DIGITAL COMPUTERS FOR REAL-TIME SIMULATION*

By MORRIS RUBINOFF
University of Pennsylvania, Philadelphia, Pa.

Since digital computers can solve certain systems of equations faster than analog computers, the question arises whether a digital computer can be used for real-time simulation. The advantages of a digital simulator are its flexibility in transferring from one simulation to a second, its universal application without mechanical alteration, its greater precision and accuracy, and its ability to simulate high frequency systems. The feasibility of digital simulation depends upon the speed permitted by numerical computational techniques, the existence of an ultra-high-speed digital computer, and very fast encoding and decoding equipment. The paper discusses these problems and describes their solution, with particular consideration to the simulation of airplane flight in connection with a digital operational flight trainer.

Introduction. The recent advent of large-scale digital computing machines has led to the study of the role of digital devices in real-time simulation and industrial process control. Several advantages appear to be attainable. Digital computers are capable of indefinite precision and hence are at least potentially capable of greater accuracy than analog devices. More important, digital computers are universal machines, lending themselves to mass production; the end use of a digital computer can generally be decided at any time before or after its construction. This is because digital computers are inherently flexible; in order to switch over from the simulation of one device or process to a second, it is only necessary to read new numbers and instructions into the computer memory either from punched cards or magnetic tape or other input device. Finally, a digital computer can be used for the simultaneous actuation of several devices. These relative advantages of digital computers have received detailed consideration previously [1] and they will not be pursued any further in the present paper.

The expression "real-time simulation" is interpreted in a number of ways by a variety of groups interested in somewhat different problems. Throughout this paper, real-time simulation refers to simulation of the performance of a process or device at normal operating speed, with the responsibility resting on the digital simulator to keep pace with the simulated process or device. For example, in an Operational Flight Trainer,

*Presented at the meeting of the Association, June 23–25, 1954. This work was done as part of a project sponsored by Special Devices Center, O.N.R., Port Washington, L.I., N.Y.

it is required that the digital simulator sense the pilot's actions, solve the flight equations, and actuate the cockpit instruments sufficiently fast that the pilot cannot distinguish between simulator response and true airplane response.

A block diagram of such a digital Trainer is shown in FIGURE 1. The heart of the device is the real-time digital simulator consisting of a universal digital computer and auxiliary equipment common to all Trainers. Conversion from one airplane type to a second requires only (1) momentary transfer of the aerodynamic coefficients, engine constants, etc., from the appropriate tape in the tape library to the digital computer memory unit, and (2) connection of the simulator to the desired cockpit through a conventional multiwire cable connector. The same real-time digital simulator, without alteration, may be used to simulate any device or control any process as long as the associated computations do not tax the speed of the digital computer.

The feasibility of real-time simulation using digital computers rests on three requirements. The first of these is ultrahigh speed of computation. Digital computers are (currently) limited by size and cost considerations to a single arithmetic unit capable of performing only one primitive arithmetic operation at a time. All computations must, therefore, be performed sequentially. Thus, even though a computer be megacycle

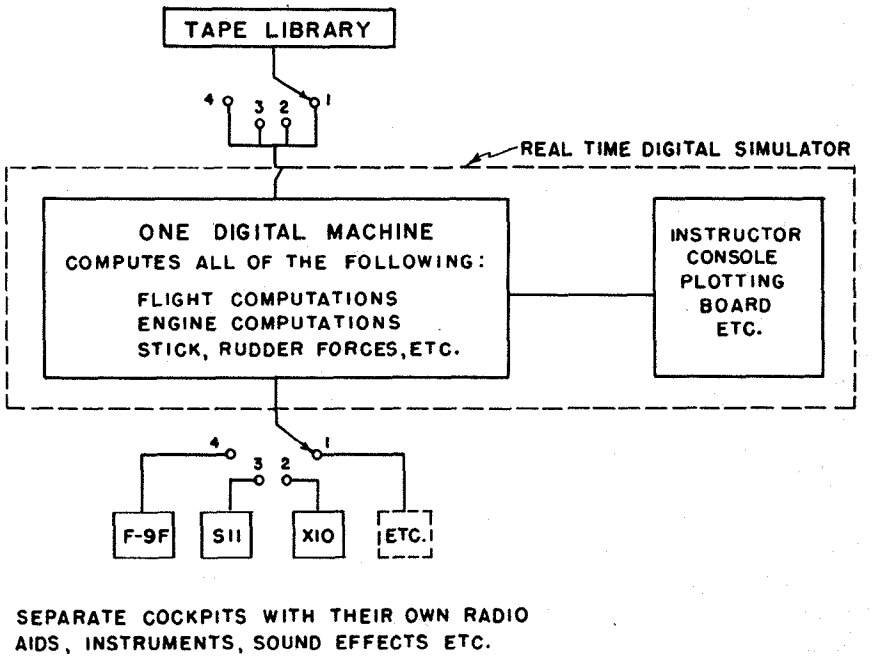


FIGURE 1. DIGITAL TRAINER BLOCK DIAGRAM

fast at addition and multiplication, the effect of time-sharing the computer is to reduce its effective speed by several orders of magnitude. Intrinsically high speed is therefore a fundamental prerequisite.

The second requirement for real-time simulation is high-speed encoding and decoding equipment. In a digital OFT, about 20 shaft positions must be sensed and 30 instruments actuated at a rate of 20 or more samplings of each per second. This imposes a conversion time of 1 millisecond or better on the encoding and decoding equipment.

The third prerequisite for real-time simulation is an assurance of mathematical stability and "reasonable" accuracy. A digital simulator may be considered stable when (1) the simulator solution does not grow indefinitely for conditions under which the true solution remains bounded, and (2) the ratio of simulator solution to true solution remains close to unity whenever the true solution grows indefinitely. Accuracy is reasonable when the deviation of the simulator solution is small enough to be considered inconsequential; for example, in an OFT, accuracy is reasonable when an experienced pilot is incapable of distinguishing the simulator solution from the corresponding "true flight".

This paper outlines a study conducted at the Moore School of Electrical Engineering which succeeded in meeting these requirements. An ultrahigh-speed computer has been designed, capable of 5 microsecond addition using conventional diode OR-AND-OR switching circuits (although at reduced power)[2]; the improvement in speed is primarily the result of novel logical interconnection. Conventional Gray-coded commutator discs permit 5 microsecond encoding. A novel diode multiplexer[3] yields 100 microsecond total decoding time per instrument, with only 5 microseconds of computer time required for each decoding.

The stability-accuracy prerequisite was the most difficult to assure. The forces acting on a simulated system give rise to (ordinary) differential equations with time as independent variable. The numerical solution of these differential equations requires the use of quadrature formulas in a step-by-step process, where the steps must be kept small to insure accuracy and stability. But in real-time simulation, the quadrature step is a real time-interval; it must therefore be long enough to permit computation of all the arithmetic entering into the quadrature step.

