + O(h^{p+1}),
$$

and the error in approximating $e_n$ by $e(nh)$ is of an order in $h$ one higher than the order of the method.
Since (51) approximates pointwise error at all time, it is reasonable to consider a variable $y^*(t)$, defined so that $y^*(nh) = y_n$, which represents the solution of the approximating difference equation at all times. In analogy to (32), $Y(t)$, $y^*(t)$, and $e(t)$ are related by

$$Y(t) = y^*(t) - e(t). \quad (52)$$

It is useful to note that the function $\phi(\mu, Y)$ of (43) can be expanded in a Taylor series about a point $(t, y^*)$ to obtain

$$\phi(\mu, Y) = \phi(t, y^*) - e \frac{\partial \phi}{\partial Y} + (\mu - t) \frac{\partial^2 \phi}{\partial \mu^2} + \ldots. \quad (53)$$

If $\frac{\partial \phi}{\partial Y}$ and $\frac{\partial^2 \phi}{\partial \mu^2}$ are bounded, as is usually the case, then for $e$ at least $O(h)$ and $\mu - t \leq h$

$$\phi(\mu, Y) = \phi(t, y^*) + O(h). \quad (54)$$

Thus, to the accuracy being used, (43) can be expressed as

$$e(t) = \frac{\partial f}{\partial Y} e(t) + hP \phi(t, y^*). \quad (55)$$

Using (55), (52) and the equation resulting from differentiating both sides of (52), it is a simple matter to obtain the equation

$$\dot{y}^*(t) = \frac{\partial f}{\partial Y} y^* + Y(t) - \frac{\partial f}{\partial Y} Y(t) + hP \phi(t, y^*) \quad (56)$$

the solution of which closely approximates the solution of a difference equation obtained by the one step or multistep methods. For the test problem to be discussed in the next section, (56) gives a tractable expression for
evaluating accuracy.

5. EVALUATION OF APPROXIMATIONS WITH TEST PROBLEMS

The discussion up to this point pertains to the general class of differential equations given by (1). To proceed further with the development of a tractable procedure for evaluating difference equation approximations to differential equations, it is necessary to make simplifying assumptions. The approach which has been chosen is that of using a tractable linear constant coefficient test problem to replace the given differential equation in the evaluation of difference equation approximations. Such an approach, which is not uncommon in engineering analysis, can be justified in several ways. For example, the linear constant coefficient equation can be chosen to represent the incremental behavior of a general system described by (1) about some average trajectory.

Lomax [7] gives an extended discussion justifying the linear test problem approach. In addition, he shows that errors associated with each natural frequency of a coupled system of linear constant coefficient equations and with each forcing function can be treated separately so that it is sufficient for the test problem to contain only a single (possibly complex) eigenvalue and a single forcing function. Thus, an adequate test problem is given by

\[ \dot{y}(t) = \lambda y(t) + \gamma e^{\omega t} \]  \hspace{1cm} (57)

where \( y(t) \) is a scalar and \( \lambda \) and \( \omega \) can be either real or complex.

When using (57) to study a difference equation approximation for a specified problem, \( \lambda \) and \( \omega \) are chosen to correspond to extreme or "worst case"
natural frequencies or forcing function components in a manner which will be illustrated below.

Two considerations, namely stability and accuracy are necessary in evaluating an approximation and these will now be treated separately using the test problem.

5-1 Stability

Limiting stability as defined above can be studied without recourse to a test problem and it can be noted that all one step methods are stable in this sense. In practical work, however, the question of stability for non zero values of \( h \) must be faced and one approach is to study such a situation using the test problem of (57). When studying stability a forcing function is not required and thus \( \gamma \) can be set equal to zero.

Generally speaking, stability can be studied by formulating the difference equation resulting from a particular algorithm applied to the test problem of (57) with \( \gamma = 0 \). Due to the nature of the test problem, the difference equation thus obtained is linear with constant coefficients. Since the test problem has a single root, one step methods produce a difference equation with a single root. Multistep methods on the other hand produce extraneous roots. In any case, the solution of the difference equation is the sum of terms of the form \((r_1^n)\), where the \( r_1 \) are the roots of the difference equation. It follows that the condition \(|r_1| \leq 1\), (with all roots for which \( r_1 = 1 \) simple), insures stability. Note that the \( r_1 \) depend on \( \lambda \) and \( h \) so that boundaries on \( h\lambda \), which insure that \(|r_1| \leq 1\), can be obtained. One way to present these boundaries is to plot the real versus the
imaginary part of $h\lambda$ for the condition that $\left| r_{\text{max}} \right| = 1$. Curves of stability boundaries for a number of methods are given by Benyon. 4

5-2 Accuracy

The test problem given by (57) can also be used to investigate the accuracy of a given algorithm. After the algorithm has been shown to be stable, the differential equation (56) applies and gives a good approximation to the solution that is obtained by solving the difference equations corresponding to the algorithm being investigated. Thus, to study accuracy, (56) is formulated for the test problem by evaluating $\frac{\partial f}{\partial Y}$ and $\phi(t, y^*)$ for this equation and algorithm being investigated. The quantity $\partial Y$ is obtained easily as

$$\frac{\partial f}{\partial Y} = \lambda. \quad (58)$$

The principle error function $\phi(t, y^*)$ is obtained as follows. For one step methods applied to the test problem, the $a_i (\mu, Y)$ in (27) are found to be independent of $\mu$ and $Y$ and $\phi(t, Y)$ can be expressed as

$$\phi(t, Y) = -[C_1 \lambda^{D+1} Y(t) + C_2 Y e^{\alpha t} \sum_{i=0}^{D} a_i \lambda^i u^{D-i}] \quad (59)$$

where for one step methods $C_1 = C_2 = C$

For example in considering the Runge-Kutta methods for $N = 1$ and $3$ as given in Table III, the only nonzero partial derivative of $f = \lambda Y + Ye^{\alpha t}$ is $\frac{\partial f}{\partial Y}$ which is equal to $\lambda$. Thus, for $N = 3$ the $a_i$ of (27) become $a_5 = \frac{1}{2880}$, $a_4 = \frac{-\lambda}{576}$, $a_3 = \frac{\lambda^2}{288}$ and $a_2 = \frac{-2 \lambda^3}{192}$. For $N = 1$

the values are $a_3 = \frac{1}{8\alpha} - \frac{1}{6}$, $a_2 = \frac{-\lambda}{8\alpha}$. In each case

$$C_2 = \sum_{i=2}^{p+1} a_i$$

$$A_i = \frac{1}{C_2} \sum_{i=p+1-i}^{p+1} a_i ; i = 0, 1, \ldots, p-1$$

$$A_p = 1$$

For the multistep methods, $\phi(t,Y)$ as given by (28), becomes equal to the expression of (59) with $C_1 = C_2 = C$ and the $A_i = 1$, $i = 0, 1, \ldots, p$.

