1983

An Extension to the Best Numerical Integration Formula Development

Jorge Medina
j.medina@ieee.org

AN EXTENSION TO THE BEST NUMERICAL INTEGRATION FORMULA DEVELOPMENT

BY

JORGE MEDINA
B.S.E., Florida Technological University, 1974

THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Engineering in the
Graduate Studies Program of the College of Engineering
University of Central Florida
Orlando, Florida

Spring Term
1983

ABSTRACT

A mathematical analysis seeking an accurate measure of the worth of numerical integration techniques used for real-time digital flight simulation problems is presented.

This investigation allows the subject of "best" integration methods to be pursued making emphasis on the choice of practical steps and the use of available mathematical techniques to illustrate and evaluate a potential root matching approach involving a selected first-order differential system.

This study allows certain evaluational techniques to be developed. Notable among these are the schemes for comparing roots of sampled ideal integrators to roots of sampled approximated integrators, the development of an integration and of an iteration formula, and the creation of a computer program.

ACKNOWLEDGEMENT

This thesis was made possible by the continued support from Dr. Benjamin W. Patz. I owe a great debt of gratitude to Dr. Patz for his enthusiasm and the outstanding example of excellence he set for me.

PREFACE

The study presented in this thesis seeks an accurate measure of the worth of numerical integration methods used in flight simulation problems. The emphasis here is on the choice of practical steps and the use of available mathematical techniques to evaluate a potential root matching approach involving a selective first-order differential system. This study assumes a knowledge of numerical analysis terminology and techniques. A computer program is developed with the aid of an HP-41CV hand-held calculator system which is then written in Fortran and run in the Harris 800 computer.

Section I explains how integration methods have been evaluated in the past and how typical textbooks today compare them. The criteria for comparison of the various methods is outlined along with a brief description of the program capabilities.

Section II is an illustration of the theoretical basis for the root comparison techniques. Included are arguments and techniques as to how to compare roots of sampled ideal integrators to roots of sampled approximated integrators.

Section III shows the development of a form of integration and of iteration formulas from the closed quadrature

formula. There follows a discussion of the z-transform method to describe a numerical method which can be used to approximate the desired solution of a given differential equation.

Section IV discusses the four methods of integration which were used to investigate the root error problem. Included is a summary of their key characteristics from a general and comparative point of view.

Section V illustrates the results of the computer programs with plots to compare the stability range, the percent constant root error and the transient root errors of the four methods under investigation. Included are arguments and techniques as to how to match the numerical transient root to its analytic counterpart.

Section VI explains the advantages and disadvantages of the root matching techniques as a tool for measuring the accuracy of integration methods and finally, in Section VII recommendations of other potential areas of investigation in the field of the evaluation of numerical integration techniques are presented.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SECTION I

INTRODUCTION

A significant task in flight simulation is to numerically solve the differential equations of motion for the vehicle of concern. For this task a method of numerical integration is required.

Although a theoretically infinite number of numerical integration techniques exist, fewer than a half dozen are popularly chosen as candidates each time a flight simulator is developed. According to Knoop (1) the final selection of one method of integration is normally based on two considerations:

1. The apparent past success of the method for other simulations.
2. The apparent accuracy of the method for present simulation as judged by several empirical tests.

Classical methods like those analyzed and developed by Knoop (1) and Nigro (2) were basically evaluated on the stability, the truncation error, the round-off error and the propagation error. Their tests are time consuming, laborious and lacking root consistency, not permitting the integration methods to be effectively compared so that an easy selection can be made of a method best suited for the problem at hand.

On the other hand, today's typical textbooks do not generate adequate techniques for root measures. Ball (3), Forsythe (4), Kreyszig (5), Linz (6) and Scheid (7), for example, based the comparison of the accuracy of integration methods in the case of the initial value problem, on the absolute value of error. To others, like Ferziger (8), it is more appropriate to discuss numerical methods in terms of amplitude and phase errors rather than the apparently simpler truncation error when one is interested in problems with oscillatory solutions.

As a result of this, in 1980 a study was begun to test a proposed measure of accuracy of numerical integration techniques for application of complex numbers to a selected first-order differential system.

This test consists of matching the percent root error to the transient root error. This objective was conducted in two phases over a two-year period, one phase being devoted exclusively to the % root error and % transient root error analysis and the second phase to the implementation of a Fortran computer program.

During the first phase, the theoretical bases for the root comparison techniques were set, the iteration formula for calculating the root accuracies was developed from the quadrature formula as well as a form of integration formula for computing the numerical and analytical transient root

errors. Four candidate methods of integration were chosen to implement this test and they are: O_{33} Mod Gurk, O_{12} Second-Order Adams, MTH .01-.9 and MTH .0-1. The assumptions made in order to perform this test are as follows:

1. The system does not have to be stable but the integration method should be stable.

2. The system will be excited by a single input driving delta function (unit impulse).

3. The roots will be bounded in a complex z-plane and should be independent of the system.

The criteria for comparison of the four integration methods, in order of importance is:

1. The stability of the method.

2. The order of accuracy provided by each of the methods being compared.

The iteration formula developed from the quadrature formula is used to compare the four methods. The iteration formula calculates the sampled approximated root A'T, which is also called the computed, iterated or exact root. This root is then compared to the sampled ideal root AT, also called the ideal root, and recalculated until their absolute value is within a predetermined limit or % accuracy. The final iterated A'T root is then used to initialize the integration formula, ultimately calculating the transient root error, Rss.

The Rss error plot of the integration formula will give insight about the error due to the unit impulse driver. Also, the transient could be identified, which in turn is a good measure of response of a system. On the other hand, the resonance response could be analyzed via the constant root error plot.

These relative errors versus its root locations may be sufficient enough to help predict and evaluate ahead of time the accuracy of an integration technique and lead us into a solution of the root error problem being investigated.

The second phase of this study is the generation of a Fortran computer program to implement the theoretical work performed during the first phase. The resulting program REAP, Root Error Analyzer Program, is capable of evaluating integration methods for applications to first-order differential systems by calculating the constant root errors and the transient root errors.

Originally, the program was implemented in RPN (Reverse Polish Notation) for the Hewlett Packard HP-41CV hand-held calculator system but later programmed and executed in Fortran 77 using the Harris 800 computer at the University of Central Florida via batch processing.

The calculations were performed in single-precision, floating-point arithmetic.

In using this program, one must supply certain input

data which describes both the integration method and the accuracy desired for the constant root error.

The program computes and outputs key characteristics of the constant root error and the transient root error. From these the user can easily discern the advantages or disadvantages afforded by the use of the method for the particular root error problem of concern.

A listing of the Root Error Analyzer Program which consists of the % Root Error, Numerical Rss Error and Analytic Rss Error routines appears in Appendix B.

SECTION II

THEORETICAL BASES FOR ROOT COMPARISON TECHNIQUES

The theoretical work in this analysis which forms a basis for the computer program is presented in this section, i.e., the arguments as to how to compare "roots" of continuous integrators to "roots" of sampled integrators.

Continuous Integrator Concepts

We shall restrict ourselves to a system having an exponential response, so in order to illustrate the root comparison techniques, a first-order system with

$$y_i(t) = Ay_i(t) \tag{1}$$

is selected as the significant test function, where A is a complex number and represents the true or natural resonant frequency.

We are interested in frequencies with non-positive real parts, i.e., $Re(A) < 0$. If the differential equation 1 is solved analytically, its solution is found to be

$$y_i(t) = e^{At}$$

The real part of A determines the rate at which the solution grows or decays and its imaginary part is 2π times

the frequency of the oscillations in the true solution. Hence A is a natural frequency of equation 1.

The Laplace transform of $y_i(t)$ is:

$$Y_i(s) = \frac{1}{s-A}$$

An example of a continuous integrator is shown in Figure 1. Note that continuous functions are special cases of discrete time functions. Also we will limit ourselves to the special case of a step response (since the general case is too complex).

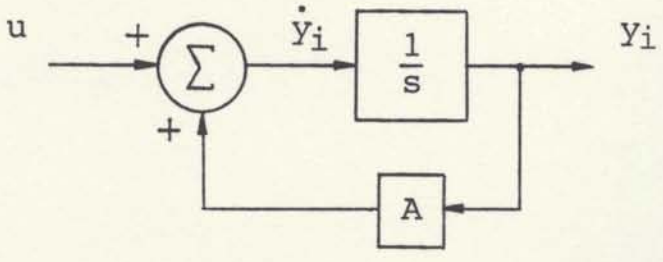

Figure 1. Continuous Ideal Integrator

The transfer function of this system is

$$H_i(s) = \frac{Y_i(s)}{U(s)} = \frac{1}{s-A} \qquad (2)$$

where A is a complex number, $Re(A)<0$, and a dominant root, $s=A$

Therefore the complex exponential time function becomes

$$y_i = e^{At}, \text{ where } A=\sigma+jw,$$

or

$$e^{At} = e^{\sigma t}\left[\cos(wt) + j\sin(wt)\right]$$

Sampled Integrator Concepts

An example of a sampled integrator is shown in Figure 2. The effects of sampling on an integrator system are best illustrated through the normalized z-transform.

$$\underline{f(t)} \xrightarrow[T]{} \underline{f^*(t)} \text{ sampler}$$

The sampler takes the time function f(t) and generates a sequence of delta functions f*(t). The area under each delta function corresponds to f(t) at the sampling instant, so

$$f^*(t) = \sum_{n=0}^{\infty} f(t)\cdot\delta(t-nT) = \sum_{n=0}^{\infty} f(nt)\cdot\delta(t-nT)$$

where T is the sampling interval (i.e., in our case the computational interval, T).

We define a z-transform, so $\mathcal{Z}[\,\delta(t-nT)\,] = z^{-n}$. NOTE: $\delta(t-nT) \leftrightarrow e^{-nTs}$ under the Laplace transform, hence when in the s domain where the characteristic roots are easily determined, we also see

$$z^{-n} = e^{-nTs}\Big|_{e^{Ts}\triangleq z}$$

Examples of z-transforms from Karworski (9)

(a) $$f(t) = e^{-At} \leftrightarrow \frac{1}{s+A}$$

$$f^*(t) = (e^{-At})^* \leftrightarrow \frac{1}{1-z^{-1}e^{-AT}}$$

(b) Unit step $u(t) \leftrightarrow \dfrac{1}{s}$

$$u^*(t) \leftrightarrow \frac{1}{1-z^{-1}}$$

If the roots of z are within the unit circle in the z plane, then the solution is stable.