A method for displaying the *maximum permissible quadrature step* for any specified quadrature formula was developed. The method employs "stability charts" analogous to the Nyquist plot, and requires only that the natural (complex) frequencies of the simulated device be known. It is particularly noteworthy that the solutions to be encountered in any specific simulation need not be known. The basis for the stability chart method is described elsewhere[4] in greater detail than is given below.

For the particular Operational Flight Trainer under study, the stability charts for several quadrature formulas permitted a quadrature step of

50 milliseconds. The longest possible routine needed to evaluate all the computations for one quadrature step required 11 milliseconds using the novel computer. Thus, available computer speed is four times greater than required.

The sequential digital computer. A fundamental requirement for the digital computer actuating a real-time simulator is extreme reliability. In an Operational Flight Trainer, for example, a computer error in the midst of a training flight can interrupt continuity and consequently destroy the illusion of true flight. In order to attain the utmost assurance of reliability, novel switching circuits were ruled out. Ultrahigh speed was to be achieved only through logical rearrangement of circuits which have proved reliable in existing digital computers.

The first decision concerned the number of addresses per instruction. The flight equations for a particular airplane were programmed as a basis for evaluation, and the result of the study showed one-address code preferable to two-, three-, or four-address codes as regards both speed and cost. The one-address code was therefore adopted.

The second decision was to use separate memories for numbers and instructions. Independent memories gain a factor of two in speed, since they permit access to an instruction $I(k)$ simultaneously with access to the number called for by the preceding instruction $I(k\text{-}1)$. Thus, the arithmetic unit operates at 100% duty factor to yield its maximum inherent speed. It is noteworthy that the increased cost of separate memories is offset by savings in the control unit, which no longer needs to distinguish instructions from numbers.

A five-fold increase in speed of multiplication was achieved through adoption of a whipple-tree multiplier and a serial computer.[5] Although this demanded about 30% additional equipment, the time saving was essential for real-time simulation. The whipple-tree multiplier made possible an additional time saving. By employing the divisor register to store a product temporarily, under command of a special "multiply-add" instruction, the output of the whipple-tree multiplier could be added and/or stored within the arithmetic unit where it is immediately accessible. This is especially attractive in computations of the form $ab+cd+ef+gh$, which occur frequently in the solution of differential equations.

Until this point in the study, it had appeared that the optimum simulator required a serial computer and a serial memory unit. The fastest switching circuit for the purpose was the diode pyramid, perhaps with reduced power.[2] Reliability considerations limited the pulse repetition rate to about 1 megacycle (actually 1.2 megacycles at reduced power) with 5 clock phases per pulse time. It is important to observe that the time interval between successive clock phases is designed to be slightly longer that the maximum possible transmission delay through the "worst" diode pyramid, tube, and transformer. Thus, if a switching circuit is

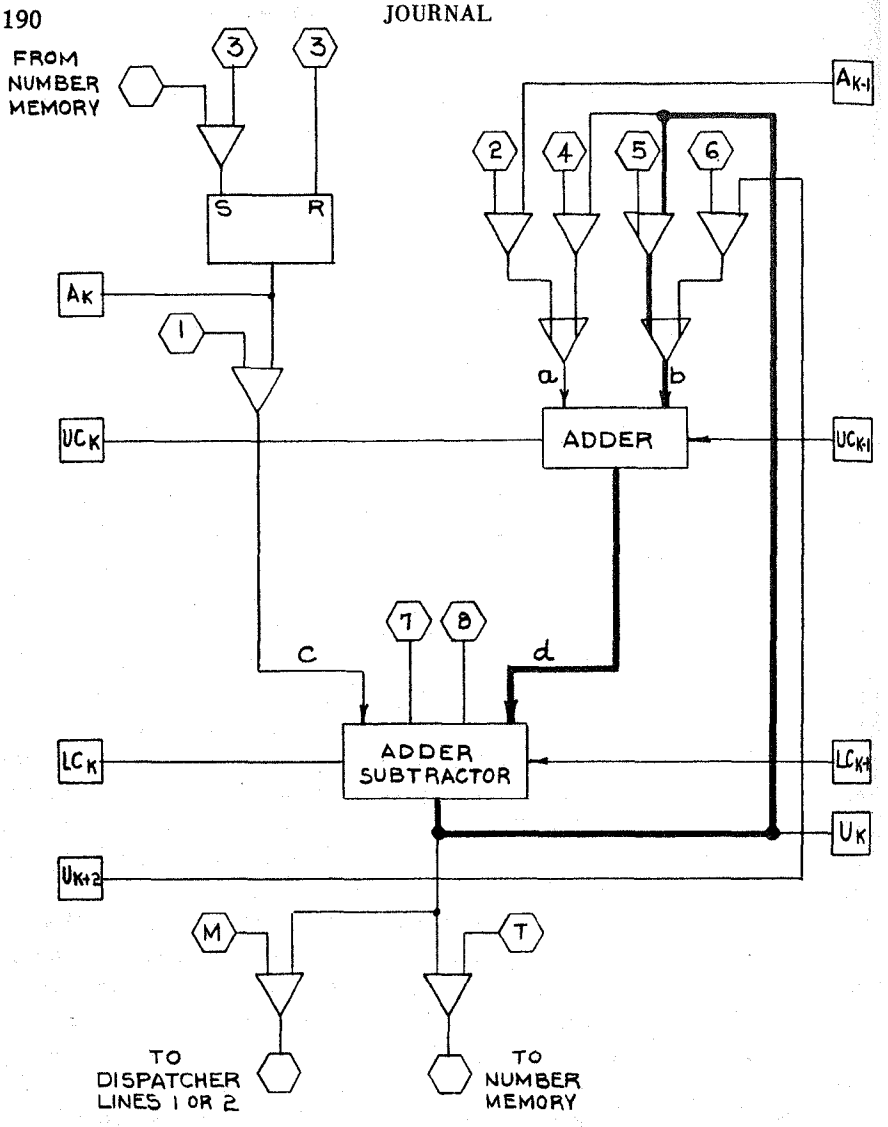


FIGURE 2. K-TH STAGE OF PARALLEL ARITHMETIC UNIT

timed at its input by a clock pulse at time T, its output pulse is certain to be available one clock phase (one-fifth of a pulse time) later.

It was decided to investigate the possibilities of a "parallel" computer using the synchronized serial transmission described in the preceding paragraph. The result was a "sequential" computer, which combined the better features of serial and parallel techniques to gain an additional factor of five in speed. Since the sequential principle is apparently novel, a brief description will be attempted here [11].