There are methods with digital filter equations being the prime example, for which $C_1$ and $C_2$ are not equal. In fact, for the digital filter case $C_1 = 0$. For all the algorithms investigated, however, $\phi(t,Y)$ can be evaluated and expressed in the form of (59).

When (58) and (59), with $y^*$ replacing $Y$, are introduced in (56) the equation becomes

$$y^*(t) = [\lambda - C_1 h^p \lambda^{p+1}] y^*(t) + \gamma e^{\omega t} - C_2 h^p \gamma e^{\omega t} \sum_{i=0}^{p} A_i \lambda_i^{p-i}$$

Note that the solution of (61) approximates the solution of a difference equation for the test problem obtained using an algorithm described by an order $p$, constants $C_1$ and $C_2$ and $A_i = 0, 1, \ldots, p$.

If (61) is compared to (57), it can be seen that using a difference equation to approximate a differential equation has produced two types of error, namely:

1. The root $\lambda$ has been shifted to the new position $\lambda - C_1 h^p \lambda^{p+1}$ and
(2) The forcing function has been perturbed from $\gamma e^{\alpha t}$ to

$$\gamma e^{\alpha t} \left[ 1 - C_1 h^{p} \sum_{i=0}^{p} A_i \lambda^i \right].$$

These effects will be considered separately by obtaining the "root shift" with no forcing function and the "forced error" assuming no root shift.

5-2.1 Root Shift: If fractional root shift, $E_r$, is defined in an obvious fashion there results

$$E_r = \frac{\left( \lambda - C_1 h^{p} \lambda^{p+1} \right) - \lambda}{\lambda} = -C_1 h^{p} \lambda^{p}$$

where it should be recalled that both $E_r$ and $\lambda$ can be complex. This equation can be expressed in the following equivalent and useful ways:

$$h^{1/p} \lambda = \left( \frac{|E_r|}{|C_1|} \right)$$

$$\frac{1}{h|\lambda|} = \left( \frac{|C_1|}{|E_r|} \right) 1/p$$

$$\ln|E_r| = p \ln h|\lambda| + \ln|C_1|.$$  

Note that the units of $\frac{1}{h|\lambda|}$ are steps/radian.

Equation (68) shows that $|E_r|$ increases with $h$. Therefore, if $|E_r|$ is regarded as maximum permissible root shift, $h|\lambda|$ must satisfy

$$h|\lambda| \leq \left( \frac{|E_r|}{|C_1|} \right) 1/p$$

(66)
and similar expressions can be obtained to correspond to (64) and (65).

Equation (65) plots on log-log paper as a straight line with slope $p$ intercepting $h|\lambda| = 1$ at a point equal to $|C_1|$. The normalized quantity $|E_r|/C_1$ is plotted versus $h|\lambda|$ and $2\pi/h|\lambda|$ for various values of $p$ in Figure 1.

5-2.2 Forced Error: A fractional forced error, $E_f$, is defined as

$$E_f = \frac{Y^*(t) - Y(t)}{Y(t)}.$$  \hspace{1cm} (67)

Since (57), and (61), whose solutions are $Y(t)$ and $y^*(t)$, are linear, it follows that

$$y^*(t) = [1 - C_2 h^p \sum_{i=0}^{p} A^i \omega^{-i}] Y(t).$$  \hspace{1cm} (68)

(Recall that root shift is being ignored in considering forced error so that the homogeneous parts of (57) and (61) become identical.) Substituting (68) into (67) yields

$$E_f = -C_2 h^p \sum_{i=0}^{p} A_1 \omega^{-i}.$$  \hspace{1cm} (69)

Note that $E_f$ is independent of time since $y^*(t)$ is a constant times $Y(t)$.

It is reasonable to consider three cases which result in the following tractable approximations:

Case 1: $|\lambda| < |\omega|$

$$E_f \approx -A_o C_2 h^p \omega^p.$$  \hspace{1cm} (70)
Figure 1. Normalized error versus $|\lambda|$ and $2\pi/h|\lambda|$ for various values of $p$. 
Case 2: \(|\lambda| >> |\omega|\)

\[ E_f \approx - A P C_2 h P \lambda^P \]  \((71)\)

Case 3: \(|\lambda| \sim |\omega|\)

\[ E_f \approx - C_2 h P \lambda^P \sum_{i=0}^{P} A_i \]  \((72)\)

In all three cases, \(E_f\) has the same form as \(E_r\) given by (62). In cases 2 and 3, \(E_f\) is proportional to \(h P \lambda^P\) just as is \(E_r\) but with different constants in the two cases.

In case 1, \(E_f\) is proportional to \(h P \omega^P\) which has the same form as \(E_r\) with \(\omega\) replacing \(\lambda\). Thus the expressions and curves for \(E_r\) can also be used for \(E_f\) with appropriate changes in the constants and with \(\lambda\) replaced by \(\omega\) for case 1.

Applications of the accuracy equations in studying difference equation approximations will be given in the next section.

6. EXAMPLES

6-1 General Example:

To illustrate the techniques presented above, consider the problem of choosing a numerical algorithm for digital simulation of an aerospace vehicle using the simplified model discussed by Ryan et al [8]. This model can be represented as a forth order linear differential equation with a time varying coefficient which is proportional to vehicle attitude. The differential equation has a random forcing function (caused by wind perturbation) whose power spectral band width is also proportional to attitude.

In order to study various algorithms, the problem can be regarded as quasi-stationary with roots whose position depends on the time varying altitude.
The ranges of these roots and of the power spectral band width of the forcing function are tabulated in Table IV.

**Table IV.** Frequencies associated with the Space Vehicle Problem.

<table>
<thead>
<tr>
<th>Root Type</th>
<th>Range of Root Location or Maximum Bandwidth</th>
</tr>
</thead>
<tbody>
<tr>
<td>Real Roots</td>
<td>$\lambda_1 = 0 ; ; ; -0.05 \leq \lambda_2 \leq 0$</td>
</tr>
<tr>
<td>Complex Roots</td>
<td>$\lambda_{3,4} = \omega_c \sqrt{\pm127^\circ} ; ; ; 0.5 \leq \omega_c \leq 2.0$</td>
</tr>
</tbody>
</table>

In considering various algorithms, it is clear that stability is required for successful approximation. For purposes of illustration let it also be required that fractional root shift, $E_r$, and fractional forced error $E_f$ be limited in magnitude to 1%.

