AD NUMBER

ADA800038

LIMITATION CHANGES

TO:

Approved for public release; distribution is unlimited.

FROM:

Distribution authorized to DoD only; Administrative/Operational Use; 21 MAR 1951. Other requests shall be referred to Office of Naval Research, 875 North Randolph Street, Arlington, VA 22203. Pre-dates formal DoD distribution statements. Treat as DoD only.

AUTHORITY

ONR ltr dtd 27 Apr 1966

AD NUMBER
ADA800038
CLASSIFICATION CHANGES
TO: **unclassified**
FROM: **restricted**
LIMITATION CHANGES
TO: **Distribution authorized to DoD only; Administrative/Operational Use; 21 MAR 1951. Other requests shall be referred to Office of Naval Research, 875 North Randolph Street, Arlington, VA 22203. Pre-dates formal DoD distribution statements. Treat as DoD only.**
FROM: **Controlling DoD Organization: Office of Naval Research, 875 North Randolph Street, Arlington, VA 22203.**
AUTHORITY
E.O. 10501 dtd 5 Nov 1953; Pre-dates formal DoD distribution statements. Treat as DoD only.

FLIGHT TRAINER DIGITAL COMPUTER STUDY

UNIVERSITY of PENNSYLVANIA

Moore School of Electrical Engineering

PHILADELPHIA, PENNSYLVANIA

21 March 1951

Research Division Report 51-28

RESTRICTED

Final Report on

FLIGHT TRAINER DIGITAL COMPUTER STUDY

Report of Work under Contract N6Onr 24913

Project 24-F-1

Between

Office of Naval Research

Special Devices Center

Port Washington, New York

and

University of Pennsylvania

Moore School of Electrical Engineering

Philadelphia 4, Pennsylvania

This report was prepared by;

C. N. Weygandt

H. Sohon

M. Rubinoff

H. J. Gray, Jr.

E. A. Knobelauch

21 March 1951

Research Division Report 51-28

The authors of this report wish to acknowledge with thanks the contributions to this report of Eleanor Pfefferle, Lillian Snyder, Elizabeth P. Schoff who did the computation described in Section 4; of W. DeW. Horrocks and his staff who prepared the drawings; and of Jean Schwitters, Mary Davis and Dorothy McIlhatton of the Moore School Secretarial Staff who helped in the preparation of the manuscript.

C. Eastman W. L. Fitzgerald
Project Supervisor

W. G. Dow
Section Head

W. G. Dow
Supervisor of Research

SYNOPSIS

This is the final report of work done on this contract. Sections 2, 3, 5 and 6 are substantially the same as sections of the same number in the interim report dated 15 December, 1950. Sections 1 and 4 are different from those of the interim report and contain additional information.

The contents of this report are:

1. INTRODUCTION

1.1 History of Problem

Operational flight trainers have been used for many years in the training of airplane pilots. The original Link Trainer is the most familiar of these. Present day flight trainers are complex devices which involve computers which solve, among other things, the equation of motion of the airplane in space in order to obtain information necessary to provide not only realistic but theoretically correct instrument readings and stick and rudder forces in the cockpit of the trainer. The computers used in all trainers which have been built up to the present time are of the analogue type. The question naturally arises that, since digital computers have been built which can obtain more accurate solutions of certain systems of equations in less time than analogue computers, why cannot digital computers be used to advantage in the construction of operational flight trainers? It has also been suggested that possibly a single digital computer might be associated with a number of cockpits, and so make it possible for a group of trainees to receive simultaneous instruction. Another and very real advantage of a digital computer would be the ease with which it could be changed from simulation of the performance of one plane to that of another plane and back again.

In its early stages Project Whirlwind was a more ambitious program along these same lines. It was hoped at one time that the Whirlwind computer could be made to simulate the performance of an airplane so well, even including structural deformations in flight, that it could be used to a certain extent in place of actual experi-

ments on pilot models to guide the design of new airplanes.

The present project is a little more definite and less ambitious in its scope. The Moore School of Electrical Engineering has been asked by the Special Devices Center Office of Naval Research to make a detailed study of the requirements of a digital computer for use in an operational flight trainer to simulate the Grumman F-9-F airplane. This plane was chosen partly because a successful operational flight trainer for this plane has been built by the Engineering Research Corporation, Riverdale, Maryland. The equations used by ERCO in the design of their trainer are the starting point of the present study.

1.2 Special Aspects of a Flight Trainer Computer

The requirements of an Operational Flight Trainer impose a number of conditions upon a computer that are not present in most general purpose computers.

a. Real time simulation

The solution of the equations must be accomplished at a rate such that changes in variables must make their effects felt as quickly as the corresponding effects would be felt by the pilot of a real plane in flight. That is the solutions must keep pace with real time.

b. Provision for the introduction of arbitrary quantities at any time.

Since an airplane in flight is a controlled system in which the pilot is a part of the system, provision must be made for introducing the effects of pilot caused motions of the control surfaces at any time. As explained more fully in Section 4, these motions, which

cannot be predicted on the basis of past history, have an important bearing on the numerical methods of integration used in a digital computer.

c. Conversion from digital to continuous information and back.

The information coming from the pilot (stick, rudder, and throttle motion) are continuous, and the information delivered to the pilot (instrument readings and control loading) are also continuous. Therefore, conversion apparatus in both directions must be provided if a digital computer is used. This apparatus is similar to input-output equipment for general purpose digital computers, except that for a flight trainer it must be fast enough to keep up with real time.

d. Change of equations for ground and air operation.

An operational flight trainer must simulate the plane when warming up and taxing on the ground, and during take off and landing as well as in the air. For these various conditions the equations are in some cases entirely different and in most cases not exactly the same, so that provision must be made for solving the proper equations under the proper conditions, and for making reasonably smooth transition from one solution to the other.

1.3 Accuracy Required for Flight Trainer

Let us consider what is meant by accuracy of solution. Consider three hypothetical experiments.

(a) Suppose that a pilot in actual flight executes a certain maneuver a number of times and that we were able to record both the

movements of the stick and the resultant motion of the plane.

By "maneuver" is meant a series of control motions by the pilot starting and ending in undisturbed level flight. Examples would be an Immelman turn or a dive and pull out.

Call the records of the resultant motions the true motion. Now suppose that a digital flight trainer were to be built and two experiments performed upon it.

(b) Let the recorded motion of the stick for one of the real maneuvers be fed into the trainer and the simulated motion of the trainer be recorded.

(c) Let the same pilot get into the trainer and be asked to execute the same maneuver a number of times, and the simulated motion of the trainer recorded for each maneuver.

Now let us make comparisons of the hypothetical records obtained in all of these cases.

The records of the true motion (a) and the simulated motion in experiment b could be expected to differ for three reasons:

1. The equations used do not entirely describe the motion.

2. In the real flight certain disturbances such as changes in wind velocity were present that the trainer did not attempt to simulate.

3. The trainer did not solve the equations with perfect accuracy.

However, this comparison would be of little use since it would be entirely impossible to distinguish between these causes of inaccuracy.

Now suppose that all of the records of the real maneuvers experiment (a) be considered. They will differ among themselves because the environmental conditions were not the same for all maneuvers, and because it would be impossible to expect the pilot to produce exactly the same stick rudder and throttle motions each time.

The records of the simulated motions in experiment c. would be expected to differ less among themselves because in this case the environmental conditions could be made the same for each simulated maneuver.

Now suppose that some kind of statistical study were made of the records of the real flight experiment (a) such that upper and lower bounds could be determined for each of the motion variables as functions of time. If all of the records of simulated motion in experiment c. fell within these bounds for all possible maneuvers then the flight trainer could be considered a perfect simulator of the actual motion of the plane. The conclusion to be drawn from this reasoning is that the computer may introduce inaccuracies in the simulated motion variables so long as these inaccuracies do not cause the motion of the simulator to exceed the limits to be expected for the same maneuver in actual flight.

Since, because of varying environmental conditions and pilot motions the values of motion variables might well differ by as much as ten percent in two actual executions of the same maneuvers by the same pilot in the same airplane, high accuracy is not necessary in a flight trainer. On this basis a computer which produced a solution

which differed from the true solution by as much as ten percent at times would be entirely acceptable provided the ten percent represented the maximum deviation from the true solution. In terms of a digital computer the maximum accumulated error should be less than ten percent in a time sensible to pilot. This means that the error in a single step in computation must be much less than ten percent. How much less is a function of the equations being solved and the length of the computation interval and the number of intervals comprising the manoeuvres

1.4 SUMMARY OF RESULTS OF STUDY

1.4-1 Manipulation of Given Equations

As mentioned before, the equations used by Erco formed the starting point of this study. As is explained fully in Section 2, it was found possible to rewrite the equations in a form which does not contain the Euler angles explicitly, and thus to eliminate the infinite values of functions of these angles that occur for certain aspects of the airplane. Also as explained in Section 3, P.3.3 some trigonometric functions, which are time consuming to compute digitally, were approximated. The computations in Section 4 show that these approximations did not alter the accuracy of representation. Lastly, as is explained in Section 2, digital machines are usually designed to handle only numbers less than one. On this account new variables were introduced such that no number appearing in any equation is greater than one. A complete list of the final equations starts on page 2-28.

1.4-2. Programming of Equations

The equations mentioned above were then programmed for digital

computation. By programming is meant the process of breaking down all complex algebraic formulae, and numerical integration formulae into the basic operations of addition, subtraction, multiplication, and division, plus some other operations like shift and transfer, peculiar to digital computers. Programming should be distinguished from encoding, which is the translation of a program into the particular language of a digital computer. The programming is explained in detail in Section 5.2 and is given in chart form in figures 5.2-2 to 5.2-9. From the program a count of the total operations and the total memory space can be obtained. These are summarized in the following table:

	Hurricane	Machine of Section 3.9
Time for one computing cycle	0.22 secs.	0.1 secs.
Memory cells	2768-2960	5232-5424

This count is based on the use of the equations as listed in Section 2.7 and on a representative integration formula of the Adams-Bashforth type. The equations listed provide continuity for all transitions from air to ground and ground to air. However, since the usual integration formulae call for a certain amount of "overlap" or remembering of past values, the programming of transitions is dependant on the integration formula selected. Hence, if a different integration formula were to be used it might make a considerable change both in the time and memory space. However, the numbers given are representative of order of magnitude, and for the equations used would not vary by as much as a factor of two.

1.4 Accuracy of Digital Solutions

The times for one computing cycle given in the table of the preceding section are representative of time consumed by typical digital computers in carrying out one step of computation. If this time is less than a real time increment which, for the method of computation used, assures sufficient accuracy of the solution, then a digital computer will perform satisfactorily; if not, the computer will of course fail to keep up with real time. In section 4.4 the theoretical computations of the errors in a method of integration, called the modified Moulton Method are compared with hand computations of several manoeuvres using the Flight trainer equations presented in this report. The conclusions of this section may be summarized as follows:

a. The errors occuring in the hand computation are much greater than those predicted taking into account the usual round-off and truncation error.

b. The cause of this additional error is two-fold. First, there is the inherent inability of any open method of integration to predict the effect of a change introduced into into the equations (by the action of the pilot) after the integration for a particular step in computation has started. Second, when (as in the case of the flight trainer) there are many simultaneous equations involved, a disturbance introduced into one equation may take many steps of computation to make its effect felt in the other equations. The second source of error (called compliance error in this report) is

identical with errors caused by the time delays in the components of an analogue computer.

c. These errors can be reduced to acceptable values in three ways. First, by decreasing the real time increment corresponding to a step in computation. Second, by using repeated closures for the same real time increment. This is analogous to using feedback in analogue computers. Third, by using a more complicated method of integration that takes into account in one step of computation the effect of a disturbance on all the equations involved, instead of only the equation into which it is directly introduced. Obviously, any of these methods will lengthen the time taken by the computer for one cycle of computation.

The experimental computations indicate that the Moulton method with no more than three closures at 1/16 second time intervals would be satisfactory for the manoeuvres computed. Three closures would triple the time for one computation cycle estimated in Table of Section 1.4-2, that is 0.3 seconds for the faster machine. Since this is greater than 1/16 second (.0625 seconds) this machine could not keep up with real time. Similar remarks hold for the Modified Moulton Method.

1.4 Conclusions

These conclusions are based on the equations of the F-9-F airplane as developed by the Engineering Research Corporation which formed the starting point of this study.

a. No existing or soon to be completed digital computer can solve the given equations of a single airplane in real time using methods of integration commonly used on digital computers. In the case of the Moulton Method of Integration (which appears to the writers to be the best of the commonly used methods) the fastest digital computer considered would be too slow by at least a factor of three. In addition, a memory capacity in the neighborhood of 5000 binary numbers would be necessary, which exceeds that of most existing or projected machined.

b. In view of conclusion (a), it follows that no existing digital computer can solve the given equations for more than one airplane simultaneously in real time.

SECTION 2

MANIPULATION OF GIVEN EQUATIONS

S Y M B O L S

In the list of symbols that follows, generally the dimensions are given rather than the units involved. In order to find what the units are, please refer to the list of transformations.

A	quantity computed to make take-off decision
B	quantity computed to make two-wheel decision
C_L, C_{L_1}, C_{L_2}	lift coefficient
C_{m_1}, C_{m_2}	coefficient, functions of M_a; used to compute M_t.
C_{m_3}	coefficient used to compute M_t.
D	quantity computed to determine thrust T
E	quantity computed to determine thrust T
F_B	resulting brake force
F_{BR}	force applied to right brake pedal
F_{BL}	force applied to left brake pedal
H	quantity computed determine thrust T
I_x, I_y, I_z	moments of inertia of airplane with reference to the X, Y, Z axes respectively
J	quantity introduce in determination of thrust, equals a definite positive number or zero, depending on whether or not water is injected.
K_F, K_G, K_N	factors that show effect of flaps, landing gear, and nose droop.
L_tV_t, M_tV_t, N_tV_t	moments about the wind tunnel axes X_t, Y_t, Z_t respectively. Positive directions determined by right hand rule.
M_a	mach number
R_x, R_y	ranges of the plane North and East of the origin during a problem.
S_i	rate of ice formation
S_a	rate of firing ammunition

SYMBOLS

S_{Fb} rate of fuel consumption from body tank.

S''_{Fb} rate of dumping body tank.

S'''_{Fb} rate of filling body tank.

S_{FW} rate of transferring fuel from wing tank to body tank.

S'_{FW} rate of dumping wing tanks.

S''_{FW} rate of filling wing tanks.

T thrust.

T_B resulting brake torque.

V_t airspeed.

W weight of plane.

W_M maximum weight of plane in pounds (figure used in equations is 15,000).

W_i weight of ice.

W_a weight of disposable portion of ammunition.

W_{fb} weight of fuel in body tank.

W_{fW} weight of fuel in wing tanks.

X airplane longitudinal (roll) axis, or force along this axis, positive forward.

Y airplane lateral (pitch) axis, or force along this axis, positive to starboard.

Z airplane normal (yaw) axis, or force along this axis, positive downward when airplane is in normal upright attitude.

X_t, Y_t, Z_t wind-tunnel axes, X_t and Z_t differ from X and Z by the angle of attack while Y_t coincides with Y.

X_tV_t, Y_tV_t, Z_tV_t forces along wind-tunnel axes.

SYMBOLS

a	function of M_a used to compute X_t.
b	function of M_a used to compute X_t.
b_{ix}	amount of ice collected on the plane.
c	function of M_a used to compute X_t.
d	distance in feet of airplane center of mass aft of reference point.
g	acceleration of gravity 32.18.
h	altitude above sea level.
h_o	altitude of landing field.
k	coefficient appearing in ϵ^{-hk} which in turn accounts for the variation of air density with altitude above sea level.
l_1, m_1, n_1, l_2, m_2, n_2, l_3, m_3, n_3	direction cosines relating the airplane axes to axes fixed to the earth.
m	slope of airplane lift curve.
n	engine speed (proportional to RPM), used to compute thrust T.
n_o	minimum idling speed.
p, q, r	angular velocities of the plane about X, Y, Z axes respectively.
t	time in seconds.
u, v, w	velocities along X, Y, Z axes respectively.
x_o	displacement of plane to the North of starting point.
y_o	displacement of airplane to East of starting point.

S Y M B O L S

Dot over a quantity indicates a function proportional to the time derivative of the quantity.

α angle of attack. Angle between X axis and projection of relative wind on the XZ plane, i.e. angle between X and X_t axes. Positive when positive X_t lies between positive X and positive Z.

δ surface deflection, subscript indications as follows:

- a Aileron, positive when left aileron is down.
- e Elevator, positive downward.
- fd Dive flap
- r Rudder, positive to the left.

θ Eulerian angle of pitch. Angle between X axis and a horizontal plane positive for nose up.

ψ Eulerian angle of heading. The angle a vertical plane through the X axis makes clockwise from true North.

ϕ Eulerian angle of roll. After ψ and θ have put the X axis in the correct space position, the airplane is rolled about the X axis through ϕ to put the Y and Z axes in proper position.

ϵ, ϵ_2, ϵ_3 corrections applied to direction cosines to normalize them.

ϕ_t angular setting of throttle.

2. MANIPULATION OF GIVEN EQUATIONS

The equations of motion of an airplane were modified by the Engineering Research Company to make them suitable for solution by an analogue-type computer. These modified equations are referred to in this report as the ERCO equations or the given equations. It is evident that the ERCO modifications might be desirable for an analogue computer while other modifications might be desirable for a digital computer. It has in fact been found desirable to change the ERCO equations in some respects to make them more suitable for solution by means of digital computers, and it is the purpose of this section to discuss these alterations.

For example, it was felt that the presence of Euler angles would present difficulties because the kinematic equations might lead to division by small quantities. One of the first steps, therefore, was to eliminate the Euler angles from the equations as outlined below in P. 2.3.

The determination of functions such as sines, cosines, exponentials, and other arbitrary functions can be handled easily by an analogue computer, but not by a digital machine (see Section 3). Consequently for the digital simulator contemplated here, these functions are determined by a different procedure. The sine and cosine of the angle of attack can be computed directly. A small angle can be equated to its sine as was done by ERCO. Other functions can be handled as sets of straight lines.

For motion on the ground it was decided to modify the ERCO equations to eliminate forces and moments that are neutralized by the

ground reaction. This leads to three types of manoeuvres for solution: manoeuvres in the air, manoeuvres with all three wheels on the ground, and manoeuvres with the front wheel raised. This is discussed in P. 2.4.

The effect of winds can be taken care of by simple devices while the airplane is in the air. The method of taking account of wind while the airplane is on the ground has not been developed at this time.

The symbols and equations as they appear at the time of this report are listed in P. 2.7.

2.1 Normalization of Parameters

The general purpose digital computer has the point (decimal, binary, or otherwise) at one end of the register. The machine therefore works with either integers or fractions but not with mixed numbers. Of the two extreme positions it is more common to find a machine working with fractions than with whole numbers. If our problem is to be worked on a computer designed to handle fractions it is necessary that we change all equations so that no number as large as unity will be encountered.

To make all quantities less than one we could divide all our equations by sufficiently large numbers. The effect might be to make some of the variables considerably less than unity. Under these circumstances, if each variable is to be represented by the same number of significant figures, the number of digits in the machine would be made quite large. The more digits in each number the longer each process becomes. Therefore where speed is so important, the numbers should be adjusted to remain less than unity, but each variable should

have its maximum as close to unity as possible. A more detailed discussion of this matter is to be found in Section 3.

Consequently, each variable in the equations was divided by a number somewhat larger than its own maximum. The new coefficients in the equations were then examined to ensure that no number exceed unity. This necessitated that numbers be shifted either to the right or to the left. Shifting has the effect of multiplying or dividing the number by a power of two in a binary machine, just as shifting the decimal point corresponds to multiplying or dividing by a power of ten in the decimal system; thus it will be noted in the equations of 2.7 that factors equal to powers of two have frequent occurrence.

2.2 Simplification of Given Equations

The equations solved by the ERCO simulator are empirical equations that approximate test results. Since these equations lead to a satisfactory trainer one would expect that a satisfactory trainer could be obtained if the digital computer made use of the same equations. On the other hand it is agreed that the ERCO equations in many cases are only empirical approximations. If one empirical approximation is replaced by another that is not too different, satisfactory results should be expected.

The ERCO equations contain the angle of attack, its sine, and its cosine. The ERCO computer in many places replaced the sine by the angle and replaced the cosine by unity. When these equations were prepared for solution on a digital computer it was decided to compute the sine and cosine and to replace the angle by its sine instead of replacing the sine by the angle. Numerical calculations have been performed

which indicate that these approximations are justified.

The angle of yaw appears in some of the ERCO equations. In our equations it has been replaced in all cases by the approximation v/V_t. As a result of the angle of yaw does not appear in any of our equations.

The determination of thrust was made by an empirical stydy of block diagrams supplied by the contracting agent. It is believed that the formulas used are reasonable approximations for the present purpose. That is, a more satisfactory set of formulas would probably involve the same amount of instruction - and memory-space.

The block diagrams supplied appear to indicate that the fuel consumption is constant for any throttle setting! This is hard to believe. However, it is felt that the machine space and computation time for rate-of-fuel determination will be small. Hence this arbitrary assumption will be adopted if the information is not made available before the final report is written.

2.3 Elimination of Eulerian Angles

The equations of motion of a rotating rigid body are usually simplified by using a set of axes attached to the rigid body. The simplification results from the fact that this choice of axes permits the use of constant moments and products of inertia.

The orientation of the rotating axes OX, OY, and OZ relative to a set of fixed axes OX_o, OY_o, and OZ_o may be defined by the Eulerian angles ψ, θ, ϕ. For convenience assume OZ_o downward, OX_o North, and OY_o East. ψ is the angle measured about OZ_o from the X_oZ_o plane to

the (vertical) plane containing OZ_o and OX; it is positive when OX is toward the East. θ is the angle between OX and the X_oY_o plane; it is positive when OX is above the X_oY_o plane. It is measured about an axis in the X_oY_o plane. Call this axis OY'.

ϕ is the angle measured about the OX axis from OY' to OY; it is positive for clockwise rotation as viewed from the origin along the positive OX axis.

If the angular velocity of the moving axes has components p, q, and r about the moving axes OX, OY, and OZ, respectively, then the following three differential equations show the relations among the angles and the components of angular velocity.

$$\dot{\theta} = q \cos \phi - r \sin \phi \qquad (2.3 - 1)$$

$$\dot{\phi} = \frac{1}{\cos \theta} (p \cos \theta + q \sin \theta \sin \phi + r \sin \theta \cos \phi) \qquad (2.3 - 2)$$

$$\dot{\psi} = \frac{1}{\cos \theta} (q \sin \phi + r \cos \phi). \qquad (2.3 - 3)$$

The presence of cos θ in the denominator makes two of these equations indeterminate when θ is $\pm 90^\circ$. This makes the equations difficult to use especially where point by point numerical computations are being performed. One common method of evading the difficulty is to limit the variable θ say to the range $\pm 85^\circ$. This is avoiding the problem rather than solving it.

This indeterminism is a product of the mathematical approach to a physical problem. A different choice of axes for example would present the same difficulty at a different orientation of the rotating body. Therefore if there is a way out of the difficulty it may well be to try a new mathematical point of view.

If a point has coordinates x, y, z, in the moving system, then its coordinates x_o, y_o, z_o in the fixed system are given by the matrix equation,

$$\begin{Vmatrix} x_o \\ y_o \\ z_o \end{Vmatrix} = \begin{Vmatrix} l_1 & m_1 & n_1 \\ l_2 & m_2 & n_2 \\ l_3 & m_3 & n_3 \end{Vmatrix} \times \begin{Vmatrix} x \\ y \\ z \end{Vmatrix} \qquad (2.3 - 4)$$

where the direction cosines in the transformation matrix are related to the Eulerian angles by the formulas

$$\begin{aligned} l_1 &= \cos\theta \cos\psi \\ l_2 &= \cos\theta \sin\psi \\ l_3 &= -\sin\theta \\ m_1 &= \sin\theta \sin\phi \cos\psi - \cos\phi \sin\psi \\ m_2 &= \sin\theta \sin\phi \sin\psi + \cos\phi \cos\psi \\ m_3 &= \cos\theta \sin\phi \\ n_1 &= \sin\theta \cos\phi \cos\psi + \sin\phi \sin\psi \\ n_2 &= \sin\theta \cos\phi \sin\psi - \sin\phi \cos\psi \\ n_3 &= \cos\theta \cos\phi \end{aligned} \qquad (2.3 - 5)$$

When the equations of motion contain the Eulerian angles they appear in combinations equal to one or more of the direction cosines. That is, the Eulerian angles can be eliminated from the mechanical equations of motion by introducing the direction cosines in the transformation matrix as parameters. The kinematic differential equations for the direction cosines are quite simple and have no indeterminate points. This is the principal gain.

The new differential equations can be derived by the following method. The transformation that was introduced above to obtain the coordinates of a point in the fixed system in terms of its coordinates in the moving system can be used to transform the unit vectors i, j, k along the OX, OY, OZ axes into the unit vectors i_o, j_o, k_o along the OX_o, OY_o, OZ_o axes; that is

$$\begin{Vmatrix} i_o \\ j_o \\ k_o \end{Vmatrix} = \begin{Vmatrix} l_1 & m_1 & n_1 \\ l_2 & m_2 & n_2 \\ l_3 & m_3 & n_3 \end{Vmatrix} \times \begin{Vmatrix} i \\ j \\ k \end{Vmatrix} \qquad (2.3-6)$$

In particular

$$i_o = il_1 + jm_1 + kn_1. \qquad (2.3-7)$$

Differentiate this with respect to time

$$0 = \frac{di}{dt} l_1 + i\dot{l}_1 + \frac{dj}{dt} m_1 + j\dot{m}_1 + \frac{dk}{dt} n_1 + k\dot{n}_1. \qquad (2.3-8)$$

If the moving axes are rotating with an angular velocity ip + jq + kr

$$\frac{di}{dt} = jr - kq \; ; \; \frac{dj}{dt} = kp - ir \; , \; \frac{dk}{dt} = iq - jp. \qquad (2.3-9)$$

Substituting these values for the derivatives we obtain

$$i(\dot{l}_1 - rm_1 + qn_1) + j(\dot{m}_1 - pn_1 + rl_1) + k(\dot{n}_1 - ql_1 + pm_1) = 0 \qquad (2.3-10)$$

which requires that

$$\dot{l}_1 = rm_1 - qn_1 \qquad (2.3-11)$$

$$\dot{m}_1 = pn_1 - rl_1 \qquad (2.3 - 12)$$

$$\dot{n}_1 = ql_1 - pm_1. \qquad (2.3 - 13)$$

A similar treatment of j_o and k_o will give similar results

$$\dot{l}_2 = rm_2 - qn_2 \qquad (2.3 - 14)$$

$$\dot{m}_2 = pn_2 - rl_2 \qquad (2.3 - 15)$$

$$\dot{n}_2 = ql_2 - pm_2 \qquad (2.3 - 16)$$

$$\dot{l}_3 = rm_3 - qn_3 \qquad (2.3 - 17)$$

$$\dot{m}_3 = pn_3 - rl_3 \qquad (2.3 - 18)$$

$$\dot{n}_3 = ql_3 - pm_3 \qquad (2.3 - 19)$$

Since there were three independent differential equations involving the Eulerian angles we expect only three of these nine new differential equations to be independent of the 21 identities that obtain for such a transformation matrix. These 21 identities include six of the form (one for each row and one for each column):

$$l_1^2 + m_1^2 + n_1^2 = 1 \qquad (2.3 - 20)$$

There are six of the form

$$l_1 l_2 + m_1 m_2 + n_1 n_2 = 0, \qquad (2.3 - 21)$$

one for each pair of rows and one for each pair of columns. And there are nine of the form

$$l_1 = m_2 n_3 - m_3 n_2, \qquad (2.3 - 22)$$

one for each of the nine elements.

If p, q, r are eliminated from equations (2.3 - 14, 15, 16), one obtains

$$l_1 \dot{l}_1 + m_1 \dot{m}_1 + n_1 \dot{n}_1 = 0 \qquad (2.3 - 23)$$

which of course can be obtained from equation (2.3 - 20),

$$l_1^2 + m_1^2 + n_1^2 = 1. \qquad (2.3 - 20)$$

if r and q are eliminated from the three equations

$$\dot{l}_1 = r m_1 - q n_1 \qquad (2.3 - 11$$

$$\dot{l}_2 = r m_2 - q n_2 \qquad (2.3 - 14)$$

$$\dot{l}_3 = r m_3 - q n_3 \qquad (2.3 - 17)$$

the equation that results is

$$(m_2 n_3 - m_3 n_2)\dot{l}_1 + (m_3 n_1 - m_1 n_3)\dot{l}_2 + (m_1 n_2 - m_2 n_1)\dot{l}_3 = 0. \quad (2.3 - 2$$

By means of identities of the form

$$l_1 = m_2 n_3 - m_3 n_2, \qquad (2.3 - 22)$$

equation (2.3 - 21) reduces to

$$l_1 \dot{l}_1 + l_2 \dot{l}_2 + l_3 \dot{l}_3 = 0 \qquad (2.3 - 24)$$

which of course can be derived from

$$l_1^2 + l_2^2 + l_3^2 = 1, \qquad (2.3 - 25)$$

one of the identities mentioned above, see equation (2.3 - 20).

To choose three equations independent of transformation matrix identities, choose equations all of which do not have the same subscripts nor do all involve the time derivative of the same letter.

For example

$$\dot{l}_1 = rm_1 - qn_1 \qquad (2.3-13)$$

$$\dot{l}_2 = rm_2 - qn_2 \qquad (2.3-14)$$

$$\dot{m}_2 = pn_2 - rl_2 \qquad (2.3-15)$$

is a set of independent equations which with six of the usual identities can be used to obtain all nine direction cosines. If desired, more than three differential equations and fewer than six identities can be used. For some applications, such as analogue computers, the differential equations are more suitable than the identities. Even for digital computers the differential equations may be preferred to identities of the form of equations (2.3 - 20) and (2.3 - 21).

2.4 Computation Procedure for Direction Cosines

We have shown that the nine direction cosines in a rotational matrix satisfy twenty-one identities of which only six are independent. We have shown also that there are nine kinematic differential equations of motion, of which three are independent. As a grand total there are thirty equations for nine unknown direction cosines. It is planned to make use of the great wealth of equations to check on the computations and make adjustments as time goes on. The idea is that the twenty-one identities should be obeyed at the expense of satisfying the nine differential equations.

Before discussing the plan of computation, it will be well to discuss a fundamental change that has been made in the matrix. Some of the direction cosines will be computed by integrations. When a

cosine is large, the computation error may give a result greater than unity. Since there is no unity, this number cannot be held in the machine. Indeed, if a direction cosine were computed correctly to be unity, it would appear in the machine as zero. To remove this difficulty, we use one-half the direction cosine in each case. Therefore the identities are of the form

$$l_3^2 + m_3^2 + n_3^2 = \frac{1}{4} \qquad \text{(6 of these)} \qquad (2.3 - 20a)$$

$$l_3 l_2 + m_3 m_2 + n_3 n_2 = 0 \qquad \text{(6 of these)} \qquad (2.3 - 21)$$

$$l_1 = 2(m_2 n_3 - m_3 n_2) \qquad \text{(6 of these)} \qquad (2.3 - 22a)$$

We first use three differential equations to compute l_3, m_3, n_3. We assume that the relative magnitudes are correct and that we must perhaps apply a percentage correction to normalize them to M (which may be unity, one-half or any number), according to the equation

$$l_3^2 + m_3^2 + n_3^2 = M^2 \qquad (2.4 - 1)$$

To explain the method of normalization, let the computed values be l'_3, m'_3, n'_3. Define ε by the equation:

$$l'^2_3 + m'^2_3 + n'^2_3 = M^2 (1 + \varepsilon). \qquad (2.4 - 2)$$

Note that this can be rewritten

$$\frac{l'^2_3}{1 + \varepsilon} + \frac{m'^2_3}{1 + \varepsilon} \quad \frac{n'^2_3}{1 + \varepsilon} = M^2 \qquad (2.4 - 2a)$$

Hence, let

$$l_3^2 = \frac{l_3'^2}{1 + \varepsilon} \quad \text{etc.} \qquad (2.4 - 3)$$

By using two terms in the binomial expansion of $(1 + \varepsilon)^{-\frac{1}{2}}$ we have, sufficiently accurately,

$$\begin{aligned} l_3 &= l_3' \left(1 - \frac{\varepsilon}{2}\right) \\ m_3 &= m_3' \left(1 - \frac{\varepsilon}{2}\right) \qquad (2.4 - 4) \\ n_3 &= n_3' \left(1 - \frac{\varepsilon}{2}\right) \end{aligned}$$

Assuming now that the corrected values of l_3, m_3, and n_3 are the true values, we procede to compute l_2, m_2, and n_2. Let us first consider a rather elegant way of correcting the computed values and then we shall show that a quicker way is almost as elegant. Since we have no way of deciding which is more accurate, we choose to use the quicker method.

Let us compute l_2', m_2', n_2' from three more differential equations. Let us now consider the two vectors.

$$r_3 = \mathbf{i}\, l_3 + \mathbf{j}\, m_3 + \mathbf{k}\, n_3 \qquad (2.4 - 5)$$

$$r_2' = \mathbf{i}\, l_2' + \mathbf{j}\, m_2' + \mathbf{k}\, n_2' \qquad (2.4 - 6)$$

The vector r_3 has a magnitude M. The vector r_2' should have a magnitude M and should be perpendicular to r_3. If r_2' is not perpendicular to r_3 it must be rotated through an angle that will make it so.

We have arbitrarily assumed the computed direction of r_3 to be the true direction. Now it seems reasonable to assume that the computed direction of r_2' is very nearly correct. Then r_2' should be turned through as small an angle as possible to make it perpendicular to r_3. To do this we need look for r_2 in the plane of r_2' and r_3.

$$r_2 = (1 + a)\, r_2' + b\, r_3. \qquad (2.4 - 7)$$

The scalars a and b should be selected to make the magnitude of r_2 equal to M, and the direction normal to r_3. Assuming that the original computations were good, the errors are small and the corrections a and b should be small.

Define ε_1 and ε_2 by the equations

$$l_2'^2 + m_2'^2 + n_2'^2 = M^2 (1 + \varepsilon_1) \qquad (2.4 - 8)$$

$$l_2' l_3 + m_2' m_3 + n_2' n_3 = M^2 \varepsilon_2 \qquad (2.4 - 9)$$

We may write r_2 in the forms

$$r_2 = i\, l_2 + j\, m_2 + k\, n_2$$

$$= i\, [(1 + a)\, l_2' + b\, l_3] + j\, [(1 + a)\, m_2' + b\, m_3]$$

$$+ k\, [(1 + a)\, n_2' + b\, n_3]. \qquad (2.4 - 10)$$

Since r_2 equals M in magnitude

$$[(1 + a) l_2' + b\, l_3]^2 + [(1 + a) m_2' + b\, m_3]^2 + [(1 + a) n_2' + b\, n_2]^2 = M^2$$

$$(2.4 - 11)$$

When this is expanded and use is made of equations (2.4 - 1,8,9) the result is

$$(1 + a)^2 (1 + \varepsilon_1) + 2 (1 + a) \varepsilon_2 + b^2 = 1. \qquad (2.4 - 12)$$

If the errors ε_1 and ε_2 are small, the corrections a and b will be small and we assume that it will be sufficiently exact to neglect second degree terms in a, b, ε_1, ε_2. We therefore obtain

$$2 a + \varepsilon_1 = 0 \qquad (2.4 - 13)$$

$$a = - \frac{\varepsilon_1}{2} \qquad (2.4 - 14)$$

To obtain a value of b we make use of the fact that r_2 is normal to r_3. Therefore

$$[(1 + a) l'_2 + b l_3] l_3 + [(1 + a) m'_2 + b m_3] m_2 + [(1 + a) n'_2 + b n_3] n_3 = 0. \qquad (2.4 - 15)$$

which reduces to

$$(1 + a) M^2 \varepsilon_2 + b M^2 = 0. \qquad (2.4 - 16)$$

Neglecting a ε_2 which is a second degree term,

$$b = - \varepsilon_2. \qquad (2.4 - 17)$$

As a result, the corrected values may now be written

$$\begin{aligned} l_2 &= l'_2 \left(1 - \frac{\varepsilon_1}{2}\right) - l_3 \varepsilon_2 \\ m_2 &= m'_2 \left(1 - \frac{\varepsilon_1}{2}\right) - m_3 \varepsilon_2 \\ n_2 &= n'_2 \left(1 - \frac{\varepsilon_1}{2}\right) - n_3 \varepsilon_2 \end{aligned} \qquad (2.4 - 18)$$

The correction involving ε_1 adjusts the magnitude while the correction involving ε_2 corrects the direction of r_2. The results obtained show that to the first order either correction may be applied first or they may both be applied together.