Numbers are transmitted to the accumulator on many wires, one wire per columnar position, as in a parallel computer; but the digits in suc-

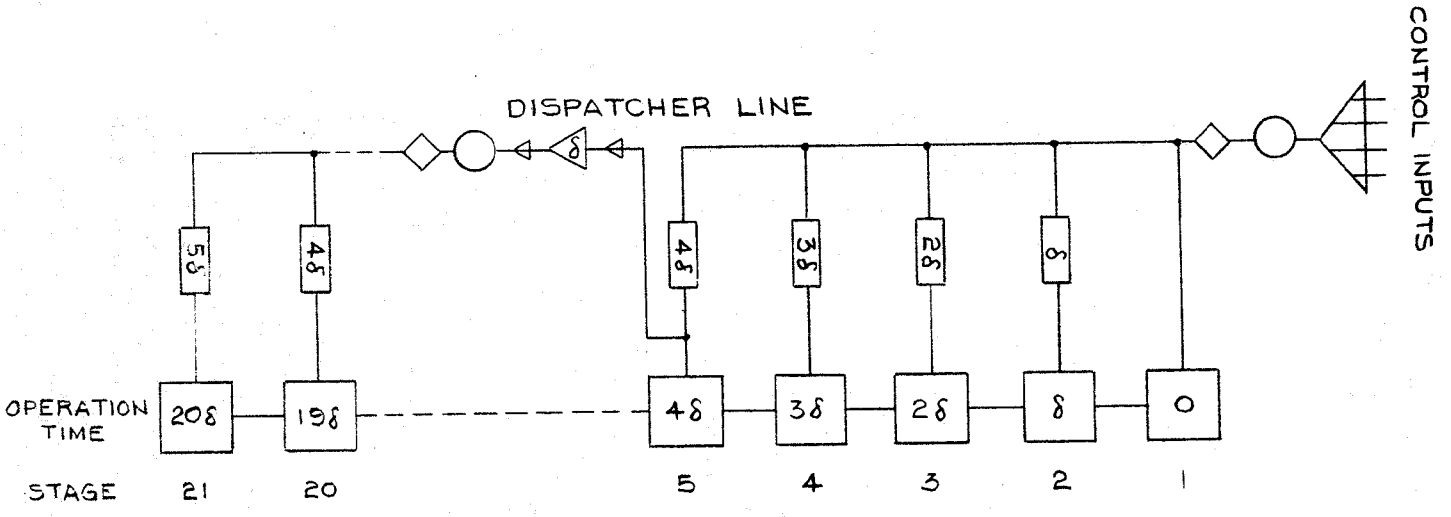
cessive columns appear in time sequence one clock *phase* apart, at just the right time to permit immediate columnar carry. Again, the serial synchronous transmission on each wire permits additions to follow each other at the 1.2 megacycle rate; successive "ripple" carries from right to left are clocked and kept distinct at all times. Thus, in multiplication the partial products can be added in at the *pulse* rate of 1.2 mc. In the O.F.T. computer design, the accumulator loop is provided with two adders so that two partial products are accumulated at a time.

FIGURE 2 shows the *k-th* column of the arithmetic unit. The circulating storage of the digit in the accumulator is outlined in heavy lines and is seen to include an adder and an adder-subtractor. In addition or subtraction, the digit in column k arrives from the number memory and, if a one, sets the augend flip-flop; since the dynamic flip-flop is "set dominant", the reset control pulse 3 is ignored. The augend digit is transmitted by control pulse 1 to point c, where it is added or subtracted under control of pulse 7 or 8. If carry is generated, it is transmitted to column $(k+1)$ through output LC_k at just the right phase to be accepted in that column; in the same way, the carry pulse is received from the column $(k\text{-}1)$ through input LC_{k-1} just as the pulses reach c and *d*.

In multiplication, the multiplier *MP* is read out of the accumulator into dispatcher lines 1 or 2 under control of pulse *M*; the least significant *MP* digit goes to line 1, the next to line 2, the third to line 1, etc. The multiplicand *MC* is stored by a single pulse at 3; pulse 3 is then turned off. If the least significant *MP* digit is unity, it appears as pulse 1 and transmits the *MC* digit to point *c* to start accumulation as in addition; at the same time, if the second *MP* digit is unity, it appears as pulse 2 and transmits the *k-th MC* digit through output A_k in column $(k+1)$ into the adder in column $(k+1)$. Similarly, just one clock phase earlier, pulse 2 has allowed *MC* digit $(k\text{-}1)$ to enter column k via input A_{k-1} and reach the adder at point *a*; carry propagation for this adder is through UC_{k-1} and UC_k. As a result, two partial products accumulate during the first pulse time. Two-columnar shift is accomplished via output U_k to an input in column $(k\text{-}2)$ to prepare for *MP* digits 3 and 4. The corresponding input in column k is U_{k+2} from column $(k+2)$; the pulse enters the accumulator through control pulse 6, pulse 5 being absent.

The accumulation of the product proceeds, two *MP* digits at a time, until multiplication is complete. All *MP* units in odd (even) columnar positions generate control pulses 1 (2) spaced one clock *pulse* apart; accumulation proceeds at a 1.2 megacycle rate in each column. Since ten pairs of additions are required for 20-binary digit numbers, multiplication takes ten clock-pulse times.

FIGURE 3 shows a typical dispatcher line for transmitting the control pulses 1, 2, 3, . . ., 9, down the accumulator at clock *phase* inter-



TIMING BETWEEN STAGES OF PARALLEL ARITHMETIC UNIT, AND CONTROL DISPATCHER LINE

FIGURE 3. TYPICAL DISPATCHER LINE

vals. Delay lines are used for 5 phases (one pulse time), after which the pulse is amplified and reshaped. Nine dispatcher lines are of course required.

Computing speeds for the sequential machine are as follows:

Addition or Subtraction	5	microseconds
Multiplication	10	"
Multiplication and Addition	10	"
Division	105	"

For example, the evaluation of the sixth degree polynomial $a\,x^6 + b\,x^5 + c\,x^4 + d\,x^3 + e\,x^2 + f\,x + g$ takes 65 microseconds.

A sequential computer is about 70 times as fast as UNIVAC even though it uses less than half the pulse repetition rate. Nevertheless, only 1750 tubes and 25000 diodes are required for the arithmetic unit, control, both memories, and all the encoding and decoding equipment for an O.F.T.

Conversion equipment. A considerable amount of attention has been given to both encoding and decoding devices.[6,7,8] These devices vary considerably in both speed and complexity. It was felt that for real-time simulation, reliability dictated that the simplest satisfactory devices be adopted.

Since only shaft positions needed to be encoded for the Trainer under study, it was decided to use mechanical commutator discs, coded in cyclic (reflected binary) to eliminate ambiguities. Ten bit precision was considered adequate. Data can be read into the computer at machine speed, 5 microseconds per number.