In Figure 2 the integrator function is I(z) which represents the z-transform of the integration formula is an approximation to the ideal integration function.

The integration formula is developed in Section III.

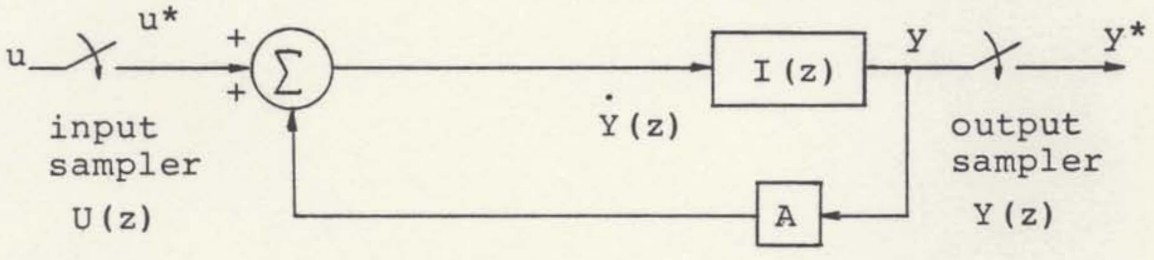

Figure 2. Sampled Approximated Integrator

$$Y(z) = I(z) \cdot [U(z) + AY(z)]$$

Therefore the transfer function for this sampled approximated integrator becomes

$$H(z) = \frac{Y(z)}{U(z)} = \frac{I(z)}{1-AI(z)} \qquad (3)$$

It is the intent of this section to introduce the root comparison techniques of sampled ideal and approximated integrators; in other words, the roots of $Y_i(z)$ versus $Y(z)$, where $y_i^*(t)$ is defined as the sampled ideal solution and $y_a^*(t)$ is defined as the sampled approximated solution. For simplicity we will let $y_a^*(t) = y$ in our analysis.

If we want the primary root of $\frac{1}{\frac{1}{I(z)} - A}$ to be at s=A for the best integration formula, then the arguments for doing this are as follows:

The z-transform of $\frac{1}{s-A}$ is,

$$Y_i(z) = \frac{1}{1-z^{-1}e^{AT}}$$

where the exact root of the sampled ideal integrator then becomes $z=e^{AT}$ and the comparison of $H(z)$ and $H_i(z)$ becomes

$$\frac{1}{\frac{1}{I(z)} - A} \cong \frac{1}{1-z^{-1}e^{AT}}$$

or

$$\frac{1}{I(z)} - A \cong a-z^{-1}e^{AT}$$

We could substitute $z=e^{AT}$ into $1-z^{-1}e^{AT}$ then

$$\frac{1}{I(z)} \Big|_{z=e^{AT}} - A \cong 0$$

but

$$\frac{1}{I(z)} \Big|_{z=e^{AT}} \neq A$$

But instead, let

$$\frac{1}{I(e^{AT})} = A'$$

so that A'-A is a measure of the small error in the root position, where A' is the root of the sampled integrator, that is, the iterated root.

We choose not to solve $\frac{1}{I(z)} - A=0$ since this yields, in general, a complicated polynominal.

But if we solve for A',

$$\frac{1}{I(z)}\Bigg|_{z=e^{AT}} -A' = 0$$

the calculations are easy and the distance (A'-A) should, in concept, be the same. Hence comparing the roots of Y_i and Y should yield some root matching and if the primary root of Y_i matches the primary root of Y, the damping and frequency characteristics of solutions to equations 2 and 3 will be similar.

The calculation of A' is done by means of an iterative process in which the percentage of error is computed as described in Section V, using successively different (larger) and smaller) values of A until the desired percentage is attained.

Using the z-transform method we have been able to exactly set up the numerical integration scheme in a form which lends itself for exact analysis, since it provides the basis for analyzing all equal interval integration schemes--not only of stability analysis, but also for root error analysis.

SECTION III

FORM OF INTEGRATION FORMULAS

The iteration and the integration formulas were derived from the closed quadrature formula described in (10),

$$y_n = \sum_{i=1}^{K} a_i y_{n-i} + T \sum_{j=1}^{K} b_j \dot{y}_{n-j}$$

where T is the quadrature interval, a_i and b_i are the known coefficients of the quadrature formula and, K is the order of the formula.

If we let K=3 the quadrature formula becomes

$$y_n = a_1 y_{n-1} + a_2 y_{n-2} + a_3 y_{n-3} + T(b_1 \dot{y}_{n-1} + b_2 \dot{y}_{n-2} + b_3 \dot{y}_{n-3}) \qquad (4)$$

Since this equation is a recursion relation involving four sequences of numbers, y_n, y_{n-1}, y_{n-2}, y_{n-3}, then, since n is arbitrary, the above equation may also be written

$$y_{n+3} = a_1 y_{n+2} + a_2 y_{n+1} + a_3 y_n + T(b_1 \dot{y}_{n+2} + b_2 \dot{y}_{n+1} + b_3 \dot{y}_n) \qquad (5)$$

Development of the General Integration Formula

The z-transform method is used to describe a numerical method which can be used to approximate the desired solution of a system of ordinary differential equations.

This method is a technique for deriving a difference equation whose solution approximates the solution of a given

differential equation. The difference equation so derived is required to have the following two properties:

(a) It must have a solution that has the same damping and frequency characteristics as the continuous solution (solution of the differential equation).

(b) It must have a solution with a final value corresponding to the solution of the differential solution.

Property (a) is satisfied if the roots of the discrete representation match the roots of the approximate discrete representation.

Property (b) is satisfied if the final value of the output function of both representations (given above) are the same.

In employing this method we are matching our sampled ideal system with the sampled approximated system.

If we take the z-transform to equation 5 we get

$$z^3Y(z) - z^3y_0 - z^2y_1 - zy_2 = a_1[z^2Y(z) - z^2y_0 - zy_1]$$

$$+ a_2[zY(z) - zy_0] + a_3[Y(z)] + T(b_1[z^2\dot{Y}(z) - z^2y_0 - zy_1]$$

$$+ b_2[z\dot{Y}(z) - zy_0] + b_3[\dot{Y}(z)])$$

Assuming initial conditions are set to zero, i.e.,

$y_0 = y_1 = y_2 = 0$, we now obtain

$$z^3Y(z) = a_1z^2Y(z) + a_2zY(z) + a_3Y(z) + T[b_1z^2\dot{Y}(z) + b_2z\dot{Y}(z) + b_3\dot{Y}(z)]$$

which may be written as

$$Y(z)\ [z^3 - a_1 z^2 - a_2 z - a_3] = T[b_1 z^2 + b_2 z + b_3]\dot{Y}(z)$$

or

$$\frac{Y(z)}{\dot{Y}(z)} = \frac{T[b_1 z^2 + b_2 z + b_3]}{z^3 - a_1 z^2 - a_2 z - a_3}$$

Let $I(z) = \frac{Y(z)}{\dot{Y}(z)}$ denote the integration formula,

therefore equation 6 becomes

$$I(z) = \frac{T[b_1 z^{-1} + b_2 z^{-2} + b_3 z^{-3}]}{1 - [a_1 z^{-1} + a_2 z^{-2} + a_3 z^{-3}]}$$

or

$$I(z) = \frac{T \cdot \sum_{j=1}^{3} b_j z^{-j}}{1 - \sum_{i=1}^{3} a_i z^{-1}}$$

where $a_1+a_2+a_3=1$ and $1+a_2+2a_3=b_1+b_2+b_3$,

but $\frac{1}{I(z)}\Big|_{z=e^{AT}}=A'$, where A' is the root of the sampled

approximated integrator; then

$$A' = \frac{1 - \sum_{i=1}^{3} a_i z^{-i}}{\sum_{j=1}^{3} b_j z^{-j}}\Bigg|_{z=e^{AT}} = \frac{1}{I(z)}\Bigg|_{z=e^{AT}}$$

multiplying both sides by the normalizing T we get the

iteration formula;

$$A'T = \left. \frac{1 - \sum_{i=1}^{3} a_i z^{-i}}{\sum_{j=1}^{3} b_j z^{-j}} \right|_{z=e^{AT}} = \left. \frac{T}{I(z)} \right|_{z=e^{AT}} \tag{7}$$

the left side is the root of the sampled approximated integrator while AT is the root of the sampled ideal integrator.

From our sampled approximated integrator system we found the z-transform function to be,

$$Y(z) = \frac{U(z)I(z)}{1-AI(z)}$$

Expanding Y(z) we get the solution of the integration formula,

$$Y(z) = \frac{U(z)T\ (b_1 z^{-1} + b_2 z^{-2} + b_3 z)^{-3}}{1-(a_1 z^{-1}+a_2 z^{-2}+a_3 z^{-3})-AT(b_1 z^{-1}+b_2 z^{-2}+b_3 z^{-3})} \tag{8}$$

One root of the denominator of the above formula will be in the vicinity of z=+1, if $AT \rightarrow 0$. The significance of this is that if we first eliminate this root, then we will have a quadratic to solve, and we will be able to see how the roots of the sampled approximated integrator system compare to the sampled ideal integrator (11).

SECTION IV

NUMERICAL INTEGRATION METHODS

There are four methods of integration which the author used to investigate the root error problem, and two of them are known to be popularly selected for use in flight simulation. These are not self-starting methods. Therefore, some other procedure must be used to obtain sufficient past values.

These are discussed below from both a general and a comparative point of view. Table I summarizes the key characteristics of these four methods, taken from Nigro (2) and Knoop (1) documents, which are arranged in order of increasing stability range. The MTH methods were derived by the SAP program (1).

$\underline{O}_{33}$ Mod Gurk Method*

$$y_{n+3} = 1.1462084y_{n+2} - 0.201087y_{n+1} + 0.0548787y_n +$$

$$h(1.6415864y'_{n+2} - 1.0080130y'_{n+1} + 0.2750971y'_n)$$

---

* The double subscript, e.g., O_{ij} means that the method of integration uses i past values of the variable and j past values of its derivative to compute the present value of the dependent variable. The "O" signifies an open integration method, i.e. one which does not use the present value of the derivative (13).