Use is now made of the normalized curves of Figure 1 and Benyon's [2] results on stability to obtain limits on the step size for stable and accurate operation using representative algorithms. Limits on $h|\lambda|$ for stability are tabulated for a test problem with a real root and a root location along the 127 degree line, in the complex plane for five representative methods in Table V. The table also gives the limit on $h|\lambda|$ for $|E_r| \leq 1\%$ and the value of $h\Omega$ for $|E_f| = 1\%$ under the three assumptions $|\lambda| < \Omega$, $|\lambda| \sim \Omega$ and $|\lambda| > \Omega$ where $\Omega$ is the single forcing frequency and $\lambda$ is the single natural frequency of a test problem.

It can be noted from Table V that for the one-step methods, (Euler, RK-2, RK-4), the limits on $h|\lambda|$ (or $h\Omega$) for stability tend to be significantly
Table V. Limits on $h|\lambda|$ or $h \Omega$ for various conditions.

| Algorithm | Real Roots | $\lambda = \lambda_c / 127^\circ$ | $|E_f| \leq 1\%$ | $|\lambda| \ll \Omega$ | $|\lambda| \simeq \Omega$ | $|\lambda| \gg \Omega$ |
|-----------|------------|----------------------------------|-----------------|-----------------|-----------------|-----------------|
| Euler     | 2.0        | 1                                | .02             | .023            | .01             | .023            |
| AB-4      | .3         | .3                               | .4              | .41             | .3              | .41             |
| AM-4      | 1.8        | .9                               | .9              | .9              | .6              | .9              |
| RK-2      | $\alpha = 1/2$ | 2.0                  | .25             | .3              | .15             | .2              |
|           | $\alpha = 1$ | 2.1                               |                 |                 |                 |                 |
|           | $\alpha = 3/4$ | 2.0                               |                 |                 |                 |                 |
| RK-4      | 2.7        | 2.6                              | 1.05            | 2.3             | .9              | 1.05            |
larger than the limits imposed by $E_r$ or $E_f$. On the other hand for the multistep methods, (AB-4, AM-4), the limits imposed by stability tend to be approximately the same as those imposed by $E_r$ and $E_f$.

For the space vehicle problem, the complex roots will be the limiting factor due to the relative sizes of the four roots.

The forcing function for the problem covers a range of frequencies such that the results for the single test frequency, $\Omega$, approximately equal to the magnitude of the complex root would seem to be the best guide to accuracy of the forced problem. Thus setting $\Omega = B$ and $|\lambda| = \omega_c$ yields the numbers given in Table VI which lists bounds on $h$ caused by $\omega_c$ and by the forcing function. The table also gives final bounds on $h$ assuming the two conditions $\omega_c = 2$, $B = 6$, and $\omega_c = B = 2$. The table shows that the RK-4 algorithm has the largest limiting step size for both of the conditions cited.

6-2 Digital Filter

A recursive digital filter is an algorithm, usually implemented in real time, which can be expressed as a difference equation. One important digital filter design technique involves tailoring the difference equation defining the digital filter so that its response closely approximates the solution of a differential equation defining the response of an analog filter, (see for example Rader and Gold [9]). Using this point of view, the general results of this paper apply and will give, for example, the accuracy with which the digital filter response approximates the analog filter response in the region of excitation for which the product of step size and frequency is small. The latter point should be noted since digital filter responses are
Table VI. Limits on $h$ for Various Conditions.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Limit resulting from $\omega_c$</th>
<th>Limit resulting from forcing function</th>
<th>Final Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>limit on $h$</td>
<td>Cause</td>
<td>$\omega_c = 2$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$B = 6$</td>
</tr>
<tr>
<td>Euler</td>
<td>$\frac{0.02}{\omega_c}$</td>
<td>$E_r$</td>
<td>$\frac{0.01}{B}$</td>
</tr>
<tr>
<td>AB-4</td>
<td>$\frac{0.3}{\omega_c}$</td>
<td>Stability</td>
<td>$\frac{0.3}{B}$</td>
</tr>
<tr>
<td>AM-4</td>
<td>$\frac{0.9}{\omega_c}$</td>
<td>Stability or $E_r$</td>
<td>$\frac{0.6}{B}$</td>
</tr>
<tr>
<td>RK-2</td>
<td>$\frac{0.2}{\omega_c}$ $\alpha = 1/2$</td>
<td>$E_r$</td>
<td>$\frac{0.15}{B}$</td>
</tr>
<tr>
<td></td>
<td>$\frac{0.2}{\omega_c}$ $\alpha = 1$</td>
<td>$E_r$</td>
<td>$\frac{0.15}{B}$</td>
</tr>
<tr>
<td></td>
<td>$\frac{0.2}{\omega_c}$ $\alpha = 3/4$</td>
<td>$E_r$</td>
<td>$\frac{0.15}{B}$</td>
</tr>
<tr>
<td>RK-4</td>
<td>$\frac{1.05}{\omega_c}$</td>
<td>$E_r$</td>
<td>$\frac{0.9}{B}$</td>
</tr>
</tbody>
</table>
sometimes examined in the frequency region where this restriction
does not apply.

The considerations involved in obtaining accuracy for digital
filters can be adequately illustrated by considering the realization
of a single complex pole pair. An impulse invariant digital filter
corresponding to an analog filter satisfying the differential equation

\[ \ddot{y} + 2a \dot{y} + (a^2 + b^2) y = a \chi(t) + \dot{\chi}(t) \quad (73) \]
satisfies the difference equation \[ (9) \]

\[ y_{n+2} = 2e^{-ah} \cos bh y_{n+1} - e^{-2ah} y_n + h[\chi_{n+2} - e^{-at} \cos bh \chi_{n+1}] \quad (74) \]

Equation (73) corresponds to the transfer function

\[ T(s) = \frac{s+a}{(s+a)^2 + b^2} \]

which has a pair of complex poles

\[ \lambda_{1,2} = -a \pm jb. \]

It is convenient in using the results of section 5 to consider a
first order equation with \( y(t) \) complex, instead of (73) in which \( y(t) \)
is real. It is straight forward to show that the first order equation

\[ \ldots \]

\[ ^5 \text{Digital filter designs are typically normalized to } h = 1. \text{ This does not seem approximate here and an } h \text{ multiplying the braketed term in (74) is added to Rader and Gold's equation.} \]
\[ y^*(t) = \lambda y(t) + \chi(t), \quad (75) \]

with \( y(t) \) and \( \chi(t) \) complex and \( \lambda = -a+jb \)

has the property that \( \Re y(t) \) in (75) satisfies (73) if the excitation is \( \Re \chi(t) \). The difference equation corresponding to (74) can similarly be shown to be

\[ y_n = K y_{n-1} + h \chi_n \quad (76) \]

where \( y_n \) is complex and \( K = e^{\lambda h} \).