The decoder selected for the real-time simulator is shown in FIGURE 4. A single 1000 volt supply serves as a source for the required current generators (12 in the FIGURE). The binary number is stored in static flip-flops (e.g. Eccles-Jordan) which maintain cathode followers at +6 v. or -6 v. according as the digit in the corresponding column is 0 or 1, resp. Thus, a current proportional to 2^k flows through the summing resistor only when the bit in column k is a one; the total current through the summing resistor is proportional to the binary number.

The precision of the decoder shown in FIGURE 4 is clearly one part in 4096. Precision must be high in real-time simulation to eliminate large discrete steps in instrument readings. On the other hand, accuracy need not be as high. The 0.1% accuracy of this decoder was therefore acceptable. Decoding time depends mainly on the speed of the flip-flop-cathode-follower combination, about 1 microsec.

The output of the decoder ranges from zero to -1 v., and a proportional amplifier generates the larger range, -5 v. to + 5 v. Since the decoder-amplifier combination is relatively expensive, only one such unit was provided. A multiplexer, shown in FIGURE 5, was designed to distribute the amplifier output to the various instruments. As with the decoder,

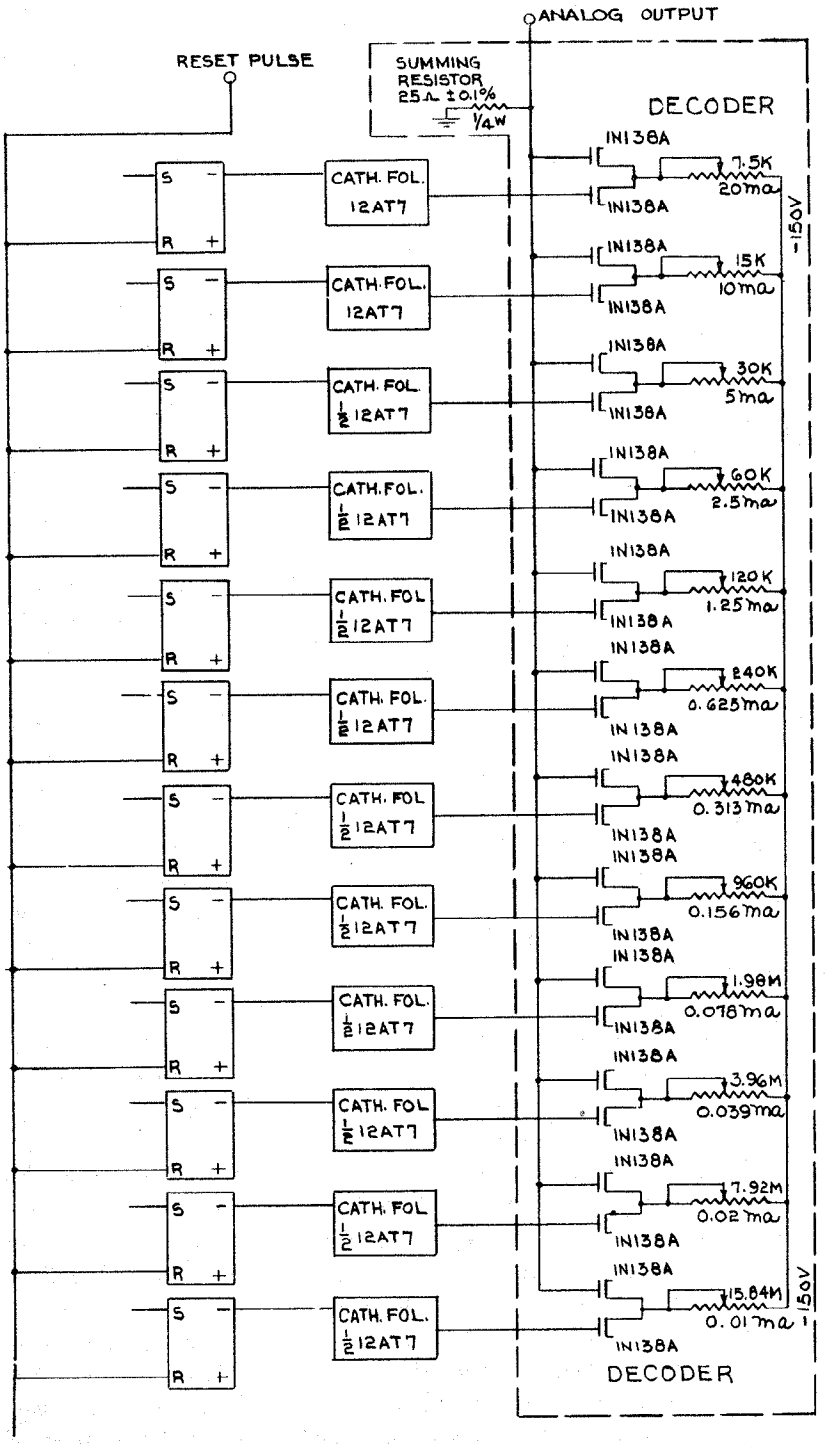


FIGURE 4. DIGITAL-TO-ANALOG DECODER

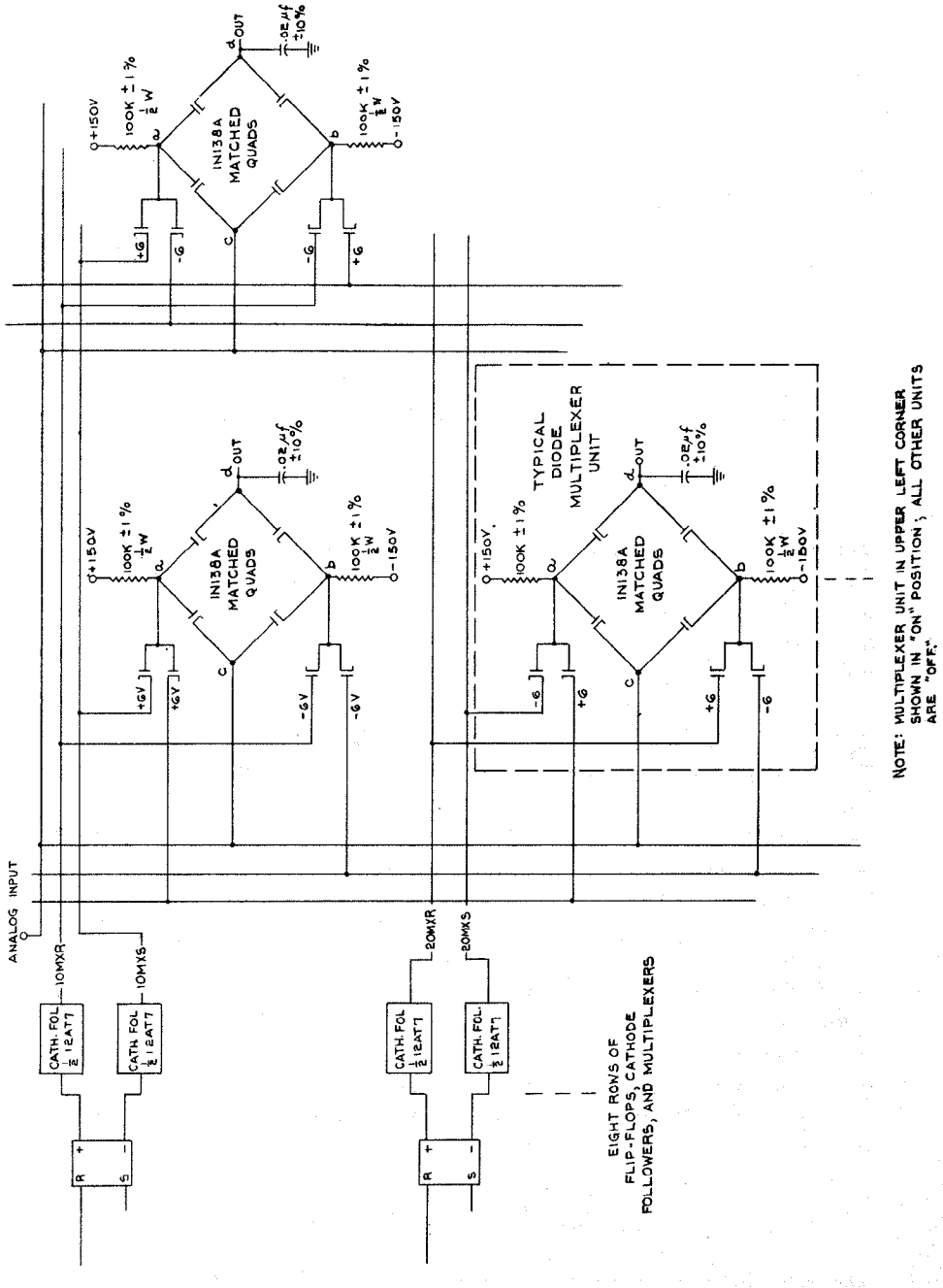


FIGURE 5. DIODE MULTIPLEXER

selection is made through cathode followers driven by static flip-flops. The amplifier output is connected to each diode bridge at the point horizontally opposite the condenser. Normally the bridges are cut off by the diodes held at ± 6 volts by the selection cathode follower; the condensers thus maintain their charge (except for the slight leakage through the silicon junction diodes[3]). In FIGURE 5, the top left multiplexer is shown turned on, i.e., the gating diodes are cut off by the cathode followers and current flows through the diode bridge from the + 150 v. supply. Since the bridge diodes have only small voltage drops across them and are matched, and since the amplifier output has negligible resistance, the condenser charges (or discharges) to the amplifier output voltage.

Note that this multiplexer allows diode gating of conventional type, which permits reduction of selection equipment.

Mathematical considerations. As mentioned in the introduction, real-time simulation using digital computers is essentially equivalent to numerical solution of a set of simultaneous non-linear ordinary differential equations at a pace sufficient to keep up with real time. The numerical solution consists of evaluating the magnitudes of the parameters of motion at discrete instants in time through integration of the differential equations over short time intervals, called quadrature steps. Thus, the state of the system at any time $\tau + h$ is obtained in one step from the state at time τ and the differential equations of motion. In real-time simulation the quadrature steps are preferably all of equal duration h.