The O_{33} Mod Gurk method is so named because it is a nonclassical, open three-step method derived by H.M. Gurk (12) specifically for the F-100 A flight simulation problem (2). It is classical to the second degree, has a truncation error coefficient of 0.03297, and has a stability range of (-0.82134,0). Its stability and propagated-error damping rate characteristics make it excellently suited for first-order problems of the form

$$y_i = +Ay_i, \text{ for } Re(A) < 0$$

Whereas many three-step methods tend to oscillate about solutions, the O_{33} Mod Gurk method produces a smooth, nonoscillating solution.

As a three-step method with mixed coefficients, it is time-consuming to compute.

The O_{33} Mod Gurk method is possibly the most often used and probably the most over-rated method in flight simulation work according to Knoop (1). Useful for solutions of the aerodynamic equations of the helicopter rotors, its single exceptional characteristic is its smoothness in approximating solutions of the form

$$y_i = e^{AT}, \text{ for } Re(A) < 0$$

It has enjoyed great popularity primarily because, first, it is of the only three-step methods ever suggested or recommended for use in flight simulation (2), and second, it

appeared to produce satisfactory results for the F-100A simulation problem (1).

O_{12} Second-Order Adams Method

$$y_n=y_{n-1}+h(1.5\ y'_{n-1}-0.5y'_{n-2})$$

Possibly second in popularity in simulation work is the Second-Order Adams method. Submarines and most aircraft dynamic equations use this method, which is often mistakenly referred to as the Trapezoidal Method according to Nigro (13). This is a two-step O_{12} method which, because of its simple coefficients, is extremely rapid to compute. While having a larger coefficient of truncation error (.417) than the O_{33} Mod Gurk method, it is classical to the third degree and thus matches three terms of the Taylor series. Thus depending on the integration interval, it could produce a smaller truncation error term and usually does with the Δt's popularly used in simulation.

The Second-Order Adams method has a stability interval of (-1,0), slightly larger than that for Mod Gurk. The Adams method, therefore, could use an integration interval up to .085 for the F-100A problem, considering stability alone. Like the O_{33} Mod Gurk method, the Second-Order Adams method has excellent propagated-error damping characteristics.

MTH.01-.9 Method

$$y_n = 0.614318\ y_{n-1} + 0.0097546\ y_{n-2} + 0.375927\ y_{n-3}$$

$$+ h(1.707964\ y'_{n-1} - 0.161177\ y'_{n-2} + 0.214822\ y'_{n-3})$$

The MTH.01-.9 is a three-step nonclassical method with a stability range of (-.950,0) and a truncation error coefficient of .012. It is classical to the second degree. As a three-step method with mixed coefficients, it is time-consuming to compute. This integration formula shifts the roots toward increased stability. This can be disastrous if the actual system is mildly unstable. Then one would not observe the instability. They could see only a weakly damped oscillatory system. Therefore, MTH.01-.9 should never be used to investigate a system. However, if the continuous system to be simulated is known to be unconditionally stable, then MTH.01-.9 may be an effective formula.

MTH.01-1 Method

$$y_n = 1.006667\ y_{n-1} - 0.006667\ y_{n-2}$$

$$+ h(1.488056\ y'_{n-1} - 0.494722\ y'_{n-2})$$

The MTH.01-1 method is a two-step nonclassical method with a stability range of (-1.020,0), fairly close to that of Second-Order Adams. Like the O_{33} Mod Gurk method, the

MTH.01-1 method is classical to the second degree but slightly faster to compute. It has a smaller truncation error coefficient (.009) than the MTH.01-.9 method.

TABLE I

TABLE OF NUMERICAL INTEGRATION TECHNIQUES*

Method	Order	Coefficients a_1 b_1	Coefficients a_2 b_2	Coefficients a_3 b_3	Stability Range	Truncation Error Coefficient	Degree to which Classical
Mod Gurk O_{33}	3 step non-classical	1.145208 1.641586	-0.201087 -1.008013	0.54879 0.275097	-0.821	0.03297	2
MTH .01-.9	3 step non-classical	0.614318 1.707964	0.0097546 -0.161177	0.375927 0.214822	-0.95	0.012	2
Second Order Adams O_{12}	2 step classical	1.0 1.5	0.0 -0.5	0.0 0.0	-1.0	0.417	3
MTH .01-1	2 step non-classical	1.006667 1.488056	-0.006667 -0.494722	0.0 0.0	-1.020	0.009	2

* Taken from Knoop's document (1)

SECTION V

EMPIRICAL RESULTS

In order to be able to judge the relative merits of one integration method over another, we must certainly consider stability and accuracy. However, stability alone only determines whether a method is usable or not.

Finding the Stability Range of an Integration Method

A short computer program is given in Appendix A of a method whereby the stability range* may be determined for any method of integration. The results of the four candidate methods are shown in Figure 3.

This stability plot affords us a method by which we may evaluate the stability of a given numerical integration technique.

The equation used for finding the stability range is the iteration formula developed in Section III, where $A'T \triangleq Y$.

$$Y = \frac{z^3 - [a_1 z + a_2]z - a_3}{b_1 z^2 + b_2 z + b_3}$$

where $z = e^{jw}$ and $w = \frac{n\pi}{30}$ for $n = 1,2,\ldots,60$

---

* The stability range is the largest real value of Y such that all roots lie within the unit circle.

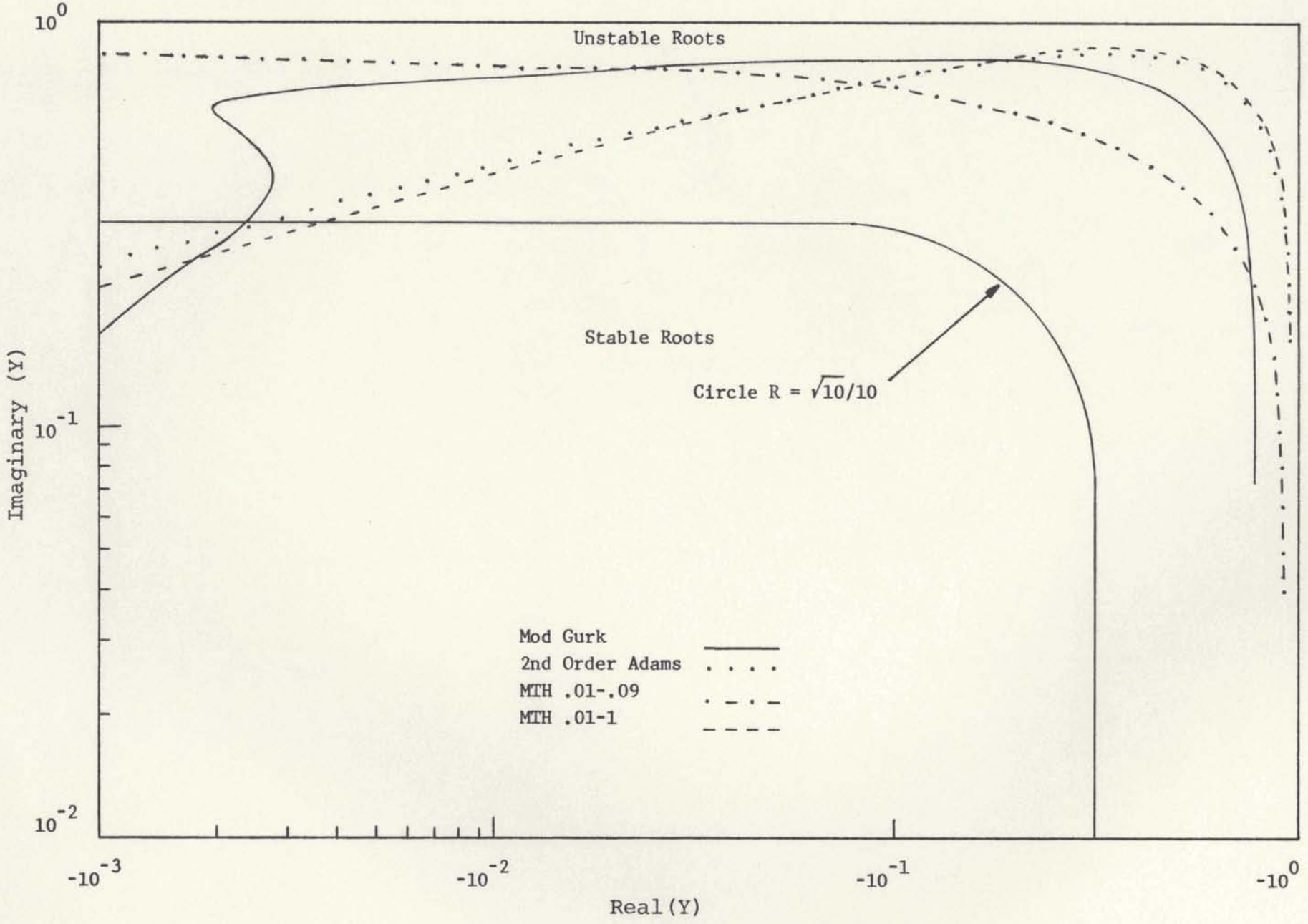

Figure 3. Stability Boundaries of Integration Methods

Here the imaginary part of $Y \to 0$ where $|Y|$ is large.

The stability range for the four methods under investigation were found to be as follows:

(a) Mod Gurk: -0.82134

(b) Second-Order Adams: -1.0000

(c) MTH.01-.9: -0.95035

(d) MTH.01-1: -1.0154

Looking at the stability plot in Figure 3 we see that MTH.01-.9 should show best results of the four methods compared if all roots inside the $\sqrt{10}/10$ circle are equally likely. However, unstable systems can yield improper results by appearing stable.

Therefore, it is better to have

(a) a log-log plot showing the stability boundary, guaranteeing "good" simulation, i.e., never forcing an unstable system to appear stable; and

(b) a log-log plot of the A'T plane mapping equal relative accuracy regions.

% Root Error Calculations

The calculation of A' is done by means of an iterative process in which the percentage of error is computed using successively different (larger and/or smaller) values of A until the desired percentage is attained. The program for this procedure is presented in Appendix B.

The algorithm which is used in solving for the % Root Error via the iteration formula, which provides the specified accuracy, is summarized and given below as a step by step procedure.