The techniques of section 5 will now be used to determine the accuracy of the solution of (76) as an approximation to the solution of (75). Equation (75) can be identified as being identical to the test problem of (57) if the excitation is chosen to be \( ye^{\omega t} \). Hence, the equations for \( E_r \) and \( E_f \) given in (62) - (72) can be used for accuracy by identifying \( C_1, C_2, \) and \( p \) for the algorithm of (76).

The principle error function \( \varphi(t,\mu) \), which is given by (59) for differential equations in the form of (75) or (57), gives implicitly the required constants \( C_1, C_2 \) and \( p \).

The principle error function for (76) is evaluated as follows. The chain of approximations

\[ \varphi(t,\mu) \sim \varphi(\mu,\nu) \sim -h^{(p+1)} T_{n+1}, \quad (77) \]

where \((n+1)h \approx \mu \geq nh\), has been established and discussed above. Thus, what is required is \( T_{n+1} \) given by
\[ T_{n+1} = \hat{Y}_{n+1} - Y[(n+1)h] \tag{78} \]

where \( \hat{Y}_{n+1} \) and \( Y[(n+1)h] \) are defined with (23) above.

From (78), \( \hat{Y}_{n+1} \) is expressed as

\[ \hat{Y}_{n+1} = e^{-i(jb)h} Y(nh) + hX[(n+1)h] \tag{79} \]

The true solution of (79) at \((n+1)h\), \( Y[(n+1)h] \), can be expressed in terms of the solution at \( nh \) using the well known analytic solution for linear constant coefficient equations. The result is

\[ Y[(n+1)h] = e^{-i(jb)h} Y(nh) + \int_{nh}^{(n+1)h} e^{-(a+jb)[(n+1)h-v]} X(v) \, dv. \tag{80} \]

Using (79) and (80), (78) becomes

\[ T_{n+1} = hX[(n+1)h] - \int_{nh}^{(n+1)h} e^{-(a+jb)[(n+1)h-v]} X(v) \, dv \tag{81} \]

The integrand of the integral appearing in (81) can be expanded in a Taylor series and the integration carried out term by term to obtain

\[ T_{n+1} = -\frac{1}{2} h^2 \left\{ (a-jb) X[(n+1)h] + X[(n+1)h] \right\} + \text{higher order terms} \tag{82} \]

The fact that \( T_{n+1} \) is \( O(h^2) \), shows that the algorithm of (76) is first order so that \( p = 1 \). Thus using (77),

\[ \phi(t,Y) \approx + \frac{1}{2} \left[ (a-jb) \chi(t) + \chi(t) \right]. \tag{83} \]

Note that with \( \chi(t) = e^{\omega t} \), \( \phi(t,Y) \) becomes
\[ \phi(t,Y) = + \frac{1}{2}(\omega - \lambda)ye^{\omega t} \] (84)

which is in the form of (59) with \( C_1 = 0 \) and \( C_2 = \frac{1}{2} \).

It is interesting to note that since \( C_1 = 0 \), \( \phi(t,Y) \) is independent of \( Y \), there is no root shift and \( E_r = 0 \). This of course results from making use of the analytic solution of the differential equation in obtaining the homogeneous part of the difference equation in the digital filter design.

The forced error \( E_f \) is not zero and, in fact, in this example is that of a first order method. Equation (69) gives \( E_f \) for this example as

\[ E_f = \frac{1}{2} h(\lambda - \omega) \] (85)

and the approximations of (70) through (72) become

\[ |E_f| \approx \frac{1}{2} h|\omega| , \ |\lambda| < < |\omega| \] (86)

\[ |E_f| \approx h|\lambda| , \ |\lambda| \approx |\omega| \] (87)

\[ |E_f| \approx \frac{1}{2} h|\lambda| , \ |\lambda| >> |\omega| \] (88)

Among other conclusions that can be drawn from these expressions is the fact that for a fractional forced error of less than \( 1\% \), \( h|\omega| < .01 \), or 628 samples must be made per cycle of the highest forcing frequency.

6-3 Stiff Equations

In numerical analysis, equations with widely separated eigenvalues are referred to as "stiff" equations. Difference algorithms corresponding to stiff differential equations usually require a considerable computing time for solution. The reason for this fact can be illustrated by considering algorithms
for solving an equation whose solution is given by

\[ y(t) = A_1 e^{-\lambda_1 t} + A_2 e^{-\lambda_2 t} \]  \hspace{1cm} (89)

where \( \lambda_1 \) and \( \lambda_2 \) will be assumed to be real with \( \lambda_2 = \alpha \lambda_1 \) and \( \alpha \) large.

In solving most problems, a solution over at least one time constant of the lowest frequency is typically required. Thus the solution time \( T \) must satisfy an equation of the form

\[ T = \frac{1}{\lambda_1} \]  \hspace{1cm} (90)

At the same time the solution must be both stable and accurate in the sense defined above and this imposes an upper limit on \( h \lambda \) for all eigenvalues. The latter requirement can be expressed for this example by the equation

\[ h \lambda_2 = \alpha h \lambda_1 \leq \beta \]  \hspace{1cm} (91)

where \( \beta \) is determined by a particular algorithm.

A relation for the number of steps, \( N \), required to obtain a solution under the condition stated results from combining (90) and (91) to obtain,

\[ N \geq \frac{\alpha \beta}{\lambda_1} \]  \hspace{1cm} (92)

Values for \( \beta \) can be obtained from Table V which shows limits on real roots for stability and reasonable root shift for typical algorithms. The values range from 1.05 for the RK-4 to 0.02 for the Euler method. If \( \alpha \) is assumed to be \( 10^3 \) the required \( N \) becomes 1050 for the RK-4 and 43,500 for the Euler method.
REFERENCES


FINAL REPORT

PROJECT A-588

DEVELOPMENT OF NEW METHODS AND
APPLICATIONS OF ANALOG COMPUTATION

Joseph L. Hammond, Jr., Project Director

Contract NAS8-2473

12 September 1961 to 30 September 1970

Prepared for

George C. Marshall Space Flight Center
National Aeronautics and Space Administration
Huntsville, Alabama
INTRODUCTION

Georgia Tech Research Project No. A-588 was established under Contract NAS8-2473 on 12 September 1961 and continued through various modifications of the original work statement until 30 September 1970. The overall project aim has been to assist the Simulation Branch of the Computation Division at George C. Marshall Space Flight Center in the investigation and development of new methods and applications of analog and hybrid computation within several areas of mutual interest.

Throughout the period of the contract Dr. W. K. Polstorff has been the Contracting Officers' Representative.