The classical approach to the numerical solution of differential equations is to select a quadrature formula which introduces negligible "truncation" or "curtailment" error at each step. For real-time simulation, this approach leads to the difficulty that, since the solution depends upon the external driving forces and since the latter are introduced in an *unpredictable* manner, it is virtually impossible to determine the truncation error at every step in the solution.

The approach developed at the Moore School is based upon *a priori* evaluation of the natural (complex) frequencies of the simulated system in all its physically permissible (or conceivable) dynamic states. The effect of the curtailment error for any specified quadrature formula, Q, is displayed in a "stability chart" which shows, for any true frequency λ, the incorrect frequency μ which results from numerical solution using Q.

More precisely, the system of differential equations can be displayed in the form:

$$\dot{x}_i = g_i (x_1, x_2, \ldots, x_n, t); \; i = 1,2, \ldots . n \qquad (1)$$

where x_i are the motion parameters (e.g., in the case of airplane flight, these are the linear velocities, angular velocities, direction cosines, and

height), t is real time, and the dot represents differentiation with respect to time. Questions of stability of these equations can usually be answered by applying the Liapounoff criterion of stability[9,13] to the equations of the first approximation:

$$\dot{x}_i = \sum_{j=1}^{n} a_{ij} x_j + f_i(t);\ i = 1, 2, \ldots, n \tag{2}$$

$$a_{ij} = \frac{\partial g_i}{\partial x_j}$$

The a_{ij} are evaluated of course in the vicinity of the motion considered; they are therefore different for different instantaneous states of motion. Bounded true solutions of the exact equations 1 are assured if the n eigenvalues λ_k of the matrix $A = (a_{ij})$ all have negative real parts. The question of stability of numerical solutions may therefore be approached from a study of the solution of the equivalent linear equations 2.

Consider first the numerical solution $x'(t)$ of the single linear equation

$$\dot{x} = \lambda x;\ \lambda \text{ a complex constant} \tag{3}$$

For simplicity, let us employ the primitive quadrature formula O_{11}* with quadrature interval h, thus

$$x'(\tau + h) = x'(\tau) + h\dot{x}'(\tau);\ \text{for all } \tau \tag{4}$$
$$= (1 + h\lambda)\ x'(\tau)$$

Beginning at $\tau = 0$ and applying the formula m times, we obtain the numerical solution

$$x'(mh) = x(0)\ (1 + h\lambda)^m \tag{5}$$
$$= x(0)\ e^{h\mu m}$$

where the new complex number μ is defined by

$$e^{h\mu} = 1 + h\lambda \tag{6}$$

*O_{NM} is an open formula (predictor) using N ordinates and M derivatives. Similarly C_{RS} is a closure formula (corrector) using R ordinates and S derivatives.

Now the true solution of equation 3 yields at time $t = mh$:

$$x(mh) = x(0)e^{\lambda mh} \tag{7}$$

Comparing equations 5 and 7, the only effect of the quadrature solution is to replace the true frequency λ by the incorrect frequency μ. If we write $z = h\lambda$, $w = h\mu$, then equation 6 defining μ in terms of λ may be expressed as a mapping of the dimensionless frequencies z and w in the complex plane.