As mentioned above, there is a second method of orthogonalizing that we find preferable because it is quicker. In this method we assume that the computed values of two of l_2, m_2, n_2 are correct and adjust the third to make r_2 perpendicular to r_3. Assume that n_3 is greater than either l_3 or m_3, then we compute l_2 and m_2 by integrating the two differential equations; we then determine n_2 from

$$n_2 = \frac{l_2 l_3 + m_2 m_3}{-n_3} \qquad (2.4 - 19)$$

Note that if we had computed n_2' from the differential equation we would have

$$l_2 l_3 + m_2 m_3 + n_2' n_3 = \varepsilon_3 M^2 \qquad (2.4 - 20)$$

But

$$l_2 l_3 + m_2 m_3 + n_2 n_3 = 0 \qquad (2.3 - 21)$$

Therefore

$$n_2 = \frac{n_2' n_3 - \varepsilon_3 M^2}{n_3} = n_2' - \frac{\varepsilon_3 M^2}{n_3} . \qquad (2.4 - 21)$$

What we are doing amounts to making a correction of

$$-\frac{\varepsilon_3 M^2}{n_3}$$

to a computed value n_2'.

Obviously if we decided to correct l_2' instead of n_2' the correction would be

$$-\frac{\varepsilon_3 M^2}{l_3}$$

To make the correction small we make sure that the largest of l_3, m_3, n_3 appears in the denominator. This will also prevent us from being embarrassed by a zero in the denominator as might occur if we decided to correct n_2 under all circumstances.

After the direction of r_2 has been established perpendicular to r_3, we must now adjust its magnitude. We procede as we did for r_3 above.

Let $$l_2'^2 + m_2'^2 + n_2'^2 = M^2 (1 + \varepsilon_4) \qquad (2.4 - 22)$$

$$l_2 = l_2' \left(1 - \frac{\varepsilon_4}{2}\right) \qquad (2.4 - 23)$$

etc.

The three remaining unknown direction cosines are computed from the formulae

$$\begin{aligned} l_1 &= \frac{m_2 n_3 - m_3 n_2}{M} \\ m_1 &= \frac{n_2 l_3 - n_3 l_2}{M} \\ n_1 &= \frac{l_2 m_3 - l_3 m_2}{M} \end{aligned} \qquad (2.4 - 24)$$

which do not introduce any further inaccuracy.

The method of adjusting the integrated direction cosines to ensure that the twenty one identities included in equations (2.3 - 20, 21, 22) are satisfied does not make a corresponding correction on the integrands. We accept the fact that the direction cosines need correction; therefore we should agree that the integrands need correction. However we save several incorrect integrands to determine subsequent direction cosines instead of using the corrected integrals.

If this normalization and orthogonalization were performed at each step there would be a small discontinuity at each step. If correction is made only once in a number of cycles, there will be a greater discontinuity which may introduce instability into the system. This difficulty is intimately connected with the overall problem of integration and no satisfactory answer to the problem has as yet been devised (see Section 4).

2.5 Motion on the Ground*

When the airplane is on the ground the reactions of the ground on the wheels are in general indeterminate. We assume that there will be enough force in the Y direction to overcome any tendency the plane has to move in the Y direction. Therefore instead of computing Y_t we arbitrarily assume that v equals 0. The upward force on the three (or two) wheels is sufficient to balance the downward force of gravity and (positive) Z_t. As long as the plane is being supported by a ground

* It has been assumed that the runway or carrier deck is horizontal.

reaction, the resultant velocity will be horizontal and it is simpler to use a set of axes fixed to the earth instead of the plane axes as reference. The upward reaction of the earth on the wheels, which cannot be computed easily, has no component in the horizontal direction and therefore does not enter into the computation of V_t.

Similarly, while the airplane has three wheels on the ground there will be an indeterminate moment supplied by the reactions of the earth on the wheels which will prevent the plane from nosing down. Otherwise the front wheel would dig a trench in the ground. Now, the two back wheels are only a short distance behind the center of mass, therefore it is possible for the front wheel to be raised and the airplane run on the ground on only two wheels. This is the condition just before take-off and immediately on landing in the ideal case.

The two rear wheels are reacted upon by the earth in such a way that any tendency for rotation about the horizontal axis determined by V_t is cancelled. If the angular velocities p, q, r were to be computed first, it would be necessary to determine just what unbalance exists in the reactions of the earth on the wheels. It is simpler to assume that any remaining rotation of the airplane will be about a vertical axis and to compute only torques about it. The unbalanced ground reactions do not produce torques about the vertical axis and therefore do not enter the equations.

In addition to the in-flight equations of motion, equations have been derived for the case of two wheels on the ground and three wheels on the ground. The two-wheel equations involve V_t, q, and $\dot{\psi}$

while the three-wheel equations involve only V_t and $\dot{\psi}$. The angular velocity q involves a rocking motion on the rear wheels which is impossible as long as the front wheel is on the ground.

An inspection is made to determine whether there is a moment about the Y axis sufficient to lift the front wheel. This decision leads to either the two or three-wheel equations.

An inspection is made to determine whether a negative Z_t is available to lift the plane. This is the manner of deciding whether to use the in-flight equations or the ground equations.

When landing, if the computed value of altitude is equal to or less than the altitude of the landing field, the plane is on the ground. If the computed value of the altitude is less than the altitude of the landing field, a landing bump is experienced in which the computed value of the altitude is replaced by the altitude of the landing field. At the same time that the plane reaches the ground, it is required that the two rear wheels strike the ground together, which may produce another bump. The direction cosine m_3 is replaced by zero if the computed value of m_3 is different from zero.

If the computed value of l_3 indicates that the front wheel is off the ground then two-wheel equations will be used. However if the computed value of l_3 indicates that the front wheel is on the ground then 3 -wheel equations will be used. If the computed value of l_3 indicates that the front wheel is in a trench, then a front wheel bump is experienced in which the computee value of l_3 is replaced by the value corresponding to three-wheel contact and three-wheel equations are used.

For some comments regarding the effect that the changing of variables during landing and take-off has on the integration process, see Section 5.

2.6 Plotting Board

Part of the problem to be solved by the computer is the displacement of the airplane from its starting point. For example a training problem may be to fly from Washington to Newark. In order to locate the plane on the plotting board the speed to the north $\dot{x}_o$ and the speed to the east $\dot{y}_o$ are integrated with respect to time. Remembering that the digital computer holds only proper fractions it is necessary to adjust the numbers by introducing R_x and R_y as maximum expected displacements.

We could equally well use polar coordinates say with the origin at the point of starting the problem. However if the plane circled the field several times the angle coordinate would appear to increase giving several coordinate pairs for the same point in space. The digital computer, it is interesting to note, takes care of this apparent difficulty automatically. If we let the angular coordinate be equal to

$$\frac{\psi}{2\pi},$$

and if ψ were to increase to 2π the number stored would be equal to zero. There would be a duplication however because values of the angle coordinate corresponding to ψ between -2π and $+2\pi$ could be stored. This cuts down the number of coordinate pairs representing

the same point only to two.

Because of this remnant duplication and because of the fact that the equation of a straight line, which is needed in radio range representation, is so much more complicated in polar coordinates than in rectilinear coordinates, no further consideration is given to pola coordinates.

Knowing the amount of fuel the airplane is carrying an estimat of maximum cruising range can be made. If R_x and R_y are slightly mor than this maximum range then a computed value of x_o or y_o greater than 1/2 would indicate that the airplane could not return to point of take-off. However, because of manoeuvres and possible directions of travel the fact that both x_o and y_o are less than 1/2 does not imply that sufficient fuel remains to return.

To include radio range simulation it is necessary to know the location of the range transmitter and the direction of the beam. For an omnidirectional radio range the instrument information can be supplied by knowing the location of the range transmitter and the location of the airplane.

For radio compass indication it is necessary that the location of the transmitter and the heading of the plane be known. The direction cosines l_1 and l_2 contain the information regarding the heading.

If there is no wind x_o and y_o give the speed of the plane relative to both the air and the ground. However, if there is a wind the wind components should be taken into account.

To do this x_o should be increased by the north component of

wind velocity and $\dot{y}_o$ should be increased by the east component of wind velocity. Under these conditions, for long distance flight, it might be necessary to use larger values of R_x and R_y.

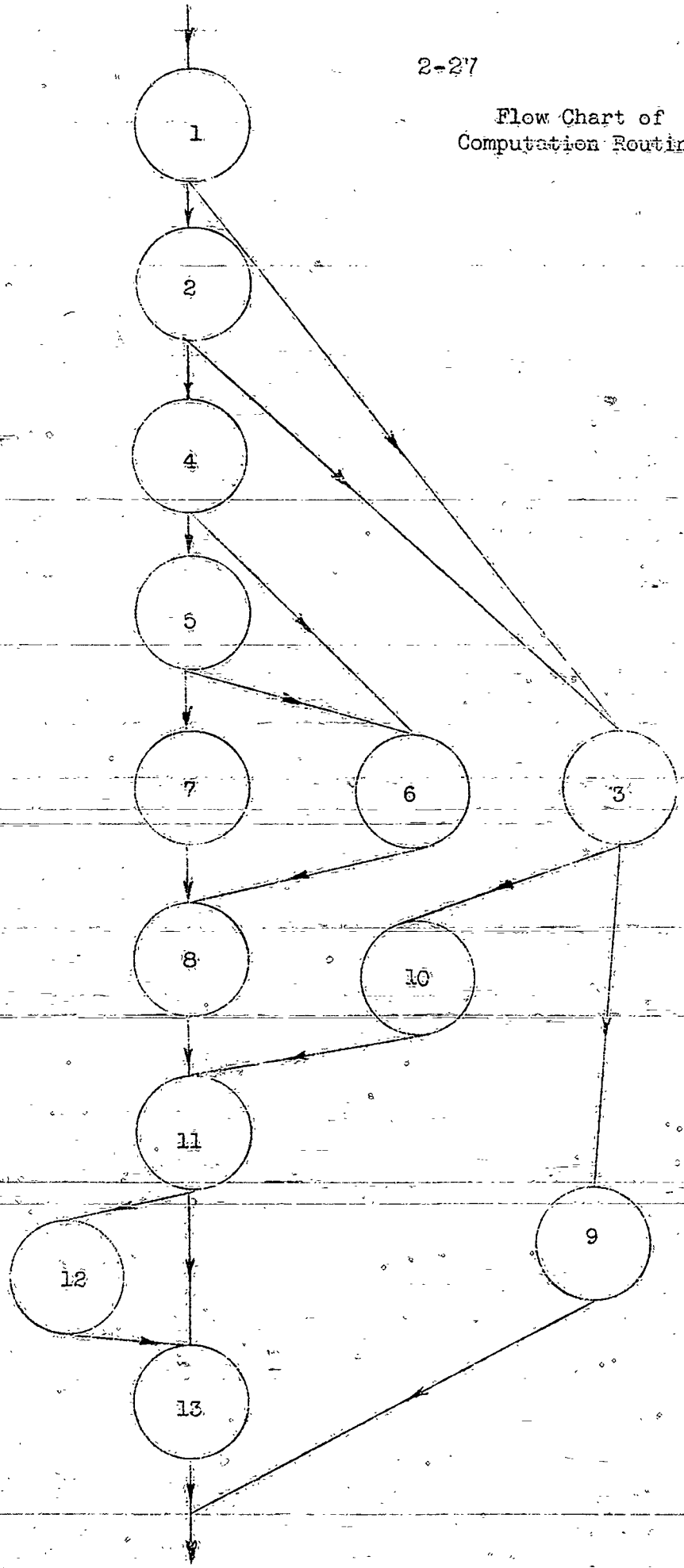
Inclusion of the effect of wind in the ground equations is complicated by the fact that not only may v be different from zero but v may become much larger than u and w. However, if take-off and landing always are into the wind then the problem becomes somewhat simpler. A take-off or landing across wind would be disastrous to the airplane, and if these possibilities are eliminated the problem can be handled by the method of Durand by assuming that the cross-wind v has no effect on the lift and drag of the wings.

At this time equations have not been derived to simulate radio ranges and radio compass indications. Neither has the effect of wind on the ground equations been taken into account.

2.7 List of Equations

A complete list of the equations divided into a number of subroutines follows.

Flow Chart of
Computation Routine



CONVERTED INFORMATION

$\delta_e < 0.91$

$\delta_a < 0.472$

$\delta_r < 0.091$

$\delta_{fd} < 0.28$

$K_G = 0 \text{ or } 0.2651$

$K_F = 0 \text{ or } 0.9492$

$K_N = 0 \text{ or } 0.03107$

$S_i < 0.0002665$

$S_{Fb} = 0 \text{ or } 0.001073$

$S_{Fb}'' = 0 \text{ or } 0.004453$

$S_{Fb}''' = 0 \text{ or } 0.004453$

$S_{FW} = 0 \text{ or } 0.004448$

$S_{FW}' = 0 \text{ or } 0.004448$

$S_{FW}'' = 0 \text{ or } 0.004448$

$S_a = 0 \text{ or}$

Note:- We assume that motor is always supplied from the body tanks.

TRANSFORMATIONS

Variables marked with the star are ERCC or actual values

$\dot{p}^* = 32\,\dot{p}$ $\qquad$ $p^* = 8\,p$

$\dot{q}^* = 16\,\dot{q}$ $\qquad$ $q^* = q$

$\dot{r}^* = 8\,\dot{r}$ $\qquad$ $r^* = r$

$\dot{u}^* = 4\,g\,\dot{v}$ $\qquad$ $u^* = 40\,g\,u$

$\dot{v}^* = 8\,g\,\dot{v}$ $\qquad$ $v^* = 20\,g\,v$

$\dot{w}^* = 16\,g\,\dot{w}$ $\qquad$ $w^* = 20\,g\,w$

$V_t^* = 40\,g\,V_t$

$\dot{h}^* = 40\,g\,\dot{h}$ $\qquad$ $h^* = 1280\,g\,h$

$L_t^* = 128\,W_M\,V_t\,L_t$ $\qquad$ $I_x^* = 4\,W_M\,I_x$

$M_t^* = 256\,W_M\,V_t\,M_t + 0.2613\,K_F\,W_M\,T$ $\qquad$ $I_y^* = 4\,W_M\,I_y$

$N_t^* = 256\,W_M\,V_t\,N_t$ $\qquad$ $I_z^* = 4\,W_M\,I_z$

$X_t^* = 4\,W_M\,V_t\,X_t$ $\qquad$ $W^* = W_M\,W$

$Y_t^* = 8\,W_M\,V_t\,Y_t$ $\qquad$ $Y^* = 8\,W_M\,Y$

$Z_t^* = 128\,W_M\,V_t\,Z_t$ $\qquad$ $Z^* = 128\,W_M\,Z$

$l_1^* = 2\,l_1$ $\qquad$ $m_1^* = 2\,m_1$ $\qquad$ $n_1^* = 2\,n_1$

$l_2^* = 2\,l_2$ $\qquad$ $m_2^* = 2\,m_2$ $\qquad$ $n_2^* = 2\,n_2$

$l_3^* = 2\,l_3$ $\qquad$ $n_3^* = 2\,m_3$ $\qquad$ $n_3^* = 2\,n_3$

$\alpha^* = \frac{180}{\pi}\,\alpha$ $\qquad$ $\sin\alpha = \frac{w}{2V_t}$ $\qquad$ $\cos\alpha = \frac{u}{V_t}$

$\dot{x}_o^* = 40\,g\,\dot{x}_o$ $\qquad$ $\Delta x_o^* = R_x\,\Delta x_o$

$\dot{y}_o^* = 40\,g\,\dot{y}_o$ $\qquad$ $\Delta y_o^* = R_y\,\Delta y_o$

TRANSFORMATIONS

$T^* = W_M T$

$F_{BR}^* = 0.03232\ W_M F_{BR}$

$F_{BL}^* = 0.03232\ W_M F_{BL}$

$\dot{\psi}^* = \dot{\psi}$

R_x = range in x_o direction (North)

R_y = range in y_o direction (East)

$\delta_{fd} = 0.003728\ \delta^*_{fd}$	$\delta^*_{fd} < 75^\circ$	$\delta_{fd} < 0.28$
$\delta_a = 0.03144\ \delta^*_a$	$\delta^*_a < 15^\circ$	$\delta_a < 0.472$
$\delta_r = 0.003639\ \delta^*_r$	$\delta^*_r < 25^\circ$	$\delta_r < 0.091$
$\delta_e = 0.01811\ \delta^*_e$	$\delta^*_e < 50^\circ$	$\delta_e < 0.91$ (?)

$W_{Fb}^* = 15000\ W_{Fb}$

$W_{Fw}^* = 5400\ W_{Fw}$

$W_a^* = 6655\ W_a$

$d^* = d$

$n^* = 15000\ n$

$n_o^* = 15000\ n_o$

$\phi_t^* = 80\ \phi_t$

SUBROUTINE NO. ONE

$$\Delta x_o = \frac{40\,g}{R_x} \int \dot{x}_o \, dt$$

$$\Delta y = \frac{40\,g}{R_y} \int \dot{y}_o \, dt$$

$$b_{ix} = \int S_i \, dt$$

$$\Delta W_{fb} = -\int [S_{Fb} - 0.3600\, S_{FW} + S''_{Fb} - S'''_{Fb}] \, dt$$

$$\Delta W_{fw} = -\int [S_{FW} + S'_{FW} - S''_{FW}] \, dt$$

$$\Delta W_a = -\int S_A \, dt$$

$$W = 0.613 + W_{fb} + 0.3600\, W_{fw} + 0.4437\, W_a + 0.04171\, b_{ix}$$

$$d = \frac{0.32 - 0.651\, W_{fb} + 0.3571\, W_{fW} - W_a}{W}$$

$$I_x = 0.1283 + 0.9942\, W_{fW}$$

$$I_y = 0.3083$$

$$I_z = 0.400 + W_{fW}$$

$$M_a = V_t + 0.1471\, h\, V_t$$

SUBROUTINE NO. ONE

For $0 \leq M_a \leq 0.4613$	$m = 0.623 + 0.7154\ M_a$
$0.4613 < M_a \leq 0.6962$	$m = 0.656 + 0.6769\ (M_a - 0.4613)$
$0.6962 \leq M_a$	$m = 0.815 - (0.8486)\ 2^{3}\ (M_a - 0.6962$
For $0 \leq M_a \leq 0.6092$	$a = 0.1052$
$0.6092 < M_a \leq 0.6962$	$a = 0.1052 + 0.08851\ (M_a - 0.6092)$
$0.6962 < M_a$	$a = 0.1822 + 0.9025\ (2^{3})\ (M_a - 0.6962)$
For $0 \leq M_a \leq 0.5222$	$b = 0.2727\ M_a$
$0.5222 < M_a < 0.6266$	$b = 0.6820\ (2)\ (0.6266 - M_a)$
$0.6266 \leq M_a$	$b = 0$
For $0 \leq M_a \leq 0.3481$	$c = 0.0977 + 0.4519\ M_a$
$0.3481 < M_a$	$c = 0.2550 + 0.9224\ (M_a - 0.3481)$
For $0 \leq M_a \leq 0.6962$	$C_{m_1} = 0.02504$
$0.6962 < M_a$	$C_{m_1} = 0.02504 + 0.4087\ (M_a - 0.6962)$
For $0 \leq M_a \leq 0.4569$	$C_{m_2} = 0.1936 + 0.3763\ M_a$
$0.4569 < M_a \leq 0.6962$	$C_{m_2} = 0.4569 - 0.645\ (2)\ (M_a - 0.4569)$
$0.6962 < M_a$	$C_{m_2} = 0.1483 + 0.961\ (2)\ (M_a - 0.6962)$

$C_{L_1} = 2\ ma$

$C_{L_2} = 0.7301 - 0.7093\ b_{ix} - C_{L_1} - 0.03273\ K_F + K_N$

If $C_{L_1} < C_{L_2}$	If $C_{L_1} \geq C_{L_2}$
$C_L = C_{L_1}$	$C_L = C_{L_2}$
$C_{m_3} = -0.4966\ \alpha$	$C_{m_2} = f\ (M_a)$
	$C_{m_3} = C_{m_2} - 1.1588\ \alpha$
$\mp 0.4242\ p = -0.4242\ p$	$\mp 0.4242\ p = +0.4242\ p$

SUBROUTINE NO. ONE

$$X_t = -\epsilon^{-kh}\{(a + 0.5006\, b_{ix} + \delta_{fd} + 0.8727\, K_F + K_G)\, V_t + 0.2065 \mid v \mid + [(b_{ix} + c)\, 2^5\, a - b + K_F]\, w\}$$

$$Z_t = -\epsilon^{-hk}\{C_L + 0.1118\, K_F\}\, V_t$$

$$M_t = \epsilon^{-hk}\{(C_{m_i} + C_{m^3} - \delta_e - 0.08549\, \delta_{fd} + 0.06291\, K_F + 0.1860\, K_N + 0.1780\, K_G)\, V_t - (0.01721 + 0.03395\, M_a)\, q\}$$

$$N_t = \epsilon^{-hk}\{-\delta_r\, V_t - 0.2448\, v - 0.009663\, r - 0.03964\, p\, \alpha\}$$

Stick forces =

If $h < 0.364$	If $h \geq 0.364$
$n_o = 0.173 + 0.0735\, h$	$n_o = 0.548\, h$

If $\phi_t < 0.267$

$\Delta n = \frac{1}{2} \int (n_o - n)\, dt$

If $\phi_t \geq 0.267$

If $\phi_t < 0.394$	If $\phi_t \geq 0.394$
$D = 0.8\, \phi_t - 0.215$	$D = 0.0025 + 0.2475\, \phi_t$

$\Delta n = \frac{1}{2} \int [(n_o - n) + 4D\,(0.973 - n_o)\, dt]$

If $n \leq 0.332$

$H = 0$

If $n > 0.332$

If $n < 0.664$	$n \geq 0.664$
$H = \frac{n - 0.332}{0.664}$	$H = 0.5$

If $V_t \leq 0.338$	If $V_t > 0.338$
$R = 0.00296 - 0.865\, V_t$	$R = 0.00264 - 0.778\, V_t$

$E = 0.1328\, n^3$

$T = 4\,(HR + E + J)\, \epsilon^{-hk}$

If $h > h_o$	If $h \leq h_o$
go to Subroutine No. Three	go to Subroutine No. Two

SUBROUTINE NO. TWO

Plane on ground; take-off decision

$$A = \frac{W + 2T\ 1_3}{128}$$

Compute and store $Z_t\ V_t$

If	If
$-Z_t\ V_t > A$	$-Z_t\ V_t \leqq A$
go to	go to
Subroutine No. Three	Subroutine No. Four

SUBROUTINE NO. 3a

$$\dot{p} = \tfrac{1}{4} (\ddot{\psi}\, l_3 + \dot{\psi}\, \dot{l}_3)$$

$$\dot{r} = 2 (\ddot{\psi}\, n_3 + \dot{\psi}\, \dot{n}_3)$$

$$\dot{u} = 2 (\dot{V}_t\, n_3 + V_t\, \dot{n}_3)$$

$$\dot{v} = 0$$

$$\dot{w} = -\left[(\dot{V}_t\, l_3 + V_t\, \dot{l}_3)\right] 2^2$$

$$\dot{m}_3 = 0$$

$$\dot{l}_2 = -2(\dot{m}_1\, n_3 + m_1\, \dot{n}_3)$$

$$\dot{m}_2 = (\dot{m}_1\, \dot{\psi} + m_1\, \ddot{\psi})$$

$$\dot{n}_2 = 2 (\dot{l}_3\, m_1 + \dot{m}_1\, l_3)$$

$$\dot{h} = 0$$

This subroutine lies between Subroutine No. Two and Subroutine No. Three.

SUBROUTINE NO. THREE

$$Y_t = \epsilon^{-hk} \quad 2\ (0.6583v)$$

$$L_t = \epsilon^{-hk} \quad \{ (\delta_a + 0.1411\ \delta_r)\ V_t + 0.3916\ v \mp 0.4242\ p + 0.09857\ \epsilon_L\ r \}$$

$$\Delta p = 4 \int \frac{(uL_t - wN_t) + 2^{-5}\ (I_y - I_z)\ q\ r}{I_x}\ dt$$

$$Z = \frac{X_t\ w}{2^6} + Z_t\ u$$

$$\Delta q = 16 \int \frac{2^2\ M_t\ V_t - 2\ d\ Z + 2^{-4}\ (0.06532)\ K_F\ T + (I_z - I_x)\ \frac{pr}{2}}{I_y}\ dt$$

$$\Delta r = 8 \int \frac{2^3\ (2^{-2}\ w\ L_t + u\ N_t) + (I_x - I_y)\ qp - 2^{-2}\ d\ V_t\ Y_t}{I_z}\ dt$$

$$\Delta u = \frac{1}{10} \int \left[\frac{(X_t\ u + 2^{-2}\ T) - 8\ (0.625\ W\ [wq - vr]) - 2^4\ Z_t\ w}{W} + \frac{1_3}{2} \right] dt$$

$$\Delta v = \frac{2}{5} \int \left[\frac{Y_t\ V_t}{W} + \frac{m_3}{4} - 2^6\ (0.15625\ ur - 0.625\ wp) \right] dt$$

$$\Delta w = \frac{4}{5} \int \left[\frac{2^{-3}\ X_t w - 2^4\ (0.625 W\ [vp - 2^{-2}\ uq]) + 8 Z_t u}{W} + \frac{n_3}{8} \right] dt$$

$$V_t = u + \frac{1}{8}\ \frac{w^2 + v^2}{u}$$

$$1_3 = \int (\ r\ m_3 - q\ n_3\ )\ dt$$

$$m_3 = 8 \int (\ p\ n_3 - \frac{1}{8}\ r\ 1_3\ )\ dt$$

$$n_3 = 8 \int (\frac{1}{8}\ q\ 1_3 - p\ m_3\ )\ dt$$

$$\epsilon_3 = (\ 1_3^2 + m_3^2 + n_3^2\ ) - \frac{1}{4}$$

$$\text{New } 1_3 = 2\ 1_3\ (\tfrac{1}{2} - \epsilon_3)$$

$$\text{New } m_3 = 2\ m_3\ (\tfrac{1}{2} - \epsilon_3)$$

$$\text{New } n_3 = 2\ n_3\ (\tfrac{1}{2} - \epsilon_3)$$

SUBROUTINE NO. THREE

If $\lvert n_3 \rvert < \lvert l_3 \rvert$		If $\lvert n_3 \rvert \geq \lvert l_3 \rvert$	
If $\lvert l_3 \rvert > \lvert m_3 \rvert$	$\lvert m_3 \rvert \geq \lvert l_3 \rvert$	$\lvert m_3 \rvert \geq \lvert n_3 \rvert$	$\lvert n_3 \rvert > \lvert m_3 \rvert$
$m_2 = 8\int (pn_2 - \frac{rl_2}{8})\ dt$	$l_2 = \int (rm_2 - qn_2)\ dt$	$l_2 = \int (rm_2 - qn_2)\ dt$	
$n_2 = 8\int (\frac{ql_2}{8} - pm_2)\ dt$	$n_2 = 8\int (\frac{ql_2}{8} - pm_2)\ dt$		$m_2 = 8\int (pn_2 - \frac{rl_2}{8})\ dt$
$l_2 = -\frac{m_2 m_3 + n_2 n_3}{l_3}$	$m_2 = -\frac{l_2 l_3 + n_2 n_3}{m_3}$		$n_2 = -\frac{l_2 l_3 + m_2 m_3}{n_3}$

$$\epsilon_2 = (l_2^2 + m_2^2 + n_2^2) - \tfrac{1}{4}$$

New $l_2 = 2\, l_2\ (\frac{1}{2} = \epsilon_2)$

New $m_2 = 2\, m_2\ (\frac{1}{2} - \epsilon_2)$

New $n_2 = 2\, n_2\ (\frac{1}{2} - \epsilon_2)$

$$l_1 = 2\,(m_2 n_3 - m_3 n_2)$$

$$\dot{h} = -[2\, u\, l_3 + (v\, m_3 + w\, n_3)]$$

$$\Delta h = 2^{-6} \int \dot{h}\ dt$$

If $h > h_c$	If $h \leq h_c$
go to Subroutine No. Nine	go to Subroutine No. Ten

SUBROUTINE NO. FOUR

Plane on ground, brake equations and landing decision No. 2.

$$T_B = F_{BR} - F_{BL}$$

$$F_B = 0.2423\ (F_{BR} + F_{BL})$$

If $l_s < -\frac{1}{2} \sin 5^\circ$ go to Subroutine No. Six

If $l_s \geq -\frac{1}{2} \sin 5^\circ$ go to Subroutine No. Five

SUBROUTINE NO. FIVE

Plane on ground, decision on raising front wheel

$$B = \frac{W}{256} + \frac{1}{2} V_t Z_t + \frac{1}{8} pr (I_z - I_x) - \frac{0.06532 K_F}{64}$$

Compute and store $V_t M_t$

If $V_t M_t > B$

go to Subroutine No Six

If $V_t M_t \leq B$

go to Subroutine No. Seven

SUBROUTINE NO. SIX

Plane on two wheels

$$\Delta V_t \equiv \int \frac{V_t Y_t - \frac{1}{4} TN_3 - F_B}{10 W} dt$$

$$\Delta \dot{\psi} = \int \frac{64 V_t N_t + T_B - 2 r q (I_y - I_x)}{I_z n_3^2 + I_x 1_3^2} dt$$

$$\Delta q = \int \frac{64 M_t V_t - 8 p r (I_x - I_z) - (\frac{1}{4}W + 32 V_t Z_t) - 0.06532 K_F T}{I_y} dt$$

$$\Delta 1_3 = -\frac{1}{2} \int q \, dt$$

$$n_3 = \frac{1}{2} - 1_3^2$$

$$\alpha = -2 1_3$$

Go to Subroutine No. Eight

SUBROUTINE NO. SEVEN

Plane on three wheels

$$\Delta v_t = \int \frac{0.1 W_t X_t + 0.25 T \left(\frac{1}{2} \cos 5^\circ\right) - 0.1 F_B}{W} dt$$

$$\Psi \qquad \Delta\dot{\Psi} = \int \frac{64 V_t N_t + T_B}{I_z \left(\frac{1}{2} \cos 5^\circ\right)^2 + I_x \left(\frac{1}{2} \sin 5^\circ\right)^2} dt$$

Go to Subroutine No. Eight

SUBROUTINE NO. EIGHT

Plane on two or three wheels

$$\Delta m_2 = \int m_1 \dot{\psi}\, dt$$

$$\Delta m_1 = -\int m_2 \dot{\psi}\, dt$$

$$\epsilon \equiv \left(m_1^2 + m_2^2 \right) - \frac{1}{4}$$

$$\text{New } m_1 = 2m_1 \left(\frac{1}{2} - \epsilon \right)$$

$$\text{New } m_2 = 2m_2 \left(\frac{1}{2} - \epsilon \right)$$

Go to subroutine No. Eleven.

SUBROUTINE NO. NINE

Plane in flight, miscellaneous

$$m_1 = 2 (n_2 l_3 - n_3 l_2)$$

$$n_1 = 2 (l_2 m_3 - l_3 m_2)$$

$$\dot{x}_0 = 2 l_1 u + (m_1 v + n_1 w)$$

$$\dot{y}_0 = 2 l_2 u + (m_2 v + n_2 w)$$

$$\alpha = \frac{w}{2 V_t}$$

For $0 \leq h \leq 0.3$

$$\epsilon^{-hk} = 0.67 + 0.55 (0.6 - 2 h)$$

For $0.3 < h \leq 0.7$

$$\epsilon^{-hk} = 0.88 - 0.7 h$$

For $0.7 < h < 1$

$$\epsilon^{-hk} = 0.67 - 0.4 h$$

Go to Subroutine No. One

SUBROUTINE NO. TEN

Landing bump.

$$h = h_o$$

$$v = 0$$

$$m_3 = 0$$

$$n_3 = \tfrac{1}{2} - l_3^2$$

$$m_1 = -\tfrac{1}{2}\,\frac{l_2}{n_3}$$

$$m_2 = \tfrac{1}{2}\,\frac{l_1}{n_3}$$

$$\dot{\psi} = \tfrac{1}{2}\,\frac{r}{n_3}$$

Go to Subroutine No. Eleven

SUBROUTINE NO. 10a

$$\dot{V}_t = \frac{1}{2}\left(\frac{n_3\,\dot{u} - \dot{n}_3\,u}{n_3^2}\right)$$

$$\ddot{\psi} = \frac{1}{2}\left(\frac{n_3\,\dot{r} - r\,\dot{n}_3}{n_3^2}\right)$$

$$\dot{l}_3 = -\frac{1}{2}\,q$$

$$\dot{\psi} = \frac{1}{2}\,\frac{r}{n_3}$$

$$\dot{m}_2 = m_1\,\dot{\psi}$$

$$\dot{m}_1 = m_2\,\dot{\psi}$$

This subroutine lies between Subroutine No. Three and Subroutine No. 10.

SUBROUTINE NO. ELEVEN

Decision on front wheel bump

If $1_3 > -\frac{1}{2}\sin 5^\circ$ go to Subroutine No. Twelve

If $1_3 \leq -\frac{1}{2}\sin 5^\circ$ go to Subroutine No. Thirteen

SUBROUTINE NO. TWELVE

Front wheel bump.

$q = 0$

$l_3 = -\frac{1}{2} \sin 5^\circ$

$n_3 = \frac{1}{2} \cos 5^\circ$

$a = \sin 5^\circ$

Go to Subroutine No. Thirteen

SUBROUTINE NO. THIRTEEN

$$l_1 = 2 m_2 n_3$$

$$l_2 = -2 m_1 n_3$$

$$n_1 = -2 l_3 m_2$$

$$n_2 = 2 l_3 m_1$$

$$p = \frac{1}{4} \dot{\psi} l_3$$

$$r = 2 \dot{\psi} n_3$$

$$u = 2 V_t m_3$$

$$w = -4 V_t l_3$$

$$\dot{x}_o = 2 V_t m_2$$

$$\dot{y}_o = -2 V_t m_1$$

$$\epsilon^{-hk} = \epsilon^{-h_o k}$$

Go to Subroutine No. One

SECTION 3

COMPUTER CONSIDERATIONS

3. COMPUTER CONSIDERATIONS

The airplane equations used by the Engineering Research Company in the construction of their analogue simulator were presented in the preceding section. The discussion there was confined to the adaptation of the ERCO equations for digital computation. In this section analogue and digital computers will be described and compared with special emphasis on characteristics pertinent to simulator design.

It is instructive first to denumerate the mathematical operations which constitute the computation and solution of the ERCO equations. From the algebraic point of view, these processes may be listed as follows:

1. Addition and subtraction
2. Multiplication and division
3. Evaluation of polynomials
4. Square root extraction
5. Computation of functions, namely, sine and cosine, arc sine and arc tangent, exponential
6. Solution of a set of ordinary non-linear differential equations

Whereas a sharp delineation in order of increasing complexity would be difficult to make, it is clear that the operations (1) and (2) are fundamental and rudimentary while the process (6) is a complex composite routine.