ADMINISTRATIVE SUMMARY

Research Project A-588 was originally assigned to the Special Problems Branch of the Engineering Experiment Station's Physical Sciences Division. Mr. Frederick Dixon, at that time Head, Special Problems Branch, was project director from the beginning of the contract until December 1967.

The initial phase of research was performed by personnel of the Georgia Tech Analog Computer Laboratory, principally Messrs. Robert S. Johnson, Research Engineer; Ralph D. Loftin, Research Assistant; and Frank R. Williamson, Jr., Assistant Research Engineer.

Early in 1964 the equipment of the Analog Computer Laboratory was physically located in the building housing the School of Electrical
Engineering and the project activities were divided between a group in the School of Electrical Engineering coordinated by Dr. J. L. Hammond and a group under the direction of Mr. Dixon. Principal personnel of the two groups in the period 1964-1967 were:

**Group 1**
- Mr. Frederick Dixon, Senior Research Physicist
- Mr. Frank R. Williamson, Assistant Research Engineer
- Mr. John W. Robertson, Research Assistant

**Group 2**
- Mr. Richard E. Bryan, Graduate Research Assistant
- Dr. David L. Finn, Professor
- Dr. Joseph L. Hammond, Jr. Associate Professor
- Dr. Roger P. Webb, Assistant Professor
- Dr. Thomas M. White, Associate Professor
- Various other Graduate Research Assistants active over short periods of time.

Late in 1967 the tasks assigned to Group 1 were completed and the project activity shifted completely to the group in the School of Electrical Engineering. In keeping with the change in emphasis of the work, Dr. Hammond replaced Mr. Dixon as Project Director. During the final phase of the contract Drs. Cecil O. Alford, Associate Professor; Joseph L. Hammond, Professor; and David L. Finn, Professor, were the principal participants, assisted by various Graduate Research Assistants.

---

* Titles are those applicable at the time each individual joined the project.
active over short periods of time.

During the period 1961-1964 current activity on the project was documented by quarterly and annual reports and completed tasks by technical notes, issued when appropriate. Beginning in late 1964 monthly progress letters replaced the quarterly and annual reports. One interim progress report for the period 12 March 1968 to 12 July 1969 was issued.

TECHNICAL SUMMARY

Technical progress on the contract has been documented by technical notes issued when a particular phase of the work was completed. Twenty-one technical notes, covering a variety of subjects, have been published. A listing of the titles, authors and abstracts of each technical note is given in the Appendix.

This section of the Final Report summarizes the research objectives and results of the various tasks assigned to the project. Reference is made in each case to the appropriate technical note for the details of research accomplishments.

Using the liberty of a certain amount of unification in viewing past work, the scope of the project can be said to cover six areas of indepth study and three specialized preliminary studies. These areas, the approximate dates of each study and the applicable technical notes are listed in Table I.

Each of the study areas is briefly summarized below.

1. Nonstationary Noise Studies: The general objective of studies
Table I. Analysis of Areas of Study - Research Project A-588

<table>
<thead>
<tr>
<th>Area</th>
<th>Applicable Technical Notes</th>
<th>Dates of Study</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Nonstationary Noise Studies</td>
<td>1,3,5,10,11,15</td>
<td>1961-1966</td>
</tr>
<tr>
<td>2. Electronic Generalized Integrator</td>
<td>6,14</td>
<td>1961-1966</td>
</tr>
<tr>
<td>4. Analysis and Control of Error in Hybrid Computation</td>
<td>4,8,12,13,17,18</td>
<td>1964-1969</td>
</tr>
</tbody>
</table>

Preliminary Studies

<table>
<thead>
<tr>
<th>Area</th>
<th>Dates of Study</th>
</tr>
</thead>
<tbody>
<tr>
<td>7. Simulation of Manned Space Flight</td>
<td>1963-1964</td>
</tr>
<tr>
<td>8. Network Analyzer Applications</td>
<td>2</td>
</tr>
</tbody>
</table>
in this area was to provide techniques for use in analog and hybrid computer Monte Carlo studies involving nonstationary random functions. An important application area is the simulation of space vehicle trajectories when the vehicle is subjected to random nonstationary wind perturbations.

The work centered around two specific problems, namely mathematical modeling of the phenomena of interest and development of techniques for generating nonstationary random processes with Gaussian distribution functions and prescribed nonstationary autocorrelation functions.

Progress made on the modeling problem is given in Technical Note No. 1 which summarizes background random process theory and discusses models appropriate to the application noted above.

A general technique for generating the required type of random process is given in Technical Note No. 3 along with a discussion of techniques for instrumenting analog computer Monte Carlo studies with nonstationary random functions. Specialized techniques for generating certain specific random functions are given in Technical Notes 5, 10, 11 and 15.

2. **Electronic Generalized Integrator**: The objective of work in this area was to design and construct an improved analog device for integrating with respect to an arbitrary variable. A central problem in constructing the generalized integrator was found to lie in the required electronic switch. A considerable amount of study documented in Technical Note No. 6 was thus devoted to electronic switches.

A generalized integrator having desirable characteristics was designed and tested. Schematic diagrams and the test results are
3. Hybrid Computer Methods for Solving Partial Differential Equations: The objective of this phase of the research was to determine what methods are available for solving partial differential equations on hybrid computers and then evaluate the advantages and disadvantages of the methods.

The first part of the objective was accomplished and documented in the bibliography of references on techniques for solving partial differential equations given in Technical Note No. 16. To obtain further insight into available methods a short course in hybrid computer methods for solving partial differential equations, was set up on the Georgia Tech campus with several authorities in the field as lecturers. Notes for this short course were provided the sponsor.

Emphasis was not given to this phase of the program and therefore an extensive evaluation of hybrid methods for solving partial differential equations was not carried out. It was established, however, that most practical methods approximate partial differential equations with sets of ordinary differential equations obtained through finite difference techniques. Since storage always seems to be a problem in solving the coupled sets of ordinary differential equations, a preliminary investigation of several methods of reducing storage requirements was made. The results are reported in Technical Note No. 21.

4. Analysis and Control of Error in Hybrid Computation: The objectives of this part of the project were to identify the major error sources in hybrid computation and study the characteristics of such error.
Preliminary studies, documented in Technical Note No. 4, indicated that for the word lengths possible with modern hybrid computers, errors due to a noninfinite sampling rate of the A/D converters are the most serious source of error. Thus a detailed study of sampling error was undertaken. This study ultimately resulted in a compact set of sampling error equations which seem to be quite general and provide considerable insight into the cause and effects of such error. Technical Note No. 17 documents the final, most general, form of the error equation, while the earlier notes give various stages in its evolution.