The implications of equation 6 form the basis of the stability chart. It can be shown that the solution of equation 3 using *any* quadrature formula can be expressed as a mapping of w onto the z-plane. Moreover, *for any quadrature formula employing only ordinate-values and derivative-values,* the mapping can always be expressed in the form

$$z = \frac{N(e^w)}{D(e^w)} \tag{8}$$

where N and D are polynomials whose coefficients depend only upon the quadrature formula in question.

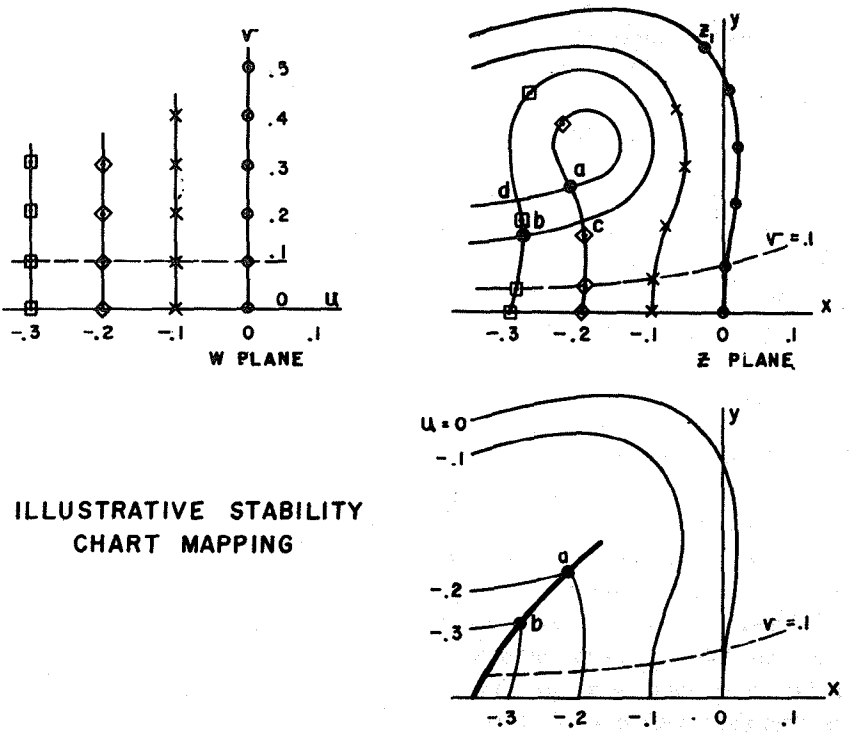
Now there is a fundamental theorem[10] in the theory of linear differential equations which states that the solution of equations 2 contains two parts, a particular solution dependent upon the forcing functions $f_i(t)$ and a complementary (transient) solution, independent of $f_i(t)$. The complementary solution of equations 2 for $f_i(t) = 0$ is

$$X(t) = \sum_{k=1}^{n} c_k X_k e^{\lambda_k t} \tag{9}$$

where $X(t)$ is the solution in vector form $(x_1, x_2, x_3, \ldots x_n)$; X_k are the eigenvectors corresponding to λ_k; and c_k are constants determined by the initial conditions

$$X(0) = \Sigma c_k X_k \tag{10}$$

It can be shown that the behavior of the true solution is strongly influenced by the frequencies λ_k in the transient solution even when $f_i(t) = 0$. Moreover, it can be shown[11] that the behavior of the numerical solution of equations 2 using quadrature Q may be determined by applying equation 8 (for quadrature Q) *separately* to each λ_k. Consequently, the study of the single equation 3 and its frequency mapping (equation 8) are basic to determining the behavior of equations 2 and



ILLUSTRATIVE STABILITY CHART MAPPING

FIGURE 6. ILLUSTRATIVE STABILITY CHART MAPPING

hence, within the limitations of the Liapounoff criterion, the behavior of the numerical solution of the original equations 1.[11, 12]

To display equation 8 in a more convenient manner, note that w is a multiple-valued function of z, but z is a single-valued function of w. In other words, for any single specified frequency λ and quadrature interval h, the numerical solution will in general contain many frequencies μ_j; on the other hand, any specified frequency in the numerical solution can result from only one true frequency. It follows that a single-valued mapping can be obtained only by mapping the w-plane onto the z-plane.

Let us first examine the many frequencies $w_j = h\,\mu_j$ which result from the single frequency z_o in the true solution. Let the $w_j = u_j + i\,v_j$ be arranged in order of decreasing damping terms, so that $u_1 > u_2 > u_3 > \ldots$ As time increases, the subdominant frequencies decay more rapidly than the *dominant* w_1; thus only w_1 is of any importance in the asymptotic solution. The *stability chart* is a mapping of the dominant frequencies w_1 *(z)* onto the z-plane.

The derivation of a stability chart may be explained with the aid of FIGURE 6. Straight lines corresponding to constant damping *(u)* and constant frequency *(v)* are shown on the w-plane, upper left. The mappings of these lines onto the z-plane for a fictitious quadrature formula Q are shown upper right. The curvature of the mapped lines is a measure

of the *inaccuracy* of Q; it will be shown that the *instability* of solutions using Q results from the excursion of the mapping for $u = 0$ into the left half-plane.

Let us first interpret the mapping of the point $u = 0$, $v = .5$ into the point $x = -.03$, $y = .48$. Clearly this indicates that if the true solution has dimensionless frequency $z_1 = -.03 + i\ .48$, corresponding to a damped sinusoid, the solution using formula Q will contain the undamped frequency $w_1 = i\ .5$. Moreover, if the true solution is a damped sinusoid represented by a point slightly to the right of z_1, the numerical solution will be a sinusoid with exponentially increasing amplitude and hence unstable. In fact, stability is achieved only when the frequencies in the true solution lie to the left of and under the $u = o$ curve.

The branch points, a and b, may now be interpreted. At these points, one of the subdominant roots of equation 8 grows to have a real part equal to the real part of the dominant root. Thus, the same true frequency z produces two quadrature frequencies with equal damping. Both frequencies are of consequence since they will persist for an equally long time in the numerical solution.