The algebraic notion of operation-complexity is almost immediately applicable to digital computation. On the other hand, it does not apply at all when the computations are mechanized by analogue techniques. Because of the many variations of analogue devices, the discussion below

will be limited to those simulators of which the ERCO system is an example. The description of digital computers will likewise be limited later to special types. It is believed, however, that the basic considerations to be illuminated are almost completely unaffected by this specialization.

3.1 Analysis of Mechanization Requirements

The first decision which must be made when designing a calculating machine is the choice of a device to represent the parameters which occur. These parameters will in general consist of constants, both additive and multiplicative, and of variables, such as a, b, and x, respectively, in the expression $(a + bx)$.

Next it is necessary to invent a compatible mechanism for performing the (algebraically) basic operations, addition, subtraction, multiplication and division, listed as (1) and (2) above.

Finally, it is necessary to devise apparatus for performing the remaining operations. The latter must be divided into two groups which, at least machine-wise, are quite different. The first group consists of the processes (3), (4) and (5), or, more generally, the evaluation of functions, which we shall find is a relatively straight-forward matter. The second group consists of the solution of equations, of which process (6) is an example and which makes considerably more serious demands on computers, both human and machine.

The mechanisms used to perform these operations will now be described in some detail.

3.2 Mechanization of Parameters

In analogue computers (of the ERCO type) parameters are represented as voltages. Technical considerations necessitate restricting the voltage range between upper and lower limits, usually an equal voltage above and below ground, e.g. +50 v. and -50 v. In order to attain maximum precision* for variable parameters, it is usual to "normalize" all variables; that is , the full range of the variable is represented by the full 100 volt range available. Normalization can always be performed by altering multiplicative constants; e.g. if $-100 \leq x \leq 100$ and if $y = 0.1x$ be required, then one can write $y = 0.1x \times \frac{2}{2}$

$= (0.1 \times 2) \times (\frac{x}{2}) = 0.2\ X$ (3.2-1) and $-50 \leq X \leq 50$. (3.2-2)

The representation of variables is effected by means of potentiometers. For example, the variable x is shown in Figure 3.2-1 to be represented by the voltage delivered to the adjustable arm of the potentiometer P.

Because the arithmetic processes are performed by current

* Precision, which is the quality of being sharply defined, should not be confused with accuracy, which is the quality of being free from error. For example, a four-digit number correctly computed is more accurate but less precise than an incorrectly calculated 6-digit number.

A measure of the precision of a representation is the number of distinguishable alternatives from which it was selected; thus, a four digit decimal representation xxxx has a precision of one part in 10^4.

summation in resistors (theoretically of negligible resistance), constants are represented by resistors with large fixed resistance values. For example, in Figure 3.2-1, R_1 represents an additive constant by determining the component $i_1 = \frac{50}{R_1}$ of the current i flowing through the (small) summing resistor r. At the same time R_2 is a multiplicative constant which determines the current i_2 generated by the variable voltage x from from the potentiometer, so that $i_2 = \frac{1}{R_2}x$. Briefly, then, the current $i = i_1 + i_2 = \frac{50}{R_1} + \frac{1}{R_2}x$ represents the function $y = (a + bx$ where $a = \frac{50}{R_1}$ and $b = \frac{1}{R_2}$.

Several significant remarks are now in order. Note first that the variable x, as represented, is a continuous variable; any value of x within the range is theoretically possible. On the other hand, the value of x can never be specified exactly but only as lying within a narrow range of voltage because it is physically impossible to generate infinitely precise voltagee or to manufacture infinitely precise potentiometers. Although precisions up to one part in 10^6 can be obtained in the laboratory by exercising extreme care, the precision of analogue simulators is limited in general to one part in 10^3 or at most 10^4.

Secondly, it should be noted that the mechanization process is one involving an appreciable amount of labor, including the precise measurement of resistors and their connection physically into an electric circuit. Each parameter must be incorporated separately; moreover, if the variable x be required at more than one point in the simulator, it is generally necessary to provide separate resistors to represent the associated multiplicative constant at each point. Briefly, then, an

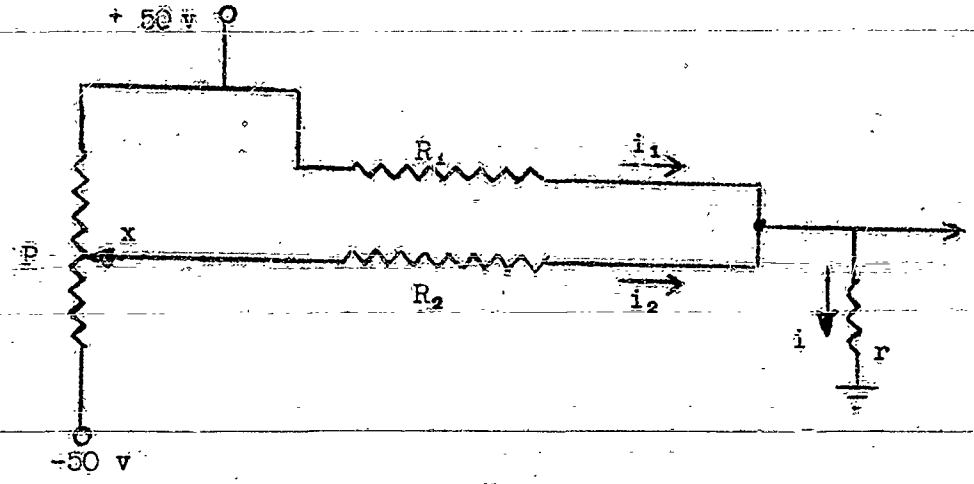


Figure 3.2-1

analogue system is inherently a simultaneous or "parallel" system, one in which a separate unit is provided for every parameter.

As a direct consequence of their parallel construction, analogue simulators lack flexibility. Once a system is assembled to perform a given task, it is restricted to the performance of that task. To alter it to perform an appreciably different task would be an undertaking of considerable magnitude.

In digital computers, parameters are represented by an ordered sequence of digits. In most cases it is understood that each digit is to be multiplied by a number or "weight" according to the position of the digit in the sequence*. Thus the parameter A, which happens to be represented by the sequence $a_n\ a_{n-1}\ a_{n-2}\text{------}a_1\ a_0$, has the magnitude

$$A = \sum_{j=0}^{n} a_j\ w_j \qquad (3.2\text{-}3)$$

where the weights, w_j, are implied from previous information. In most cases, the weights are given as powers of a "radix" r as in the expression

$$w_j = r^{j-c} \qquad j = 0, 1, 2, \text{-----}, n \qquad c = \text{(given) integer} \qquad (3.2\text{-}4)$$

In the decimal system the radix r is 10, each digit a_j is restricted to the set 0 through 9, and the constant c locates the decimal point. Thus, for the decimal number $A_{10} = 63.859$, n = 4, r = 10, c = 3, and $A_{10} = 6 \times 10^2 + 3 \times 10^0 + 8 \times 10^{-1} + 5 \times 10^{-2} + 9 \times 10^{-3}$. (3.2-5)

In most digital computers of recent design, parameters are repre-

* Other codes are discussed in some detail in P3.8.

sented in the binary notation, with radix r = 2. This notation is particularly simple since the digits a_j are restricted to only two possible values "0" and "1". Thus, the number $A_2 = 1101.100$ has the (decimal) significance

$$A_2 = 1 \times 2^3 + 1 \times 2^2 + 0 \times 2 + 1 + 1 \times 2^{-1} + 0 \times 2^{-2} + 0 \times 2^{-4} \qquad (3.2\text{-}5)$$

$$= 13.5$$

It is generally believed that the simplicity of the binary system lends itself to digital mechanization, since such mechanization most frequently utilizes devices which have two stable states, e.g. high vs. low voltage, open vs. closed relay contacts, conducting vs. non-conductive tubes, pulse vs. no pulse. The discussion which follows will be restricted to binary machines because they are most amenable to clarity of exposition, but the remarks apply to all digital systems with only minor modification.

It has just been pointed out that binary digits are mechanized by means of bi-stable devices. When used in an arithmetic unit the latter are commonly referred to as binary cells. In one class of digital computers, a single parameter is represented by a set of (n+1) binary cells called a register. The magnitude of the parameter is given by the states of the binary cells, the digit value a_j being determined by the state of the corresponding cell. The binary point (the value of (c) is "built into the machine". Since an arithmetic unit which utilizes (n+1) -- celled registers usually provides transmission of the digital information over (n+1) separate lines, it is referred to as a parallel unit.

Clearly, since the number of binary digits is independent of the

location of the binary point, the precision is determined only by (n+1), and is, in fact, exactly one part in 2^{n+1}. Thus, any desired precision can be obtained by supplying a sufficient number of binary cells. On the other hand, every digital computer requires at least three registers in its arithmetic unit, and the associated equipment rises roughly in proportion to (n+1). Consequently, the size and cost of a parallel arithmetic unit rises appreciably with increase of precision. However, where only low precision is demanded the size and cost may prove to be smaller than in the serial unit now to be described.

In a serial arithmetic unit only one binary cell is provided for each parameter. The parameter digits are presented to the binary cell one at a time and in temporal sequence, with the lowest-order digit a_o leading and with the highest-order digit a_n last to leave. The weight w_j is here determined by a clock which "counts" the digits as they pass and simultaneously directs the computer to establish a 1-1 correspondence between digit j and "machine time j". The rate of flow of digits must be synchronized with (and is usually synchronized by) the master clock; this rate is referred to as the pulse repetition frequency (PRF) of the computer. Thus, the computer requires a time interval t = (n+1) × PRF to scan each parameter as it flows through the binary cell. Clearly, a serial machine effects a saving in equipment (at least where high precision is required) at the expense of speed of computation.

Regardless of the organization of the arithmetic unit, it would be folly both financially and technically to attempt to incorporate enough registers in the machine to store simultaneously all the

parameters required by any one simulator. Consequently every modern computer includes an internal storage unit or memory, and operates by selecting a parameter from its memory only when the parameter is required, and by returning the results of the operations of its arithmetic unit to the memory as soon as their immediate usefulness has expired.

A computer memory unit may also be classified as a parallel or a serial device. A parallel memory unit is one from which all digits of the requested parameter are transmitted simultaneously over separate lines to the specified register in the arithmetic unit, and similarly in the reverse direction. A complete parameter transmission is thus accomplished in the time required to transmit only one digit.

A serial memory unit is one from which the parameter digits are removed in time sequence at PRF over one line. It does not follow, however, that the time, t_a, for access to a parameter is equal to the time $t_t = (n+1) \times PRF$ required for transmission of the $(n+1)$ digits of that parameter. This is because serial memory units are generally "circulatory" in character, storing parameter digits in the form of a time sequence of pulses. Two examples of such memory units are (1) acoustic delay lines and (2) rotating (magnetic) drums carrying many "lines" of information.* Technical considerations decree that the lines be long enough to accommodate many parameters (say m in number, $m \gg 1$). Consequently, in order to read a parameter into or out of the memory it is necessary to indicate in advance which of the m locations within the line

* Although other kinds of serial memories are used, the succeeding remarks on speed are applicable without major modification.

is in demand. A time interval t_s is required to activate the number-selection equipment and a second time interval t_w is wasted in waiting for the desired m location to arrive at the end of the line. The access time is given by the equation

$$t_a = t_t + t_s + t_w \qquad (3.2\text{-}7)$$

It is presumably possible to eliminate the waiting interval t_w in a simulator because of the predetermined and fixed character of its instruction routine; consequently, t_w will be assumed negligible. Also, most machines provide a (usually negligible) "dead" interval for number-selection purposes, so that in general $t_s \ll t_t$. Note, however, that in the absence of this dead interval, access to the memory must await a full additional transfer interval on account of the manner in which the lines are synchronized with the arithmetic unit. It follows that, with the above assumptions,

$$\begin{aligned} t_a \approx t_t &= (n+1) \times (\text{PRF}) \quad \text{[dead interval provided]} \\ t_a &\geq 2t_t \quad \text{[no dead interval provided]} \end{aligned} \qquad (3.2\text{-}8)$$

A pictorial representation of 6-digit parameter transmission in an all-parallel and in an all-serial computer is portrayed in Figure 3.2-2a and 2b, respectively. A computer using serial memory and parallel arithmetic units has been proposed and is perfectly feasible. The reverse mixed arrangement is probably feasible but is unlikely to offer technical advantages.

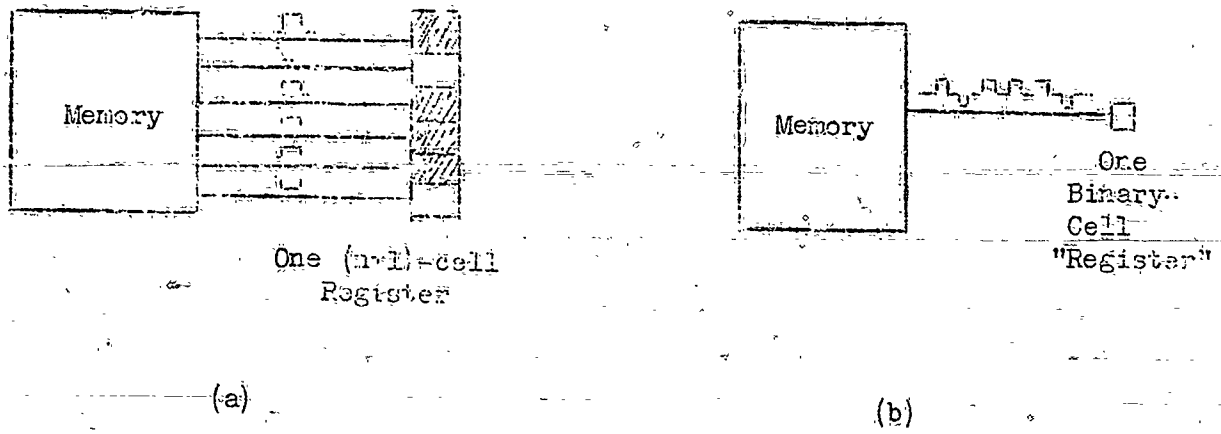


Figure 3.2-2

From the preceding remarks several characteristics of digital parameter representation are evident. Note first that, in contrast with analogue representation, every parameter is represented as a discrete magnitude; if the parameter is a time-dependent variable it must be represented as a discrete set of numbers whose time coordinates are known to the computer.

On the other hand, parameters are here represented not by sizes or physical components but by instantaneous states of the latter. Any parameter may be erased from the memory unit at will or may be replaced by the insertion of a different parameter in its number-location. Consequently, the computer is completely flexible; the performance of diverse routines associated with completely unrelated mathematical equations calls for preliminary pencil and paper work together with the translation of equation-information into machine language through equipment external to the computer itself. Once several problems have been prepared in this manner, it is a trivial task to switch from the computation of any one

problem to the computation of any other one; the machine need not be altered physically in any way.

The range of the parameters need not be restricted as severely in the case of digital computers as with analogue computers, because of the theoretically-unlimited precision available. However, for technical reasons many digital machines require each variable x to be normalized, generally so that $|x| < 1$. The question will be discussed more thoroughly in P 3.6.

3.3 Mechanization of Basic Operations and Function Evaluation

In the preceding paragraph 3.2 it was pointed out that parameters are represented in analogue computers by resistors and potentiometers. At the same time, the connections for simulating addition of and multiplication by a constant were shown in Figure 3.2-1. In this paragraph it is intended to indicate the mechanization of the basic arithmetic operations +, −, ×, ÷, of sudden discontinuity or "transfer", and (briefly) of function evaluation.

Referring to Figure 3.3-1 the small resistor r_o is the summing resistor for the variable x. Since the voltage across r_o is too small to be used directly, an amplifier A_x is provided. The output of the amplifier feeds the motor M_x which drives the arms of two linear potentiometers P_1 and P_2. Because of the manner of connection, the arm of P_1 carried a voltage representing +x while the arm of P_2 generates -x. Clearly, then, r_1 is the summing resistor for ax-b, where a and b are determined by R_1 and R_2, respectively. Similarly, the summing resistor

r_2 computes cy - dx, where c and d are determined by R_5 and R_6, respectively, and y is obtained from the motor-driven arm of P_3.

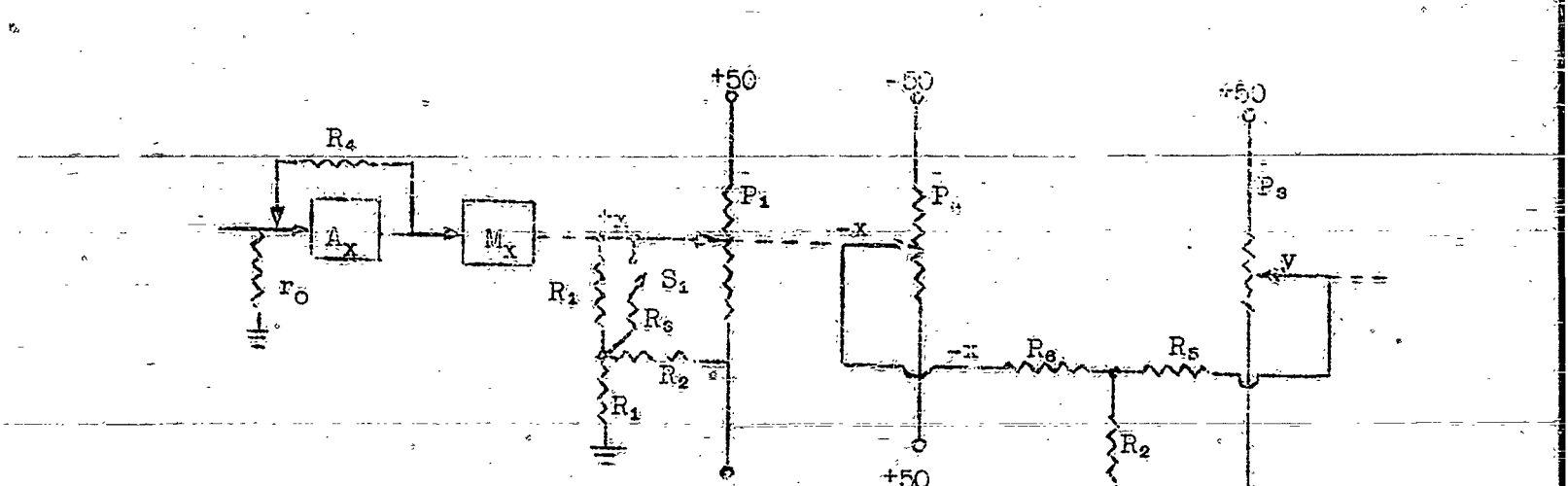


Figure 3.3-1

It often happens that a function z is represented most conveniently by two or more straight lines, e.g.

$$z_- = ax - b \qquad -50 \leq x < 0$$

$$z_+ = Ax - b \qquad 0 \leq x \leq +50$$

$$A \neq a$$

Discontinuities of this kind are easily incorporated by the inclusion of a switch operated by a cam on the shaft of a motor; e.g. S_1 in Figure 3.3-1 is held closed by a cam on motor M_x's shaft when $x > 0$, and R_1 and R_5 in parallel represent the new slope A of the line z_+. All disconti-

nuities can be handled by this technique.

Multiplication is also easily performed, as indicated in Figure 3.3-2. Here the voltage x is applied across potentiometer P, the arm of which is positioned by the y-generating motor M_y. Clearly the arm carries a voltage proportional to (xy).

Monotonically varying functions may be computed as simply as linear functions by using nonlinear potentiometers. For example, sin α can be mechanized over the interval $-\frac{\pi}{2} \leq \alpha \leq \frac{\pi}{2}$ by means of a potentiometer

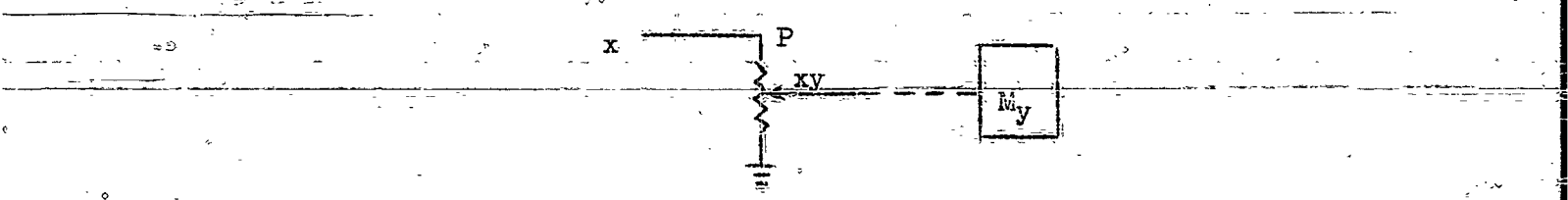


Figure 3.3-2

whose resistance P varies with arm rotation θ $(\pi \leq \theta \leq \pi)$ according to to the relation $P = \sin \frac{\theta}{2}$ and whose arm is driven by a motor M_α representing α. Similarly, any part of sin α within that range can be mechanized through an appropriately-designed non-linear potentiometer, as can also any monotonic piece of the cosine, the arc sine and tangent,

the exponential, and even a given polynomial and the square root. In other words, any member of function of a single variable x can, if desired, be mechanized extremely simply merely by replacing the linear potentiometer driven by M_x with the appropriate non-linear one.

Where the function being evaluated is dependent on two variables or where function of one variable is not monotonic, the function must be generated by extended connections of adders and multipliers. Clearly, as functions become more complex (i.e. consist of a larger number of basic operations), the bulk and the cost of equipment rises, roughly in proportion to the number of units required. At the same time, the computing system becomes increasingly looser. This looseness is directly attributable to (1) the compliance of motor shafts and the backlash in potentiometer arms, (2) the inertia of motors and their associated mechanical systems, and (3) the delay in voltage transmission through amplifiers, (electrical compliance). The first two of these are clearly sources of lag; the amplifier delay will be discussed later in this paragraph.

As a consequence of this looseness in the connections between units the function represented is computed inaccurately. For suppose that z is a function of several variables of which one variable x is machinewise related to z through many connections producing a delay τ. Then, if x should rise and the return to its original value in a time less than τ, its influence on z will be felt only after the termination of its excursion, and the eventual z-excursion will differ radically from the one decreed by the functional relationship (see Figure 3.3-3). Clearly, if the looseness in the system be decreased, τ decreases and the accuracy of

functional computation improves. Conversely, accuracy can be improved by restricting the rate at which variables are permitted to vary and/or the frequencies at which the variables may oscillate. From the machine point of view this can only mean slowing down the computer. It follows that, for analogue computers, accuracy is limited by the intrinsic maximum permissible speed of computation.

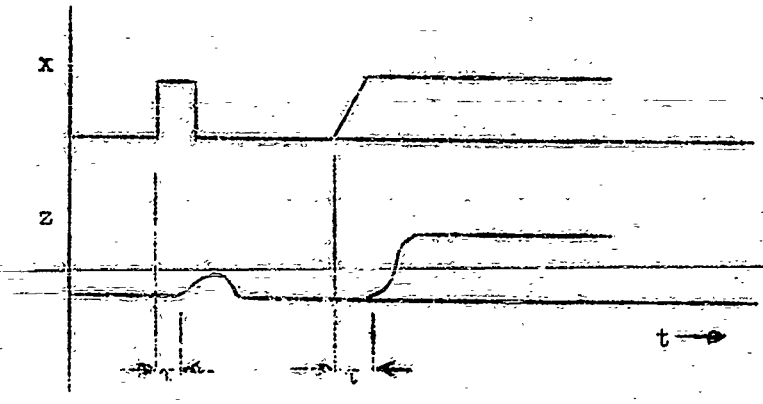


Figure 3.3-3

In the discussion of Figure 3.3-1 above, no attention was paid to resistor R_4 which is seen to connect the output of A_x to its own input. This is a feedback resistor introduced to decrease another source of inaccuracy by comparing the amplifier with its own input. When the adder was described above, it was assumed that the resistors r were small in comparison to the resistors R. Engineering difficulties prevent the direct realization of this assumption, but the use of feedback reduces the effective resistance of r to a satisfactory value. The feedback serves

also to decrease the intrinsic non-linearity of the amplifier, an additional source of inaccurate computation.

The mechanization of division has now to be described. Advantage is taken of the fact that division is the inverse operation to multiplication so that if $z = \frac{y}{x}$, then $y = xz$. Thus, the heart of the analogue divider is a multiplier (Fig. 3.3-4) whose potentiometer arm is driven by x and whose input, z^*, should be but differs from the initially unknown z. The output of the multiplier, xz^*, is delivered to a comparison element CE whose other input is fed by the known instantaneous value of y. The output of CE, which is proportional to the error term, $y - xz^*$, is fed to amplifier A_z whose purpose it is to supply a voltage z to the multiplier to be fed back to CE. The system is arranged to seek the condition where the voltages at the two inputs to CE agree, i.e. where $y = xz$, as desired. Note that the divider unit is a feedback unit which incorporates another arithmetic unit, viz. the multiplier. This is the first instance of a principle which will be discussed at greater length below, namely, that the solution to an equation $(z = \frac{y}{x})$ must always incorporate one feedback loop around the arithmetic units. The combination of comparison unit and feedback system is called a servomechanism.

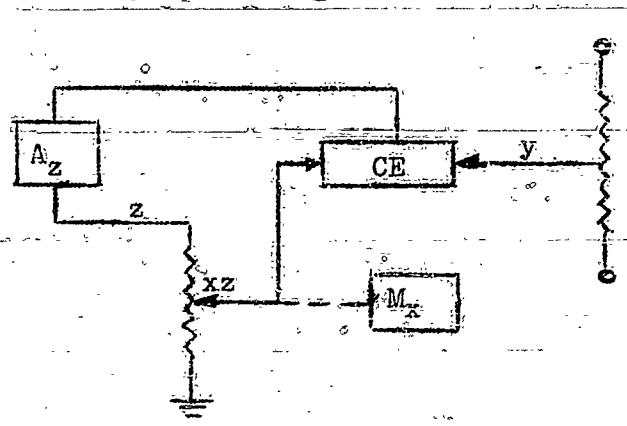


Figure 3.3-4

The square root can be mechanized by noting that $z = \sqrt{y}$ implies $y = z^2$. Thus in figure 3.3-5 if M_x is driven by z so that $x = z$, the output of the multiplier unit is z^2 and the feedback unit drives z^2 to agree with y as required.

One more unit remains to be discussed. Analogue computers are able to generate the derivative of a variable in a simple unit. For this purpose a motor-driven tachometer-generator is used, whose operation is based on the electromagnetic law that the voltage e generated in a wire moving with velocity v through a magnetic field is

$$e = kv$$

where k is a constant. The derivative unit is so designed that the motor M_x drives the tachometer simultaneously with x and in such a way that v is always proportional to $\frac{dx}{dt}$ and thus that $e = \frac{dx}{dt}$. The mechanization of the derivative is easily extended to include integration by noting that if $y = \int^t x(\tau)\, d\tau$, then $x(t) = \frac{dy}{dt}$. By forming $\frac{dy}{dt}$ and comparing with x through a servomechanism, y is obtained.

The physical unit required for analogue addition or multiplication or differentiation is a reasonably small package. In practice several such units may be incorporated into a space of about one-half cubic foot.

In contrast with the compactness of analogue units, an arithmetic unit for a digital computer requires an appreciable quantity of equipment and a correspondingly large space. In order to perform addition, a parallel digital machine calls for at least two (n+1) single-digit adders requiring about four tubes per digit, and a small control unit to direct

the addition process. The necessity for providing this wealth of equipment is imposed by the desire to perform the addition in one pulse time over (n+1) parallel lines.

A serial machine requires three registers for addition, but it will be recalled that each register consists of only one binary cell in association with a compact memory unit having a capacity of only one number. Nevertheless in the case of low precision ($n+1 \approx 10$, i.e. about 0.1%), serial registers employ approximately as much equipment as do parallel ones. Moreover, the control unit is somewhat larger because it is called upon not only to direct the gross operation of addition but also to keep count of the flow of digits and to start and stop the process. Consequently, there is little choice between parallel and serial computers as regards quantity of equipment.

It would appear that addition in serial computers should take considerably longer than in parallel computers, but this is not necessarily true. In a parallel computer the addition of digits in corresponding columns of addend and augend requires only one pulse time, but a much longer interval is needed to allow for the propagation of the full carry from the lowest to the highest column.* On the other hand, in a serial computer the addition could proceed as the addend and augend emerge from the memory unit, so that execution of the addition does not increase computation time. Simultaneous memory read-out and addition is provided in some computers and is recommended when feasible. Other computers wait

* For example, this occurs when 0000,0001 is added to 0111,1111 to yield 1000,0000 in an 8-digit computer (n=7).

until both numbers are extracted before proceeding to add, and in this case an extra computation interval of (n+1) pulse times is required. In general, (n+1) pulse times is longer than the addition time for an equivalent parallel unit.

Both parallel and serial computers employ three registers for multiplication and for division, and both accomplish these arithmetic processes through a sequence of repeated additions or subtractions, respectively. Neither is able to claim any advantage as regards additional requisite equipment. However, the parallel computer is faster since (n+1) additions in it require less time than (n+1) additions in a serial machine.

Discontinuities are programmed by means of the transfer instruction which permits the machine to make its own choice between the routines corresponding to the two branches of the discontinuous function. The transfer instruction will be discussed in greater detail in P.5.1.

Regardless of whether their internal organization is serial or parallel, virtually all digital computers employ at most one or two arithmetic units on account of the appreciable equipment involved, and these units can perform only +, −, ×, ÷, and transfer. As a result the evaluation of the basic operations defining a function has to proceed sequentially rather than simultaneously as in analogue machines; in any case the nature of digital computation precludes simultaneous evaluation. For example, the function

$$z = ax + bxy$$

is programmed as the following time-sequential routine:

$$z_1 = b \times x$$

$$z_2 = z_1 \times y = bx \times y$$

$$z_3 = a \times x$$

$$z = z_2 + z_3 = ax + bxy, \text{ as required.}$$

Since the numbers in the memory unit are invariant (until erased) there is nothing in a digital machine to correspond exactly to compliance in an analogue machine. The evaluation of a function $f(t)$ at time t_c yields $f(t_1)$ accurately so long as the data available is accurate. However, the digital machine is unable to calculate $f(t)$ continuously for all t; the calculation is limited to discrete values, t_k, spaced from each other at some (usually uniform) interval Δt. So long as Δt is small, the completely unknown values between the computed $f(t_k)$ may be assumed to be approximated closely by simple formulas derived for the purpose. On the other hand, if $f(t)$ is complex and requires a large interval Δt for its evaluation, the accuracy of these approximation formulae decreases. It follows that in real time simulation, the implicit accuracy of a digital computer is limited by the speed at which it can compute. Thus the interval Δt is responsible for objectionable effects in digital computation corresponding to those effects in analogue simulation for which the delay τ is accountable.

Feedback for analogue division and square-root has its digital counterpart, called iteration. Iterative division of y by x consists of (1) guessing at an approximate reciprocal z_1 for x, (2) repeatedly improving the guess by the recursion formula $z_{i+1} = z_i(2-xz_i)$, $i=1,2,3,\ldots$ until the result z_r equals $\frac{1}{x}$ to the requisite accuracy, and (3) multiplying z_r by y to obtain $z = yz_r = \frac{y}{x}$. Iterative square root differs only in

the recursion formula used.

It was implied above that function evaluation must be performed as a sequence of basic operations. However, it is possible to incorporate function tables into a digital computer to yield some functions such as sine, cosine, exponential, polynomials, without the need for extended computation. Such function tables are excessively expensive to build if all computation is to be avoided; the saving effected by limited tables is questionable, particularly where low precision is acceptable. Consequently, when the ERCO equations were reorganized in Section Z, such functions were approximated by one or more straight lines. Further pertinent remarks are to be found in P. 5.1.

The simplicity of analogue differentiation and integration unfortunately cannot be matched by digital mechanisms.

3.4 Mechanization of Solution of Equations

It is important to note at the outset that there are two possible interpretations of the sign of equality in expressions such as

$$w = f(x) = x^6 - x \sin x \qquad (3.4\text{-}1)$$

On the one hand, this expression may be an identity which states that the dependent variable w is equal by definition to the function f(x) of the independent variable x. In this case, x may take on any value x_k within its range of existence; the expression (3.4-1) then prescribes the procedure for evaluating the corresponding w_k. On the other hand, w may be a constant. In this case the expression is an equation; moreover, it imposes a restrictive condition on x, limiting it to a set of discrete

values whose magnitudes cannot be found through straightforward evaluation of a function.

It might be argued that on occasion the solution of an equation can be reduced to function evaluation, as in the following cases:

$$x^2 + 2ax + b = 0 \quad \text{implies} \quad x = -a \pm \sqrt{a^2 - b} \qquad (3.4\text{-}2)$$

$$\frac{d^2y}{dx^2} + k^2y = 0 \quad \text{implies} \quad y = A \sin (kx + b) \qquad (3.4\text{-}3)$$

In these cases the solution would be obtained in a computer by direct evaluation as described in the preceding paragraph P. 3.3. Unfortunately, these cases are the exception, and in general other techniques must be employed.

In an analogue computer the solution of an equation is obtained by means of the comparison element (see Fig. 3.3-4). A schematic diagram for mechanizing equation 3.4-1 (with w = 0) is shown in Fig. 3.4-1. The connections are observed to be of two distinct types; (1) the lower set of interconnections of many units, which serves to evaluate the function

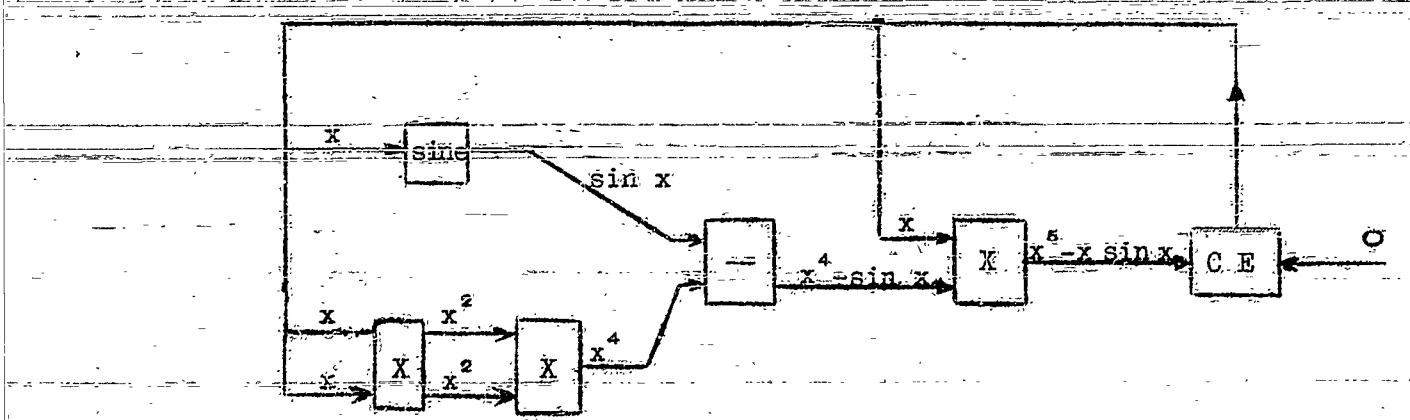
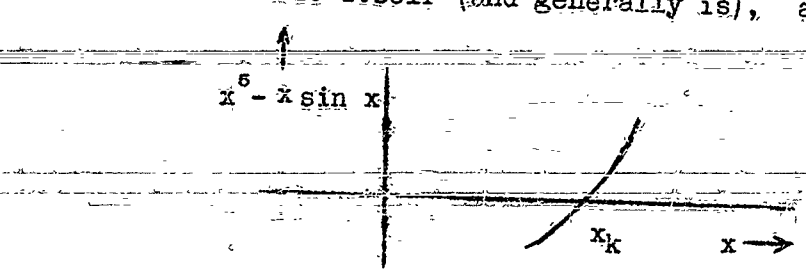


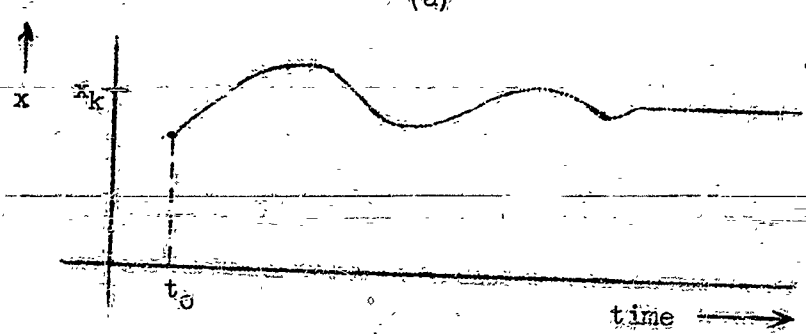
Figure 3.4-1

$x^6 - x \sin x$, and (2) the comparison element CE, which directs x to vary until the function becomes zero. Clearly, this network is a servo-mechanism whose feedback loop embraces an appreciable number of analogue units. Consequently, there is a considerable delay around the loop. Assume that when the function is negative the CE is connected so as to increase x and vice versa, i.e. the function is assumed to increase with x in the neighborhood of the solution x_k as shown if Fig. 3.4-2a. Then, if at time t_o the indicated analogue value of x is less than x_k, the CE will make x increase; however, because of the delay of transmission, τ, through the evaluating network, the CE will learn of the arrival of x at x_k only after x has exceeded x_k. Finally the CE will recognize its error, proceed to decrease x, with resultant "overshoot" on the low side. This sequence of trials during which CE hunts for the correct value x_k is depicted in Figure 3.4-2b.