1. Set the Rss flag to zero, RSSFLG= 0

2. Choose the accuracy desired, error parameter, (in %) for the constant error curve.

3. Select the form of the integrator of the type,

$$A'T = \frac{1 - \sum_{i=1}^{3} a_{i,k} z^{-i}}{\sum_{i=1}^{3} b_{i,k} z^{-i}}$$

where k = 1 → O_{33} MOD GURK METHOD

k = 2 → MTH 0.01 - 0.9 METHOD

k = 3 → O_{12} SECOND-ORDER ADAMS METHOD

k = 4 → MTH 0.01 - 1.0

4. Pick the real axis starting point Xnew. There are 40 possible values equally spaced on the real axis.

5. Select the imaginary axis starting point, START.

6. The formula is normalized by setting the integration interval to one, T = 1.

7. From the above information a candidate exact root A'T is generated where its real part is (Xnew)T and its imaginary part is (Ynew)T.

8. Now substitute $z = e^{AT}$ and expand the resulting polynomial until we get conditions such that the calculated root $A'T = AT + \varepsilon$, where ε is the absolute error which is bounded.

Furthermore, we restrict the roots of

$$\left(1 - \sum_{i=1}^{3} a_{i,k} z^{-i} \right)$$ to be within the unit circle.

9. We seek a relative error ε as $\frac{A'T - AT}{AT}$.

10. If the absolute error $|A'T - AT|$ is within the 6% of the error parameter a new exact root is generated, otherwise Ynew is iterated via a half interval search until the exact and desired roots, A'T and AT respectively, satisfy the accuracy check being sought.

11. Step 10 is repeated until all the possible roots for the chosen method are calculated.

Note: The following approach was originally used to calculate the absolute error,

$$DELTA_1 = Re(Y) - Xnew$$

$$DELTA_2 = Im(Y) - Ynew$$

$$DELTA = \sqrt{(DELTA_1)^2 + (DELTA_2)^2}$$

where Y= A'T, the exact root. But this was not a good approach because one of the parts can have a large relative error, yet the other part contributes most to the root's magnitude.

The Numeric Development of a Technique for Evaluating an Integration Method

In this section a procedure for computing the largest transient root error and when it occurs for constant root errors is presented.

One of the main features of this technique is that the transient root error of the system can be obtained without first solving the difference equation which characterizes its behavior.

In order to match the integration formula to a first-order problem with e^{At} as its solution, we need to assume exact starting values. Let

$$y(n) = e^{A'nT}$$

where $y(n)$ is the integration formula based upon a quadrature formula which computes its next result based upon old data.

If we let $y(1)=e^{A'T}$ be the present value when $n=1$, then the exact starting values may be assumed as follows,

$$y(0) = 1 \quad , \text{ for } n = 0$$

$$y(-1) = e^{-A'T}, \text{ for } n = -1$$

$$y(-2) = e^{-2A'T}, \text{ for } n = -2$$

which can be defined as the three old values of $y(n)$ and its initial conditions for our analysis.

The basic algorithm which is used in finding these errors is summarized and given below as a step by step procedure.

1. Set the Rss flag to one, RSSFLG = 1

2. Repeat steps 2 through 10 of the % Root Error Program.

3. The exact root of the ideal integrator that satisfies the accuracy check being sought is inserted into the integration formula, i.e.,$z = e^{A'T}$ is substituted in the integration formula of the form,

$$y(n) = \sum_{i=1}^{3} a_{ik} \; y(n-i) + T \sum_{j=1}^{3} b_{ik} \; y(n-j)$$

4. Compute local error, i.e., the error in one subinterval or sampling instant.

$$XC(nT) = EX(nT) - y(nT)$$

where $EX(nT) = e^{A'nT}$ is the exact root and $y(nT)$ is the calculated solution from the integration formula.

5. Compute true absolute error by multiplying XC(nT) by its conjugate

$$YL = XC(nT) \cdot XC(nT)'$$

6. Compute summation of errors

$$SS_1 = SS_1 + YL$$

7. Find arithmetic mean of summation of errors

$$Rss_2 = \frac{SS_1}{nT} = \frac{\sum_{i=1}^{nT} YL_i}{nT}$$

8. Find the upper limit of integration. If the present arithmetic mean is less than or equal to one-half of

its previous value or if the product of the sampling interval times the real part of the exact root is greater than or equal to one, the upper limit of integration has been found; that is, if $Rss_2 \leq \frac{AMX}{2}$ or $(-nT) \cdot Re(A'T) \geq 1$ where AMX is the old largest arithmetic mean and Rss_2 is the new arithmetic mean.

9. Take square root of the summation of all arithmetic mean value

$$Rss = \sqrt{Rss_2} = \sqrt{\sum_{1}^{nT} \frac{|EX(nT) - y(nT)| \cdot |EX(nT) - y(nT)|'}{nT}}$$

where nT represents the number of A'T's (upper limit of integration).

10. Repeat step 11 of the % Root Error Program until all the possible roots of the chosen method are calculated.

This scheme is a way to test the transient response of a system. Given the initial conditions, in our case the value of the output terms y(0), y(-1), and y(-2) of the integration formula, iterate this output versus nT and the response should match that of the analytic Rss_i error,

$$Rss_i(t) = \sqrt{\int_0^t \frac{(\text{EXACT SOLUTION} - \text{INTEGRATION FORMULA})^2 \, dt}{t}}$$

The Analytic Development of a Technique For Evaluating an Integration Methodp

In this section a procedure for computing the largest

Rss error bounded by the upper limit previously found by the numerical solution for constant root errors is presented but this time is solved analytically.

The basic algorithm which is used in finding these errors is presented in Appendix B and summarized and given below as a step by step procedure.

1. Find the roots of the denominator of the integration formula under investigation. The integration formula is of the form

$$Y(z) = \frac{(b_1 z + b_2 z + b_3)}{z^3 - (a_1 + A'b_1)z^2 - (a_2 + A'b_2)z - (a_3 + A'b_3)}$$

For simplicity let

$$p = -(a_1 + A'b_1)$$

$$q = -(a_2 + A'b_2)$$

$$r = -(a_3 + A'b_3)$$

then the complex cubic equation may be equated to zero

$$z^3 + pz^2 + qz + r = 0$$

which may be reduced to the form (5),

$$(root)^3 + a(root) + b = 0$$

by substituting for z the value, root - (p/3).

Here

$$a = \frac{1}{3}(3q - p^2) \text{ and } b = \frac{1}{27}(2p^3 - 9pq + 27r)$$

For solution let,

$$u = \sqrt[3]{\frac{b}{2} + \sqrt{\frac{b^2}{4} + \frac{a^3}{27}}} \quad , \quad v = \sqrt[3]{\frac{b}{2} - \sqrt{\frac{b^2}{4} + \frac{a^3}{27}}}$$

then the values of root will be given by,

$$root_1 = u + v$$

$$root_2 = -(\frac{u+v}{2}) + j(\frac{u-v}{2})\sqrt{3}$$

$$root_3 = -(\frac{u+v}{2}) - j(\frac{u-v}{2})\sqrt{3}$$

therefore the roots of the complex cubic equation become

$$z_1 = root_1 - (p/3)$$

$$z_2 = root_2 - (p/3)$$

$$z_3 = root_3 - (p/3)$$

Now the complex coefficients p, q and r may be double-checked as follows:

$$p = -(z_1 + z_2 + z_3)$$

$$q = (z_1 z_2) + (z_1 z_3) + (z_2 z_3)$$

$$r = -(z_1 z_2 z_3)$$

and the integration formula becomes

$$Y(z) = \frac{(b_1 z^2 + b_2 z + b_3)}{(z-z_1)(z-z_2)(z-z_3)}$$

2. Solve the above integration formula by the partial fraction expansion method.

$$\frac{Y(z)}{z} = \frac{(b_1 z + b_2 + b_3 z^{-1})}{(z-z_1)(z-z_2)(z-z_3)}$$

$$\frac{Y(z)}{z} = \frac{K_1}{z-z_1} + \frac{K_2}{z-z_2} + \frac{K_3}{z-z_3}$$

Without solving for the complex constants K_1, K_2 and K_3 we get,

$$Y(z) = K_1\left(\frac{z}{z-z_1}\right) + K_2\left(\frac{z}{z-z_2}\right) + K_3\left(\frac{z}{z-z_3}\right)$$

now take the Inverse Laplace Transform to Y(z),

$$\mathcal{L}^{-1}[Y(z)] = [y(t)]^* = [K_1(z_1)^{nT} + K_2(z_2)^{nT} + K_3(z_3)^{nT}], \text{ with } T=1$$

then the first three points of $y^*(t)$ are

$$y(1) = K_1(z_1) + K_2(z_2) + K_3(z_3)$$
$$y(2) = K_1(z_1)^2 + K_2(z_2)^2 + K_3(z_3)^3$$
$$y(3) = K_1(z_1)^3 + K_2(z_2)^3 + K_3(z_3)^3$$

3. Equate the first three points of the numerical solution to the last three equations above and solve for the complex constants K_1, K_2 and K_3.

$$\begin{bmatrix} e^{-2A'} \\ e^{-A'} \\ 1 \end{bmatrix} = \begin{bmatrix} z_1 & z_2 & z_3 \\ (z_1)^2 & (z_2)^2 & (z_3)^2 \\ (z_1)^3 & (z_2)^3 & (z_3)^3 \end{bmatrix} \cdot \begin{bmatrix} K_1 \\ K_2 \\ K_3 \end{bmatrix}$$

In this way the analytic solution is initialized as the numerical solution was. For simplicity let $y \triangleq f$.