Consider now the lower intersection c of the lines $u = -.2$, $u = -.3$. Here the loop of the latter intersects the rising portion of the former at a point corresponding to the same true frequency. However, since $u = -.2$ is dominant over $u = -.3$, only the former is to be retained. By similar argument, it is concluded that the $u = -.3$ curve is subdominant at the upper intersection d and in fact at every point on its loop. The entire looped

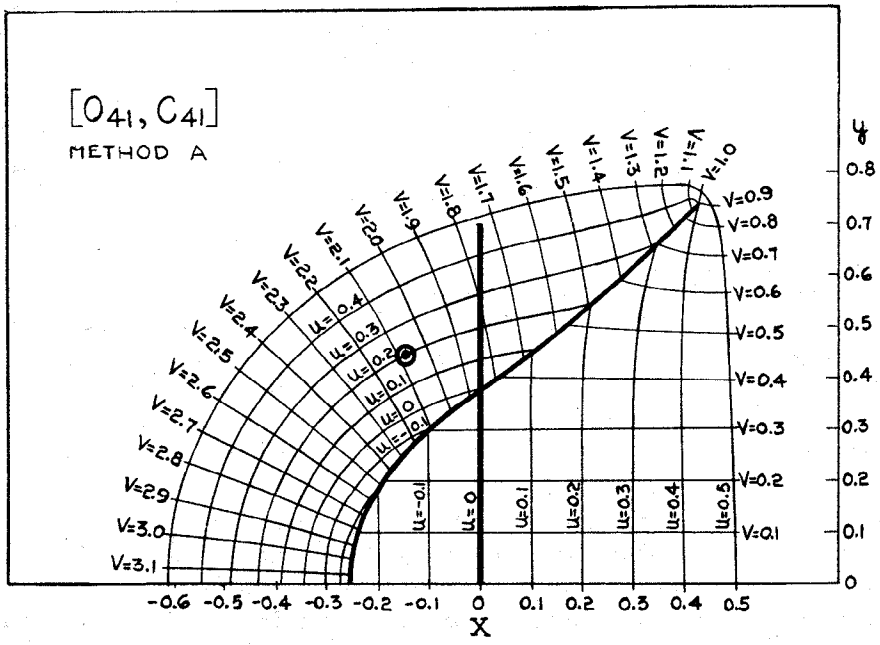


FIGURE 7. STABILITY CHART FOR METHOD A

portion of the $u = -.3$ curve is subdominant at the upper intersection d and in fact at every point on its loop. The entire looped portion of the $u = -.3$ curve may therefore be erased. Similarly the loop on $u = -.2$ and all such loops may be erased. Note the resulting discontinuity at each branch point.

The final stability chart is shown bottom right in FIGURE 6. The locus of the branch points, called a *branch contour*, is shown as a heavy line passing through a and b. The accuracy of formula Q is directly related to the squareness of the stability chart; in this case, the accuracy is only fair underneath the branch contour and very poor above the branch contour. Stable solution using this quadrature formula can be obtained only by choice of a quadrature interval small enough to locate all true dimensionless frequencies z inside the curve $u = 0$.

FIGURE 7 displays the stability chart for quadrature formula O_{41}, C_{41} ("Method A"), specified by the equations

$$\begin{aligned} O_{41}&: \; x'(h) = \frac{1}{3}[-x(0) + 18x(-h) - 6x(-2h) + x(-3h)] + 4h\,\dot{x}(0) \\ C_{41}&: \; x(h) = \frac{1}{25}[48x(0) - 36x(-h) + 16x(-2h) - 3x(-3h)] + \frac{12}{25}h\,\dot{x}'(h) \end{aligned} \tag{11}$$

Note that this formula is very accurate below the branch contour and that the $u = 0$ line never reaches as high as $v = .4$.

FIGURE 8 displays the stability chart for O_{11}, C_{41} ("Method N"), specified by the equations

$$O_{11}: \; x'(h) = x(0) + h\,\dot{x}(0)$$

$$C_{41}: \text{ as in equations 11}$$

Note that this formula is not too accurate below the lower branch contour but the $u = 0$ line reaches nearly to $v = 1$.

The circled dots on FIGURES 7 and 8 correspond to the highest natural frequency (about 0.7 cycle/sec) of a typical airplane in a rather violent dive maneuver. The maneuver was computed accurately using a very small interval and then recomputed using first Method A and then Method N, both with quadrature interval of 1/8 second. Note that the stability chart for Method A predicts an unstable solution with $u = +.2$; on the other hand, Method N predicts a stable solution, somewhat inaccurate during transients where the highest frequency is excited but rather good for steady flight where only the lower frequencies participate.

The computed results for pitching velocity, q, are shown in FIGURE 9. The instability of Method A is demonstrated within 5 seconds of flight. The stability and essential correctness of Method N is also apparent, and the error during fast transients is clear, It is interesting to