Until now it was assumed that w was a constant and that x had ample time at its disposal to seek its s dution. However, w could conceivably be a variable itself (and generally is), and the time available



(a)



(b)

Figure 3.4-2

for x-hunting would then be determined by the rate at which w varies. Moreover, if w were to vary sinusoidally at a frequency comparable to the hunting frequency, the overshoot error in x would increase cumulatively. These cumulative overshoots cause the system to break into spontaneous oscillation or to behave in a manner which does not bear any immediate relationship to the solutions sought. Clearly, the system will be even more susceptible to this unruly behaviour when the equation becomes complex (e.g., if f(x) were a more complex function) and/or when many equations must be solved simultaneously.

To summarize, the solution of equations requires interconnections between units which involve feedback paths. This feedback may cause system instability even though all individual units are stable. The tendency toward instability increases with increasing complexity of the equations to be solved and with the number of feedback loops. It should be noted that the latter increase rapidly with the number of equations (roughly as the square).

Just as it was possible to improve the accuracy of function evaluation by slowing down the rate of analogue machine computation, so it is possible to restore stability when solving equations by increasing the solution time. On the other hand, the rate of computation in a real-time simulator is forced upon the analogue system by the characteristics of the device being simulated. It follows that there is an upper limit to the complexity which can be incorporated into an analogue simulator required to compute in real time.

In a digital computer the solution to equation 3.4-1 might be

obtained by iterating using the recurrence relationship

$$x_{n+1} = x_n - \frac{x_n^6 - x_n \sin s_n}{5x_n^4 - x_n \cos x_n - \sin x_n}, \quad n=1,2,3,\cdots \qquad (3.4\text{-}4)$$

where x_{n+1} is a closer approximation to the correct value x_k than the previous guess x_n. Clearly, each improvement on the preceding approximation involves considerable computation and hence an appreciable time interval. It follows that increased accuracy is obtained only at the expence of speed of computation. When w varies as x is being computed, the digital computer must clearly choose between (1) carrying the iteration procedure to its completion and falling behind in its computation, and (2) halting the iteration at low accuracy in order to keep up.

Finally, the iteration process is liable to develop oscillatory behaviour similar to that developed by analogue servomechanisms. This is particularly true where real time considerations necessitate a large interval in the solution of a complex differential equation or of a set of simultaneous differential equations such as the ERCO equations. This characteristic will be discussed in greater detail in P. 4.4; at this point it need only be noted that digital machines also are limited in their ability to compute in real time when the equations to be solved are complex.

3.5 Comparison of Analogue vs. Digital Computers

The preceding paragraphs were devoted to an extended description of analogue and digital computers with emphasis on those characteristics which affect the behaviour of real-time simulators. The succeeding paragraphs will be concerned only with digital computers, in particular with

the digital representation of numbers and its influence on computer speed. It seems appropriate, therefore, to summarize the preceding remarks by a direct enumeration of the advantages and disadvantages of both computers, and this will now be done.

(a)--Basic Units. The basic unit of the digital computer is restricted to a narrow set of operations, namely, addition, subtraction, multiplication, division, and transfer, whereas the analogue computer utilizes basic units capable of performing not only these operations but also evaluation of monotonic functions, differentiation, and integration.

(b)--Size. The size of the digital unit is many times greater than the size of any one analogue unit. On the other hand, the analogue machine is compelled by its mode of computation to include a separate unit for each and every basic operation; consequently each simulator must be judged separately before a statement of relative size of digital vs. analogue can be made. Where the computer is to simulate several similar devices, e.g. many cockpits of the same airplane type, the digital machine may afford a saving through the use of a memory unit common to all. However this possible saving requires further study.

(c)--Speed. The inherent speed of the digital arithmetic unit appreciably exceeds that of any one analogue unit. On the other hand, the analogue simulator employs a multitude of units. Thus the speeds of the two systems are about equal for simulator use, at least at the present time.

(d)--Precision. Digital machines are capable of unlimited precision whereas analogue machines are inherently low precision devices.

Where high precision is a prerequisite, only a digital system can be used. For most simulators, however, precision is not a serious factor.

(e)--Accuracy. Analogue systems are parallel systems, performing all operations simultaneously through a large set of continuously varying units. Consequently all parameters are available at all times; however, because of the multiplicity of interconnections, the computation of the parameters is generally somewhat inaccurate. Digital systems are sequential systems, performing all operations in time sequence through the one arithmetic unit. Consequently, all parameters are computed for discrete instants, at which points they are known as accurately as the component data allow; however, between these points the parameters are known only implicitly and generally must be assumed to agree with values obtained by means of approximation formulae.

(f)--Stability. Both systems perform stably either where the computation is rudimentary or where a complex computation may be performed at indefinitely low speed. Similarly, both systems tend to instability when a complex computation must be performed in real time. The region of failure is a matter of study in each case as it arises. It is frequently necessary to simplify a complex computation for real time simulation: the ERCO equations represent just such a simplification of the true laws of behaviour of the airplane they represent, the simplification being of a nature best suited to analogue simulation; the rearrangement of the ERCO equations in Section 2 above represents a similar simplification best suited to digital simulation.

(g)--Input-output. Information received by the computer is generally supplied in analogue form. Similarly, information transmitted by the computer is generally required in analogue form. For example, the stick and rudder movements are most easily obtained analogue-wise through a potentiometer while the panel instruments demand (analogue) voltage for their operation. Clearly the analogue simulator has the advantage here since it is prepared to accept and supply information in requisite form. The digital simulator must be provided with mechanisms for coding analogue parameters into digital form and vice versa. Further remarks on coding and decoding will be found in Section 6.

(h)--Flexibility. It is in this respect that digital systems show a marked superiority. In order to switch from one routine to another on analogue computers it is necessary to rearrange and/or replace a large number of physical components, requiring appreciable effort and time. In the case of digital computers each routine is represented by instructions and numbers recorded on an external storage device such as a punched or magnetic tape, and the tapes are conveniently filed away in compact form in a library of routines. The alteration of a routine (and hence of the complete simulator) is accomplished simply by changing tapes.

It might be argued that in simulator use the process of switching a digital computer from the simulation of one device to another requires connecting the computer physically to a second unit, e.g. from the cockpit of one airplane to that of another, for input-output purpose. However these connections are relatively few and would probably be handled swiftly through switches or multiple-contact plugs. In any case the same

problem occurs of necessity using any simulation device, including analogue ones.

It might further be argued that switches could be provided for rapid transfer of the analogue simulator from routine to routine. However, for real time simulation of an airplane, such a switching scheme would probably prove so complicated and expensive as to be technically undesirable.

(i)--Future Possibilities. The probability of marked improvements in analogue computers is rather low, partly because analogue devices have already gone through many stages of development in the past, partly because modern improvements in speed and accuracy appear to depend on increased size of units. On the other hand, digital computers are still in their infancy and all indications point to smaller, faster machines in the not too distant future; it is particularly important to note that digital computation speed appears to improve with decreased size of units.

From the above, it follows that a choice between analogue and digital simulation of one airplane is not clear cut. It is to be hoped that the results of this and similar projects which compare the capabilities of the two computer types for performing particular tasks, will bring to light and emphasize the relative advantages and disadvantages of both, and will stimulate further research in the application of digital computers to real time simulation.

3.6 Parameter Normalization

It was pointed out in P. 3.2 that analogue parameters were always "normalized" to obtain maximum possible precision from the low-precision

analogue units. Furthermore, it was mentioned that digital parameters need not be normalized owing to the theoretically unlimited precision available. On the other hand there are two major reasons for normalization in digital computers which will be discussed here.

The first reason is associated with binary point location. Consider the extended multiplication of y by x in a 6 binary digit machine with binary point between columns 3 and 4. If y = 100.000 and x = 010.000, the product is given by xy = 1000.000. The significant digit "1" cannot be accommodated by the machine since it is too far to the left; the loss of this digit results in complete misinterpretation of the magnitude of xy. On the other hand, if all numbers are restricted to a value less than unity the loss of digits occurs on the right hand side; these digits may be cast aside without serious consequence, e.g., .100,000,111,111 differs from .100,000 by less than 3%. (Note however that numbers may "move off the left end of the machine" even in this case, namely, when addition is performed).

The second reason for normalization is in order to obtain efficiency of representation. Consider the two decimal numbers 257 and 0.500 whose binary representations are 100,000,001 and .100,000,000, respectively. Both numbers are recorded by means of nine significant binary digits so that they are equally precise (in binary form). However, to accommodate both numbers would require at least an 18 digit machine with centered binary point, as follows:

100,000,001.000,000,000

and 000,000,000.100,000,000

Clearly this system is wasteful of digits, providing precision far in excess of demand. Normalization of every parameter p, say to $|p| < 1$, in such a way that all constants and the maxima of all variables are just less than unity, results in a number system wherein every parameter has a minimum of excess precision and hence in a machine with the minimum sequence, n+1, of binary digits for number representation.

Efficiency of representation is important because it permits the size and cost of both the arithmetic and the memory units to be reduced. More important, it speeds up computation since, as has been described in P. 3.2 and P. 3.3, all the basic operation speeds increase with decreasing (n+1).

It is always possible to normalize a complete routine, although it may be (and generally is) necessary to invoke the aid of the shifting process [multiplication by 2^k]. In fact, the manipulation of the ERCO equations in Section 2 is an example of normalization procedure.

3.7 Precision

The precision to be adopted for the digital airplane simulator is dependent on the accuracy demanded for representing airplane motion and on the method adopted for numerical integration of the equations of motion. Since neither of these has yet been crystallized, the remarks of this paragraph are subject to modification in a later report.

In Section 1, it was estimated that an accuracy of ten percent in the computation of a manoeuvre should be sufficient for a satisfactory simulator. Somewhat better short-interval accuracy is required, perhaps 1% or maybe even 0.1%. This corresponds to between 7 and 10 binary

digits, inclusive.

The height, h, of the airplane requires special consideration. In flight the value of h is of the order of 25,000 feet; consequently a ceiling of 40,000 feet was adopted in the normalization process in Section 2. At these heights there is no need to indicate h to better than several hundred feet, a precision of about 1%. On the other hand when the airplane is near ground, it is usual to indicate height to within about ten feet. Although the percent accuracy is still about 1%, the precision required is about 10 in 40,000 which calls for 12 binary digits. To overcome this difficulty in the airplane cockpit, the altimeter, which is an instrument incapable of better than 1% accuracy, is provided with two scales with the lower scale appropriately expanded. In the digital simulator a similar method may be employed by providing a second normalization of h, say at 400 feet; however, this method makes additional demands on the basic computation interval, Δt, and is to be avoided.

It appears then that at least 10 binary digits are required to obtain satisfactory precision. On the other hand the normalization procedure introduced the left-shift; for example, after $V_t Z_t$ is computed in the equation for Δq in subroutine No. six, it is shifted left 5 columns to be multiplied by $2^{-\epsilon}$; to ensure its precision of 2^{-10}, an additional 5 binary digits must be provided, making about 15 in all.

Even in the absence of the shifting process it would be necessary to allow extra digits because of round-off error. This need becomes particularly clear in the case of integration. To illustrate, consider a 4 binary-digit machine in which the airplane speed at time t_o is $u_o = .0100$ and the increase in speed per interval Δt is $\Delta u = .0000111$,

which the machine recognizes as zero. Then after 8 cycles the speed u, would be:

	True Result	Machine Computation
u_o	.0100	.0100
$8 \times (\Delta u)$	.0111	.0000
u_1	.1010 (=.625)	.0100 (=.25)

Of course the relative error is exaggerated in this example and would not be as large in a 10-digit machine; nevertheless, an appreciable error would occur in the latter machine as well.

From the above it follows that a real time digital simulator should carry a span of about 15 binary digits plus one binary digit for the algebraic sign.

3.8 Representation of Magnitudes

In the preceding paragraphs the parameters were assumed to be represented in conventional form, for example, the magnitude A was given by

$$A_2 = a_n\, a_{n-1}\, a_{n-2} \cdots\cdots a_o = \sum_{j=o}^{n} a_j\, 2^{j+c} \qquad (3.8\text{-}1)$$

In order to minimize the number of digits (n+1) it was found advisable to normalize all parameters. However, the absolute minimum could not be achieved on account of the accompanying requirement of left shift which necessitated the use of about 5 extra digits, a 50% increase. Since these excess digits require extra machine equipment and slow-down the arithmetic processes, consideration was given to other possible representations of magnitudes.

Attention was first directed to the semi-logarithmic or floating decimal-point system, in which magnitudes appear in the form

$$A_{SL} = a_1\, a_2 \text{ ------} a_l;\ \pm\, b_m\, b_{m-1} \text{ --- } b_o \qquad (3.8\text{-}2)$$

$$= \sum_{i=1}^{l} a_i\, 2^{-i} \times 2^{\pm \sum_{j=o}^{m} b_j\, 2^j}$$

In words, each parameter is represented by a fractional number (binary point at left hand side of number) and by an index which determines the actual location of the binary point.* There is some question as to whether this system would result in a saving in digits; in fact it is probably that the total requirement of $[(l+1) + (m+1)]$ digits + two algebraic signs would exceed the previous $(n+1)$ digits + one algebraic sign in most digital simulators. In any event there would result a serious decrease in speed because the arithmetic processes become more complex; for example, in the Raytheon Hurricane computer the speed is reduced by a factor of about four.

A little consideration was given to the logarithmic system where the magnitude A would first be normalized and then be represented in the simulator by its logarithm (to any prescribed base). This representation has the advantage that multiplication is accomplished through the addition operation, a much faster process. A second advantage is that magnitudes increase at constant percentage rate so that the ratio of successive numbers is a constant fraction. On the other hand the system is

* For example, the decimal number 63.845 would appear in the form 0.63845; + 2, meaning $0.63845 \times 10^{+2}$

unable to render the number zero; moreover there is no simple way in which to mechanize addition. Consequently the representation was not studied in any detail.

No other simple systems could be discovered. As a result it was concluded that the conventional representation (in the binary notation) was the most promising.

3.9 Speeds of Current Digital Computers

The purpose of this paragraph is to evaluate the time required by digital computers currently nearing completion to perform the basic operations of addition, multiplication, division, and shift. Subtraction requires the same time as addition in almost all modern machines. Two computers will be considered, (1) the Raytheon "Hurricane", whose time of computation of the complete routine is recorded in Section 5, and (2) the Institute for Advanced Study "IAS" machine. The first is a serial machine using a mercury delay line memory unit; the second is a parallel machine using a cathode-ray tube (Williams) memory unit.

It is characteristic of all high-speed digital computers that the instructions which sequence the routine are stored in the internal memory along with the numbers on which they operate.* Consequently, it is necessary to take into account the time spent in removing the instruction(s) from the memory when calculating the total time to perform an

* In fact there is nothing to distinguish between an instruction and a number in the memory. On occasion it is convenient to use the arithmetic unit to alter an instruction; an example of this is in the handling of minor cycles (see Section 5).

operation. Note also that almost all operations require the specification of four addresses as well as the nature of the operation; for example, the four addresses required to complete an addition are those of the addend, the augend, the sum, and the location of the succeeding instruction.

The Raytheon Hurricane computer uses a pulse repetition frequency (PRF) of about 3.6 megacycles and a precision of 30 binary digits plus algebraic sign. However, in order to make the machine self-checking, the number of digits used to represent magnitudes was increased to 36. Consequently the number length and hence the fundamental cycle interval is $36 \times \frac{1}{3.6} = 10$ microseconds (μsec.). Because no "dead" interval is provided (see P. 3.2), the access time is two cycles or 20 μsec per number. On the other hand each instruction carries four addresses requiring a length of 72 digits, two number lengths; hence the access time for an instruction is three cycles or 30 μsec. Again, it should be noted that only one number may be removed from or inserted into the memory during any one cycle so that access to the memory must be performed sequentially.

Clearly, then, the time to perform one operation, $A_oB = C$, is calculated as follows*:

(a)	Removal of instruction	= 30 μsec.
(b)	Removal of A	= 20 μsec.
(c)	Removal of B	= 20 μsec.
(d)	Operation time	= t_o (see below)
(e)	Insertion of C	= 20 μsec.

* The symbol o should be interpreted here as +, −, ×, or ÷; in the case of the transfer operation the o implies subtraction and then the third address C(≠A−B) refers to the location of the alternative succeeding instruction.

Thus, the total operation time is $(90 + t_o)$ μsec, where t_o depends on the operation being performed. Note that there is a substantial saving if the operations are sequenced so that one number is already available from the preceding operation. For example, consider the operations (A+B) × D. There is no longer the need to insert C = A+B (e above) nor to remove it for the operation C×D (b above); thus the total interval required is only $(140 + t_A + t_M)$ μsec, a saving of 40 μsec. Note also that there is a similar saving of 20 μsec in the transfer operation because the result of the comparison is needed only to make a choice between two subroutines and hence is not inserted into the memory.

Finally, the time required to consummate addition, multiplication, division and shift exclusive of removal and insertion is $t_A = 50$, $t_M = 250$, $t_D = 400$, and $t_S \approx 85$μsec., respectively. Note that $t_M \neq (n+1)\ t_A$, as might have been anticipated as a result of the discussion in P. 3.3. This is because part of the computation time is devoted to starting, stopping and self-checking, and these occur only once in t_M.

The IAS machine provides for a 4) binary-digit number (including algebraic sign). Access time to any number in the memory is about 10 μsec., and numbers are withdrawn (inserted) sequentially. Operation times are $t_A = 10$, $t_M = 480$, $t_b = 500$, and $t_S = 2m$ μsec, where m is the number of columns shifted. These times are estimated, since official values are not immediately available; in particular, (1) the maximum value has been used for t_M since in a time simulation problem advantage can probably not be taken of the speed increase due to zero multiplier

digits, and (2) the estimate of t_D may be somewhat high.

Instructions require only 20 binary digits but provide only one address. Thus two instructions are withdrawn from the memory at one time but three instructions are needed to complete each operation. No address is required to locate successive instructions since the latter are stored in consecutively-numbered locations in the memory; it is sufficient to provide a binary counter to direct the sequence of commands. Consequently the execution of two independent operations requires that instructions be withdrawn from the memory only three times. The duty of the transfer operation is to set the binary counter (discontinuously) to the first address of either two new subroutines.

Since both the Hurricane and the IAS machines provided more precision than is required by a simulator, the operation times given above are longer than they would be in a specially-designed digital simulator. Crude estimates of the speeds of 16 binary-digit computers modelled after Hurricane and IAS, respectively, are given in Table 3.9-1 alongside the summarized results of the existing values. Note that the relative improvements differ (1) because the former is a serial whereas the latter is a parallel computer, and (2) because the self-checking feature of the former was omitted.

Table 3.9-1 Total Operation Times

OPERATION	Hurricane time (μsec)		IAS time (μsec)	
	As is	16 digits*	As is	16 digits*
A + B = C	140	70	60	45
A × B = C	340	145	530	120
A ÷ B = C	490	235	550	190
A+B+C = D	240	125	90	80
A×B×C = D	440	270	830	300
$A = 2^B \times B$	155	85	45	45

* Crude estimates

SECTION 4

METHODS OF NUMERICAL INTEGRATION

4. METHODS OF INTEGRATION

The integration of the differential equations is one of the most difficult problems in the solution of the flight trainer equations and one on which a large amount of time has been spent. However, no decision has yet been made as to the most suitable method of integration. This section outlines the difficulties involved in the integration, gives reasons for the rejection of certain methods, and describes a modified Moulton method which at present appears to be the most promising.

The difficulties which occur may be outlined as follows:

1. There is a system of 10 interdependent differential equations.

2. The derivatives are given by complicated expressions the evaluation of which occupies a major part of the computing time. See for example Subroutine No. 3.

3. These derivatives contain terms depending on the deflections of the control surfaces, δ_e, δ_a, etc, and other pilot- or instructor-introduced quantities, as well as terms independent of the pilot and instructor. The terms independent of the pilot and instructor are in general mathematically well-behaved, but the others, called the δ-terms are unfortunately ill-behaved.

4. Various equations are changed or partially changed depending on whether the airplane is in the air or on the ground, in normal or stalled flight, and the like, so that any integration method used must handle these transitions properly.

5. We are concerned with real time simulation, so that any integration method using a variable time interval, or an indefinite number of repeated operations which result in a non-uniform total computation interval, is unsuitable.

In general the discussion will be confined to the single differential equation $\dot{x} = f(t, x, \delta)$ or even $\dot{x} = f(t,x)$, where knowing x_n, then x_{n+1} is to be found. On occasion it may be necessary to note explicitly the existence of many equations of the form $\dot{x}_i = f_i(t, x_j, \delta_j)$; even then, on account of the complexities of the problem, the profuse notation may be contracted in many places. The notation used is listed and explained in P. 4.6.

It must be kept in mind that this is a real-time simulation problem. Hence rapidity of solution is of primary importance, taking preference over high-percentage accuracy. Where a method of integration calls for computation of more than one set of derivatives per interval it is discarded in favor of a method involving longer integration time but only one derivative evaluation.

As yet, the most suitable method of integration has not been determined. Hence the discussion has been kept brief and should be interpreted only as a report of progress to date.

4.1 Conventional Methods of Integration

The most important methods available are the following:

1. Taylor's series
2. The method of Gauss
3. Runge-Kutta

4. Milne

5. Adams-Bashforth (a) by differences or (b) by derivative-ordinates

6. Moulton

7. Relaxation

The Taylor's series method makes use of the expression

$$x_{n+1} = x_n + \frac{\Delta t}{1!}\dot{x}_n + \frac{(\Delta t)^2}{2!}\ddot{x}_n + \dots \qquad (4.1-1)$$

$$\Delta t = t_{n+1} - t_n = \text{constant}$$

This method is ruled out because of the time required to compute the second and higher order derivatives. In any case it would require knowledge of the δ-derivatives, which are not easily obtained.

Difference methods using variable interval . $(t_{n+k} - t_{n+k-1} \neq t_{n+1} - t_n)$ such as the Gaussian method cause trouble since a time simulation problem is treated most successfully when equal time intervals are used.

In the Runge-Kutta method

$$x_{n+1} = x_n + (k)_n \qquad (4.1-2)$$

where $(k)_n = \frac{1}{6}\left[(k_1)_n + 2(k_2)_n + 2(k_3)_n + (k_4)_n\right]$

and $(k_1)_n = \Delta t\left[(\dot{x})_n\right] = \Delta t\, f\left[t_n, x_n\right]$

$$(k_2)_n = \Delta t\, f\left[\left(t + \tfrac{1}{2}\Delta t\right)_n, \left(x + \tfrac{1}{2}k_1\right)_n\right]$$

$$(k_3)_n = \Delta t\, f\left[\left(t + \tfrac{1}{2}\Delta t\right)_n, \left(x + \tfrac{1}{2}k_2\right)_n\right]$$

$$(k_4)_n = \Delta t\, f\left[(t + \Delta t)_n, (x + k_3)_n\right]$$

where $\left(x + \frac{1}{2} k_1\right)_n \equiv (x)_n + \frac{1}{2} (k_1)_n$; etc.

This method is generally considered very accurate, but clearly is very complicated. It becomes even more so for a set of differential equa- equations and must be ruled out.

In the Milne Method, if $\dot{x} = f(x,t)$ and x_n is known, then

i) A first approximation to x_{n+1} is found by

$$x'_{n+1} = x_{n-3} + (\Delta t)\, E_1 (\dot{x}_n, \dot{x}_{n-1}, \dot{x}_{n-2}) \qquad (4.1 - 3)$$

where

$$E_1 (\dot{x}_n, \dot{x}_{n-1}, \dot{x}_{n-2}) = \frac{4}{3} (2\dot{x}_n - \dot{x}_{n-1} + 2\dot{x}_{n-2})$$

ii) x'_{n+1} is substituted in $\dot{x} = f(x,t)$ to find the corresponding value of $\dot{x}'_{n+1}$.

iii) $\dot{x}'_{n+1}$ is substituted in

$$x''_{n+1} = x_{n-1} + (\Delta t)\, E_2 (\dot{x}_{n+1}, \dot{x}_n, \dot{x}_{n-1}) \qquad (4.1 - 4)$$

where

$$E_2 (\dot{x}_{n+1}, \dot{x}_n, \dot{x}_{n-1}) = \frac{1}{3} (\dot{x}_{n+1} + 4\dot{x}_n + \dot{x}_{n-1})$$

If x'_{n+1} and x''_{n+2} agree to desired accuracy, the $\dot{x}_{n+1}$ is correct; if they do not agree, in general a smaller (Δt) is used. Note that the Milne method requires the storage of both derivative-and x-ordinates; hence it was discarded for simulator use.

In the Adams-Bashforth Method

$$x_{n+1} = x_n + (\Delta t)\, D_1 (\dot{x}_n, \dot{x}_{n-1}, \dot{x}_{n-2}, \dot{x}_{n-3}, \ldots) \qquad (4.1 - 5)$$

where

$$D_1(\dot{x}_n, \dot{x}_{n-1}, \dot{x}_{n-2}, \dot{x}_{n-3}, \ldots)$$

$$= \dot{x}_n + \frac{1}{2} \nabla \dot{x}_n + \frac{5}{12} \nabla^2 \dot{x}_n + \frac{3}{8} \nabla^3 \dot{x}_n + \ldots$$

$$\nabla \dot{x}_n = \dot{x}_n - \dot{x}_{n-1}$$

$$\nabla^2 \dot{x}_n = \nabla \dot{x}_n - \nabla \dot{x}_{n-1} = \dot{x}_n - 2\dot{x}_{n-1} + \dot{x}_{n-2}$$

$$\nabla^3 \dot{x}_n = \nabla^2 \dot{x}_n - \nabla^2 \dot{x}_{n-1} = \dot{x}_n - 3\dot{x}_{n-1} + 3\dot{x}_{n-2} - \dot{x}_{n-3}$$

This method has the features suitable to a real time simulation problem, namely only $\dot{x}$ need be computed, there is a uniform interval Δt and the integration is simple. Because it uses differences it is particularly suited to hand computation. However, for machine computation it was discarded in favor of the corresponding method which uses derivative-ordinates directly instead of differences. This latter method is most suitably called a Steffensen method. It follows there-

The symbol ∇ refers to the backward difference; thus $\nabla \dot{x}_n = \dot{x}_n - \dot{x}_{n-1}$.

fore that the Steffensen method uses the formula

$$x_{n+1} = x_n + \Delta t \; O_1 [\dot{x}_n, \dot{x}_{n-1}, \ldots, \dot{x}_{n-k}] \qquad (4.1 - 6)$$

where, for example,

$$[O_1 (\dot{x})_n]_2 = O_1 [\dot{x}_n, \dot{x}_{n-1}, \dot{x}_{n-2}]$$

$$= \frac{1}{12} [23\dot{x}_n - 16\dot{x}_{n-1} + 5\dot{x}_{n-2}]$$

In the Moulton method there is

i) An open integration by the Adams-Bashforth or Steffenson Method, i.e., by use of a D_1 or an O_1.

ii) A check on i) by a closed integration D_2 or O_2.

iii) A repetition of D_2 or O_2 check until there is no change.

The use of D_2 (or O_2) to improve on D_1 (or O_1) is entirely analogous to the use of the E_2 improvement on E_1 in the Milne method, equations 4.1 - 3,4. This method was used in computing the manoeuvres to be discussed in P. 4.3 and 4.4.

Relaxation methods were discarded because by their very nature they are unsuited to a real-time simulation problem.

4.2 The Modified Moulton Method

The method to be described in this paragraph represents an initial attempt to obtain some of the advantages of the interative open and closed integration procedure just described and yet to avoid the extensive additional computation demanded in evaluating more than one derivative per interval. In principle the method consists of the

first three steps of the Moulton procedure, namely (1) guessing a value $\dot{x}'_{n+1}$ by means of the open integration formula $[\,O_1(\dot{x})_n\,]_k$, (2) evaluating the new derivative x'_{n+1} and (3) guessing a more correct x''_{n+1} by means of the closed formula $[\,O_2(\dot{x})_{n+1}\,]_k$. Here the process for the interval (t_n, t_{n+1}) is terminated, it being assumed that $x''_{n+1} = x_{n+1}$ and $\dot{x}'_{n+1} = \dot{x}_{n+1}$ near enough. Since the step immediately following the use of O_2 in (3) involves the use of O_1 in (1) for the succeeding interval, it proves possible to merge these two into one step using a closed-and-open formula denoted by O_3.

The derivation of O_3 is straightforward. The steps (1), (2), (3) result in the equations

$$x'_{n+1} = x_n + [\,O_1(\dot{x})_n\,]_k \qquad (4.2\text{-}1)$$

$$\dot{x}_{n+1} = f[\,t_{n+1}, x'_{n+1}, \delta_{n+1}\,] \qquad (4.2\text{-}2)$$

$$x_{n+1} = x_n + [\,O_2(\dot{x})_{n+1}\,]_k \qquad (4.2\text{-}3)$$

respectively. The first equation for the next interval is now written down and, by means of the above equations, is put into its final form. Thus,

$$\begin{aligned} x'_{n+2} &= x_{n+1} + [\,O_1(\dot{x})_{n+1}\,]_k \times \Delta t \\ &= x_n + [\,O_2(\dot{x})_{n+1}\,]_k \Delta t + [\,O_1(\dot{x})_{n+1}\,]_k \cdot \Delta t \\ &= x'_{n+1} - [O_1(\dot{x})_n\,]_k \Delta t + [\,(O_2 + O_1)(\dot{x})_{n+1}\,]_k \cdot \Delta t \end{aligned} \qquad (4.2\text{-}5)$$

$$\therefore x'_{n+2} = x'_{n+1} + [O_3(\dot{x})_{n+1}]_{k+1} \cdot \Delta t$$

where

$$[O_3(\dot{x})_{n+1}]_{k+1} = [(O_2 + O_1)(\dot{x})_{n+1} - O_1(\dot{x})_n]_k$$

For example the second order expression k = 2 for O_3 for the interval (t_n, t_{n+1}) is given by

$$[O_3(\dot{x})_n]_3 = \frac{7}{3}\dot{x}_n - \frac{31}{12}\dot{x}_{n-1} + \frac{5}{3}\dot{x}_{n-2} - \frac{5}{12}\dot{x}_{n-3}$$

4.3 Numerical Computations

This paragraph is concerned with the numerical computations performed throughout the course of this investigation. The initial purpose of these computations was to become familiar with airplane motion as idealized by the Erco equations and to study the behavior of the parameters of flight. As the investigation proceeded, the computations served to check the transformationsof dynamic equations and normalization of parameters (described in §2), to study the behaviour of the modified Moulton integration procedure (P.4.2), and to ascertain approximately the maximum permissable time interval and the order of approximation of a usable integration formula. Finally, the computations threw light on the difficulties introduced into the solution of the dynamic equations by pilot- and instructor-introduced discontinuities and by "compliance errors" (discussed more fully in P. 4.4).

The following computations were performed:

(1) Approximately level steady flight, using the given Erco equations. Flight was computed for three seconds from an arbitrary time zero at $\frac{1}{4}$ second intervals using the Moulton method.

(2) Level flight, using the transformed equations. The computations picked up at time 1.75 seconds from the previous computation and continued to 3 seconds at $\frac{1}{4}$ second intervals using the Moulton method.

(3) "Immelman turn". Beginning from level flight at three seconds, the stick was assumed to move back in such a way that δ_e decreased linearly with time at the rate of -.1409 units per second; this corresponds to maximum elevator deflection in three seconds. Simultaneously thrust was assumed to increase by .0039 units per sec. The Immelman turn manoeuvre had to be terminated at four seconds because the airplane was found to have reached stall. The computation was performed once at $\frac{1}{4}$ second and once at 1/16 second intervals using the Moulton method, and

at 1/16 and 1/32 second intervals using modified Moulton method.

(4) Test roll. Beginning from level flight at three seconds, δ_a was varied as shown in Table 4.3-1, and flight was computed at 1/8 second intervals using Moulton method, and at 1/8, 1/16, and 1/32 second intervals using modified Moulton method. The manoeuvre corresponds to maximum deflection of the stick to the left followed by return to center position

TABLE 4.3-1

t	3.00	3.125	3.25	3.375	3.50	>3.50
δ_a	0	.2358	.4716	.2358	0	0

in $\frac{1}{2}$ second, and is evidently a manoeuvre frequently applied to test the design of aircraft.* That the manoeuvre is a realistic one is shown by the report just referred to and also by the results of a group at the Franklin Institute who were engaged in the study of pilot reactions to simulated but realistic flight requirements. Figures 4.3-1 and 4.3-2 are transcribed from records taken by that group.