The determinant for this set of equations is

$$D = z_1[(z_2)^2(z_3)^3 - (z_2)^3(z_3)^2] - z_2[(z_1)^2(z_3)^2 - (z_1)^3(z_3)^2] + z_3[(z_1)^2(z_2)^3 - (z_1)^3(z_2)^2]$$

$$K_1 = \frac{\begin{vmatrix} f(1) & z_2 & z_3 \\ f(2) & (z_2)^2 & (z_3)^2 \\ f(3) & (z_2)^3 & (z_3)^3 \end{vmatrix}}{D} =$$

$$\frac{f(1)\left[(z_2)^2(z_3)^3-(z_2)^3(z_3)^2\right]-z_2\left[f(2)(z_3)^3-f(3)(z_3)^2\right]+z_3\left[f(2)(z_2)^3-f(3)(z_2)^2\right]}{D}$$

$$K_2 = \frac{\begin{vmatrix} z_1 & f(1) & z_3 \\ (z_1)^2 & f(2) & (z_3)^2 \\ (z_1)^3 & f(3) & (z_3)^3 \end{vmatrix}}{D} =$$

$$\frac{z_1\left[f(2)(z_3)^3-f(3)(z_3)^2\right]-f(1)\left[(z_1)^2(z_3)^2-(z_1)^3(z_3)^2\right]+z_3\left[(z_1)^2f(3)-(z_1)^3f(2)\right]}{D}$$

$$K_3 = \frac{\begin{vmatrix} z_1 & z_2 & f(1) \\ (z_1)^2 & (z_2)^2 & f(2) \\ (z_1)^3 & (z_2)^3 & f(3) \end{vmatrix}}{D} =$$

$$\frac{z_1\left[(z_2)^2f(3)-(z_2)^3f(2)\right]-z_2\left[(z_1)^2f(3)-(z_1)^3f(2)\right]+f(1)\left[(z_1)^2(z_2)^3-(z_1)^3(z_2)^2\right]}{D}$$

After solving for K_1, K_2 and K_3 substitute these constants back into the general analytic integration formula,

$$f^*(t) = \left[K_1(z_1)^t + K_2(z_2)^t + K_3(z_3)^t\right]^*$$

4. Solve the analytic Rss error, i.e.,

$$Rss_i = \sqrt{\frac{1}{t}\int_0^t [EX^*(t) - f^*(t)]^2 dt}$$

or

$$Rss_i = \sqrt{\frac{1}{t}\int_0^t \left(|EX^*(t) - f^*(t)| \cdot |EX^*(t) - f^*(t)|'\right) dt}$$

where $|EX^*(t) - f^*(t)|'$ is the complex conjugate of $|EX^*(t) - f^*(t)|$ and

$$EX^*(t) = [e^{A't}]^*$$

$$f^*(t) = [K_1(z_1)^t + K_2(z_2)^t + K_3(z_3)^t]^*$$

Here $EX^*(t)$ and $f^*(t)$ are the exact and the integration formula sampled or pulsed signals respectively at regular intervals of T seconds. This sequence of values $EX^*(nT)$ and $f^*(nT)$ may be represented by a train of impulses with the areas or strengths of the impulses equal to the magnitude of EX(t) and f(t) at the corresponding instants of time. At any given time t = nT, the impulse is $EX(nT)\cdot\delta(t-nT)$ and $f(nT)\cdot\delta(t-nT)$ and the train of impulses may be represented by the infinite summation

$$EX^*(t) = \sum_{n=0}^{\infty} EX(nT)\cdot\delta(t-nT) \quad , \; n = 0,1,2,... \tag{9}$$

$$f^*(t) = \sum_{n=0}^{\infty} f(nT)\cdot\delta(t-nT) \quad , \; n = 0,1,2,...$$

This representation of a continuous signal by impulses of varying amplitude is a mathematical convenience which simplifies much of the analysis of sampled-data systems. Whether $EX^*(t)$ and $f^*(t)$ represent a continuous signal EX(t) and f(t) respectively, at every T seconds, or if $EX^*(t)$ and $f^*(t)$ are actually sampled or pulsed signals at regular intervals of T seconds, the impulse representation may be thought of as a switch, as in Figure 4.

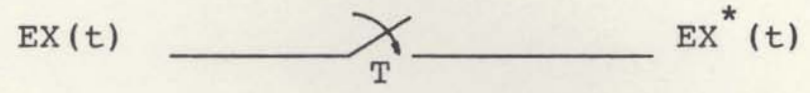

Figure 4. Ideal Sampler Model

The switch or ideal sampler closes instantaneously every T seconds, and its output at the sampling instants nT is EX(nT).

If the unit pulse train is $\delta_T(t) = \sum_{n=0}^{\infty} \delta(t-nT)$

it is observed the $EX^*(t)$ is simply the multiplication of the unit impulse train by the values of EX(t) at the sampling instants. Hence, equation 9 may be written

$$EX^*(t) = \sum_{n=0}^{\infty} EX(nT)\cdot\delta(t-nT) = EX(t)\cdot\delta_T(t)$$

The delayed delta function $\delta(t-nT)$ is an impulse of unit area occurring at time t = nT, where nT is a positive real constant. Consistent with the definition for $\delta(t)$,

$$\delta(t-nT) = \begin{cases} 0 \ , & t \neq nT \\ \infty \ , & t = nT \end{cases}$$

Also the unit area property of the impulse requires (14)

$$\int_0^t \delta(t-nT)\,dt = \int_{nT-\Delta}^{nt+\Delta} \delta(t-nT)\,dt = 1 \ , \quad t > nT$$

Another property of the delta function is that the integral

$$\int_0^t f(T)\cdot\delta(T-nT)\,dT = f(nT) \ , \quad t > nT$$

where T is a dummy variable of integration, and provided f(t) is a continuous function in an interval including T=nT.

The proof of this property follows by noting that the integrand is zero for $T \neq nT$; hence the integral may be written according to Saucedo (14)

$$\int_{nt-\Delta}^{nT+\Delta} f(T)\cdot\delta(T - nT)dT \ , \ t > nT \qquad (10)$$

where Δ is some arbitrarily small positive quantity. Using the theorem of the mean for integrals, integral 10 may be written

$$f(nT)\cdot\int_{nT-\Delta}^{nT+\Delta}\delta(T-nT)dT = f(nT)\cdot 1 = f(nT)$$

Therefore, the analytic Rss error may be written, for $t>nT$,

$$Rss_i = \sqrt{\frac{1}{nT}\sum_{nT=1}^{nT}|EX(nT) - f(nT)| \cdot \sum_{nT=1}^{nT}|EX(nT) - f(nT)|'\cdot\int_0^t\delta(t-nT)dt}$$

or

$$Rss_i = \sqrt{\frac{1}{nT}\sum_{nT=1}^{nT}|EX(nT) - f(nT)| \cdot |EX(nT) - f(nT)|'}$$

Interpretation of Results

The Root Error Analyzer Program has been used to evaluate four integration methods for application to a selected first-order differential system.

For this empirical study, the problem examined has consisted of one first-order linear differential system,

$$\dot{y}_i = Ay_i \quad , \quad y_i(0) = 1 \quad , \quad \text{Real } (A) < 0,$$

where analytical solution and behavior is well known,

$$y_i(t) = e^{At}$$

where the only eigenvalue is $\lambda = A$.

The analysis of the four integration methods was done at a normalized integration interval of T = 1. At each step of the solution, the following parameters were computed and printed during the execution of the % Root Error Program:

1. The sampled ideal root A.
2. The sampled approximated root A'.
3. The % normalized root error ε.

The problem was subsequently solved numerically and analytically using four different methods. The results were compared, and a match was found between the numeric and analytic transient root error.

For the Transient Root Error Program the following parameters were computed and printed at each step of the solution:

1. The sampled ideal solution, e^{AnT}.
2. The sampled approximated solution, $e^{A'nT}$.
3. The approximate numerical solution, y(n).
4. The approximate analytical solution, $y_i(n)$.
5. The numerical transient root error, Rss.
6. The analytical transient root error, Rss_i.

In solving the cubic equation of the integration formula it was found that the dominant or primary root, z_1, is the same sampled ideal solution, i.e., $z=e^{At}$. Looking at the other two roots, z_2 and z_3, we find that they are not quite complex conjugates of each other. The contribution of these secondary roots to the Rss error was almost negligible since their magnitudes were, in the most part, less than z_1, and their corresponding constants, K_2 and K_3, were one to two orders of magnitude smaller than K_1 whose magnitude is approximately equal to 1.

Summarizing we get,

$$y^*(t) = [K_1(z_1)^t + K_2(z_2)^t + K_3(z_3)^t]^*$$

where $z_1 = e^{At}$, $z_2 = e^{Bt}$ and $z_3 = e^{Ct}$

$$|z_1| > |z_3| > |z_2|$$

$$|K_1| \cong 1 \ , \ |K_1| >> |K_2| \ , \ |K_1| >> |K_3|$$

where $|K_2| << 1$ and $|K_3| << 1$.

It was also found that the magnitudes of the roots B and C were greater than one,

$$|B| > 1 \quad \text{and} \quad |C| > 1.$$

The plots of the results for the normalized constant root errors are provided in Figures 5 through 7 where $A'T \triangleq Y$.

In addition to providing a basis for relative comparison of different integration methods, these plots also allow