Some insight is gained by applying the error equation specifically to hybrid programs for solving linear constant coefficient equations. A study of special forms of the error equation corresponding to linear constant coefficient problem equations is documented in Technical Note No. 18.

5. **Hybrid Software Algorithms:** Since the sampling error equations give considerable insight into the nature and sources of sampling error, a logical application of these equations is the development of algorithms for reducing sampling error. Such a study was undertaken.

The error equations were used to evaluate various numerical methods for use in the digital portion of the hybrid computer. They were also used to suggest methods of compensating against sampling error through modified digital algorithms or analog programs. Three such methods are described in detail in Technical Note No. 19 which also documents the work with numerical methods noted above.
One of the compensation algorithms was studied with digital simulations of the hybrid computer programs for solving several elementary examples. This work is reported in Technical Note No. 20.

6. Methods for Digital Solution of Differential Equations:
A study of difference equations to approximate differential equations was undertaken with partial support from the contract and partial support through Dr. Hammond's teaching activities. The work is of interest since in general applications of hybrid computers the digital portion is allocated the solution of certain differential equations. Such equations are typically solved through the use of standard numerical methods.

A unified treatment of a number of numerical methods was developed and a tractable relation between sampling rate and accuracy was derived using a modified form of the sampling error equations. This work is documented primarily in Technical Note No. 22. Technical Note No. 19 has some material on numerical methods, emphasizing their performance subject to the real time requirements of hybrid computation.

7. Simulation of Manned Space Flight: In this portion of the project specific tasks concerned with flight simulators being constructed at MSFC were undertaken and documentation with technical notes was not appropriate.

8. Network Analyzer Applications: This phase of the work was a study of possible applications of network analyzers in problems of interest to NASA. The results are documented in Technical Note No. 2.
9. Error Analysis of a Spectrum Measurement System: This phase of the work was a preliminary study for the purpose of evaluating the magnitude of the task of determining the error in a spectrum measurement system utilizing a time-compression device. The results are documented in Technical Note No. 7.

SCHOLARLY ACTIVITIES

The results obtained from research on the project have been reported to the professional community on a continuing basis. Throughout the period of the contract papers have been presented at meetings of the Southeastern Section of Simulation Councils Inc. Most of the project personnel have been active in the Simulation Society, with several holding offices in that organization.

Three full length papers have been published in Simulation, namely:


"Application of State Variable Techniques to Analog Computer Programming," J. L. Hammond, Vol. 4, no. 5, May, 1965, pp. 316-323; and


Two verbal papers have been published in proceedings of meetings, namely:

"Determination of Hybrid Computer Sampling Rates," C. O. Alford and J. L. Hammond, Proceeding of the 1969 Southeastern Symposium on System Theory, V.P.I. May, 1969; and

A paper has appeared in the Transactions of the IEEE on Control, namely:


Another paper "Relations Between Accuracy and Sampling Period for Numerical Methods and Digital Filters," J. L. Hammond, has been submitted to the Proceedings of the IEEE.

The following theses in the School of Electrical Engineering at Georgia Tech have been outgrowths of work on the contract:


FINANCIAL SUMMARY

The following figures taken from the financial statements of the Georgia Tech Engineering Experiment Station summarize expenditures on the contract for the period 12 September 1961 through 30 September 1970.
<table>
<thead>
<tr>
<th>Category</th>
<th>Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Personal Services</td>
<td>$189,128.15</td>
</tr>
<tr>
<td>Materials and Supplies</td>
<td>4,384.21</td>
</tr>
<tr>
<td>Travel</td>
<td>2,057.47</td>
</tr>
<tr>
<td><strong>Total Direct Charges</strong></td>
<td><strong>$196,281.64</strong></td>
</tr>
<tr>
<td>Computer</td>
<td>5,019.95</td>
</tr>
<tr>
<td>Overhead</td>
<td>112,797.04</td>
</tr>
<tr>
<td><strong>Total Expenditures</strong></td>
<td><strong>$314,098.63</strong></td>
</tr>
</tbody>
</table>

*The figures given are taken from the Ga. Tech Engineering Experiment Station status report for the end of September 1970. They do not include charges against the project, such as processing of this Final Report, which are made after this date.*

Abstract

This Technical Note is intended as a brief introduction to nonstationary stochastic processes from the point of view of the analog computer engineer. It is assumed that the reader is familiar with stationary processes and the manner in which they are generated and used in analog computation.

Chapter I presents the principal terms, definitions, and conventions adopted throughout the report and discusses certain important functionals of nonstationary processes. Chapter II is devoted to a particular nonstationary functional—the power spectral density function; in this chapter, the various definitions of power spectra are compared and the usefulness of the spectral concept is discussed. Chapter III deals with a particularly useful class of processes and the way in which they are generated. The Technical Note is concluded in Chapter IV with an example of process synthesis using analog computer components.

Abstract

This Technical Note presents results of a brief survey of the literature on network analyzers, which was performed under Contract NAS8-2473 ("New Methods and Applications of Analog Computation") as part of a special task assignment entitled "Comparison of Passive Element Computers with Other Computers". Additional information to be gathered under this task should provide a basis for determining what size and type of network analyzer, if any, the Computation Division at Marshall Space Flight Center might acquire in order to supplement existing analog and digital computer facilities most effectively. The present report is intended to serve as a check-list or guide to the classes of problems for which the network analyzer might be considered.


Abstract

This Technical Note is concerned with certain aspects of the generation and use of nonstationary stochastic processes in Monte Carlo studies using an analog computer. It represents a continuation and extension of Technical Note 1 of the current project.

Certain aspects of the study are essentially complete. Section II of this Note outlines in detail a general method for the synthesis of analog computer circuits which when excited by stationary Gaussian white noise produce nonstationary random outputs with prescribed first and second statistical moments.
Section III of the Note describes a system for the measurement and processing of the nonstationary random processes. Also included in Section III is an error analysis which gives a confidence level criterion for use with the measured random data.

Section IV presents several examples for which computer circuits are constructed to realize given covariance functions, and for which the moments of the experimentally generated processes are measured. These examples illustrate the use of the synthesis procedure and the analog implementation, and also give an indication of the accuracy which can be achieved in this type of work.

Work is continuing on certain aspects of the analytical representation of random processes and on possible simplifications and extensions of the current results. Plans for the continuation are discussed in Section V of this note.


Abstract

This report summarizes current methods of attacking the error problems present in analog computers, sampled-data systems, and hybrid computers. It contains 286 references to the literature in this field.


Abstract

This Technical note is concerned with one aspect of the problem of generating and using nonstationary stochastic processes in Monte Carlo
studies with an analog computer. The work covered by this note is a continuation of that reported in Technical Note 1 and Technical Note 2 of the current project.