(5) Parameter derivatives for selected manoeuvres ($\dot{p}$, $\dot{q}$,..., $\ddot{p}$, $\ddot{q}$,..., etc.). For these computations the equations of motion were differentiated repeatedly and the higher order derivatives were expressed as functions of the parameters and the derivatives of lower order. The coefficients of the δ-derivatives were computed as separate entities to allow choice of realistic values at the conclusion of the computations. The selected manoeuvres and the assumed parameter values are displayed in Table 4.3-2

*

TABLE 4.3-2

Initial Values Used in Obtaining Derivatives

Manoeuver	High Speed Roll	Immelman Flipover	Inside Loop	Outside Loop	Immelman Turn t=3,0625	Test Roll t=3.125
u	0.512	0.512	0.512	0.512	0.469	0.469
v	0.05	0.05	0	0	0	0
w	0.099	0.099	0.099	0.099	0.042	0.042
p	0.750	0.750	0	0	0	0.011
q	0	0	0.2	-0.2	-0.010	-0.014
r	0.55	0.55	0	0	0	0.002
h	0.500	0.500	0.500	0.500	0.606	0.606
l_s	0	0	-0.500	0.500	0	0
m_s	0.500	-0.500	0	0	0	0.002
n_s	0	0	0	0	0.500	0.500
δ_{fd}	0	0	0	0	0	0
δ_a	0.472	0.472	0	0	0	0.236
δ_r	0.091	0.091	0	0	0	0
δ_e	0	0	-0.273	0.273	0.009	0.009
I_x	0.202	0.202	0.202	0.202	0.376	0.376
I_z	0.508	0.508	0.508	0.508	0.662	0.662
W	0.750	0.750	0.750	0.750	1.073	1.073
T	0.491	0.491	0.491	0.491	0.083	0.083

The derivative computations were performed to a precision of only three decimal places; however, the integrations were computed to six decimal places to eliminate round-off error. For the Moulton method, the integration formulas used were:

(1) Open integration

$$\Delta x = \left[\dot{x}_o + \tfrac{1}{2} \nabla \dot{x}_o + \frac{5}{12} \nabla^2 \dot{x}_o + \frac{3}{8} \nabla^3 \dot{x}_o + \frac{251}{720} \nabla^4 \dot{x}_o \right] \Delta t$$

$$= [2.64027\, \dot{x}_o - 3.8527\, \dot{x}_{-1} + 3.63\, \dot{x}_{-2} - 1.7694\, \dot{x}_{-3} + .3486\dot{1}\, x_{-4}\,]\, \Delta t \qquad 4.3\text{-}1$$

(2) Closed integration

$$\Delta x = \left[\dot{x}_1 - \tfrac{1}{2} \nabla \dot{x}_1 - \frac{1}{12} \nabla^2 \dot{x}_1 - \frac{1}{24} \nabla^3 \dot{x}_1 - \frac{19}{720} \nabla^4 \dot{x}_1 \right] \Delta t \qquad 4.3\text{-}2$$

$$= [.3486\dot{1}\, \dot{x}_1 + .8972\, \dot{x}_o - .36\, \dot{x}_{-1} + .1472\, \dot{x}_{-2} - .02638\, \dot{x}_{-3}] \Delta t$$

Closures were repeated until the successive values of corresponding parameters agreed to within .0001 (although occasionally agreement to .0002 was accepted). No more than three closures were required for any interval encountered.

For the modified Moulton integration the formula used was

$$\Delta x = [\, 2.98\, \dot{x}_o - 5.59583\, \dot{x}_{-1} + 7.1194\, \dot{x}_{-2} - 5.25\, \dot{x}_{-3} + 2.0916\, \dot{x}_{-4} - .3486\dot{1}\, \dot{x}_{-5}]\, \Delta t \qquad 4.3\text{-}3$$

Table 4.3-3. Comparison of Erco and transformed equations

Table 4.3-4. Derivatives of flight parameters for some of the selected flight conditions of Table 4.3-2

Figures 4.3-4 through 8. Graphs of Immelman turn manoeuvre

Figures 4.3-9 through 17. Graphs of test roll manoeuvre

Table 4.3-3

	ERCO Equations					Transformed Equations				
t	2.00	2.25	2.50	2.75	3.00	2.00	2.25	2.50	2.75	3.00
u	.4680	84	88	92	95	.4677	81	84	88	92
w	.0414	15	19	20	22	.0414	14	16	15	16
q	-.0144	31	33	33	42	-.0144	41	43	41	43
1_s	-.0155	20	088	053	020	-.0155	19	084	048	013
n_s	1.0000	00	00	00	00	1.0000	01	01	01	01
h	.6067	65	65	62	62	.6063	62	61	59	58

Summary of Tables 4.3-4

Table 4.3-4a - High Speed Roll Manoeuvre

Table 4.3-4b - Immelman Flipover Manoeuvre

Table 4.3-4c - Inside and Outside Loop Manoeuvre

Table 4.3-4d - "Immelman Turn"

Table 4.3-4e - Test Roll

These tables are to be read as in the following example taken from the High Speed Roll Manoeuvre:

$$u^{(4)} = \ddot{u} = -20.927 - .68\,\delta_a + 6.6\,\delta_r - 7.26\,\delta_e - .86\,\delta_{fd} + .02\,\delta_a^{(2)}$$

$$- .84\,\delta_r^{(2)} - 6.26\,\delta_e^{(2)} - .54\,\delta_{fd}^{(2)}$$

TABLE 4.3 - 4a

	Constant	δ_a	δ_r	δ_e	δ_{fd}	δ_a	δ_r	δ_e	δ_{fd}	δ_a	δ_r	δ_e	δ_{fd}	δ_a	δ_r	δ_e	δ_{fd}	$\delta_a\delta_e$	$\delta_a\delta_{fd}$	$\delta_r\delta_e$	$\delta_r\delta_{fd}$	$\delta_e\delta_{fd}$	$(\delta_e)^2$	$(\delta_{fd})^2$	$\delta_a\delta_r$	$(\delta_a)^2$	$(\delta_r)^2$
u	.51																										
u	.03																										
u	-.20																										
u	1.8	.02	-.43	1.4	.12																						
u	-20	-.68	6.6	-7.3	-.86	.02	-.84	-6.3	-.54																		
u ✓	153	22	-167	153	14	-.73	11	139	12	.02	-.43	1.4	.12														
v	.05																										
v	-.10																										
v	-.63																										
v	21	1.3	18	0	0																						
v	-119	-31	-6.5	-273	-24	1.3	35	0	0																		
v ✓	154	492	-1174	-1950	-162	-38	-24	1200	104	1.3	18	0	0														
w	.10																										
w	-.47																										
w	3.0																										
w	-17	-1.1	-.37	-39	-3.4																						
w	-21	12	-243	-282	-24	-1.1	-.56	172	15																		
w ✓	1339	-64	694	4018	352	9.6	-473	-592	-51	-1.1	-.37	-39	-3.4														
p	.75																										
p	-.16																										
p	-.28	2.7	.91	0	0																						
p	6.7	-5.6	-8.2	2.0	.18	2.7	1.4	0	0																		
p	-8.1	6.7	153	-54	-4.7	-5.9	-20	-8.6	-.75	2.7	.91	0	0														
p ✓	-333	68	-1518	-1033	-88	3.5	23	368	32	-8.1	-13	2.0	.18	2.7	.91	0	0	20	1.7	-838	-72	0	0	0	-32	.46	-9.1
q	0																										
q	3.0																										
q	-12	0	0	-29	-2.5																						
q	62	17	-99	-249	-23	0	0	127	11																		
q	314	-252	155	1412	122	17	-194	-258	-22	0	0	-29	-2.5														
q ✓	-2306	591	5579	4041	332	-328	381	-3599	-311	17	-99	53	4.6	0	0	-29	-2.5	.20	0	0	0	36	212	1.5	-2210	115	-749
r	.55																										
r	-2.3																										
r	-3.3	.88	-17	0	0																						
r	17	-10	-5.2	36	3.1	.85	-33	0	0																		
r	244	44	28	331	28	-14	-.25	-159	-14	.85	-17	0	0														
r ✓	432	-65	-766	426	36	118	97	274	24	-18	-3.7	36	3.1	.85	-17	0	0	-545	-47	-55	-4.6	0	0	0	-21	-37	57
l	0																										
l	-275																										
l	-1.1																										
l	6.4	.44	-8.6	0	0																						
l	26	-5.1	-2.6	-242	-21	.43	-17	0	0																		
l ✓	-62	179	760	-2705	-228	-7.2	-.13	1439	125	.43	-8.6	0	0														
m	.500																										
m	0																										
m	-18																										
m	13																										
m	679	-263	-69	0	0																						
m ✓	-2284	836	843	-285	-26	-329	-123	0	0	0	0	0	0														
n	0																										
n	-3.0																										
n	.64																										
n	112	-11	-3.6																								
n	-177	22	33	-32	-2.8	-11	-5.6	0	0																		
n ✓	-3986	3958	381	141	13	23	78	174	15	-11	-3.6	0	0														

TABLE 4.3 - 4b

	Constant	δ_a	δ_r	δ_e	δ_{fd}	$\dot{\delta}_a$	$\dot{\delta}_r$	$\dot{\delta}_e$	$\dot{\delta}_{fd}$	$\ddot{\delta}_a$	$\ddot{\delta}_r$	$\ddot{\delta}_e$	$\ddot{\delta}_{fd}$
q	0												
$\dot{q}$	3.0												
$\ddot{q}$	-10	0	0	-29	-2.5								
$\dddot{q}$	49	17	-99	-290	-25	0	0	149	13				
$\ddddot{q}$	312	-246	28	1520	130	17	-194	-301	-26	0	0	-29	-2.5
r	.55												
$\dot{r}$	-2.3												
$\ddot{r}$	-2.6	.88	-17	0	0								
$\dddot{r}$	15	-10	-4.4	36	3.2	.85	-33	0	0				
$\ddddot{r}$	-227	73	35	429	36	-14	.13	-187	-16	.85	-17	0	0
m	-.50												
$\dot{m}$	0												
$\ddot{m}$	18												
$\dddot{m}$	9.6												
$\ddddot{m}$	-766	136	34	0	0								

TABLE 4.3 - 4c

	Constant	δ_a	δ_r	δ_e	δ_{fd}	δ_a	δ_r	δ_e	δ_{fd}	δ_e	δ_r	δ_e	δ_{fd}	δ_a	δ_r	δ_e	δ_{fd}	$\delta_a\delta_e$	$\delta_a\delta_{fd}$	$\delta_r\delta_e$	$\delta_r\delta_{fd}$	$\delta_e\delta_{fd}$	$(\delta_e)^2$	$(\delta_{fd})^2$	$\delta_a\delta_r$	$(\delta_a)^2$	$(\delta_r)^2$
$\dot{q}$	0.2																										
$\ddot{q}$	7.2																										
$\dddot{q}$	-16			-29	-2.5																						
$\ddddot{q}$	124			-110	-9.3			52	4.5																		
$\dddddot{q}$	-2277		-0.1	542	31			-100	-8.6			-29	-2.4														

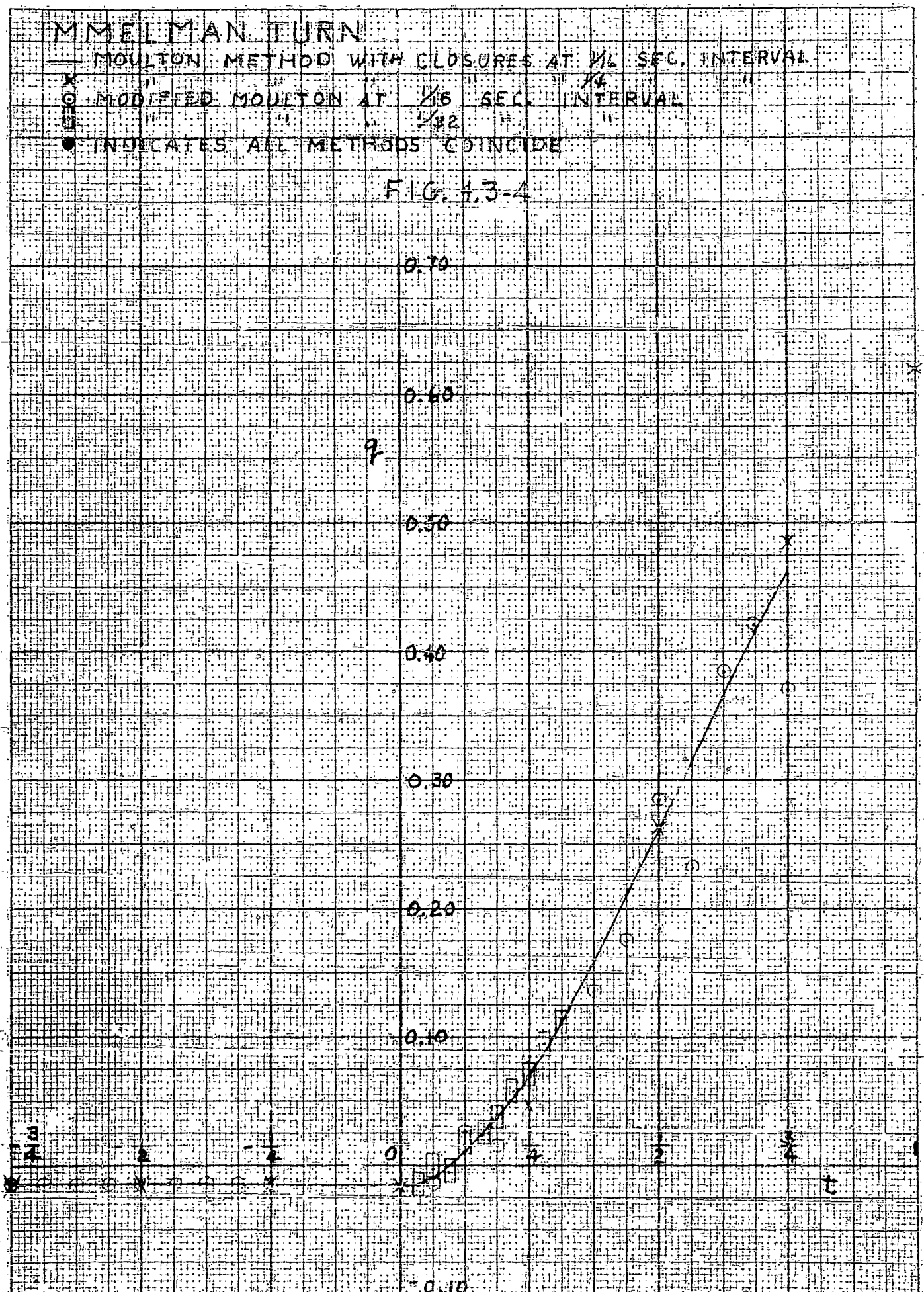
TABLE 4.3 - 4d

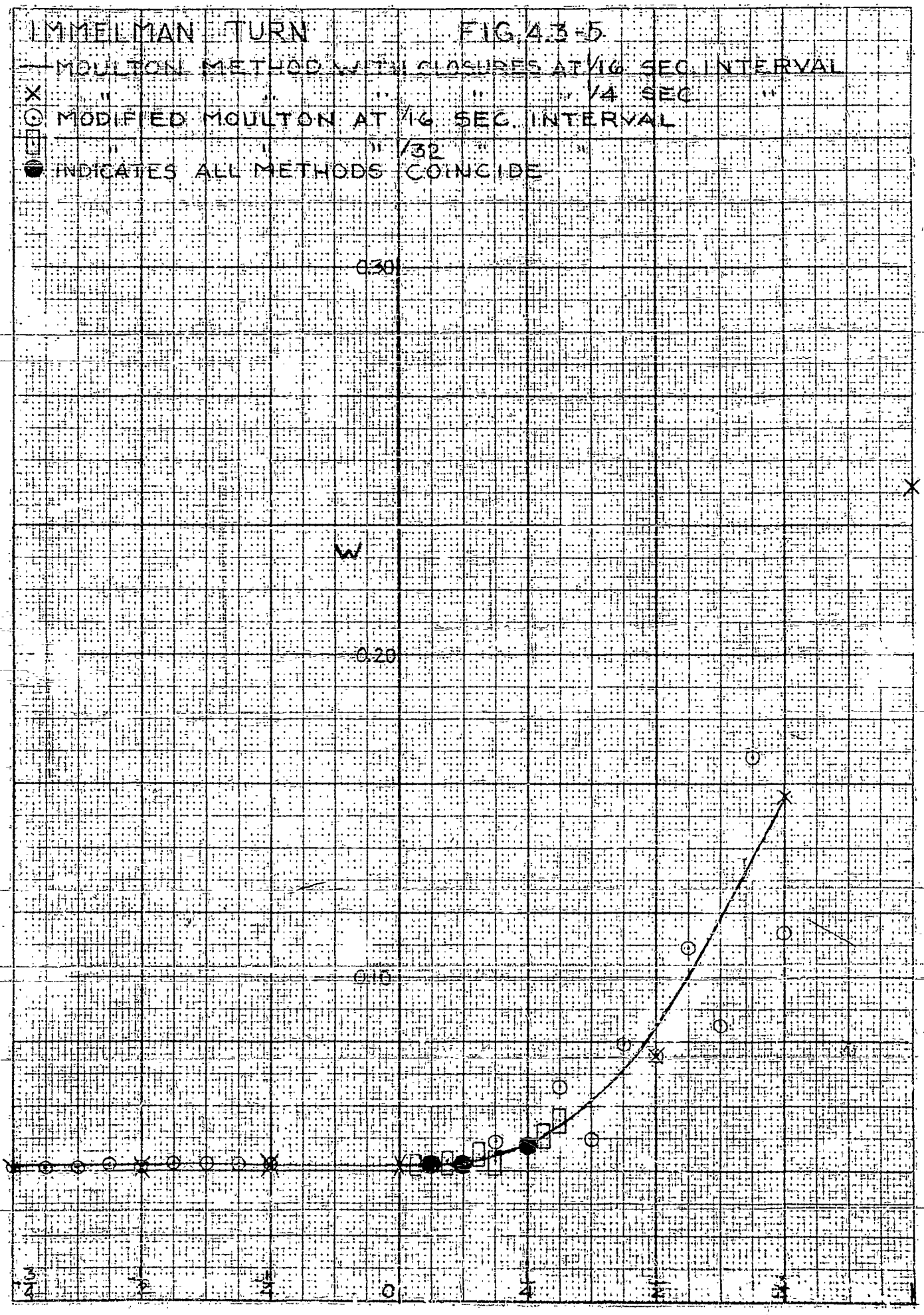
w	0.04																										
$\dot{w}$	.004																										
$\ddot{w}$	.002																										
$\dddot{w}$	-.18			-18	-1.5																						
$\ddddot{w}$	.43		-.01	44	3.7			-18	-1.5																		
$\dddddot{w}$	1.4		-.01	145	12		-.01	44	3.7			-18	-1.5														
q	-.01																										
$\dot{q}$	.21																										
$\ddot{q}$	-.21			- 21	-1.8																						
$\dddot{q}$	.31			32	2.7			-21	-1.8																		
$\ddddot{q}$	2.0		-0.02	208	18			32	2.7			-21	-1.8														
$\dddddot{q}$	-52		0.02	-931	-79		-.02	209	18			32	2.7			-21	-1.7					-1.7	-10	0	458	9.1	-106

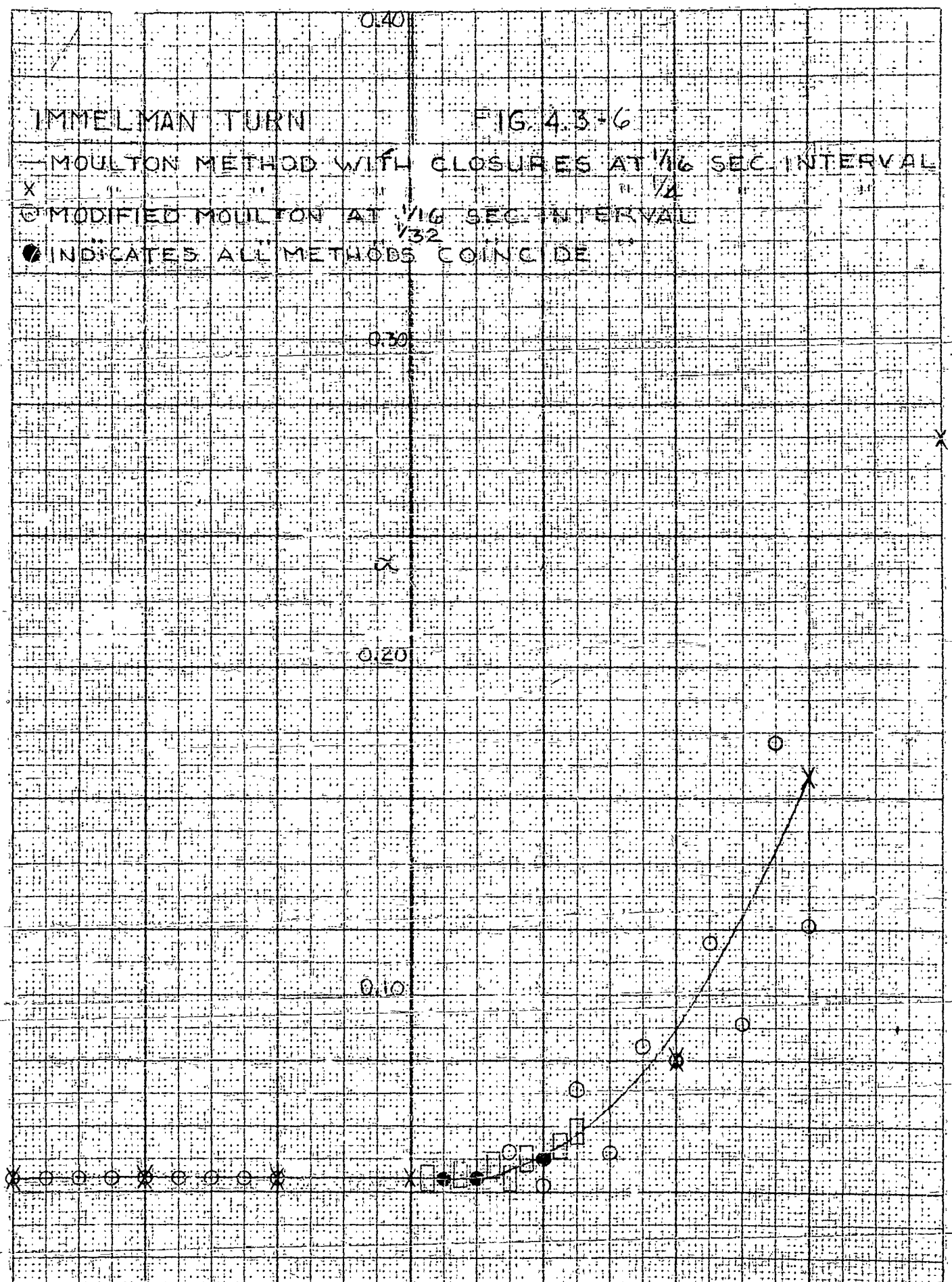
TABLE 4.3 - 4c

Constant	δ_a	δ_r	δ_e	δ_{fd}	δ_a	δ_r	δ_e	δ_{fd}	δ_a	δ_r	δ_e	δ_{fd}	δ_a	δ_r	δ_e	δ_{fd}	$\delta_a\delta_e$	$\delta_a\delta_{fd}$	$\delta_r\delta_e$	$\delta_r\delta_{fd}$
.469																				
.002																				
.000																				
-.002	0	0	.437	.04																
.01	-.001	-.04	-.59	-.05	0	0	.44	.04												
-.02	.01	-1.0	-.63	-.08	-.001	-.05	-.57	-.05	0	0	.44	.04								
.00																				
.002																				
.07																				
.04	.18	9.1	0	0																
.44	.41	-.50	-1.4	-.12	.15	9.1	0	0												
-.20	2.0	46.6	-105	-8.9	.43	-.83	-1.4	-.12	.15	9.1	0	0								
.04																				
.001																				
.04																				
.04	0	0	-18	-1.5																
-.93	-.04	-1.5	44	3.7	0	0	-18	-1.5												
.76	-3.0	-113	142	12	-.04	-1.5	43	3.7	0	0	-18	-1.5								
.01																				
.24																				
-.22	1.1	.25																		
.28	-1.0	-.38	0	0	1.1	.25														
-.23	1.1	9.3	.56	.05	-.99	-.37	0	0	1.1	.25	0	0								
.67	-.67	-9.8	-5.4	-.48	1.1	9.6	.69	.06	-.99	-.38	0	0	1.1	.25	0	0	7.1	.006	-308	-26
.002																				
.05																				
.00	.193	-9.7	0	0																
-.22	-.30	1.7	-.18	-.01	.22	-9.7	0	0												
.09	.08	-46	-93	-2.8	-.30	136	4.13	-.01	.22	-9.7	0	0								
-2.7	-2.4	5.6	209	18	.81	-45	-20	-3.2	-.30	1.6	-.18	-.01	.22	-9.6	0	0	-486	-41	-117	-9.9
.00																				
.01																				
-.01	0	0	0	0																
-.05	0	-.02	10	.88	0	0	0	0												
.41	.02	-.88	-16	-1.3	0	-.02	10	.88	0	0	0	0								
.002																				
.044																				
.98																				
-.90	4.3	.98	0	0																
1.0	-4.0	-1.3	-.02	-.00	4.3	.98	0	0	0	0	0	0								
.50																				
.00																				
-.01	0	0	0	0																
-.25	-.02	-.00	0	0																
-5.2	-1.5	-.31	-.59	-.05	-.02	-.00	0	0	0	0	0	0								
-.02																				
.02																				
.10	0	0	-21	-1.8																
-.56	.03	-.79	32	2.7	0	0	-21	-1.8												
.09	2.1	-50	206	17	.05	-.78	32	2.7	0	0	-21	-1.8								
12	-8.2	202	-933	-79	3.0	-67	206	17	.05	-.78	32	2.7	0	0	-21	-1.7	-10	-.83	9.1	-.86	-105

δ_e^2	$\delta_e\delta_{fd}$	δ_{fd}^2	$\delta_a\delta_r$	δ_r^2
-10	-.83	9.1	-.86	-105







IMMELMAN TURN

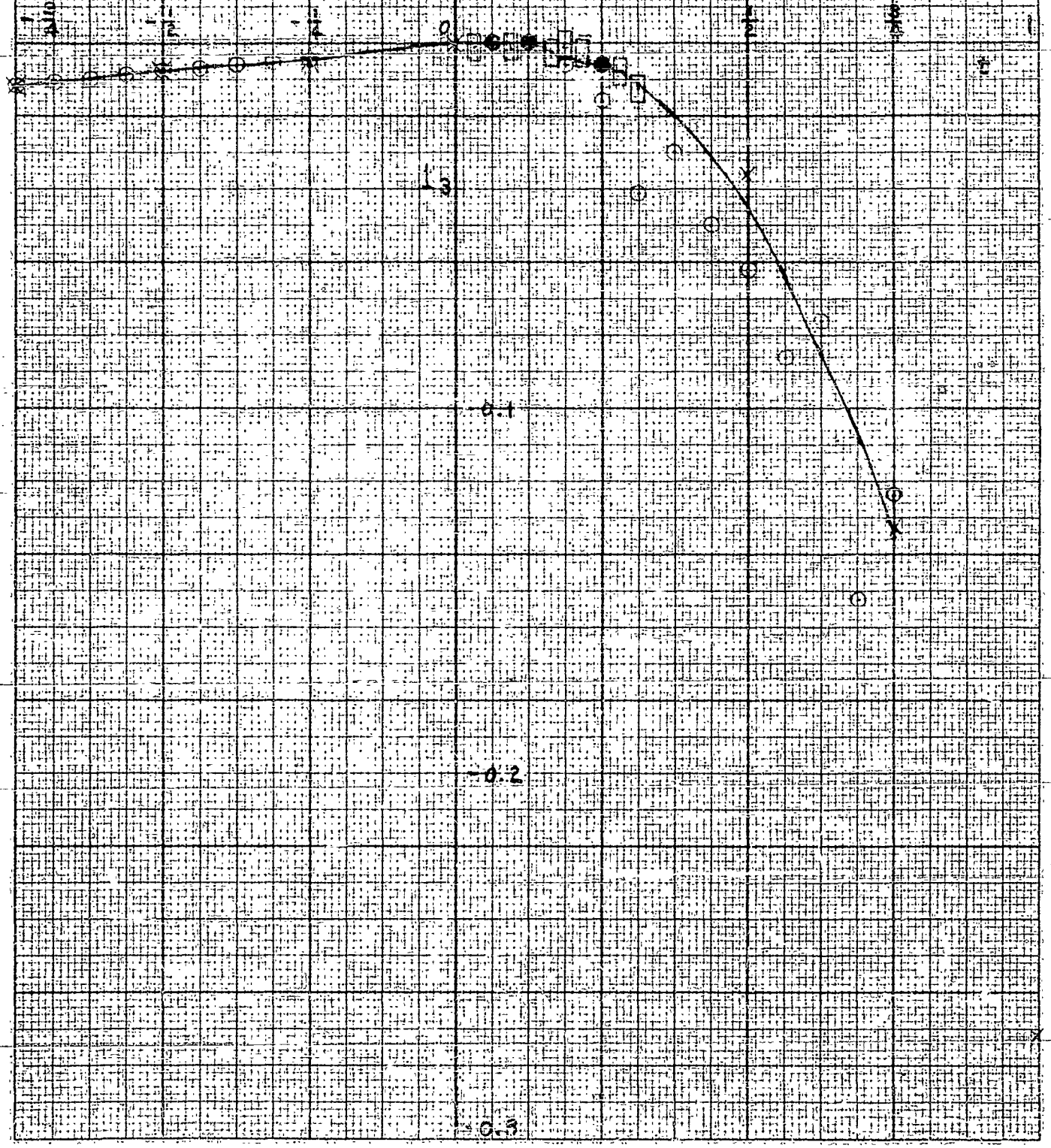
—— MOULTON METHOD WITH CLOSURES AT 1/16 SEC. INTERVAL.

X ⊙ □ MODIFIED MOULTON AT 1/16 SEC. INTERVAL

" " " 1/32 "

● INDICATES ALL METHODS GOING OF.

FIG. 4.3-7



IMMELMAN TURN FIG. 4.3-8

— MOULTON METHOD WITH CLOSURES AT 1/16 SEC. INTERVAL

× " " " " 1/4 " "

⊙ MODIFIED MOULTON AT 1/16 SEC. INTERVAL

" " 1/32 " "

● INDICATES ALL METHODS COINCIDE

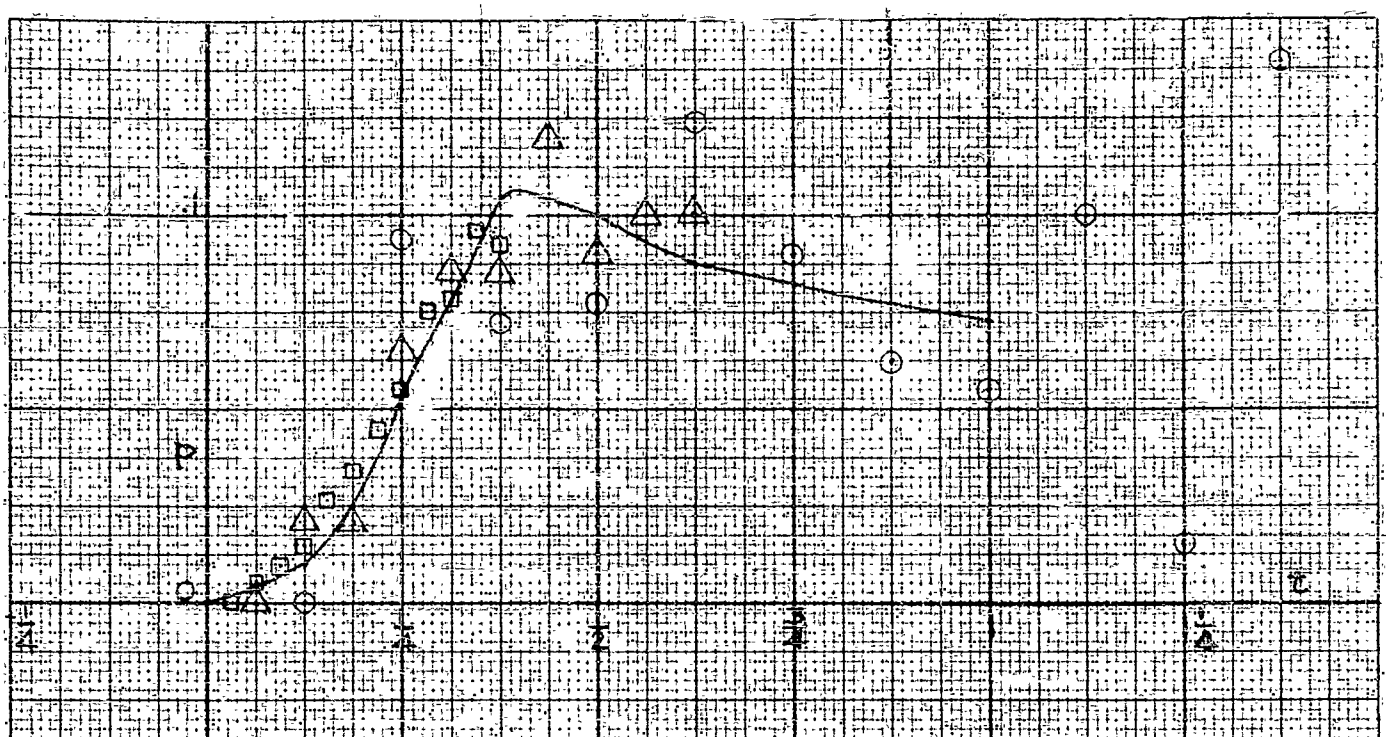


FIG. 4.3-9

TEST ROLL

MOULTON METHOD WITH CLOSURES AT 1/8 SEC. INTERVAL

⊙ MODIFIED MOULTON AT 1/8 SECOND INTERVAL

△ " " " 1/16 "