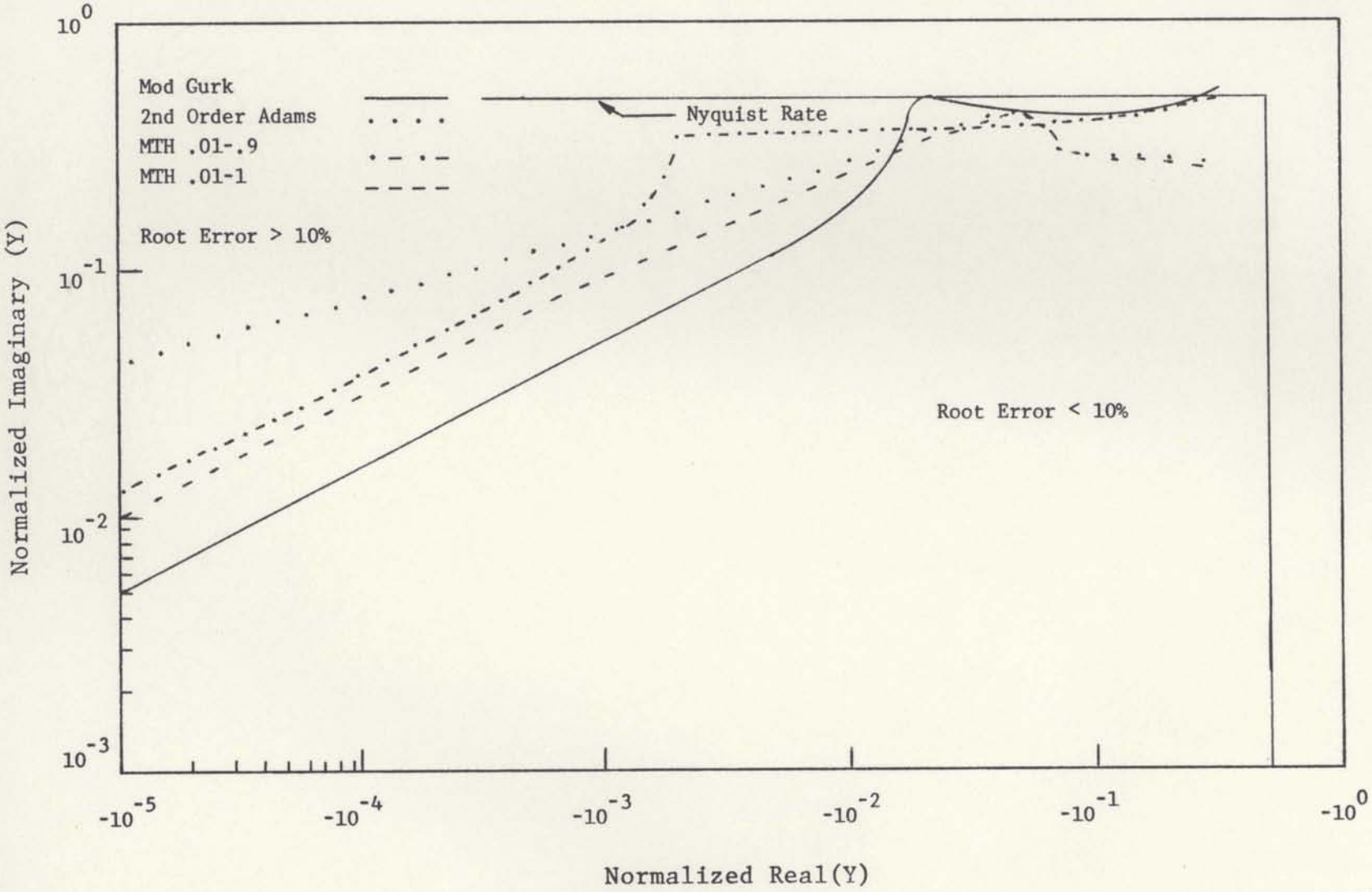

Figure 5. Comparison of 10% Root Error of the Integration Methods

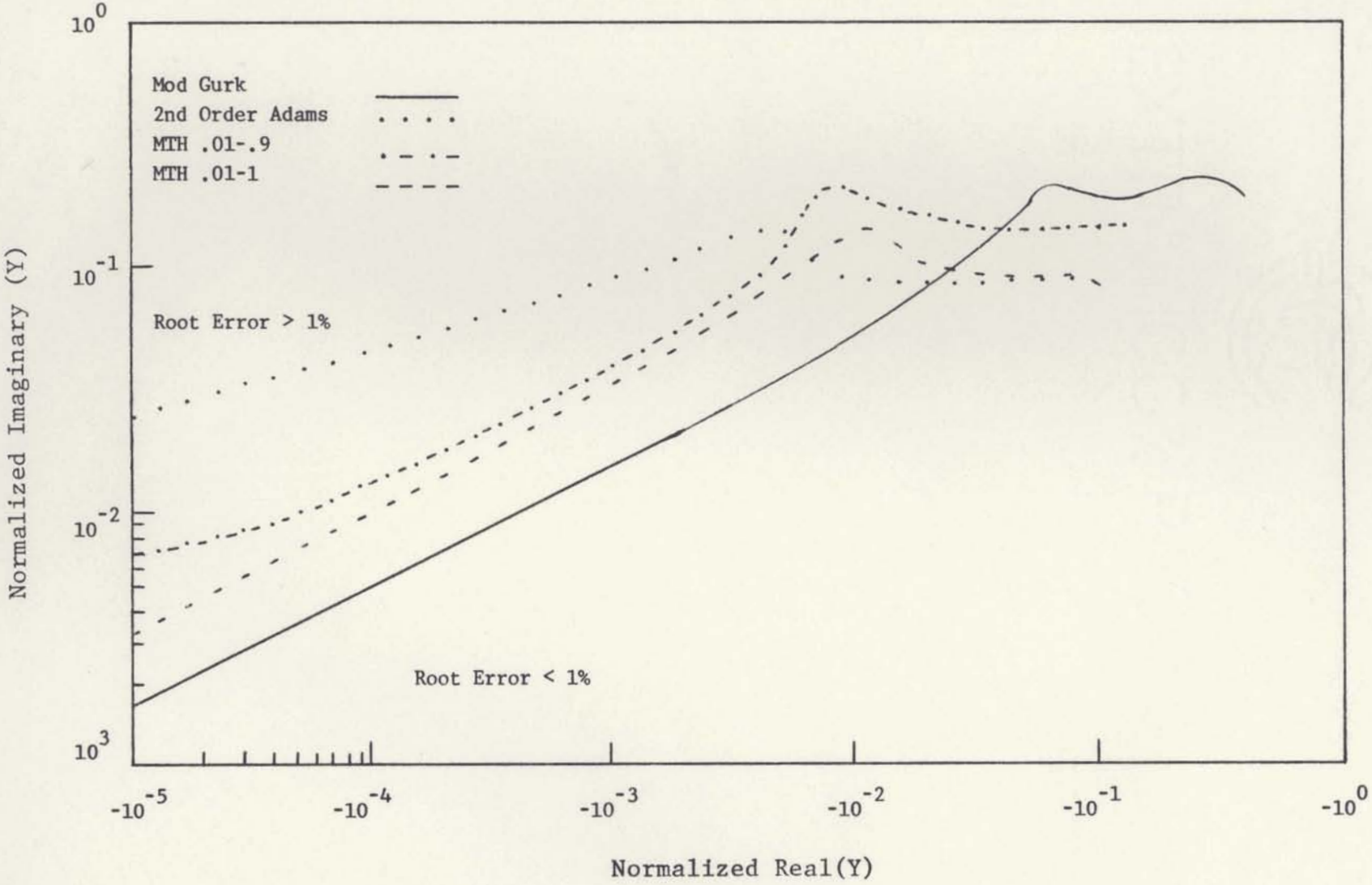

Figure 6. Comparison of 1.0% Root Error of the Integration Methods

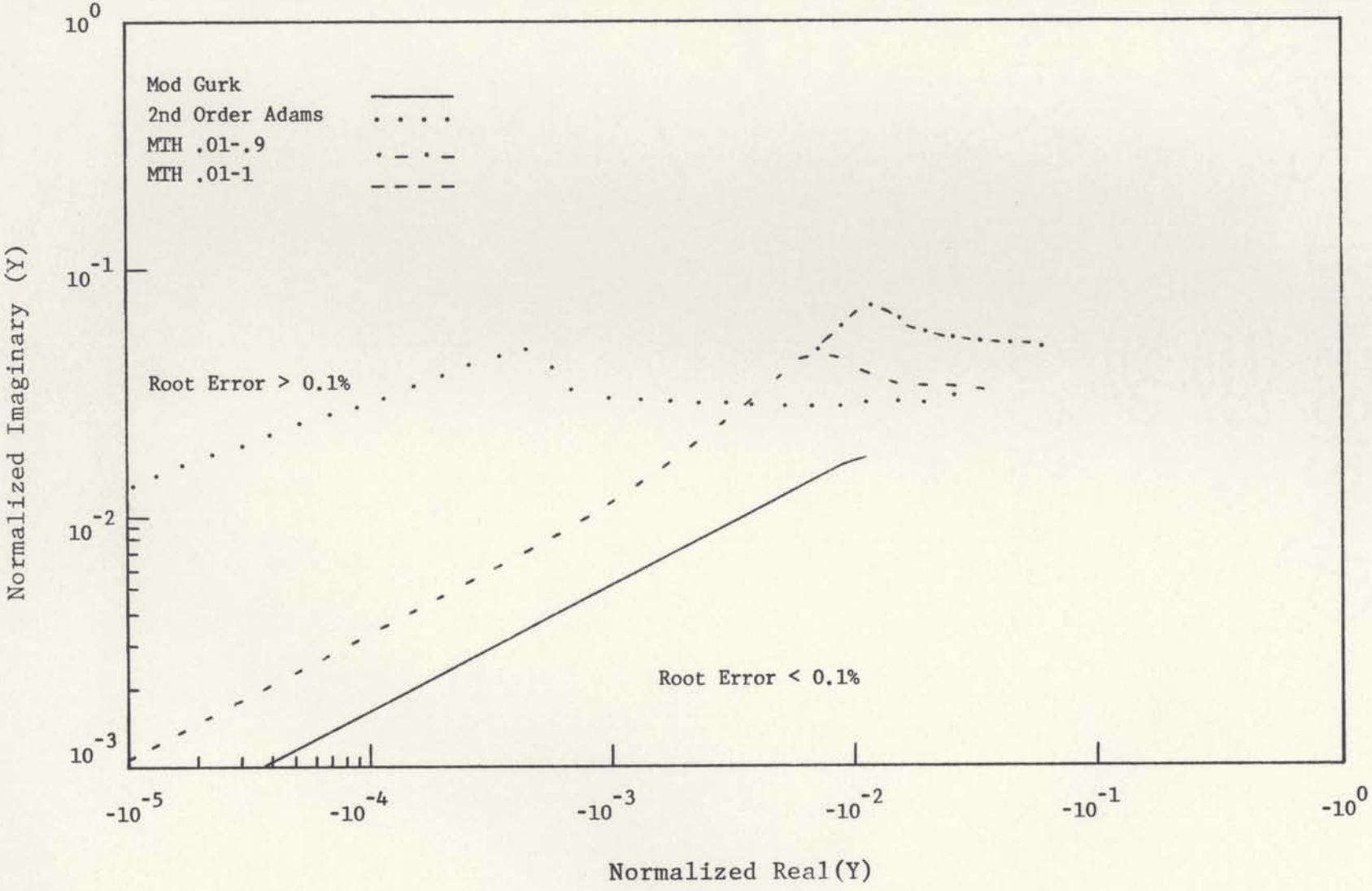

Figure 7. Comparison of 0.1% Root Error of the Integration Methods

TABLE II

FIRST PEAK POINT OF THE CONSTANT ROOT ERROR OF THE INTEGRATION METHODS

Method	10% Root [(Re(Y), Im(Y)]	1% Root [Re(Y), Im(Y)]	0.1% Root [Re(Y), Im(Y)]
MOD GURK	root #29 $(-1.7887\times10^{-2}, 5.0428\times10^{-1})$	root #33 $(-5.7419\times10^{-2}, 2.2752\times10^{-1})$	root #28 $(-1.4856\times10^{-2}, 1.9451\times10^{-2})$
2nd ORDER ADAMS	root #32 $(-4.4967\times10^{-2}, 4.4943\times10^{-1})$	root #23 $(-3.8567\times10^{-3}, 1.4744\times10^{-1})$	root #15 $(-4.4129\times10^{-4}, 4.8168\times10^{-2})$
MTH .01-.9	root #21* $(-2.0288\times10^{-3}, 3.5604\times10^{-1})$	root #25 $(-6.6117\times10^{-3}, 2.2142\times10^{-1})$	root #27 $(-1.1333\times10^{-2}, 7.1032\times10^{-2})$
MTH .01-1	root #32 $(-4.5740\times10^{-2}, 4.3439\times10^{-1})$	root #27 $(-1.1349\times10^{-2}, 1.5523\times10^{-1})$	root #25 $(-6.6035\times10^{-3}, 4.6447\times10^{-2})$

* Root #21 of MTH .01-.9 is the root location for a change in slope direction and where the curve levels off.

us to search for and recognize comparable characteristics of the candidate.