A central consideration in any use of stochastic process theory is a careful definition of a sample space and appropriate random variables. This technical note considers the physical problem of random wind perturbations of a missile trajectory, and defines a general mathematical model which describes the random wind forces. The class of stochastic processes defined by the model are termed partially-stationary stochastic processes. A single partially-stationary stochastic process which will generate the random disturbance encountered in a wide variety of individual missile flights is developed.

This technical note is subdivided into five sections. In Section II the relationship between the material presented in this note and that in previous notes is discussed.

Section III is devoted to the definition of a partially-stationary stochastic process and to the consideration of an example selected to provide physical insights into implications contained in the definition.

Some properties of first-order probability distributions for partially-stationary stochastic processes are investigated in Section IV. This is done by considering the physical problem of simulating a random disturbance affecting a rocket in flight.

The final section, Section V, is utilized for the presentation of conclusions and recommendations for further research.

Abstract

On the basis of a preliminary literature survey, selected analog switches were investigated to determine the relative performance characteristics of types considered most promising in analog or hybrid computer applications. Measured performance parameters are tabulated and evaluations given for examples of the modified Guennou chopper, the non-saturating two-and four-transistor switches, the complementary transistor bridge, the series-shunt unifet switch, and the six-diode bridge gate, all constructed from available standard semiconductor components. From these results, it is recommended that further work be done with the six-diode bridge gate and with the series-shunt field-effect-transistor switch to optimize their performance for specific analog and hybrid computer functions.


Abstract

The problem of analyzing a spectrum measurement system that utilizes a time-compression device is considered. A qualitative description of the system operation is given, and an idealized model is proposed to describe the system operation when the input is a deterministic time function. Input-output relationships are presented for each operation in the idealized model. Using the model proposed, it is estimated that approximately six additional man-weeks would be required for developing a procedure for determining the error in the measured spectrum of a deterministic function of time.
The problem of error analysis in measuring the spectrum of a more general class of functions including random processes is indicated to be much more difficult. A minimum of perhaps three times the above estimate might be required to obtain significant results.


Abstract

This report presents the theoretical portion of a study of error analysis for the solution of a class of problems on a hybrid computer. The class of problems considered are those dynamic systems which can be described by an equation of the form

\[ \dot{x} = f(x, \mathbf{y}, t) \]

where \( \mathbf{y} \) is a known vector function of time, \( x \) is an unknown state vector, \( f \) is a vector valued function, and \( t \) is time. The hybrid computer is assumed to be used in a mode for which both the analog and digital components process the dependent problem variables, with the analog components performing primarily integration and the digital components performing multiplication and nonlinear operations.

In the study a general linear equation for error is derived. This equation can be time varying and involves a vector forcing function which contains as components the errors resulting in the digital portion of each computational loop. This equation is not felt to be practical from a computational standpoint, but it can account for most dynamic errors and some types of static error.
Several approximate methods for solving the linear error equation are analyzed in the report. The most attractive of the methods investigated involves adjoining an approximation to the general error equation to the original equation and solving both equations on the hybrid computer. To make the results tractable, only dynamic errors are considered under conditions for which errors due to digital execution time and non-zero sampling interval predominate. These two sources of error are shown to have the same effect to a first approximation and thus the results involve only the sampling interval as a parameter.

Plans for evaluating the method by solving the representative problems have been made, and definite conclusions will be drawn after the next phase of the study.

Technical Note No. 9. (not issued).


Abstract

A method is presented for approximating, by use of analog computer components, any prescribed stationary Gaussian random process depending only on a position parameter $x$. The output of the analog mechanization system simulates the effect of the prescribed random process on some physical device or sensing element whose variable location is specified by the position parameter $x$. The two inputs to the mechanization system are Gaussian white noise and the first derivative of the function of time describing the position of the sensing element. Best approximation of the prescribed random process is obtained when the second derivative of the function of time describing the position of the sensing element is restricted to small values.

Abstract

Two methods are presented for the synthesis of analog computer networks that approximate a random process depending on both time and a position parameter. The networks are intended for use in the simulation of random wind disturbances that affect a rocket or other aerospace vehicle in flight. The output of the analog computer network simulates the effect of the prescribed random process on the vehicle as its position varies arbitrarily with time. The two inputs to the analog computer network are a Gaussian white-noise random process and a function of time characterizing the variable position of the vehicle.


Abstract

This report develops equations for the error caused by the nonzero sampling interval and execution time of the digital section of a hybrid computer in the solution of certain types of problems. The work applies to the solution of equations of the form $\dot{x} = f(x,y;t)$, where $x$ is a dependent vector, $y$ is a known vector of inputs, $t$ is time, and $f$ is a vector function of the indicated arguments. Integration is performed by the analog equipment, and the components of $f$ are generated digitally. Three variations of the error equation appropriate to different applications are derived. The three types of equations are as follows: (1) an equation for error with sampling rate and execution time as parameters, and with the hybrid solutions to the problem equation as
input variables; (2) an equation for error with sampling rate and execution time as parameters, and with the true solutions to the problem equation's input variables; (3) an equation for error as a function of time and equivalent sampling rate, both as independent variables. The three equations are shown to be valid with minor modifications for the case that some, but not all, of the components of $f$ are generated digitally and the others are generated by analog methods.

The report suggests the possibility that the error equations can be implemented on the hybrid computer as a means of compensating against errors caused by sampling and execution time. Advantages of such a procedure over more conventional ones are cited.


Abstract

This technical note presents recent new results obtained in a study of hybrid computer error due to nonzero sampling interval and execution time. The work is a continuation of that documented earlier in Technical Notes Nos. 8 and 12 under the subject contract. The present report is concerned primarily with the following points: (1) a computer implementation of previously derived error equations, (2) verification of each form of the error equations with specific test examples, and (3) the study of typical test examples to establish the general properties of the various equations.

A basic digital computer program for implementing the error equations is described. Among other things, this program can work from a general statement of the error equations to compute results for a
particular problem without the necessity of deriving a specific error
equation analytically. Several different types of results can be
printed out, including a table of maximum and minimum error versus
sampling interval.

Data obtained from a number of numerical solutions supports the
following conclusions:

(a) With one minor exception requiring further investigation, all
forms of the error equation are analytically accurate.

(b) The basic approximation on which the equations are based is
sound, and for reasonable conditions the error as determined from the
equations is a good approximation of true error.

(c) Little additional approximation error is introduced in the
form of the error equations which yields error versus sampling interval.
This form of the error equations is recommended for general studies.

(d) The forms of the error equations based on true solutions and
on hybrid solutions of the problem equation give comparable accuracy.