● INDICATES ALL METHODS COINCIDE

⊡ MODIFIED MOULTON AT 1/32 SEC. INTERVAL

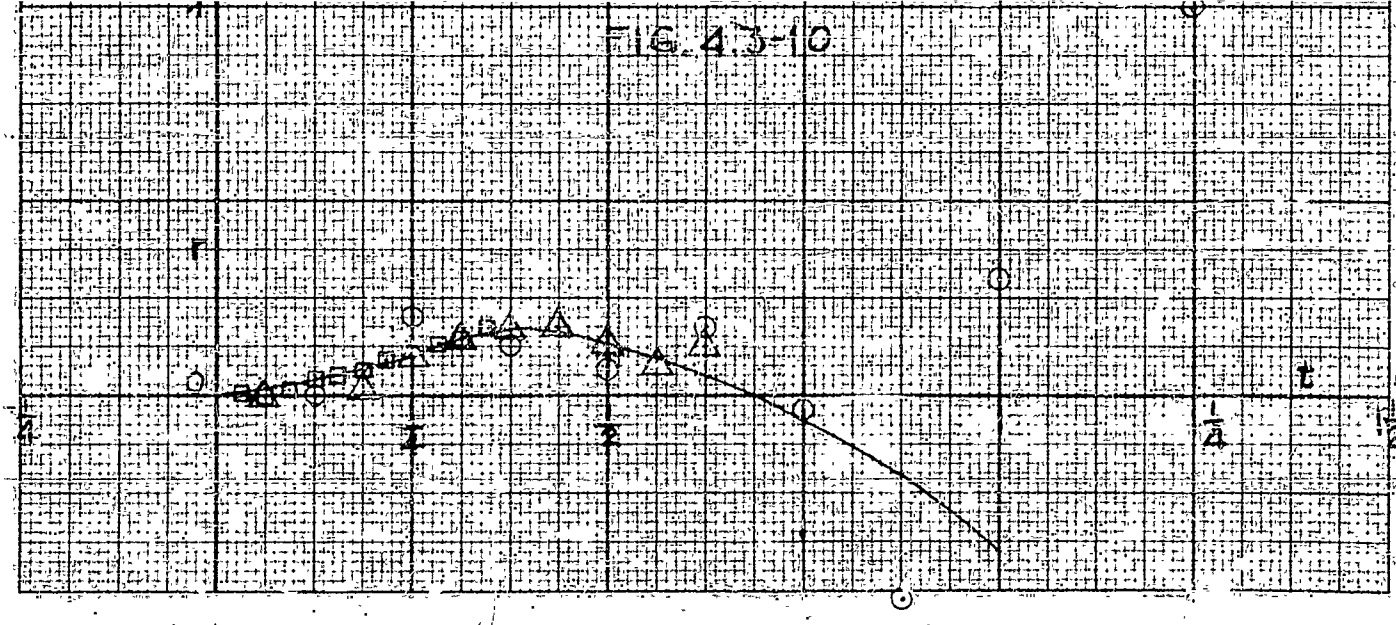


FIG. 4.3-10

TEST ROLL

— MOULTON METHOD WITH CLOSURES AT 1/8 SEC. INTERVAL

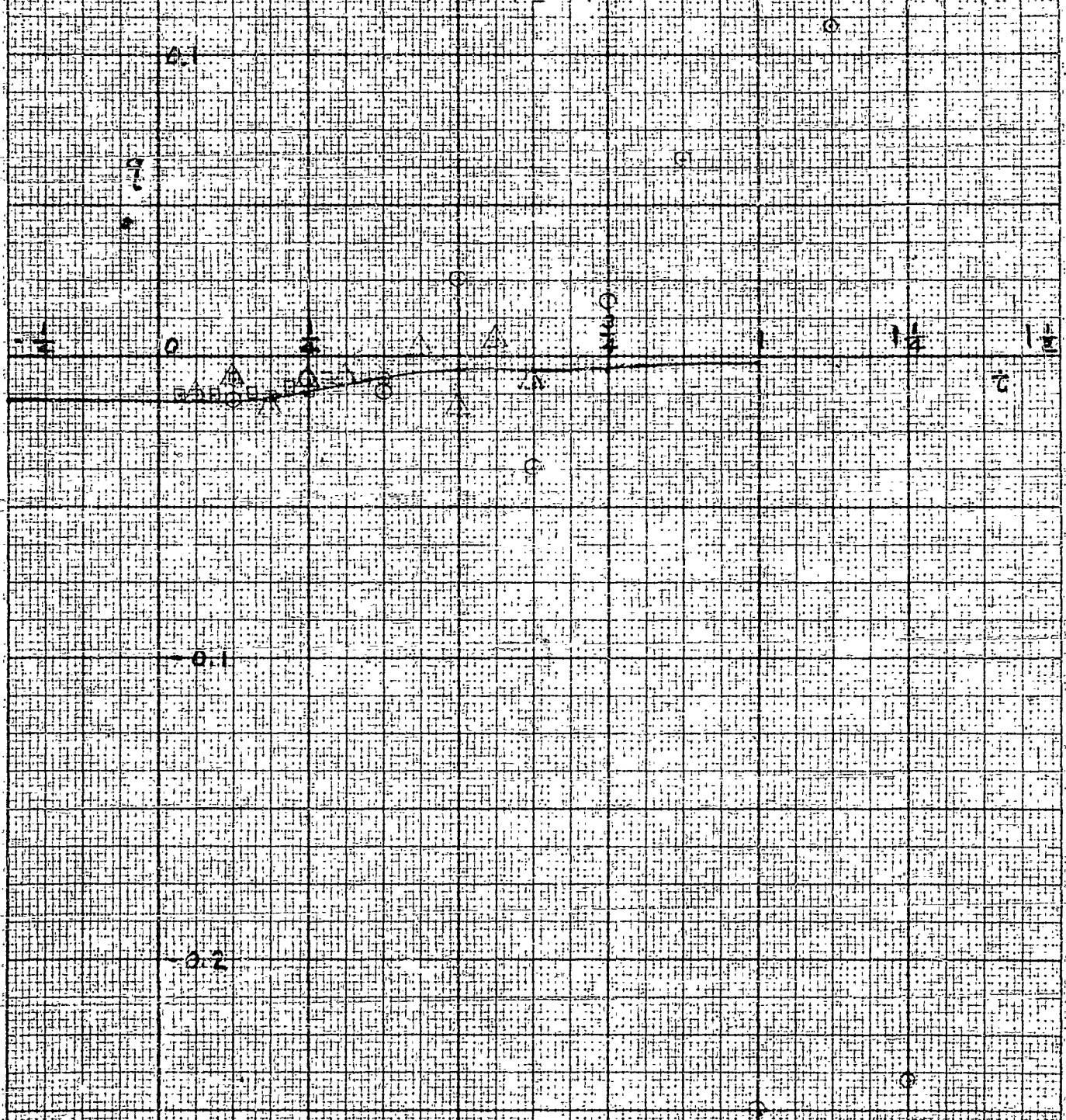
○ MODIFIED MOULTON AT 1/8 SEC INTERVAL

△ " " " 1/16 "

□ " " " 1/32 "

● INDICATES ALL METHODS COINCIDE

FIG. 4-3-11



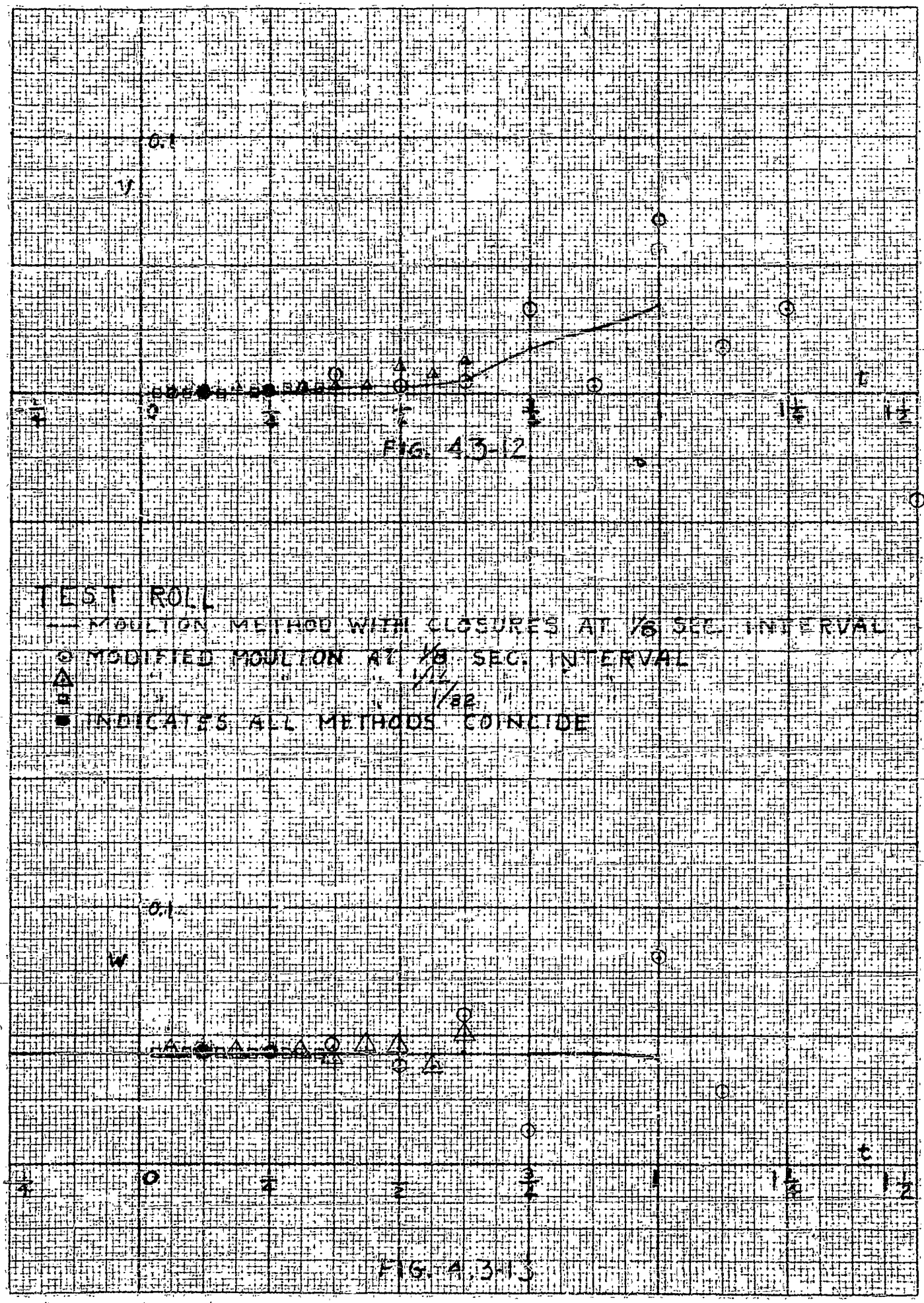


FIG. 4.3-12

TEST ROLL

— MOULTON METHOD WITH CLOSURES AT 1/6 SEC. INTERVAL

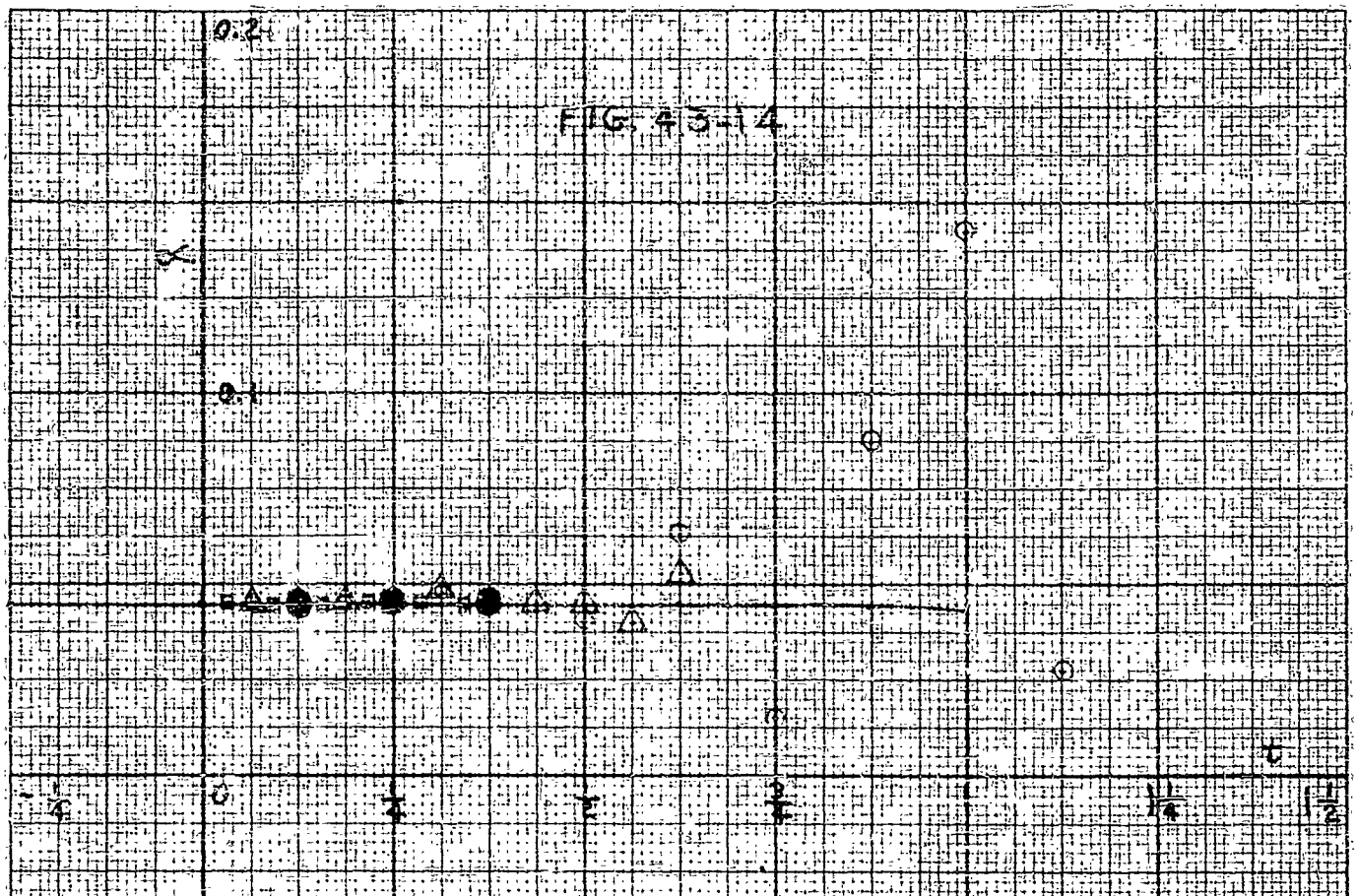
⊙ MODIFIED MOULTON AT 1/6 SEC. INTERVAL

△ " " 1/12 "

■ INDICATES ALL METHODS COINCIDE

FIG. 4.3-13

FIG. 43-14



TEST ROLL

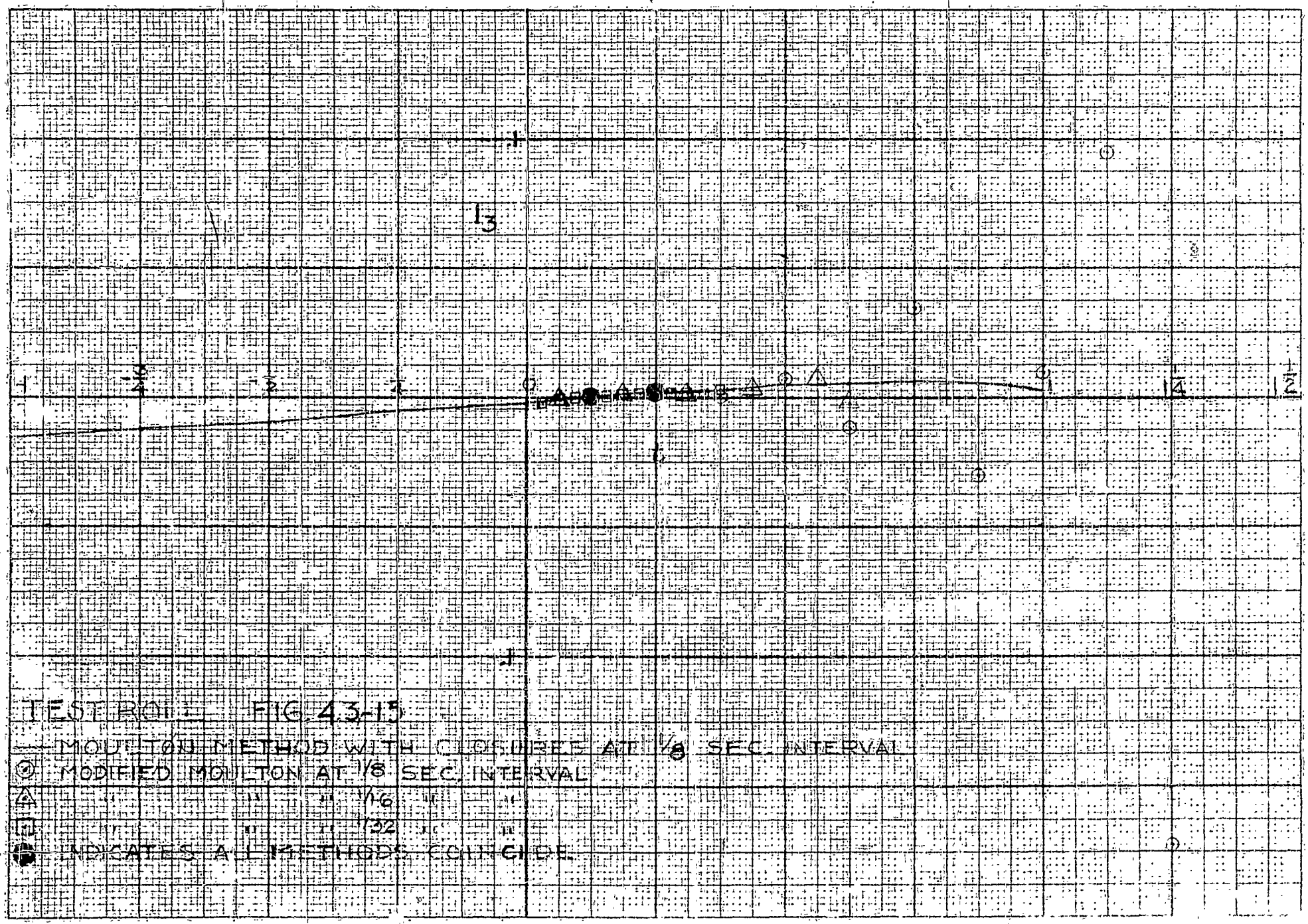
— MOULTON METHOD WITH CLOSURES AT ½ SEC. INTERVAL

○ △ MODIFIED MOULTON AT ⅛ SEC. INTERVAL

" " " " 1/16 " "

" " " " 1/32 " "

● INDICATES ALL METHODS COINCIDE



TEST ROLL FIG. 43-15

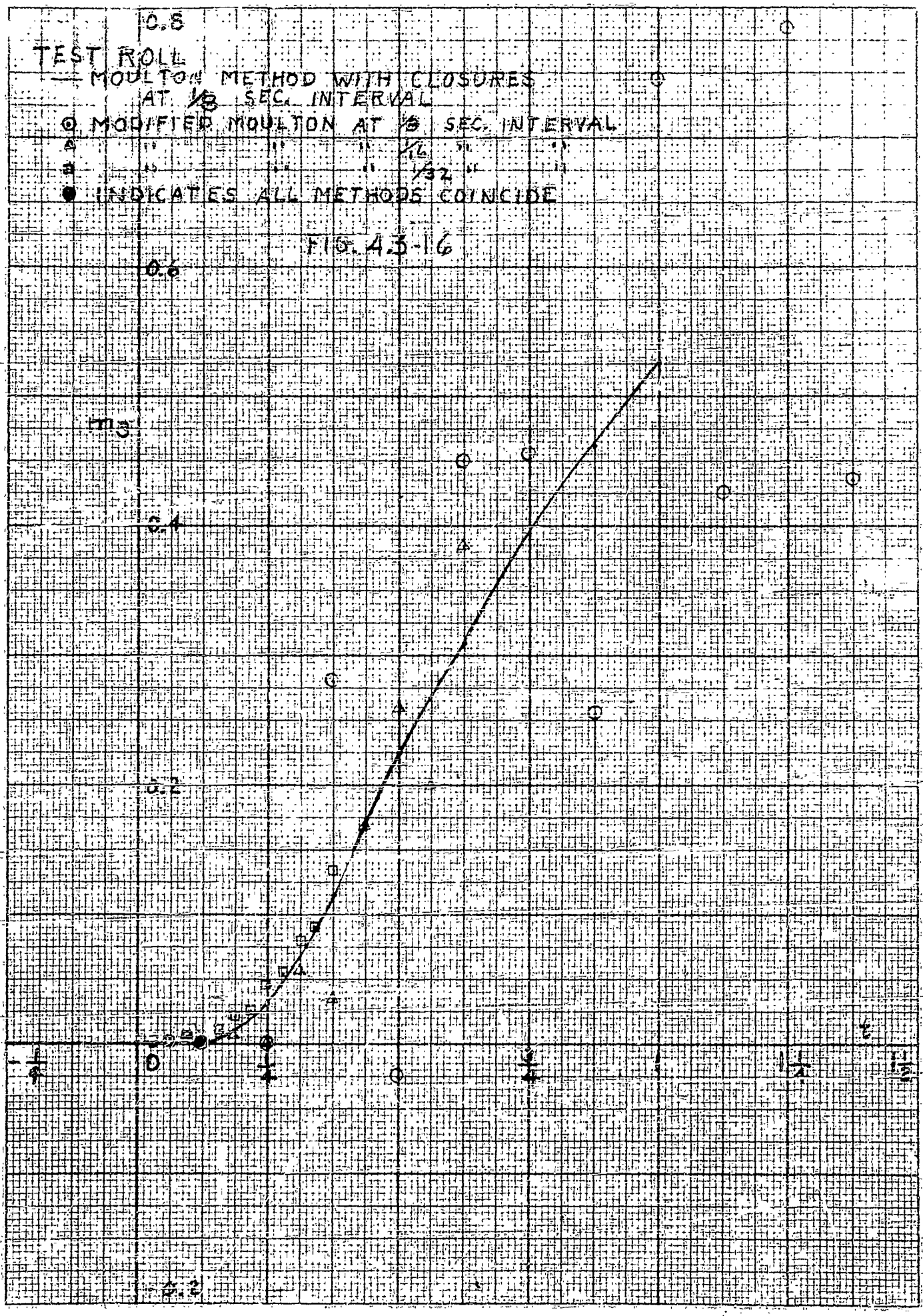
— MOULTON METHOD WITH CLOSURE AT 1/8 SEC. INTERVAL

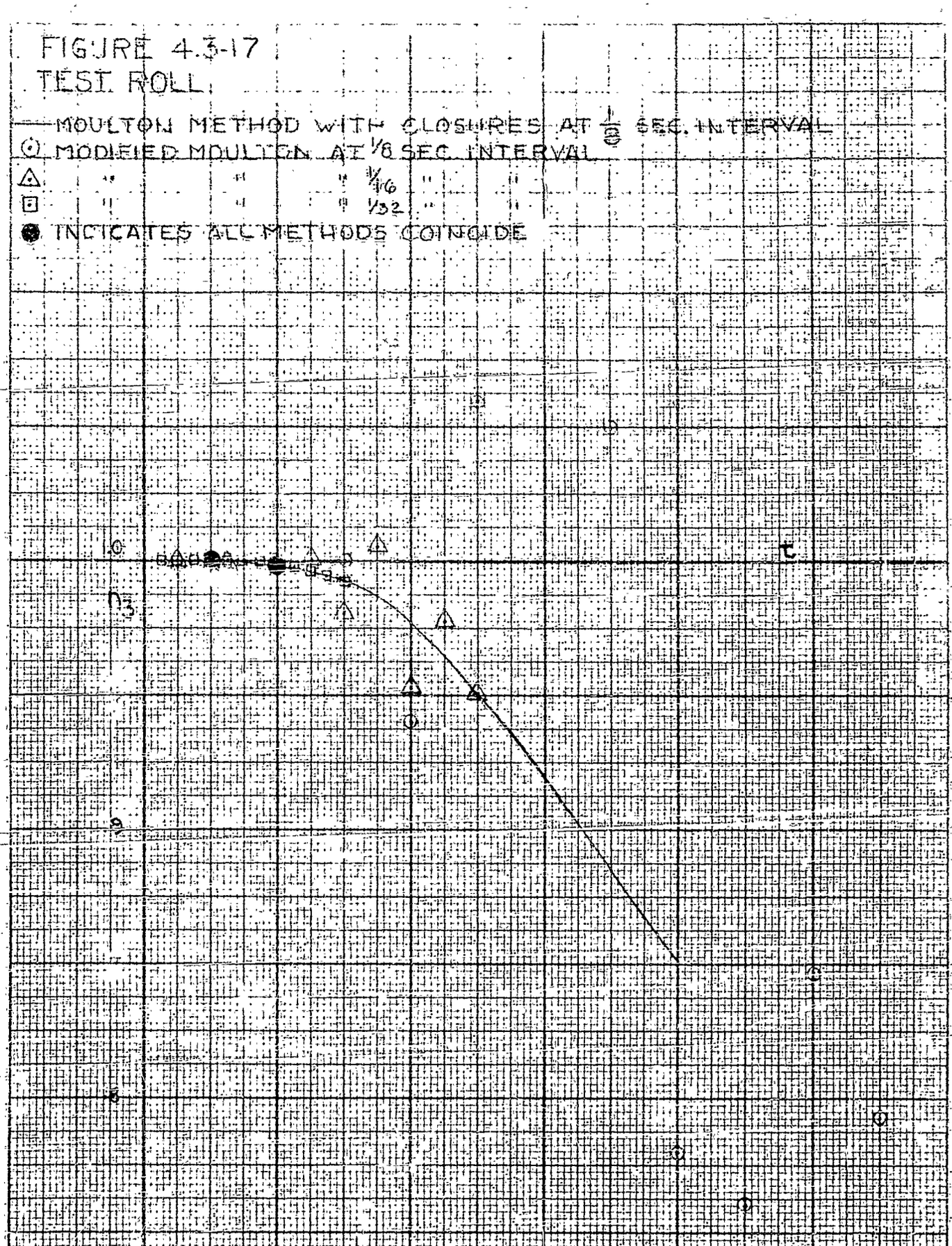
◎ MODIFIED MOULTON AT 1/8 SEC. INTERVAL

△ " " " 1/16 " "

□ " " " 1/32 " "

● INDICATES ALL METHODS COINCIDE.





4.4 Discussion of Error

This paragraph is devoted to a general discussion of all the recognized sources of error encountered throughout the problem. Since the manoeuvres were computed to six decimal places whereas the iterations were terminated with agreement to four places, it is reasonable to assume negligible round-off errors and they are ignored in the following discussion.

It was anticipated that the minor modifications introduced into the aerodynamic equations in section 2 would have negligible effect on flight characteristics. The two sets of equations were applied to level flight. The results are tabulated in Table 4.3 - 3 and are in fact in substantial agreement. Consequently, it is felt that for the purpose of flight simulation, the transformed equations of P 2.7 are equivalent to the given Frco Equations.

The next source of error is the truncation error which is inherent in the use of an approximation polynomial. This error, which depends both on the order of approximation, k, and on the selected time interval, $\Delta t = t_{n+1} - t_n$ is given by

$$E_{k,\Delta t} = a_k \cdot x^{(k+2)}(\xi) \cdot (\Delta t)^{k+2} \qquad 4.4-1$$

where a_k = a constant, dependent on k and on the formula used, $x^{(k+2)}(\xi)$ = value of the $(k+2)$'th derivative at the instant ξ and ξ = an (undetermined) instant within the time interval (t_{n-k}, t_{n+1})

The values of a_k for the open and the closed formulas 4.3 - 1, 2 are tabulated in Table 4.4 - 1. Thus, for example, the truncation error associated with equation 4.3 - 1 is

$$E_4 = 0.33 \; x^{(6)}(\xi) \; (\Delta t)^6 \qquad (4.4 - 1)$$

Table 4.4 - 1. Truncation error coefficient, a_k.

k	1	2	3	4
open	$\frac{5}{12} = .42$	$\frac{3}{8} = .38$	$\frac{251}{720} = .35$	$\frac{95}{288} = .33$
closed	$\frac{1}{12} = .083$	$\frac{1}{24} = .042$	$\frac{19}{720} = .026$	$\frac{3}{160} = .019$

The following three conclusions are a direct result of equation 4.4 - 1 and table 4.4 - 1:

(i) the closed-integration error is from five to fifteen times smaller than that of the open integration;

(ii) the error coefficient a_k decreases slowly as the order of approximation increases;

(iii) for any preassigned order of integration k, the error increases approximately as the (k + 2)'th power of the interval, i.e. as $(\Delta t)^{k+2}$. In particular, if the interval be doubled the error increases roughly by 2^{k+2}.

Hi-Speed Roll (Assuming $\frac{1}{16}$ second intervals $\delta^{(j)} = 0$ for all $j \geq 1$, all δ's)

4.4-2a

Parameter	E_1 (open)	E_2 (open)	E_3 (open)	E_1 (closed)	E_2 (closed)	E_3 (closed)
u	.00018	.00012	.00005	.00004	.00001	.00000
v	.00220	.00069	.00005	.00042	.00008	.00000
w	.00170	.00012	.00045	.00033	.00001	.00003
p	.00069	.00005	.00011	.00013	.00001	.00001
q	.00640	.00180	.00077	.00120	.00020	.00006
r	.00180	.00140	.00014	.00034	.00016	.00001
l	.00066	.00015	.00002	.00013	.00002	.00000
m	.00140	.00390	.00076	.00027	.00043	.00006
n	.01100	.00100	.00133	.00220	.00011	.00010

4.4-2b Immelman Flip-over

q	.00510	.00180	.00100	.00020
r	.00150	.00130	.00029	.00015
m	.00099	.00440	.00019	.00049

4.4-2c Inside Loop

q	.01200	.01400	.00250	.00150

4.4-2d "Immelman turn"

w	.00022	.00004	.00001	.00004	.00001	.00000
q	.00042	.00020	.00003	.00008	.00002	.00000

4.4-2e Test Roll

q	.00040	.00019	.00005	.00008	.00002
n	.00003	.00003			

On the assumption that the -derivatives do not contribute appreciably to the parameter derivatives, the truncation error for $\frac{1}{16}$ second interval for selected integrations were computed from the derivatives of Table 4.3 - 4. In particular, all the values for the High Speed Roll were computed and are shown in Table 4.4 - 2a; the worst cases for the Immelman flipover and the inside loop in 4.4 - 2b, c; and the worst cases for "Immelman turn" and test roll (which took account of δ_e and δ_a, respectively) in 4.4 - 2d, e. An examination of the table indicates that third order open integration results in a maximum error of about 10^{-3} for the $\frac{1}{16}$ second interval; with closure the error is further reduced to about 10^{-4}. The error in "q" is perhaps an exception in the case of the Inside (and Outside) Loop, since E_2 is very large in this case. However, it is not certain that the artificially chosen flight conditions are physically realizable and this error may be falsely exaggerated; computation of a portion of the manoeuvre on a high-speed calculating machine would yield a more realistic value.

In any event, it appears that third or at most fourth order closed integration is acceptable at $\frac{1}{16}$ second interval insofar as truncation error is concerned. It follows that the computations of P 4.3 by Moulton method with closures, which were repeated until agreement to 10^{-4} was obtained, are accurate to that precision.

To determine the error of the Modified Moulton method, note first that the error in 4.2 - 1 is the open integration error due to $[\dot{x}_1(\dot{x})_n]_4$, given by

$$e_4^1 \leq .33 \left| x^{(6)} \right|_{max} \cdot (\Delta t) \quad (4.4 - 2)$$

where $\left| x^{(6)} \right|_{max}$ is the maximum value of $x^{(6)}$ in the interval (t_{n-4}, t_{n+1}). If δ_{n+1} be assumed perfectly accurate for the moment, then the value obtained for $\dot{x}$ is given by

$$x'_{n+1} = f(t_{n+1}, x'_{n+1})$$

$$\approx f(t_{n+1}, x_{n+1}) + \frac{\partial f}{\partial x}(x'_{n+1} - x_{n+1})$$

$$\approx \dot{x}_{n+1} + \frac{\partial f}{\partial x} e'_4 \qquad (4.4 - 3)$$

so that the error is $\dot{x}_{n+1}$ is approximately $\frac{\partial f}{\partial x} e'_4$ Thus, in equation (4.2 - 3) the error consists of two terms, one due to the closed integration 0_2, the other due to the error in $\dot{x}$.

Finally, from equation 4.2 - 5, it follows that the full error, e_m in proceeding from $\dot{x}_{n+1}$ to x'_{n+2} is composed of the following additive terms:

(i) the error due to closed integration 0_2

$$\leq .02 \left| x^{(6)} \right|_{max} (\Delta t)^6,$$

(ii) the error introduced by $\dot{x}$, given by equations 4.3 - 3 and 4.4 - 3 to be

$$\leq 2.98\, e'_4 \left| \frac{\partial f}{\partial x} \right| \Delta t = 0.187\, e'_4 \left| \frac{\partial f}{\partial x} \right|, \text{ and}$$

(iii) the error due to the difference of two open integrations in two successive intervals.

The second error can be written as

$$.19 \times .33 \left| x^{(6)} \right|_{max} (\Delta t)^6 \left| \frac{\partial f}{\partial x} \right|;$$

inspection of the differential equations shows $\frac{\partial f}{\partial x} < 5$

always and for all parameters, so that this error cannot exceed

$$0.3 \left|x^{(6)}\right|_{max} (\Delta t)^6.$$

The third error is of the order of

$$.33 (\Delta t)^6 \left[\left|x^{(6)}\right|_{max} - \left|x^{(6)}\right|_{min}\right] \leq 2x^{(7)} \Delta t^7.$$

Consequently, assuming neglibible $x^{(7)}$,

$$e_m \leq (.02+.3) \left|x^{(6)}\right| \frac{1}{3} (\Delta t)^6 + 2x^{(7)} \Delta t^7$$

i. e.,

$$e_m < \frac{1}{3} \left|x^{(6)}\right|_{max} (\Delta t)^6 \qquad (4.4 - 4)$$

Table 4.4 - 3 lists the maximum errors for the first $\frac{1}{16}$ second interval in which the modified Moulton method was applied to the "Immelman turn" and to the test roll, using equation 4.4 - 4. At the same time the actual error (difference from the iterated Moulton method) is recorded. It is clear from the table that the errors incurred greatly exceed the anticipated maxima. It follows that the truncation error is not the only source of error; in fact, it appears that the truncation error of the modified Moulton method is not large enough of itself to have introduced excessive error.

Table 4.4 - 3. Actual error vs. computed maximum truncation error.

Manoeuvre	Parameter	Actual error	Computed Maximum	
Immelman turn	w	.000,8	.000,01	
	q	.016,3	.000,03	
Test roll	q	.003,1	.000,05	
	r	.002,6	.000,01	

There are two additional sources of error which account for the failure of the modified Moulton method at $\frac{1}{16}$ second interval.

The first of these has to do with pilot and instructor information. It was pointed out in P 4.3 that δ- behaviour can be rapidly varying or discontinuous. However, the effective δ- behaviour is implicit in the approximation formula utilized. For example, in the case of a step discontinuity the δ- behaviours implied for first and fourth orders are shown in Figure 4.4 - 1 and for small and large intervals in Figure 4.4 - 2, closed integration being assumed in both cases. Where the integration is open, the implicit δ_p behaviour is determined by previous history so that the δ- curve does not even pass through the true value of δ at time (n + 1). Thus the implied fourth-order δ-behaviour in the case of the Test Roll is shown in Figure 4.4 - 3. A method for reducing this source of error is suggested in P 4.5.