For example, the root error curve of the 2nd Order Adams method is one order of magnitude greater than the Mod Gurk method when Re(Y)→0 while the root error curves for the MTH .01-.9 and the MTH .01-1 are fairly close to each other.

In addition, Mod Gurk appears to have the largest straight line root error response of all four methods and the MTH .01-1 Method may imply similar behavior as Mod Gurk since it appears to be running in parallel to Mod Gurk during the first dozen root locations.

The peak behavior of the constant root error response as a function of the normalized iteration root is summarized in Table II.

The computed constant roots from the % Root Error Program were used for determining the numerical and analytical transient root error values.

The numerical Rss and analytic Rss_i were compared and they matched. All responses were identical with the numeric response making evident that the impulse response of the analytic solution is a sampled version of the continuous one. Then the Rss errors were compared against the constant root errors and it was found that both functions had the first peak response at the same root location although their errors did not match. This can be checked by comparing the

TABLE III

MAXIMUM TRANSIENT ROOT ERROR RESPONSE OF THE INTEGRATION METHODS

Method	Constant 10% root			Constant 1% root			Constant 0.1% root		
	Root # [Re(Y), Im(Y)]	% Rss	t	Root # [Re(Y), Im(Y)]	% Rss	t	Root # [Re(Y), Im(Y)]	%Rss	t
MOD GURK	29 $(-1.79\text{x}10^{-2}, 5.04\text{x}10^{-1})$	47.74	73	33 $(-5.74\text{x}10^{-2}, 2.28\text{x}10^{-1})$	1.20	30	27 $(-1.13\text{x}10^{-2}, 1.89\text{x}10^{-2})$	.046	149
2ND ORDER ADAMS	32* $(-4.50\text{x}10^{-2}, 4.49\text{x}10^{-1})$	31.98	35	23* $(-3.86\text{x}10^{-3}, 1.47\text{x}10^{-1})$	10.98	434	12 $(-1.96\text{x}10^{-4}, 3.78\text{x}10^{-2})$	3.58	8617
MTH .01-.9	21 $(-2.03\text{x}10^{-3}, 3.56\text{x}10^{-1})$	84.34	493	25 $(-6.61\text{x}10^{-3}, 2.21\text{x}10^{-1})$	9.76	253	25 $(-6.60\text{x}10^{-3}, 4.65\text{x}10^{-2})$	0.10	255
MTH .01-1	32 $(-4.57\text{x}10^{-2}, 4.34\text{x}10^{-1})$	28.42	35	27 $(-1.14\text{x}10^{-2}, 1.55\text{x}10^{-1})$	4.43	149	27 $(-1.13\text{x}10^{-2}, 4.02\text{x}10^{-2})$	0.19	1450

* This is the root location where the Rss error starts to drop very rapidly.

results from Tables II and III.

The Rss error transient responses are plotted in Figures 8 through 10 in terms of the % Rss error and the upper time limit, t.

From the results we may conclude that:

1. The Mod Gurk method is the only one of the four methods tested that provides the smallest constant transient root error although it is off by a factor of three from the percent root error.

2. The Second Order Adams method is the only one of the four methods tested that provides the largest constant transient root error and is off by a factor greater than three from the percent root error.

3. The MTH .01-.9 and the MTH .01-1 methods show a constantly changing transient root error with no obvious relation to the percent root error.

After careful analysis of the root error data and the Rss data and their respective plots we can conclude that they do not reveal any significant correlation between the calculated transient root error and the percent root error. Therefore these data and plots are not sufficient information to evaluate and identify the accuracy of an integration formula.

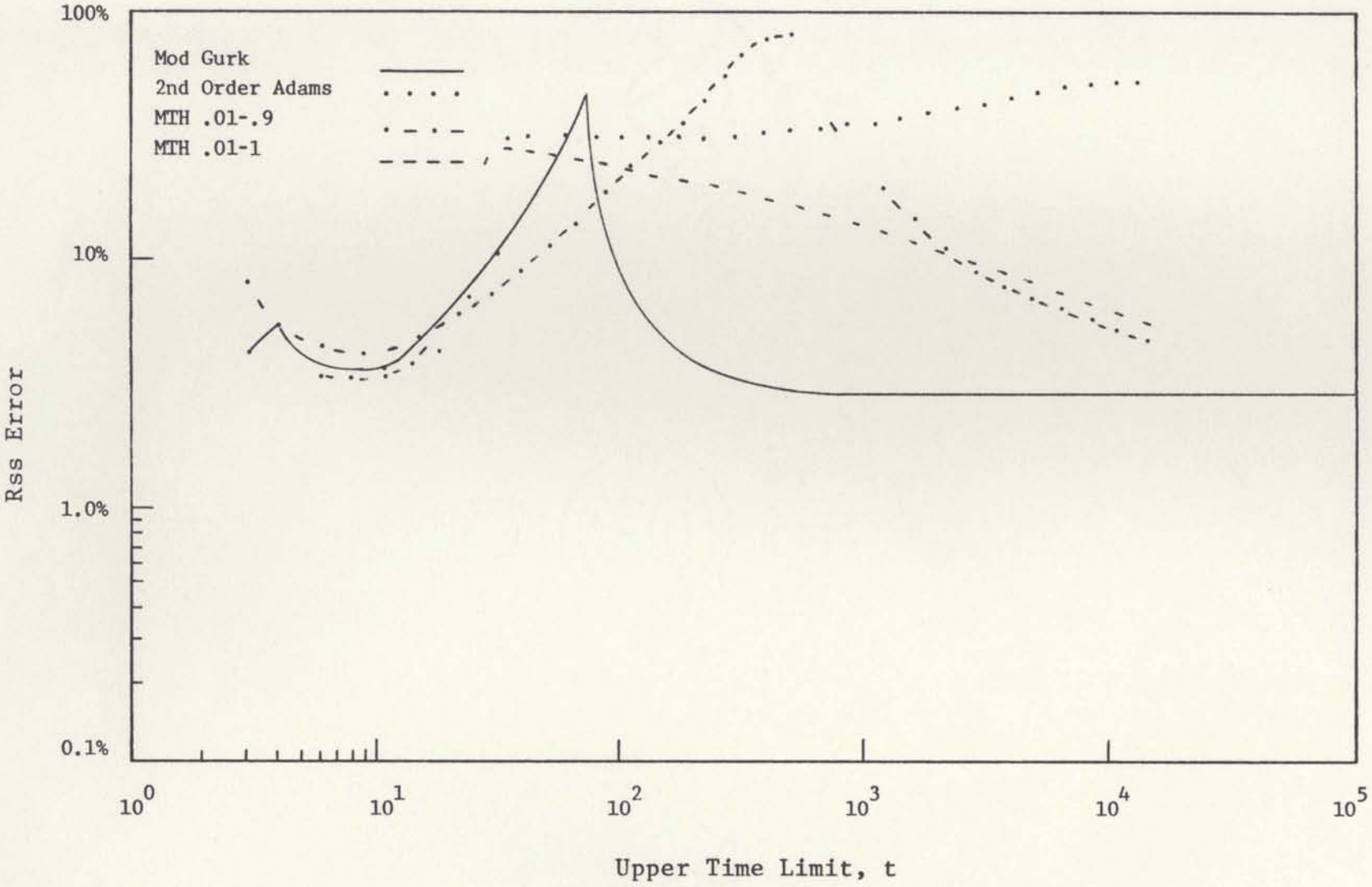

Figure 8. Comparison of the Transient Root Error Response Due to a 10% Root Error