(e) Use of the error equations to compensate a hybrid system
seems to have much potential. For the test problem used, such compensa-
tion reduced error by a factor of ten.


Abstract

This report discusses the basic concepts of the final design of
a generalized analog integrator and compares various EGI designs. The
calibration and tests performed on the final design are presented. An
Appendix contains detailed circuit diagrams.

Abstract

A mathematical model of random wind velocity is presented for use in the synthesis of an analog computer network to simulate wind turbulence. A synthesis technique, called the covariance-expansion method, is applied to the mechanization of the model. The output of the analog computer network simulates the effect of wind turbulence on a vehicle as it moves on an arbitrary path in space. The inputs to the analog computer network are (a) A Gaussian white-noise random process and (b) appropriate functions of time characterizing the variable position of the moving vehicle.


Abstract

References in this bibliography are related to the numerical solution of partial differential equations. The references are listed under three headings: Hybrid Computer Methods, Analog Computer Methods, and General Methods. Brief comments are given for most of the references on Hybrid Computer Methods.


Abstract

A vector differential equation for the error due to sampling in closed loop hybrid computer programs is developed using several approximations derived with Taylor series expansions. The equation is applicable
to general hybrid programs with no restriction on the manner in which the computing operations are allocated between the analog and digital computers. The major restrictions necessary for the error equation to apply are:

1. the problem equation must be expressible as a set of first order (linear, nonlinear or time varying) equations,
2. the digital-to-analog converters must be zero-order hold,
3. all converters and numerical methods of the digital computer must have the same sampling period,
4. all digitally generated functions must be computed during the same time period and converted D-to-A at the same time and finally,
5. the sampling period must be small.

The error equation is linear. Its homogeneous part is independent of the allocation of operations between the analog and digital computers, but its forcing function depends on the details of such allocation and certain constants, namely: the sampling period, the digital execution time and the order of numerical methods used in the digital computer. Both parts of the error equation depend on the problem solution variables, but these can be either the true solutions or the actual hybrid computer solutions.

Solution of the error equation typically requires machine computation, but several properties of sampling error are apparent from the form of the equation. For example, since the forcing function on the error equation is proportional to the first power of the sampling period, it follows that the hybrid computer is a first order computational method. It is also apparent from the error equation that the execution time of digitally generated functions has the same general effect as a non-zero sampling period but weighted twice as heavily.
The error equation is expected to be useful in studying existing hybrid computer programs, in allocating computing operations between the analog and digital computers and in compensating against sampling error.


Abstract

This technical note documents a continuation of the study of sampling error in hybrid computer programs based on the general error equation given in Technical Note No. 17 of the subject contract. Specifically attention is given sampling error for linear constant coefficient problem equations. Such equations could arise as the given problem equation or as an approximation to general nonlinear equations.

The work shows that the hybrid equation accounting for sampling error is a linear constant coefficient equation differing from the ideal equation because of perturbations in both is homogeneous and nonhomogeneous parts. Explicit equations for the solution to both the ideal equation and the hybrid equation accounting for sampling error are given in terms of allocation matrices, the sampling period and the execution time of digital operations.

Two types of errors are identified, namely: that which arises in the nonhomogeneous part of the hybrid equation and which depends primarily on the forcing function of the ideal equation and errors in the transition matrix caused by perturbations in the homogeneous part of the hybrid equation. The latter is examined in detail through a study of root shift due to sampling.
A tractable equation for computing root shift is derived along with useful bounds for this quantity. The equations are used in discussing the problem of programming a hybrid computer to minimize root shift. Three examples are presented to illustrate the determination of root shift.


Abstract

This technical note is an outgrowth of a continuing study on the sampling errors in hybrid computer programs. In particular, Technical Note No. 17 documents the error equations for a general class of hybrid programming problems, and forms the basis for this technical note. This note presents hybrid software techniques to eliminate or decrease the various error terms derived in Technical Note No. 17.

After a statement of the problem in Chapter I, attention is given to the choice of a numerical integration method when dynamic equations are programmed on the digital part of the hybrid computer. Chapter II analyzes and compares several numerical integration methods with respect to timing considerations, errors, and execution time. The chapter concludes with considerations for the optimum choice of a particular method.

Attention is next focused on errors due to sampling and the use of zero order hold devices in hybrid computers. Several schemes for the compensation of these errors are presented in Chapter III.

Another major source of error is caused by function evaluation, or execution time delay, in the digital computer. Chapter IV presents compensation methods for hybrid computers with zero-order hold devices and
with the compensated hold devices presented in Chapter III.

Concluding remarks are given in Chapter V.


Abstract

The results of preliminary studies of the digital hardware-software method, described in Technical Note No. 19, for compensating against sampling errors in hybrid computers are reported. Three examples in which the method is applied are studied with a digital simulation of a hybrid computer. In all three cases significant reduction in error is observed.


Abstract

This technical note gives the results of a preliminary study of methods for solving partial differential equations with a hybrid computer. The study deals with one form of serial solution and investigates two methods of treating stored data.

The one dimensional heat equation is used as a test case for the study. The partial differential equation is converted to a set of ordinary differential equations in the time variable using seven discrete space "stations." The ordinary differential equations are solved using a serial, iterative technique which requires storage of the complete time solution at each space point. An allocation of hybrid operations using analog integration of the differential equations
and digital storage is simulated on an all digital computer.

Results are obtained under a variety of conditions for a zero order hold approximation to stored data with and without a time shift in the data. A limited amount of data is obtained for a polynomial approximation to the stored data.

The polynomial approximation gives the best results of the methods tried. For the zero order hold, it seems necessary to shift the data in time a precise amount for the results of the computation to converge to the correct answer.


Abstract

This technical note reviews and consolidates material pertaining to one-step and multistep numerical methods with the objective of developing a tractable method for evaluating the error resulting from the use of difference equations to approximate ordinary differential equations.

A tractable approximate relation is developed between the error due to finite sampling rates and parameters describing numerical methods. The results are valid for small values of the product of the sampling period, h, and natural frequencies, λ, of the analog system.

The approach used approximates the analog system by isolated modes and the digital algorithm for each mode by its order, p, and certain constants C, A₀, ..., Aₚ. The results show that approximating an analog system by a digital algorithm results in a shift of each root and an error corresponding to each forcing function. The root shift
is given by $-Ch^p \lambda^p$, while the error caused by a forcing function, 
$\gamma_{e, \omega}^{\omega,t}$ is given by $-Ch^p \sum_{i=0}^{p} A_{i} \lambda^i \omega^{p-i}$.

The results of the study are applied to a nontrivial numerical analysis problem, a digital filter problem and to an elucidation of the problem of "stiff" equations.