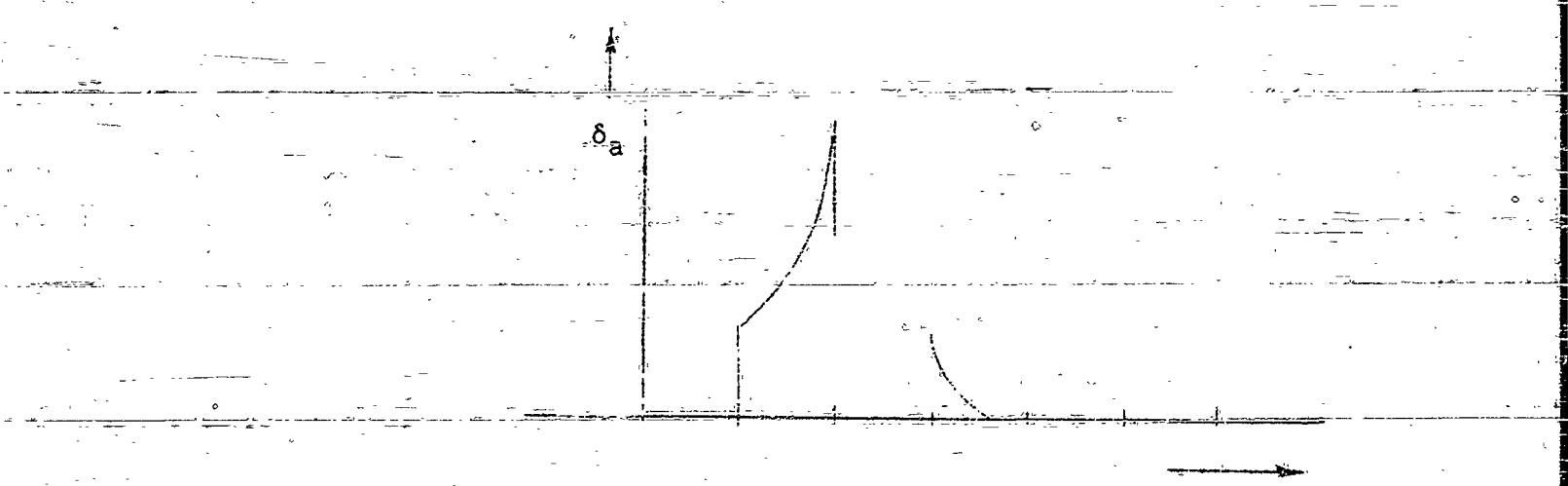


Figure 4.4 - 3

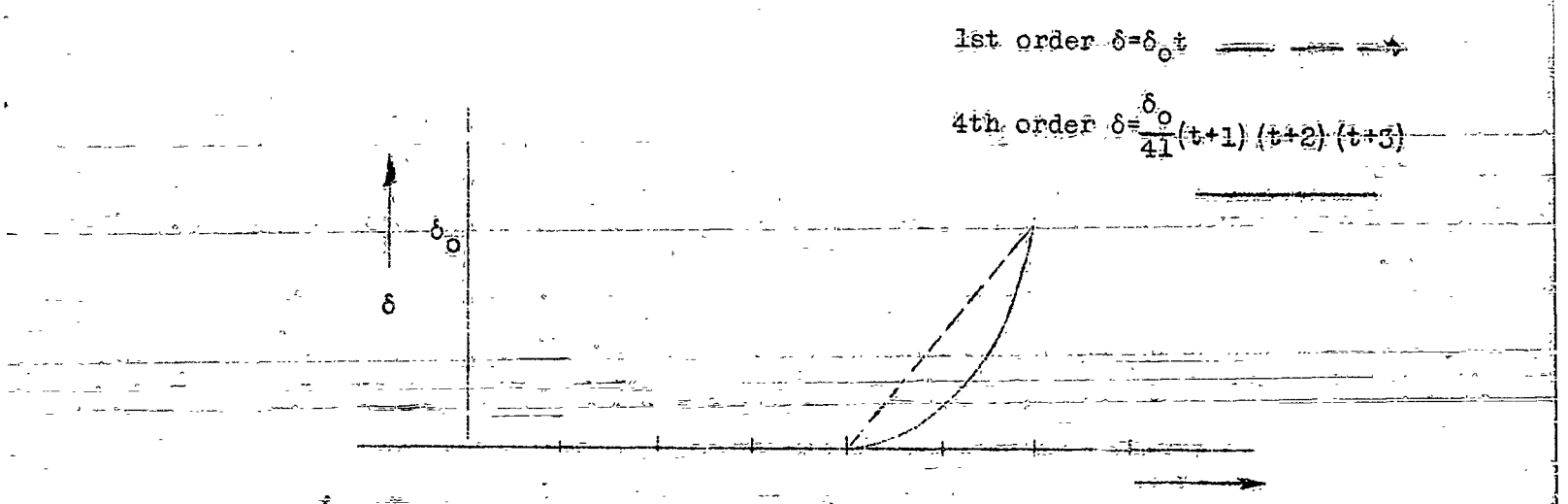


Figure 4.4-1

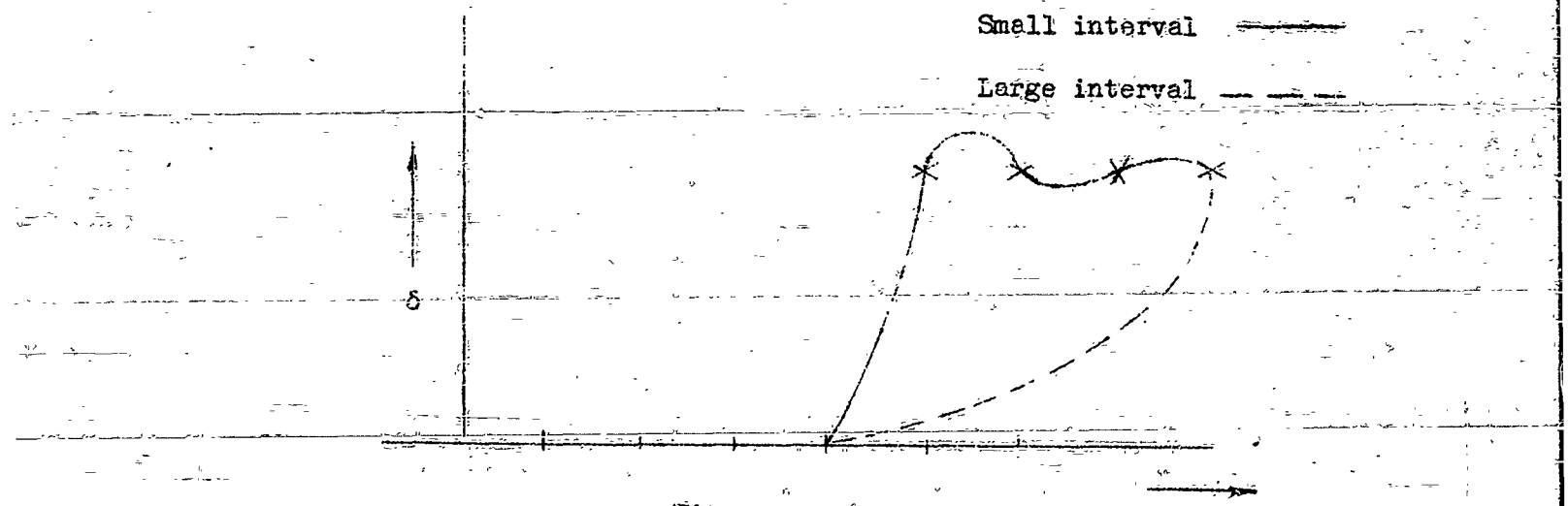


Figure 4.4-2

The second source of error is the delayed propagation of disturbances through the parameters; this is equivalent to the error caused by compliance in an analogue simulator (see P 3.3). In the case of the Moulton procedure, the compliance error is eliminated during the iterations; in fact, the major purpose of the many closures is to permit the primary disturbance computed by the open integration to filter through and execute the secondary disturbances. Examples taken from the test roll computation are shown in Table 4.4 - 4, together with the corresponding results via the modified Moulton method (all at 1/8 second intervals).

Table 4.4 - 4. Test roll computations illustrating compliance error.

Parameter	Moulton method, time and iteration number						Modified Moulton method, time	
	$3\frac{1}{8}$ -1	$3\frac{1}{8}$ -2	$3\frac{1}{8}$ -3	$3\frac{1}{4}$ -1	$3\frac{1}{4}$ -2	$3\frac{1}{4}$ -3	$3\frac{1}{8}$	$3\frac{1}{4}$
p	0	.0110	.0106	.0903	.0559	.0574	0	.0940
r	0	.0022	.0021	.0177	.0106	.0110	0	.0189
m	0	0	.0038	.0319	.0449	.0328	0	0

From Table 4.4 - 4 it is evident that the first iteration is almost but not quite fully effective in computing the unpredicted behaviour of δ_a on p and r, two of the parameters immediately affected by δ_a - behaviour. On the other hand, it is only after the alteration of p and r has occurred that the resultant effect on m_3 is obtained; clearly, this is because m_3 is only indirectly dependent on δ_a through p and/or r.

A primitive alteration of the method of integration has been suggested, and initial idealized computations indicate that this approach shows promise. Insufficient time availed to study the new method in any detail.

SECTION 5

PROGRAMMING

5. PROGRAMMING

The error introduced in the integration of the kinematic equations is proportional to some power of the interval adopted; for example, if the interval is Δt and if a quadrature formula using five values of $\dot{x}$ be employed, the error is proportional to $(\Delta t)^5$, (see P. 4.3). The interval adopted in a real-time simulator has a lower bound determined by the length of time required to perform one complete cycle of computation.

In order to obtain an approximate value for the cycle interval, the equations have been programmed in their entirety using the modified Moulton method of integration (see P. 4.2). It was found possible to decrease the amount of computation by the method of straight line approximations described in P. 5.1, and it is probable that further gains will be made by employing the minor cycle technique outlined in P. 5.6.

The program chart used for the display of the routine incorporates features which, to the best of our knowledge, are novel; it is possible to count the number of operations in the routine and at the same time it was found that the flow of operations is readily followed. The required memory space may also be obtained from the chart, but care must be taken in this case that there are no duplications in different subroutines and that account has been taken of possible multiple use of space in the memory. Since the program in this report is only a tentative one, these precautions were omitted. The program chart is given in P. 5.2 while the number of operations, the number of memory locations and estimates of the cycle interval are given in P. 5.3, 5.4 and 5.5, respectively.

5.1 Straight Line Approximations

The method of straight line approximations takes advantage of the compare (or transfer) operation to accelerate the computation of functions at the expense of memory space. The principle of the method will be explained through the use of an example, as follows:

Suppose that it is required to compute e^{-x} for $0 \leq x \leq 1.3$ to an accuracy of $g = 0.001$. One could conceivably use the series expansion

$$e^{-x} = 1 - \frac{x}{1} + \frac{x^2}{2!} - \frac{x^3}{3!} + \frac{x^4}{4!} - \frac{x^5}{5!} + \frac{x^6}{6!} - \frac{x^7}{7!} \qquad (5.1\text{-}1)$$

Note that all eight terms are necessary to ensure the desired accuracy for $x = 1.3$. To compute e^{-x} by this equation would require (1) that the number 1.3 be accommodated by the computer and (2) that time be allotted to perform 7 multiplications and 7 additions, both of which are undesirable.

It is possible to eliminate both faults by replacing equation 5.1-1 by a set of linear approximations L_1, L_2, L_3, etc. In particular, let L_1 approximate e^{-x} to the desired accuracy over the range $x = 0$ to the (as yet undetermined) value $x = a$ (see figure 5.1-1). Then it is required (1) that L_1 pass through the point $(0, 1-g)$; (2) that L_1 be tangent

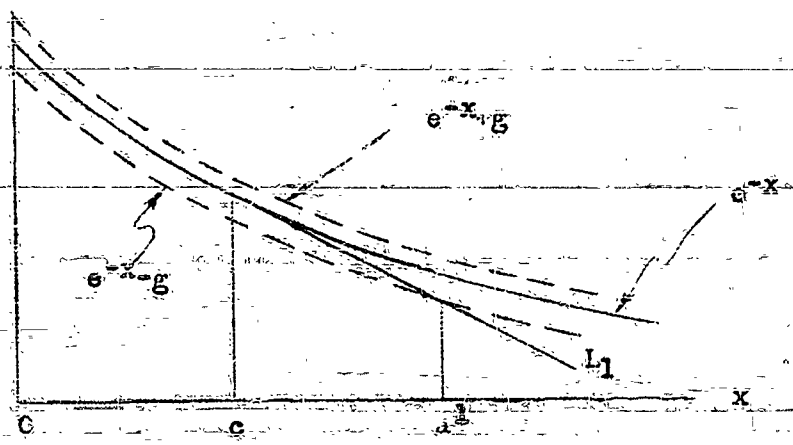


Figure 5.1-1

to the curve $y = e^{-x} + g$ at $(c, e^{-c} + g)$; and (3) that a be chosen to be less than or equal to a', the value of x at the intersection of L_1 and the curve $y = e^{-x} + g$. Thus if L_1 be given by

$$L_1: y = mx + b \qquad (5.1\text{-}2)$$

then m and b are obtained from the equations

$$e^{-0} + g = m(0) + b \qquad (5.1\text{-}3)$$

$$e^{-c} + g = mc + b \qquad (5.1\text{-}4)$$

$$-e^{-c} = m$$

and a' is obtained from the equation

$$e^{-a'} + g = ma' + b \qquad (5.1\text{-}5)$$

It is easy to verify that for the case in question, L_1 is given by

$$L_1: y = 0.9990 - 0.9375x, \quad 0 \leq x < 0.1305 \qquad (5.1\text{-}6)$$

Beginning at x = 0.1305, the process is repeated to obtain

$$L_2: y = 0.9837 - 0.8198x, \quad 0.1305 \leq x < 0.2695$$
$$L_3: y = 0.9541 - 0.7098x, \quad 0.2695 \leq x < 0.4188 \qquad (5.1\text{-}7)$$

etc.

In this manner e^{-x} can be repreeented by eight straight lines, each line requiring two numbers for its definition, the set of lines requiring seven (or at most nine) additional numbers to define their ranges as shown in Figure 5.1-2.

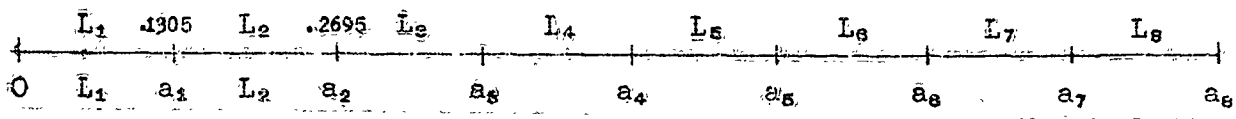


Figure 5.1-2

To determine which line is to be used in a particular evaluation of e^{-x}, the procedure adopted is outlined as follows, and its correctness is easily verified by trial:

Step 1. Compare x with a_4

	$x < a_4$				$a_4 \leq x$			
Step 2.	$x < a_2$		$x \geq a_2$		$x < a_6$		$a_6 \leq x$	
Step 3.	$x<a_1$	$a_1 \leq x$	$x<a_3$	$a_3 \leq x$	$x<a_5$	$a_5 \leq x$	$x<a_7$	$a_7 \leq x$
Step 4.	Use L_1	L_2	L_3	L_4	L_5	L_6	L_7	L_8

Clearly, the number of steps and the computation time required is independent of the magnitude of x. Three comparisons followed by one multiplication and one addition suffice in every case. The net saving over equation 5.1-1 is then at least 6 multiplications and 3 additions.*

The method has been indicated in the programming by evaluating e^{-hk} by means of three lines (see subroutine No. Nine, P. 2.7). However, no attempt has been made to apply the method there in a precise manner since there is as yet no way to decide with what accuracy the exponential need be evaluated.

5.2 Program Diagrams

In order to determine the number of arithmetical operations and the number of memory positions required for the solution of the flight trainer equations, the equations have been programmed for a hypothetical digital computer using binary representation of information. This computor is capable of the fundamental arithmetic operations of addition, subtraction, multiplication, and division. It also can perform shifts equivalent to multiplying or dividing by a power of two, transfer of information from one memory position to another (this may also be accomplished in other ways e.g. by addition with zero), comparison of two quantities to determine these relative sizes (so-called "transfer" order), and to extract magnitudes of signal numbers.

Memory positions are assigned to all constants and variables. The (Analogue) magnitudes provided by the instructor or pilot, such as δ_a, δ_r, K_F, etc. are also assigned memory positions. These δ memory positions are filled periodically by converters which generate the digital equivalent of the analogue quantities. To achieve equipment economy it might be desirable to use only one such converter with switching devices so that these memory positions would be filled in succession throughout a computing cycle. However, the integration formula used in the programming assumes that these memory positions are filled simultaneously at the "date line". It is not yet definitely known whether or not it would be preferable to fill these memory positions at any other times.

Other memory positions store quantities which must be converted to analogue information to operate external indicating devices such as

flight instruments. These are read periodically for their contents either at the date line or at some other suitable time. The most suitable time has not as yet been determined. When any memory position is read for its contents, its contents are not erased. Erasure occurs only when new information is inserted into it or when it is deliberately cleared to zero.

Computations are imagined to proceed in order, that is, sequentially. Thus when any one computation is being performed no other computation is being done. This is because electronic digital computers (at any rate, those built to date) have only one arithmetic unit with its associated error detecting circuits. To perform the operation of addition, for example, upon two numbers A and B, there must be an instruction or "order" available which specifies the memory location of A, the memory location of B, the operation of addition, the memory location where A + B is to be sent and, in some computers, the location of the next order to be executed. In some machines e.g. the Harvard Mark I, the orders are usually arranged on a "sequencing tape", so that the location of the next order need not be specified. In the MSAC and EDVAC computers, the orders are treated just like numbers and all the information concerning the location of the operands, the result, and the type of operation i.e. the "addresses" and "order type" if an order is kept together in a memory position, so that the order must contain an address specifying the location of the next order. The Hurricane machine, however, requires two memory positions to hold an order. (For a more detailed discussion, see Section 3).

The equations of motion involve, in addition to purely arithmetic operations, however, the operation of integration which corresponds to the solution of a set of first order non-linear simultaneous differential equations. Moreover, these equations contain time-varying parameters (δ) whose behavior is not completely predictable. A method of reducing the approximate solution of these differential equations to pure numerical computations is described in Section 4 of this report. Briefly, the principle of the method is the following. If the values of all the variables and constants are given at time t_o, the values of all the variables at time t_1 where $t_1 = t_o + \Delta t$ can be computed; thus, if this process is repeated the values of the variables obtained will agree sufficiently closely to those obtained by a hypothetical exact solution of the differential equations. To show this clearly consider the equation in Subroutine No. Three for Δp which can be written: $\Delta p = 4 \int \frac{\dot{p}}{4} dt.$ (5.2-1)

For Δt = 1-16 seconds, P at time t_1 is related to p at time t_o by the following equation:

$$(p)_1 = (p)_o + 4\left[.1868 \frac{(\dot{p})_o}{4} - .3497 \frac{(\dot{p})_{-1}}{4} + .4450 \frac{(\dot{p})_{-2}}{4} - .3285 \frac{(\dot{p})_{-3}}{4} + .1307 \frac{(\dot{p})_{-4}}{4} - .0218 \frac{(\dot{p})_{-5}}{4}\right] \quad (5.2-2)$$

where the quantities $\frac{(\dot{p})_o}{4}$, $\frac{(\dot{p})_{-1}}{4}$, ... $\frac{(\dot{p})_{-5}}{4}$, $(p)_o$ have been previously evaluated.

Consider now the equation for 1_s in Subroutine No. One:

$$1_s = \int (rm_s - qn_s)\, dt \quad (5.2-3)$$

1_s at time t, is related to 1_s at time t_o by an equation of the same form as (5.2-2):

$$(l_3)_1^1 = (l_3)_0 + [.1868\ (\dot{l}_3)_0 - .3497\ (\dot{l}_3)_{-1} + .4450\ (\dot{l}_3)_{-2} - .3285\ (\dot{l}_3)_{-3} + .1307\ (\dot{l}_3)_{-4} - .0218\ (\dot{l}_3)_{-5}] \qquad (5.2\text{-}4)$$

However the direction cosines, of which l_3 is one, must be normalized. This is indicated as follows:

$$(l_3)_1 = 2(l_3)_1^1\ [1 - (e_3)_1] \qquad (5.2\text{-}5)$$

where $(e_3)_1$ is computed from the values of $(l_3)_1^1$, $(m_3)_1^1$, and $(n_3)_1^1$. Note that the significance of the subscripts outside the parentheses is to show whether the variable has the value at time t_0 or time t_1. The primes outside the parentheses indicate that the variables are tentative values subject to modification such as was done on l_3 in equation (5.2-5).

The computation of l_2, m_2, and n_2 in Subroutine No. Three merits comment. All three are first computed by integration; then depending upon the relative sizes of l_3, m_3, and n_3, one of the direction cosines l_2, m_2, and n_2 is replaced by one computed by a formula to insure that they are orthogonal to the l_3, m_3, n_3 direction cosines. Then normalization is performed on l_2, m_2, and n_2 in the same way that the direction cosines l_3, m_3, and n_3 were normalized. For further discussion, see Section 2.

When Subroutine No. One is begun, all values of the variables are known at time t_0; at the subsequent time of entry into Subroutine No. One, all values of the variables have been computed for time t_1. These new values of the variables are then available to be treated in the same manner as were the variables at time t_0. The so-called date-line is therefore crossed when entering Subroutine No. One.

Figures 5.2-2 to 5.2-9 inclusive show the programming for one computation cycle. The program diagrams are divided into numbered frames, the first number of which specifies the subroutine. The significance of the symbols is explained in Figure 5.2-1.

A few other items concerning the program diagrams need mentioning. In Subroutine No. One, the quantities b_{ix}, W_{fb}, W_{fw}, and W_a are not computed digitally. It was felt that they could be computed more conveniently by analogue means, since the integrands are either zero or have constant values. (See Section 6, F. 6.13)

When going from Subroutine No. Two to air, it is found that there is no past history available on the derivatives of p, q, r, u, v, w, l_s, m_s, n_s, l_2, m_2, n_2, and h necessary for the quadratures performed in Subroutine No Three. In Subroutine No. 3a, this past history is computed from values of variables that are saved for this purpose, using the equations that follow:

$$\frac{\dot{n}_s}{8} = -\frac{1}{4} l_s \dot{l}_s \text{ or } \frac{\dot{n}_s}{8} = 0 \text{ from Subroutine No. Twelve}$$

$$\dot{l}_2 = -2(\dot{m}_1 n_s + 8 m_1 \frac{\dot{n}_s}{8})$$

$$\frac{\dot{m}_2}{8} = 2^{-3} (\dot{m}_2 \dot{\psi})$$

$$\frac{\dot{n}_2}{8} = 2^{-2} (\dot{l}_s m_1 + \dot{m}_1 l_s)$$

$$\frac{5}{4}\dot{w} = -\frac{1}{2} (\dot{v}_t l_s + v_t \dot{l}_s)$$

$$10\dot{u} = \frac{1}{.05} (\dot{v}_t n_s + 8 v_t \frac{\dot{n}_s}{8})$$

$$\frac{\dot{r}}{8} = 2^{-2} (\ddot{\psi} n_s + 8 \dot{\psi} \frac{\dot{n}_s}{8})$$

$\frac{\dot{q}}{16} = 0$ or is computed in Subroutine No. Six

$$\frac{\dot{p}}{4} = 2^{-4} \left(\ddot{\psi}\ 1_3 + \dot{\psi}\ \dot{1}_3\right)$$

Derivatives are computed from the above expressions for

times -1, -2, -3, -4, and -5.

A similar situation arises when going from air to ground (entering Subroutine No. Ten). The variables for which there is no past history available concerning their derivatives are v_t, $\dot{\psi}$, 1_3, m_2, and n_3. These are computed from values of variables that are saved for this purpose, using the following equations:

$$\dot{\psi} = \frac{1}{2} \frac{r}{n_3}$$

$$\dot{m}_1 = -m_2\ \dot{\psi}$$

$$\dot{m}_2 = m_1\ \dot{\psi}$$

$$2\ \dot{1}_3 = -q$$

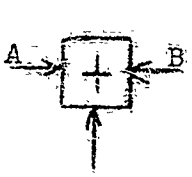
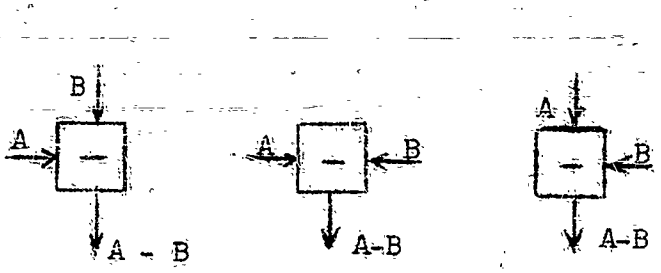
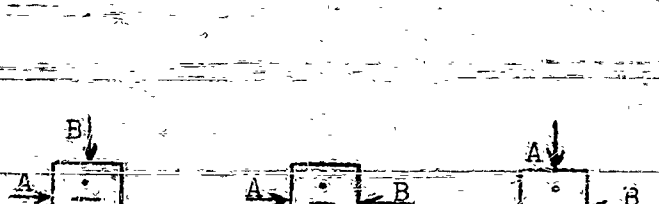
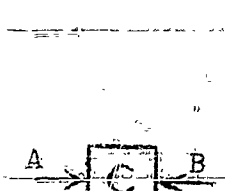
$$\ddot{\psi} = 2^2\ \frac{-n_3\ r/8 - r\ \frac{\dot{n}_3}{8}}{n_3^2}$$

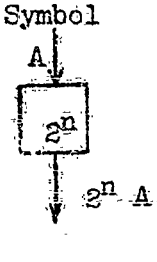
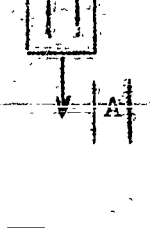
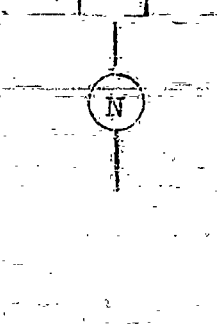
$$\dot{v}_t = 2^2\ \frac{.0125\ n_3\ 10\ \dot{u} - \frac{\dot{n}_3}{8}\ u}{n_3^2}$$

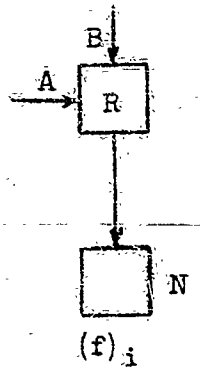
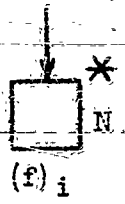
Derivatives are computed from the above expressions for times -1, -2, -3, -4, and -5.

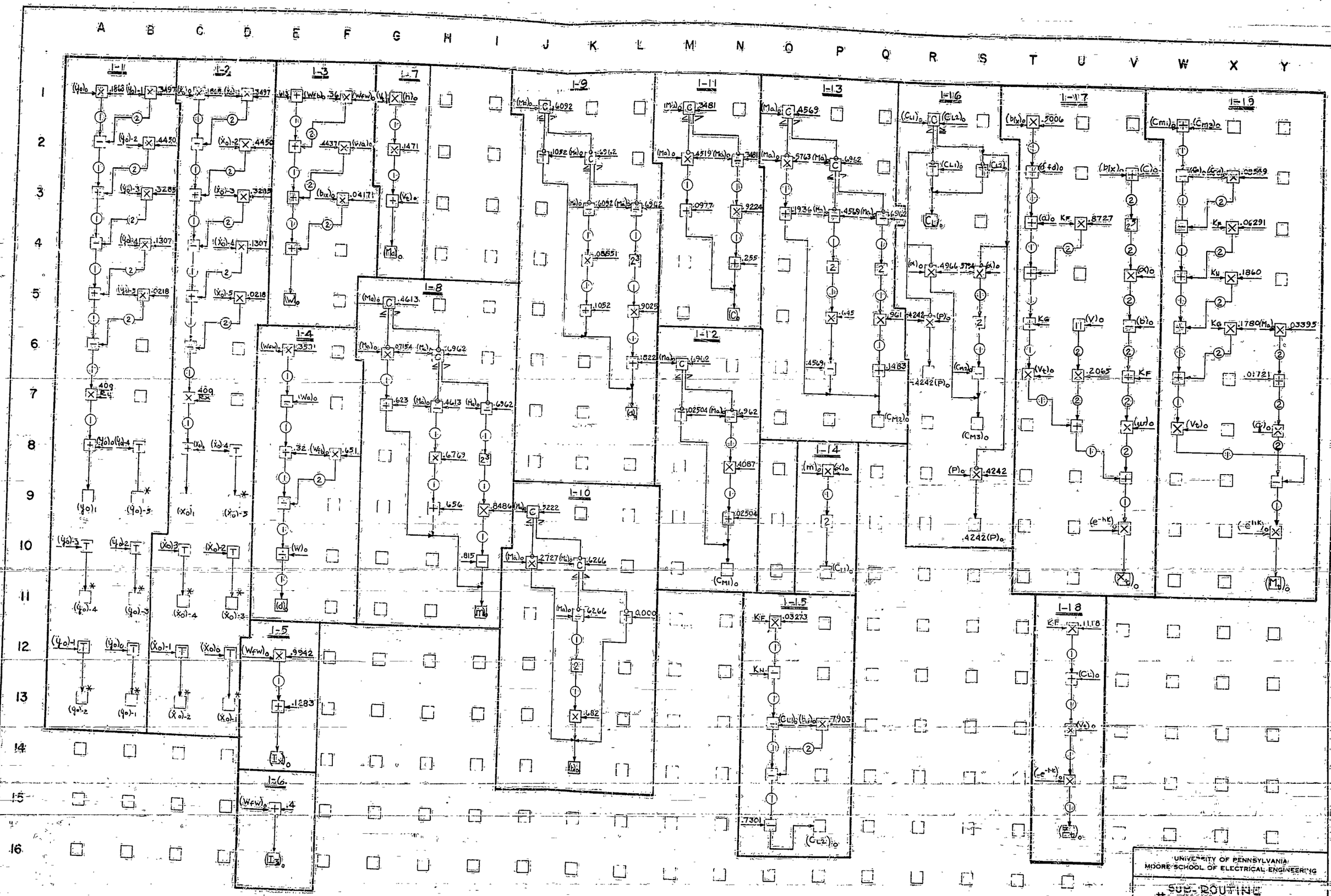
The above computations, however, are a consequence of the type of quadrature formula used. It is expected that a formula or method will be obtained that will not require this large amount of superfluous computation. (See P. 4.5)

Figure 5.2-1

Symbol	Operation
	Addition
	Subtraction
	Multiplication
	Division
	Compare Outputs are sequencing signals, not numbers.

Symbol	Operation
	Shift
	Transfer
	Sequencing signal from C operation makes the next operation performed the one indicated in this box.
	Extraction of magnitude of signed number.
	Result of operation R on A and B is temporarily stored in memory position N.

Symbol	Operation
	$(f)_i$, the result of operation R on A and B is sent to memory position N, or if N is not specified to the memory position shown by the coordinate number of the box
	Asterisk means that the subscript is the subscript the quantity would have in the next computation cycle



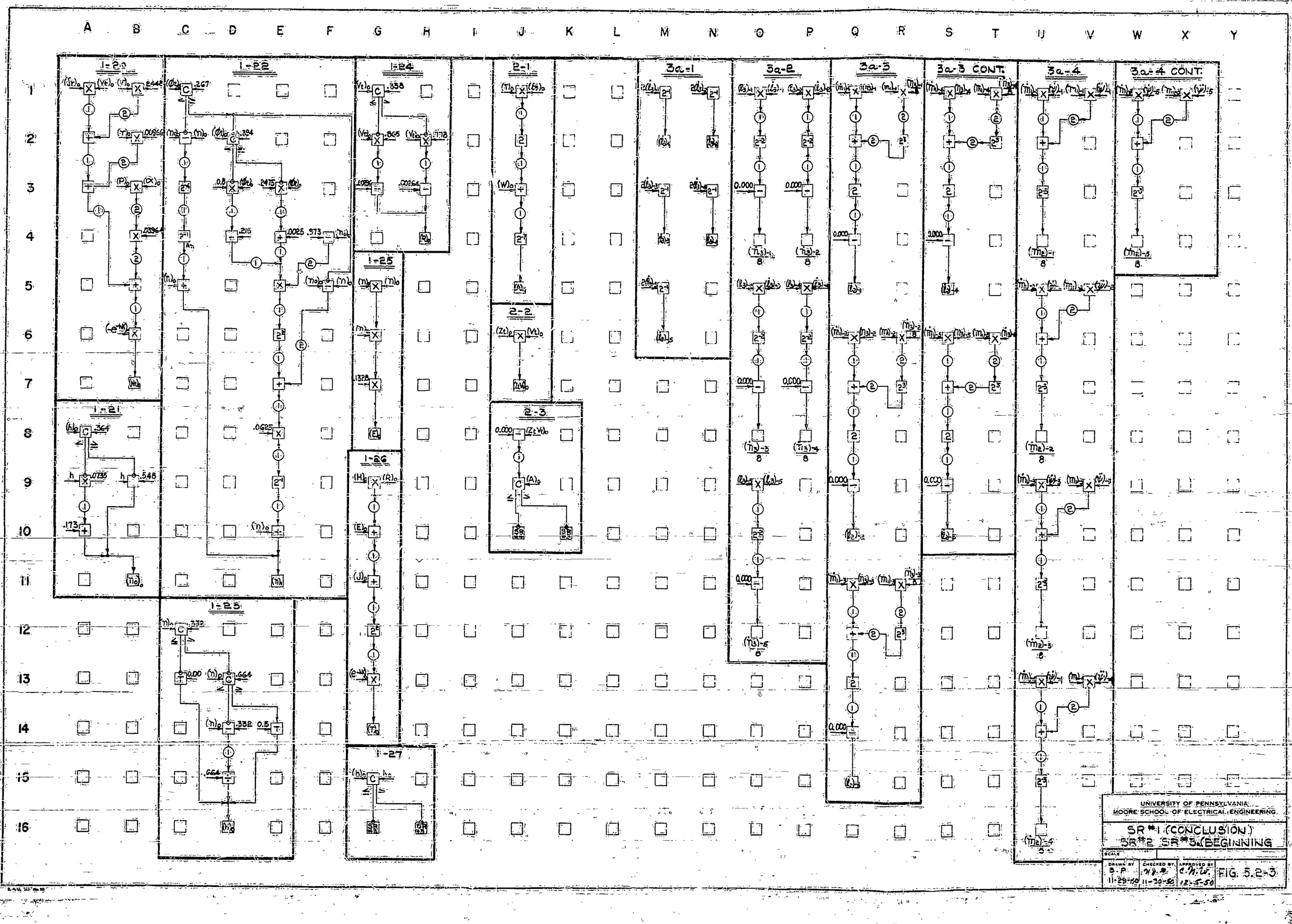
UNIVERSITY OF PENNSYLVANIA
MOORE SCHOOL OF ELECTRICAL ENGINEERING

SUB-ROUTINE #1 BEGINNING

SCALE NONE

DRAWN BY	CHECKED BY	APPROVED BY
G.B.L. 11-28-50	11-30-50	12-5-50

FIG. 5.2-2


1-20
1-22
1-24
2-1
3a-1
3a-2
3a-3
3a-3 CONT.
3a-4
3a-4 CONT.
1-21
1-25
2-2
1-26
2-3
1-23
1-27
UNIVERSITY OF PENNSYLVANIA
MOORE SCHOOL OF ELECTRICAL ENGINEERING
SR*1 (CONCLUSION)
SR*2 SR*3 (BEGINNING
SCALE
DRAWN BY S.P. 11-29-50
CHECKED BY 11-30-50
APPROVED BY 12-5-50
FIG. 5.2-3

3a-5

3a-5 (CONT.)

3a-6

3a-6 (CONT.)

3a-7

3a-7 (CONT.)

3a-8

3a-8 (CONT.)

3a-9

3a-9 (CONT.)