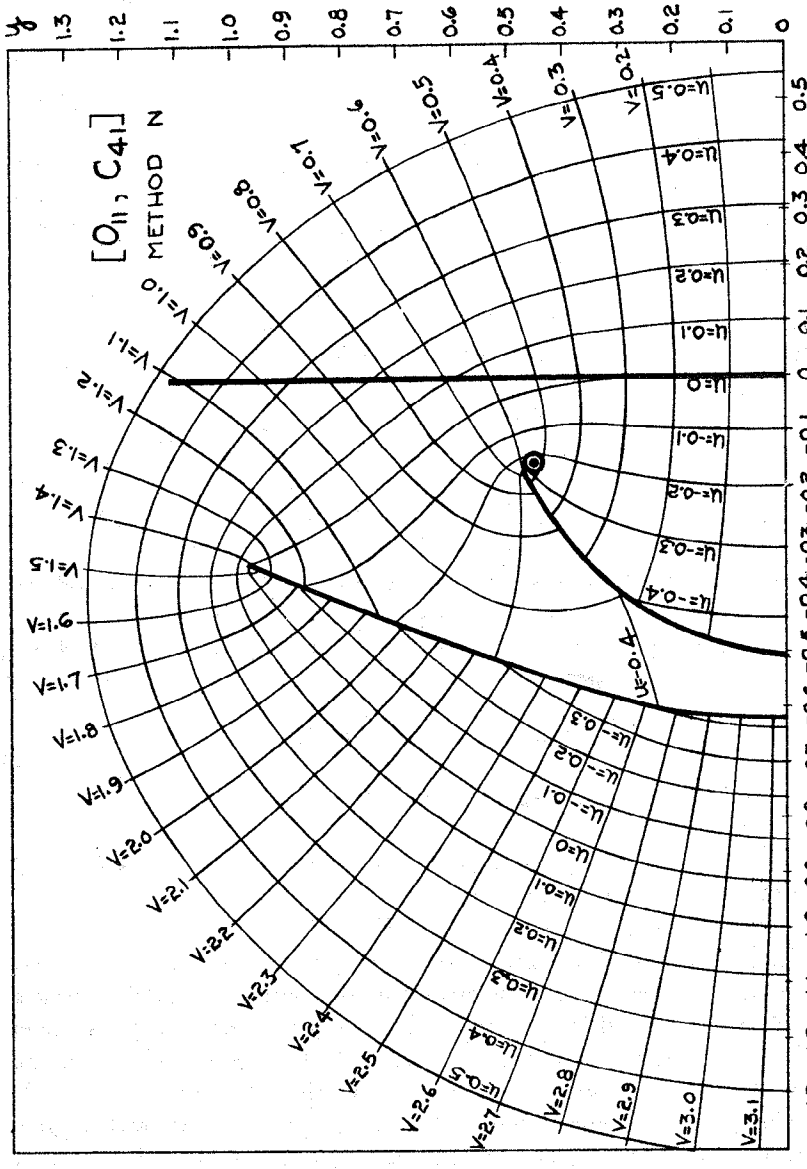


FIGURE 8. STABILITY CHART FOR METHOD N

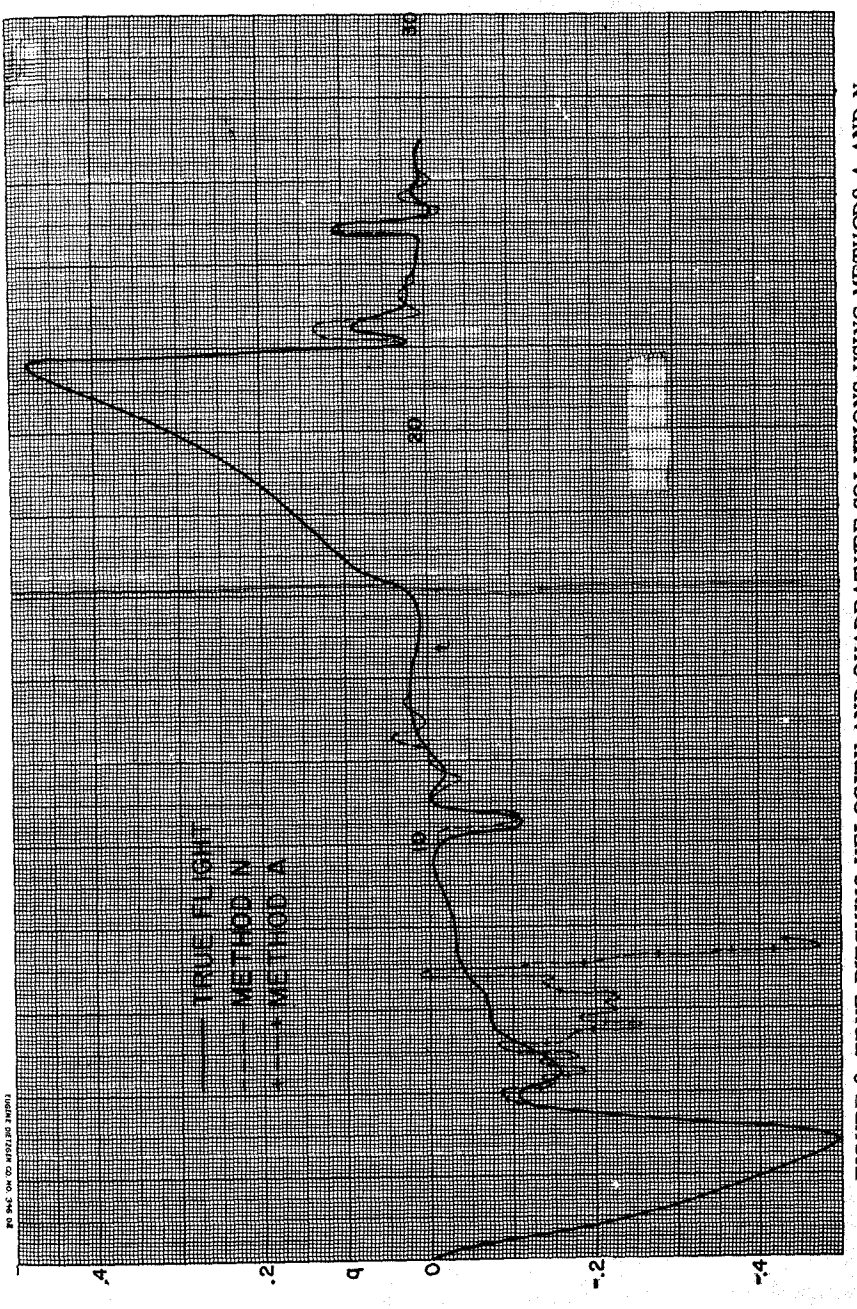


FIGURE 9. TRUE PITCHING VELOCITY AND QUADRATURE SOLUTIONS USING METHODS A AND N

observe that the change from maximum achievable q to zero q, which occurs near $t = 22$ seconds, takes places in only three quadrature steps.

Acknowledgments. The results reported in this paper were achieved through the efforts of a number of co-workers over the past several years. It is particularly proper to emphasize the contributions of C. Eldert H. J. Gray, Jr., E. Grosswald, H. Gurk, and C. T. Leondes.

References

1. Morris Rubinoff, "Analogue vs. Digital Computers – A Comparison", Proc. I.R.E., pp. 1254-1262, Oct. 1953.
2. Gluck, Gray, Leondes, Rubinoff, "The Design of Logical OR-AND-OR Pyramids for Digital Computers", Proc. I.R.E., pp. 1388-92, Oct. 1953.
3. "Universal Digital Operational Flight Trainers", University of Pennsylvania, Moore School Research Division Rept. 54-45, 30 June, 1954; Sec. 3.4.
4. H.J. Gray, Jr., "Numerical Methods in Digital Real-Time Simulation", Quart. Appl. Math., July 1954.
5. H.J. Gray, Jr., "The Organization of a Digital Real-Time Simulator", Convention Record of the Institute of Radio Engineers, Part 1 – Radar and Telemetry, 85, March 23-26 (1953).
6. Harry E. Burke, Jr., "A Survey of Analog-to-Digital Converters", Proc. I.R.E., pp. 1455-1462, Oct. 1953.
7. Gray, Levonian, Rubinoff, "An Analog-to-Digital Converter for Serial Computing Machines", Proc. I.R.E., pp. 1462-1465, Oct. 1953.
8. Bernard Lippel, "Interconversion of Analog and Digital Data in Systems for Measurement and Control", Proc.Nat.Elec.Conf., Vol. VIII, 1953.
9. N. Minorsky, "Introduction to Non-Linear Mechanics", J.W. Edwards, Ann Arbor, 1947, p. 52.
10. E.L. Ince, "Ordinary Differential Equations", p. 115, Dover Publications, New York.
11. "Design of Digital Flight Trainers", University of Pennsylvania, Moore School Res. Div. Rep. 54-09, 6 Sept. 1953.
12. Herbert M. Gurk and Morris Rubinoff, "Numerical Solutions of Differential Equations", Proceedings of 1954 Eastern Joint Computer Conference (March, 1955).
13. Richard Bellman, "A Survey of the Theory of the Boundedness, Stability and Asymptotic Behavior of Solutions of Linear and Non-Linear Differential and Difference Equations", Office of Naval Research, Washington, D.C., 1949.