3a-10

3a-11

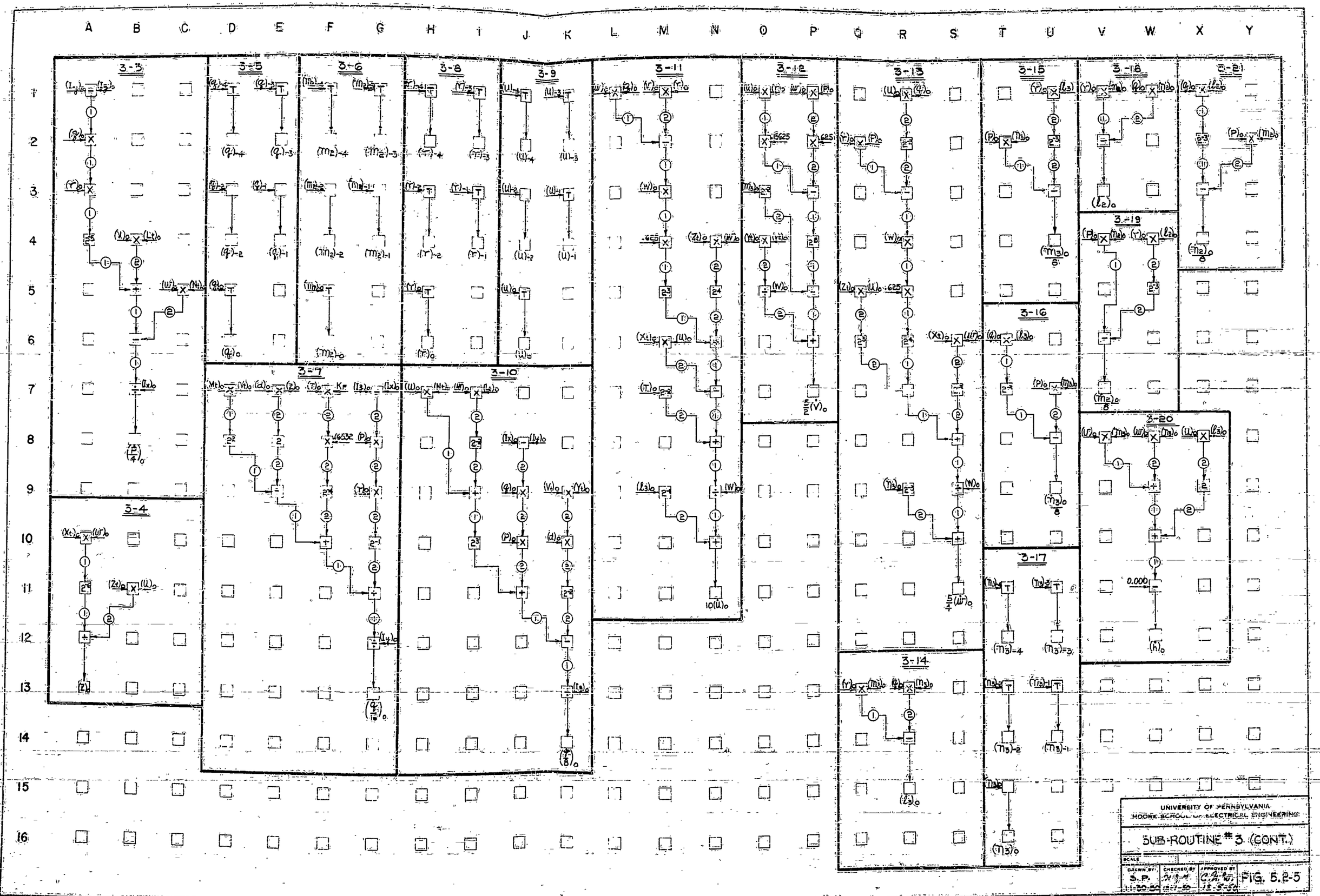
3-1

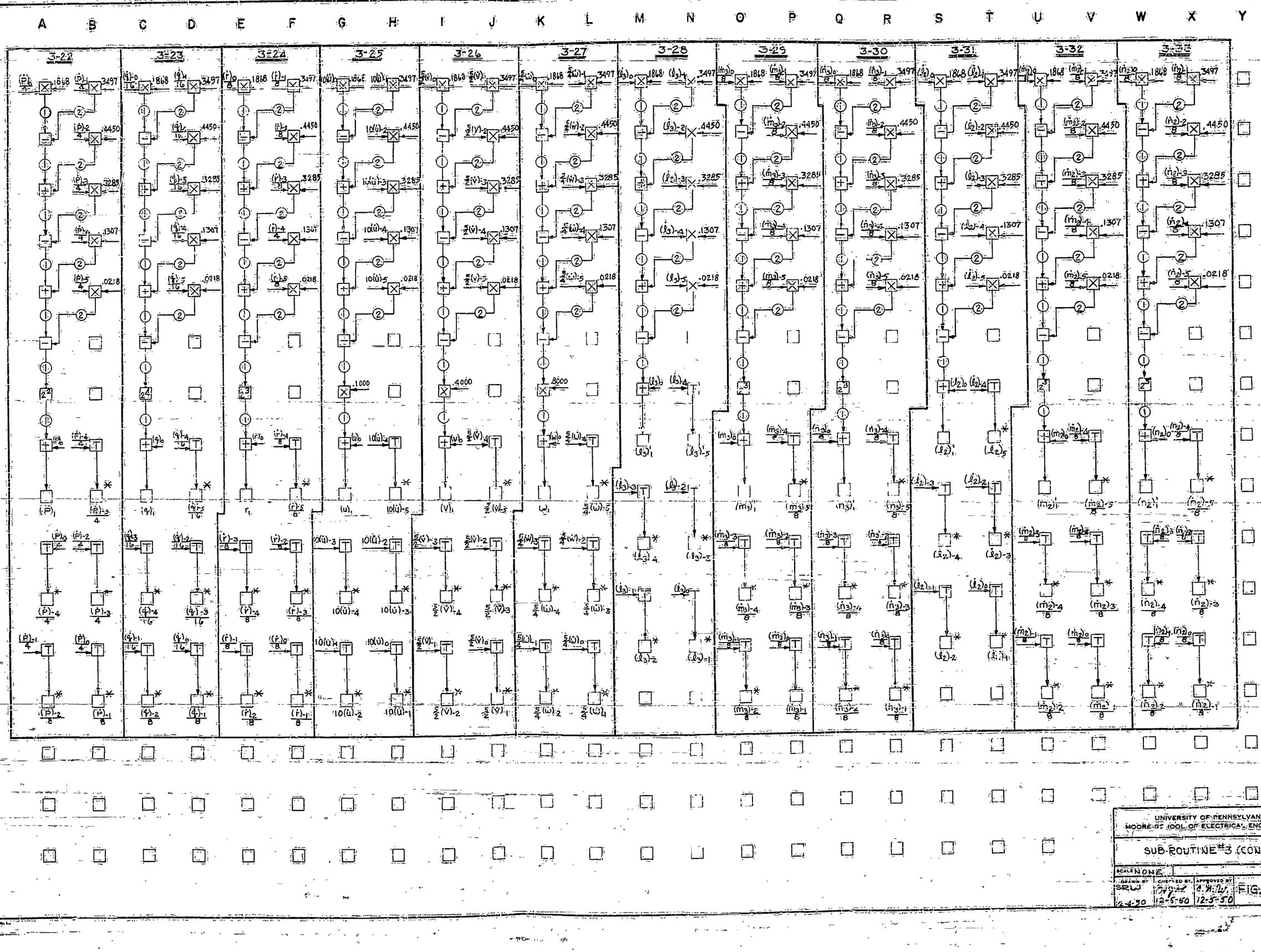
3-2

UNIVERSITY OF PENNSYLVANIA
MOORE SCHOOL OF ELECTRICAL ENGINEERING

SR 3 a (CONCLUSION)
SR 4 3 (BEGINNING)

SCALE NONE			
DRAWN BY SRW 11-25-50	CHECKED BY [illegible] 11-30-50	APPROVED BY [illegible] 12-5-50	FIG. 5.2-4


3-3
3-4
3-5
3-6
3-7
3-8
3-9
3-10
3-11
3-12
3-13
3-14
3-15
3-16
3-17
3-18
3-19
3-20
3-21
UNIVERSITY OF PENNSYLVANIA
MOORE SCHOOL OF ELECTRICAL ENGINEERING
SUB-ROUTINE # 3 (CONT.)
SCALE
DRAWN BY S. P.
CHECKED BY
APPROVED BY
FIG. 5.2-5



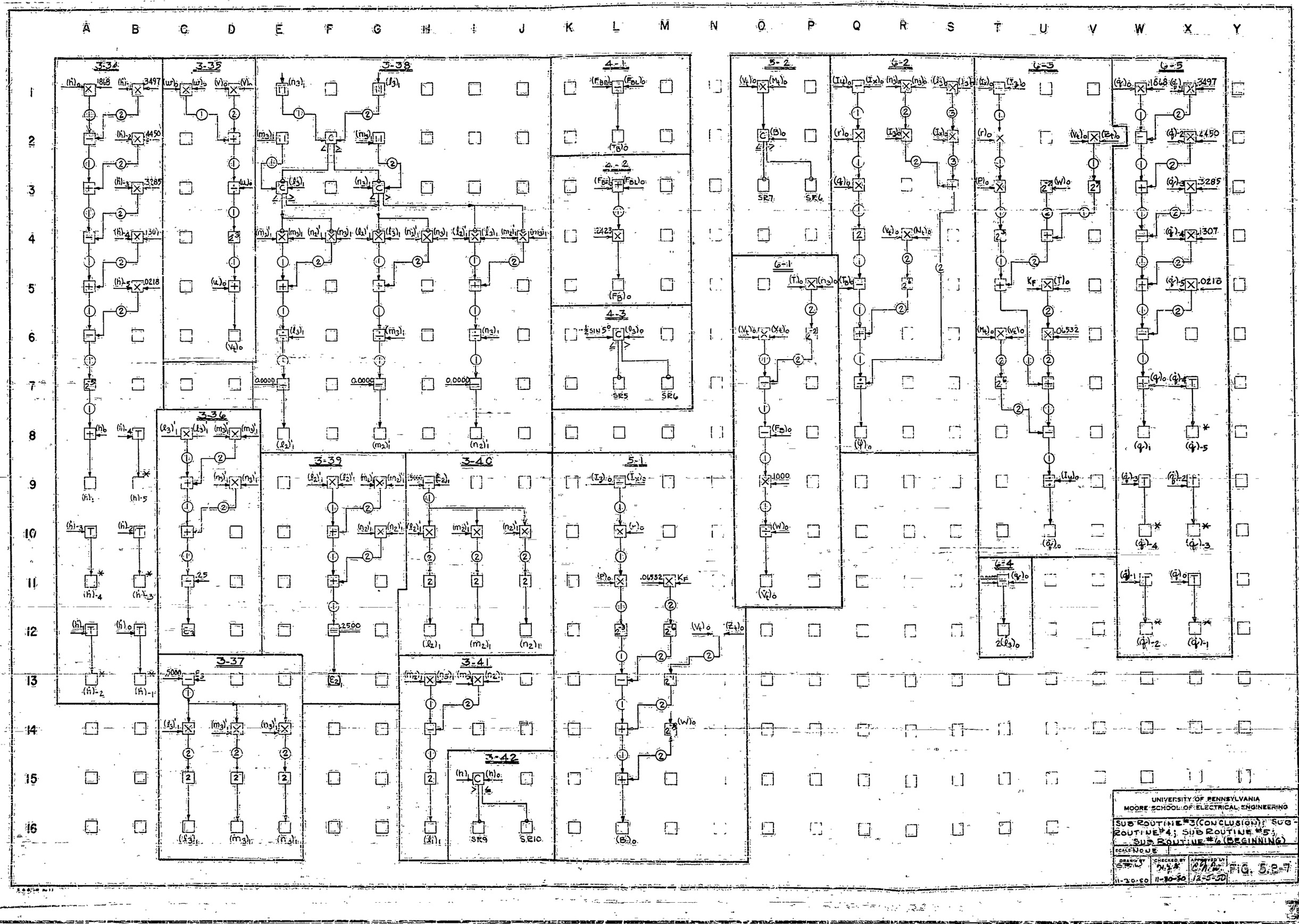
UNIVERSITY OF PENNSYLVANIA
MOORE SCHOOL OF ELECTRICAL ENGINEERING

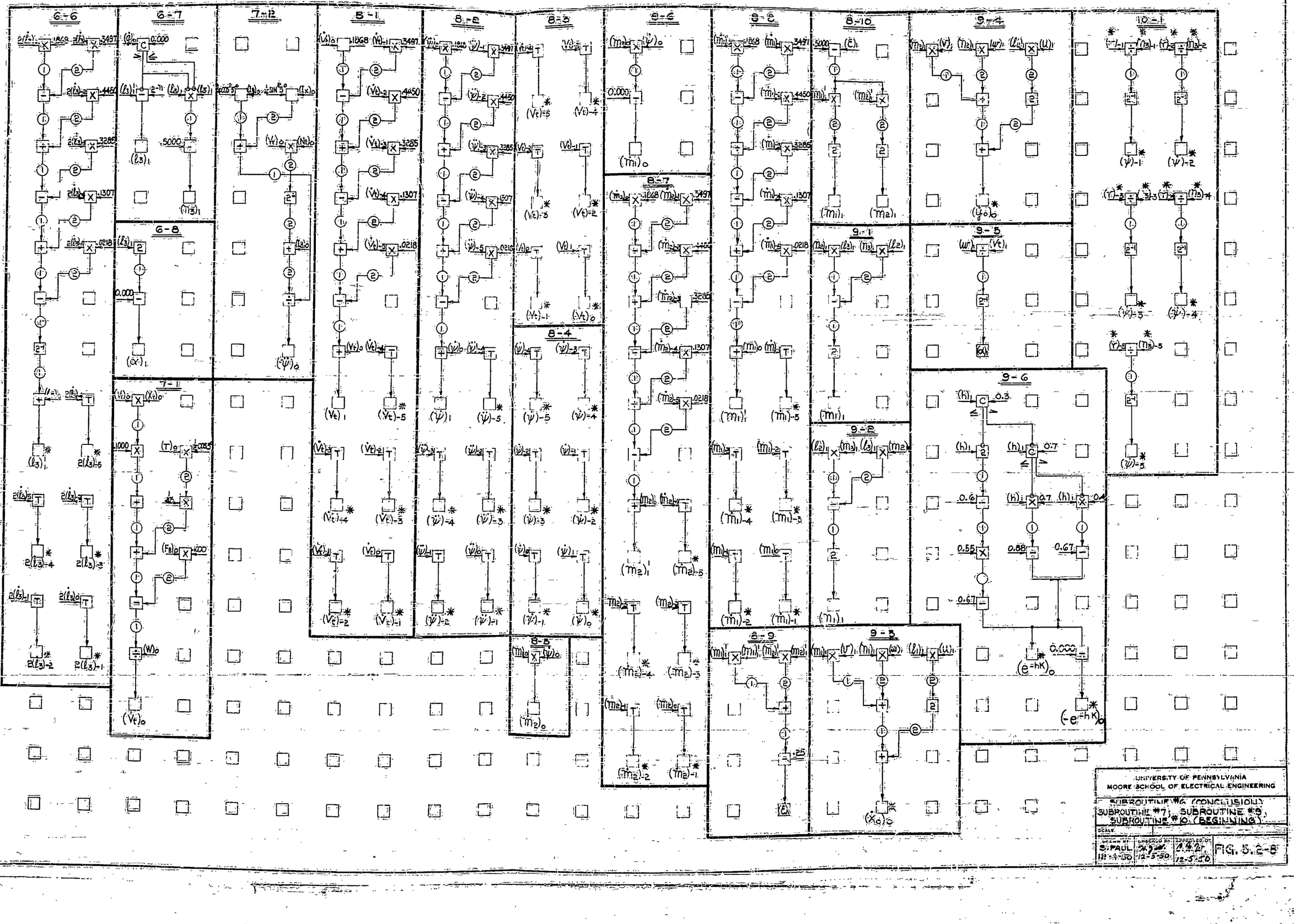
SUB-ROUTINE #3 (CONT.)

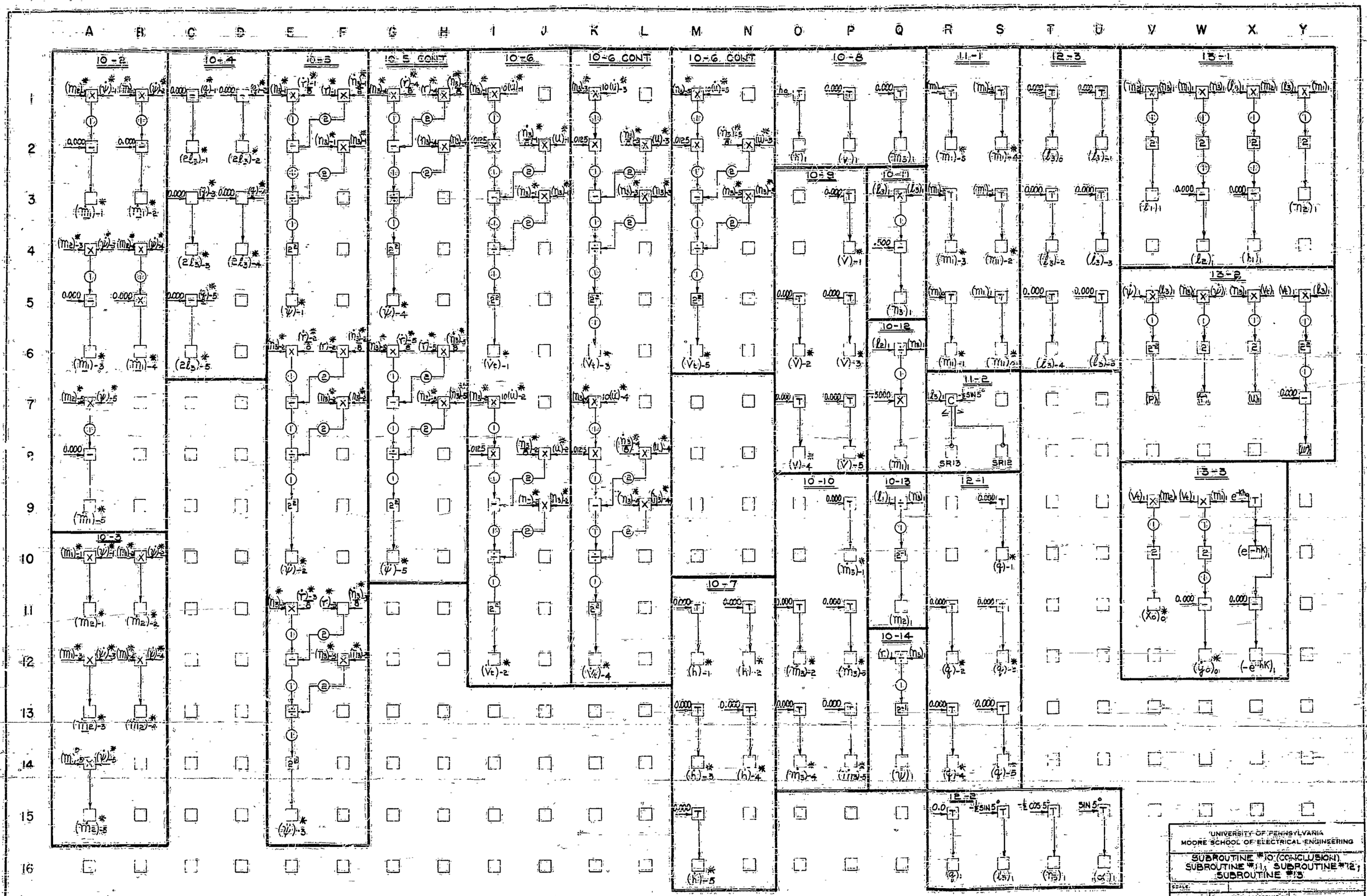
SCALE NONE

DRAWN BY	CHECKED BY	APPROVED BY
SRW 12-4-50	12-5-50	12-5-50

FIG. 5.2-6


3-34
3-35
3-38
4-1
4-2
4-3
5-2
6-2
6-3
6-5
3-36
3-37
3-39
3-40
3-41
3-42
5-1
6-1
6-4
SR7
SR4
SR5
SR6
SR8
SR9
SR10
.1868
.3497
.4450
.3285
.1307
.0218
.1423
.5000
.0000
.25
.2500
.1000
.06532
UNIVERSITY OF PENNSYLVANIA
MOORE SCHOOL OF ELECTRICAL ENGINEERING
SUB ROUTINE #3 (CONCLUSION); SUB-ROUTINE #4; SUB ROUTINE #5; SUB ROUTINE #6 (BEGINNING)
SCALE NONE
DRAWN BY C.T.C. 11-30-50
CHECKED BY 11-30-50
APPROVED BY 12-5-50
FIG. 5.2-7


6-6
6-7
7-12
8-1
8-2
8-3
8-6
8-7
8-8
8-10
9-4
10-1
6-8
7-1
8-4
8-5
9-1
9-5
9-6
9-2
9-3
UNIVERSITY OF PENNSYLVANIA
MOORE SCHOOL OF ELECTRICAL ENGINEERING
SUBROUTINE #6 (CONCLUSION)
SUBROUTINE #7; SUBROUTINE #9
SUBROUTINE #10 (BEGINNING)
SCALE:
DRAWN BY S. PAUL 12-4-50
CHECKED BY 12-5-50
APPROVED BY 12-5-50
FIG. 5.2-8


10-2
10-4
10-5
10-5 CONT
10-6
10-6 CONT
10-6 CONT
10-8
11-1
12-3
13-1
10-9
10-11
13-2
10-12
11-2
10-10
10-13
12-1
13-3
10-3
10-7
10-14
12-2
UNIVERSITY OF PENNSYLVANIA
MOORE SCHOOL OF ELECTRICAL ENGINEERING
SUBROUTINE *10 (CONCLUSION), SUBROUTINE *11, SUBROUTINE *12, SUBROUTINE *13
DRAWN BY S. PAUL 5-6-50
CHECKED BY 12-6-50
APPROVED BY 12-6-50
FIG. 5.2-9

5.3 Number of Operations

The number of operations performed in each computing cycle depends upon many things. For example, if the plane is in flight, the number of computing operations in a computing cycle will depend upon the altitude, speed, angle of attack, etc. An entirely different figure will be obtained when the plane is on the ground, and so forth. It is thought at this time that the computer will halt at the end of each computation cycle and will be started again by a signal from a central clock which occurs at a fixed repetition rate, the rate being dictated by the Δ t used in the integration formula. Thus to make sure that computations for one cycle are completed before the next start pulse, the machine must compute rapidly enough to finish the longest possible computation routine.

If reference is made to the flow chart in Section 2.7, it can be seen that there are a large number of possibilities. Consider first the following cases, 1-2-3a-3-10-11-13 and 1-2-3a-3-10-11-12-13; it is proposed to show now that they should not occur. Take-off is implied by entering Subroutine No. Two when $-Z_t V_t > A$. It follows that $\dot{h}$ computed in Subroutine No. Three should be positive. However it can be shown that $(\dot{h})_o = 0$. From Subroutine Number Three

$$(\dot{h})_o = - [2u\, l_3 + (vm_3 + wn_3)]_o \qquad (5.3 - 1)$$

where

$$(v)_o = 0$$

$$(u)_o = 2(V_t)_o\, (n_3)_o \qquad (5.3 - 2)$$

$$(w)_o = -4(V_t)_o\, (l_3)_o$$

Equations (5.3 - 2) come from subroutines Numbers 10 and 13. Substitution of (5.3 - 2) in (5.3 - 1) gives:

$$(\dot{h})_o = -\left[4(V_t)_o(n_3)_o(l_3)_o - 4(V_t)_o(n_3)_o(l_3)_o\right] = 0 \quad (5.3 - 3)$$

Therefore $h = h_o$ at the end of subroutine No. Three directing the flow of computation to subroutine 10, where the plane lands, making it impossible for the plane to engage in sustained flight. To correct this difficulty using the present scheme of integration and programming, it is necessary to make $h > h_o$ in the event of take-off. This is done in computation block 3a-10 by increasing h by an arbitrary small number. In consequence the afore-mentioned computation sequences are not possible.

A similar situation arises in subroutine No. Six in connection with the computation of l_3. In order to make it possible for the plane to go from three wheels to two wheels, l_3 is decreased by the smallest number possible in computation 6-7.

A count of the maximum estimated operations required for each of the remaining possible computational sequences is shown in Figure 5.3 - 1 on the following page.

Fiugre 5.3 - 1

Operation Sequence	+	-	X	÷	C	T	Shift	Extract Magnitude
	Maximum Number of Operations							
1-2-3a-3-9	147	109	312	20	23	101	109	4
1-3a-3-9	146	108	310	20	22	101	107	4
1-3a-3-10-11-13	141	130	356	37	22	126	128	4
1-3a-3-10-11-12-13	141	130	356	37	22	135	128	4
1-2-4-6-8-11-13	67	68	139	4	22	50	32	1
1-2-4-6-8-11-12-13	67	68	139	4	22	59	32	1
1-2-4-5-6-8-11-13	69	70	144	4	23	50	36	1
1-2-4-5-6-8-11-12-13	69	70	144	4	23	59	36	1
1-2-4-5-7-8-11-13	62	55	123	3	22	40	28	1
1-2-4-5-7-8-11-12-13	62	55	123	3	22	49	28	1

5.4 Total Memory

The number of memory positions necessary to store constants used in the computations is 110.

The number of memory positions necessary to store insturctor and pilot inputs is 18.

The total number of instructions is 232. In the Hurricane machine, the number of memory positions required to hold the instructions is about twice the number of instructions, 2464. For the hypothetical machine described in Section 3.9 the number of memory positions required to hold the instructions is 4 times the number of instructions, 4928.

The number of memory positions necessary to store intermediate values in a blocked computation is 3.

The number of additional memory positions needed to store intermediate values in the subroutines, and to store the variables is not exactly known, but a good upper bound is 365, and a good lower bound is 173. Thus the total memory required in a Hurricane type machine lies somewhere between 2768 and 2960, and for the hypothetical machine of Section 3.2 lies somewhere between 5232 and 5424.

The above comments are summarized in Table 5.4 - 1 on the following page.

Table 5.4 - 1

Memory Positions

	Hurricane Computer	Hypothetical Machine
Constants	110	110
Inputs	18	18
Outputs	--	--
Instructions	2964	4928
Variables	173 - 365	173 - 365
Working Space	3	3
Total	2768 - 2960	5232 - 5424

5.5 Estimated Machine Time

The time for one computation cycle using the Hurricane machine is 0.22 seconds.

The time for one computation cycle using the hypothetical digital machine described in Para. 3.2 is 0.10 seconds.

The computation times presented here were arrived at by using the longest possible routine: 1-3a-3-10-11-12-13 together with the operation times listed in table 3.9 - 1. Thus these times are the most probable times for computation in the longest computation cycle. Note, however, that no advantage has been taken of the saving in extended operations, such as (A+B+C), which is faster than (A+B) followed by (A+B) + C.

5.6 Minor Cycles

In the computation of the various subroutines, many quantities may not change appreciably through one computation cycle. Thus the possibility suggests itself of decreasing the total computation times of Para. 5.5 by only occasionally computing some of these slowly varying quantities.

For example, if W need be computed only every other computation cycle, something else such as $\bar{d}$ could be computed in alternate cycles. If the quantities such as $\bar{I}_x$, I_z, W, $\bar{d}$ need be computed more than say every ten computation cycles, then I_x might be computed in the first cycle, I_z in the next, W in the next, and $\bar{d}$ in the next; the possibilities are very great. However, it is not known at this time which quantities, if any, permit of this treatment; consequently, the attempt to include minor cycling in this interim program has not been made.

SECTION 6

CODING OF INPUT

AND

DECODING OF OUTPUT

6 CODING OF INPUT AND DECODING OF OUTPUT

The analogue information provided by the pilot and instructor for use in the computer, must be converted to digital information and the digital information computed must be converted to continuous information for instrument readings and control loading. The contract requires that the duty cycle of this equipment be specified but does not ask for any design data for coding and decoding devices.

6.1 Coding of Input Information

The 14 quantities listed on Page 2-28 of Section 2 must be converted from analogue to digital form. Quantities are of three different kinds.

a. Arbitrarily varying quantities controlled by pilot. (δ_a, δ_e, δ_r) Throttle angle φ_t is also in this category.

b. Quantities which are either zero or change at a constant rate with time until they reach a limiting value. (δ_{fd}, K_G, K_F, K_N)

c. Quantities that are either zero or some constant value. (S_i, S_{Fb}, S_{Fb}'', S_{FW}, S_{FW}', S_{FW}'', S_a) In addition to these, failures to be simulated which affect the equations of motion could be handled in the same way as the S's.

6.1-1 The Arbitrarily Varying Quantities

In order to get the best results in digital computation it is necessary to use values of δ_a, δ_e, and δ_r that are sampled at as close to the same time as possible. (See Section 4) The values of these quantities are available in terms of the mechanical angular position of the stick and rudder bar. It therefore seems best to provide a converter which will convert the mechanical angular position directly into digital

information, perhaps in the form of open or closed switches. Not more than ten binary digits would be necessary. There must be one mechanical converter for each quantity, but these could all use a single electronic device to produce a train of pulses identical with that coming from any memory position. By providing gates, one for each mechanical converter, all inputs could then be assigned addresses just as in the case of memory positions, and treated in the programming just like any normal memory position. A block diagram of this scheme is shown in Figure 6.1-1.

6.1-2 Quantities Varying Linearly with time.

The best way to handle these is to provide a constant speed motor and attach to its shaft one of the mechanical converters described in P.6.1-1. A limit switch would stop the motor when proper final value is reached. This enables linearly varying quantities to be handled in the same way as arbitrarily varying ones.

6.1-3 Quantities That Are Either Zero or Constant.

These can be handled in much the same way except that instead of a shaft to binary converter it is only necessary to have a switch that is either open or closed, corresponding to the binary numbers 0 and 1. The programming when called to produce the number say S_{FW} which is either 0 or 0.004448 (decimal number) will direct the machine to the memory position of the S_{FW} switch and perform a choice. If it finds 0 it will be directed to a memory position where it finds 0, if closed it will be directed to a memory position where it finds the binary equivalent of 0.004448.

Another way of handling the S quantities is used in the program

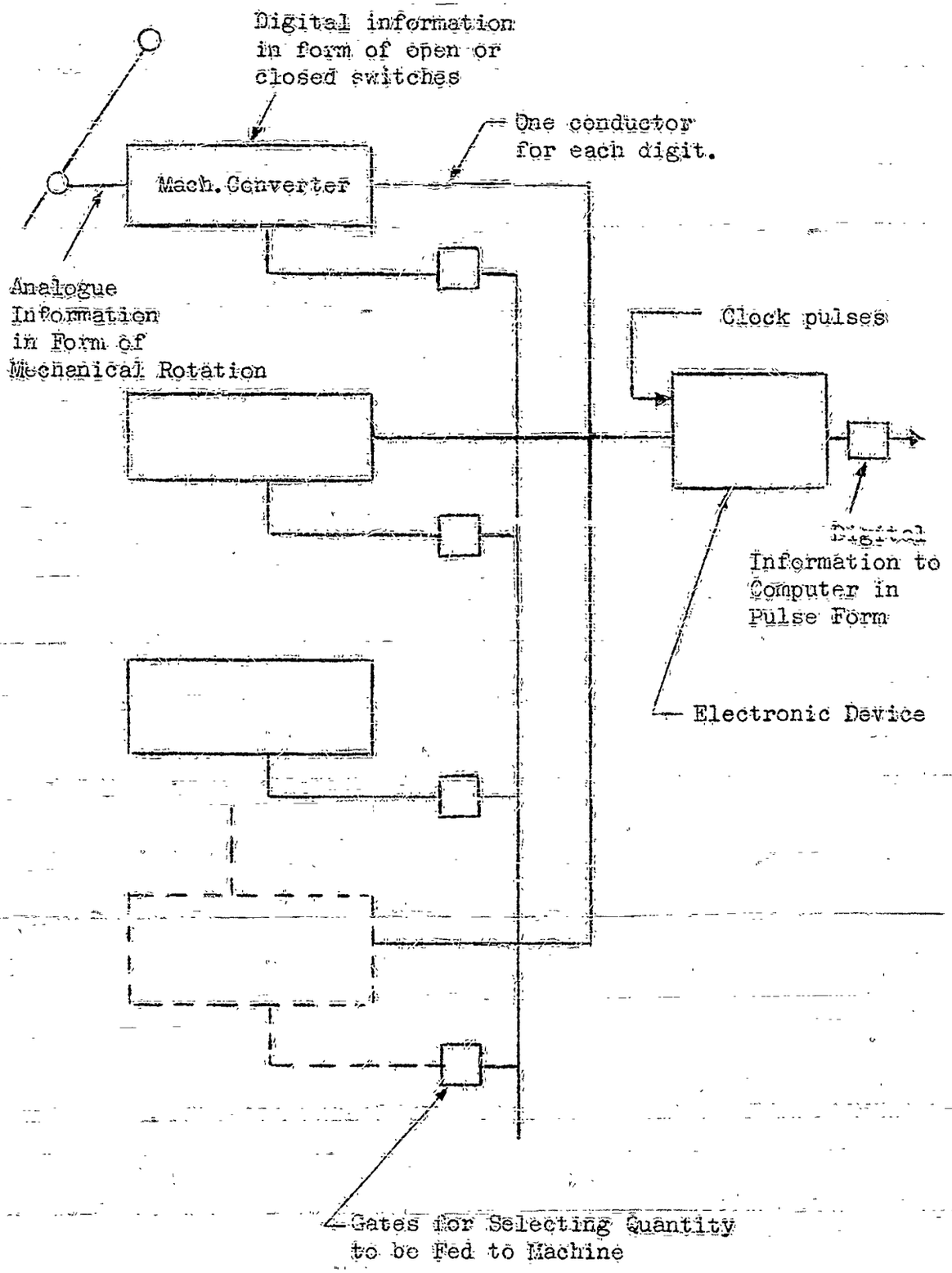

Digital information in form of open or closed switches
One conductor for each digit.
Mech. Converter
Analogue Information in Form of Mechanical Rotation
Clock pulses
Digital Information to Computer in Pulse Form
Electronic Device
Gates for Selecting Quantity to be Fed to Machine

Figure 6.1-1

described in Section 5 of this report. This method is advantageous from the standpoint of saving computing time. All of the S's (page Section 2) are rates which are integrated with respect to time. High accuracy is not warranted in these integrations so they can be performed by constant speed motors (speed proportional to values of S) whose revolutions are counted by mechanical converters of the type described in 5.6.1-1. This system also has the advantage of using an identical method of introducing all inputs.

The mechanical converters should be so designed that there would at all times be available a binary number corresponding to the input shaft position. This might necessitate two registers slightly out of time phase, so that while one register is in the process of transition the other is not, and means of inhibiting entry into the register which is changing.

The associated equipment for converting the register setting into trains of pulses must be able to scan the array of input registers in sequence in the same time that it takes to scan a similar number of memory positions in the computer.

6.2 Decoding of Output Information

6.2.1 Typical Instruments are the following:

Needle - reads	v
Dial - reads approx.	$\dfrac{m_s - V_T r}{\frac{n_s}{2b} - V_T q}$
Attitude gyro - reads	$\sin^{-1} l_s$
Altitude gyro - reads	$\sin^{-1} m_s$

Compass - reads ψ

Rate of climb - reads $\dot{h}$

Altimeter - reads h

Air Speed Indicator - reads $\sqrt{p_{o}\varepsilon^{-kh}}\,u$

The computer, if it has not already computed them for other purposes must compute numbers proportional to all instrument readings. Computations peculiar to instruments such as Ball, Attitude gyro and Airspeed Indicator have not as yet been included in the program in Section 5.

These numbers must then be converted to some analogue quantity such as a voltage suitable for actuating an instrument. Since in general instruments will make small changes in their readings in a time as short as one computation cycle, a single digital to voltage converter could be time shared by all instruments.

6.2-2 Control Loading.

Let it be assumed that the control loading is to be produced by servomechanisms that can develop stick and rudder forces proportional to voltages applied to their inputs. Then the computer must compute the stick and rudder forces and deliver to its memory numbers proportional to these forces. Then these numbers can be converted to voltages by the same mechanism used for converting instrument readings for voltages.

6.2-3 Digital to Voltage Converter

By using the scheme outlined in sections 6.2-1 and 6.2-2 the requirements of the digital to voltage converter can be stated as follows:

The converter must accept digital numbers at a rate equal to:

$$\frac{\text{Total number of instrument readings and control forces}}{\text{Time of one computation interval}}$$

and produce one continuously varying voltage for each instrument and component of control force. If this rate is excessively large it could be reduced by sampling the slowly varying instruments less frequently than every cycle.

23/5 STI-ATI-209 021 23700C UNCLASSIFIED P 1/4

30/2 Moore School of Electrical Engineering, Pennsylvania U., Philadelphia.

FLIGHT TRAINER DIGITAL COMPUTER STUDY, by C. N. Weygandt, H. Sohon, and others. Final rept., Research div. rept. no. 51-28. 21 Mar 51, 1v. incl. illus. tables. (Contract N6onr 24913)

DIV: Personnel & Training (23)
SECT: Military Training & Indoctrination (5)
DIV: Research & Research Equipment (30)
SECT: Computers (2)
DIST: Copies obtainable from ASTIA-DSC

SUBJECT HEADINGS
Computers, Digital
Trainers, Flight

CFSTI per ONR ltr, 27 Apr 66

UNCLASSIFIED

Proj. no. 24-F-1

x Flight Control Computers

23/5 STI-ATI-209 021 UNCLASSIFIED

30/2 Moore School of Electrical Engineering, Pennsylvania U., Philadelphia.

FLIGHT TRAINER DIGITAL COMPUTER STUDY, by C. N. Weygandt, H. Sohon, and others. Final rept., Research div. rept. no. 51-28, 21 Mar 51, 1v. incl. illus. tables. (Contract N6onr 24913)

DIV: Personnel & Training (23)
SECT: Military Training & Indoctrination (5)
DIV: Research & Research Equipment (30)
SECT: Computers (2)

SUBJECT HEADINGS
Computers, Digital
Trainers, Flight

DIST: Copies obtainable from ASTIA-DSC

UNCLASSIFIED

Proj. no. 24-F-1