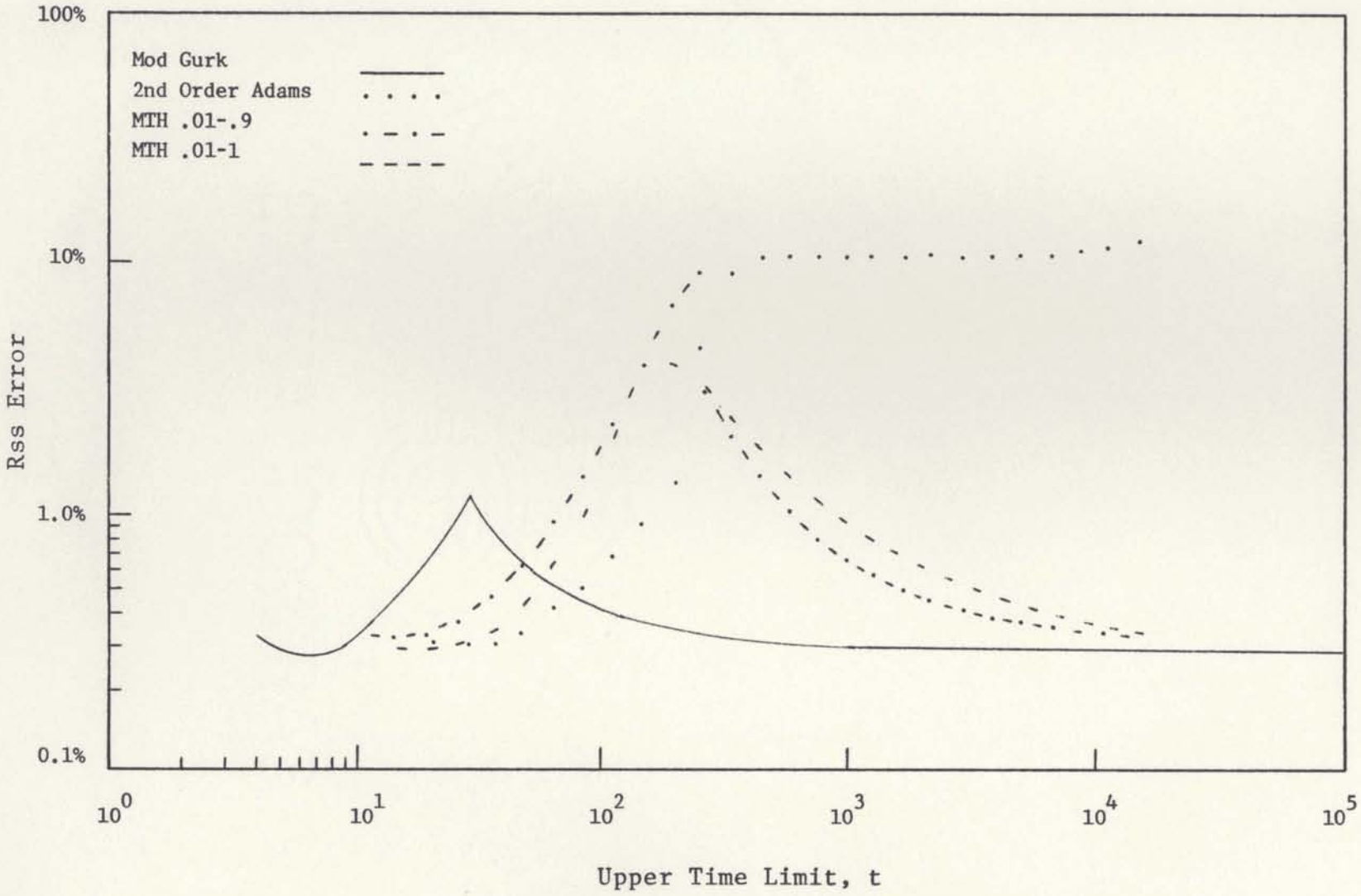

Figure 9. Comparison of the Transient Root Error Response Due to a 1% Root Error

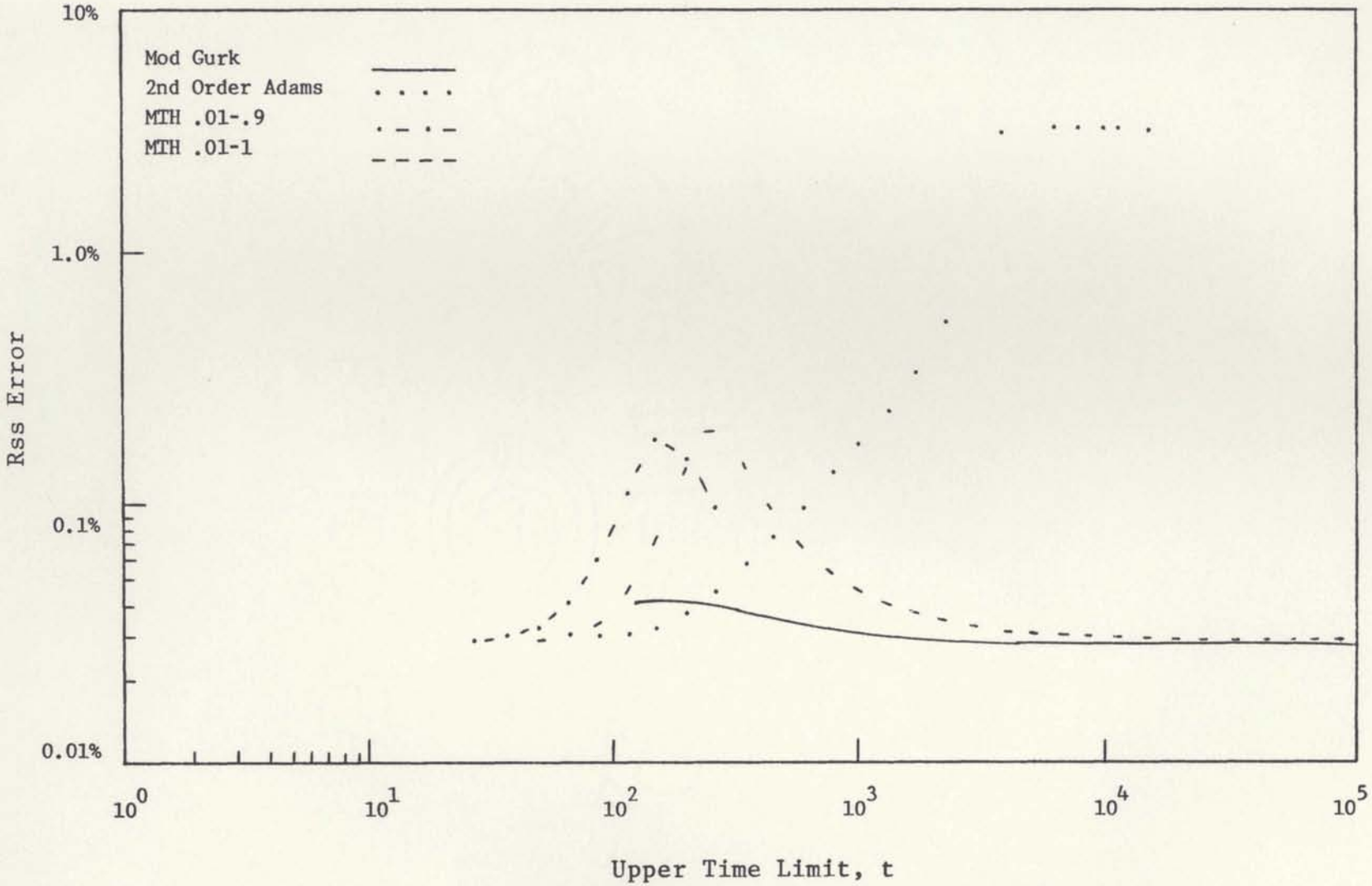

Figure 10. Comparison of the Transient Root Error Response Due to a 0.1% Root Error

SECTION VI

CONCLUSIONS

The purpose of this thesis was to illustrate and test a measure of accuracy for evaluating numerical integration techniques. Using this measure of accuracy, we have developed an evaluational algorithm applicable to numerical integration schemes. Its use permitted us to compare four different integration methods on a firm numerical and analytical basis for the specific application of a first-order differential system. The four methods tested were chosen based upon their stability range and their past performance in flight simulation work.

The technique for comparing roots of continuous integrators to roots of sampled integrators was investigated.

The development of an integration and of an iteration formula which followed standard mathematical procedures is presented.

The computer program REAP, implemented on well founded theoretical bases, was first developed for the HP-41CV handheld calculator system and later implemented for the Harris 800 computer.

The data generated by the Root Error Analyzer Program for each integration method permits one to confidently

calculate the percent root accuracy characteristics to be expected. Most important, the program allows us to find the numeric and analytic transient root error of each method.

This study has not been highly rewarding although it has produced some noteworthy results, the most important of which is that the root matching technique illustrated and evaluated in this thesis does not prove to be a very effective tool for comparing the accuracy of integration methods for application to first-order differential systems; neither is it suited for an application where a method for the development of the best integration formula is required. Although correlation of the numeric transient root error versus the analytic transient root error showed excellent computation accuracy of the computer program REAP, the transient root error failed to match the percent root error.

Admittedly, this technique has not been completely applied in its entirety and to use it in any application involving other than first-order differential problems will require further investigation.

SECTION VII

RECOMMENDATIONS

The successful creation of the REAP program has demonstrated that the art of seeking a measure of accuracy for integration methods can be mechanized to a greater degree than was formerly thought possible. However, it should be obvious to the educated reader that, considering the total field of continuous and discrete systems, there is more work to be done. For example, an interesting area of investigation would be to let A be a complex matrix with 1000:1 root variations, where the matrix is rotated so that coefficients change, but the root locations stay fixed, then use the various integration formulas and compare errors with the predicted root accuracies.

It is anticipated that use of this program will enable many integration methods used in simulation problems to be accommodated with assured (greater) accuracy. The present study was concerned with fixed integration intervals. But similar computer programs for the analysis of continuous systems -- digital simulations with variable integration intervals -- would be welcome additions, as well as nonlinear and/or variable coefficient differential equations. Similar empirical results can be conducted for second-order

differential systems. Finally, it is the hope of the author that this analysis will stimulate more investigation in the field of the evaluation of numerical integration techniques.

APPENDIX A

STABILITY RANGE PROGRAM

```
C
C       STABILITY RANGE PROGRAM
C
        DIMENSION A(3,4), B(3,4), R(4,60)
        COMPLEX Z, (4,60), W1, W2, Z3, CMPLX
C
C       TEST FUNCTION 1
C       MOD GURK METHOD
C
        A(1,1) = 1.146208
        B(1,1) = 1.641586
        A(2,1) =-0.201087
        B(2,1) =-1.008013
        A(3,1) = 0.054879
        B(3,1) = 0.275097
C
C       TEST FUNCTION 2
C       MTH .01 - .9 METHOD
C
        A(1,2) = 0.614318
        B(1,2) = 1.707964
        A(2,2) = 0.0097546
        B(2,2) =-0.161177
        A(3,2) = 0.375927
        B(3,2) = 0.214822
C
C       TEST FUNCTION 3
C       SECOND ORDER ADAMS METHOD
C
        A(1,3) = 1.0
        B(1,3) = 1.5
        A(2,3) = 0.0
        B(2,3) =-0.5
        A(3,3) = 0.0
        B(3,3) = 0.0
C
C       TEST FUNCTION 4
C       MTH .01 - 1 METHOD
C
        A(1,4) = 1.006667
        B(1,4) = 1.488056
        A(2,4) =-0.006667
```

```
      B(2,4) =0.494722
      A(3,4) = 0.0
      B(3,4) = 0.0
C
      XX = 3.141592654/30
      X = 0.0
C
      DO 200 1 = 1,60
      X = X + XX
      Z = CMPLX(COS(X),SIN(X))
      Z3 = Z * Z * Z
C
      DO 200 J = 1,4
      W1 = B(1,J)
      W2 = A(1,J)
C
      DO 150 N = 2,3
      W1 = W1 * Z + B(N,J)
      W2 = W2 * Z + A(N,J)
150   CONTINUE
C
      W2 = Z3 - W2
      Y(J,I) = W2/W1
      R(J,I) = CABS(Y(J,I))
200   CONTINUE
C
      DO 300 K = 1,4
      WRITE(6,1) K, (A(I,K), I = 1,3), (B(I,K), I =1,3)
      WRITE(6,4)
      WRITE(6,2)
      WRITE(6,3) (J, Y(K,J),R(K,J),J=1,300
C
      WRITE(6,1) K, (A(1,K), I = 1,3), (B(I,K), I =1,3)
      WRITE(6,4)
      WRITE(6,2)
      WRITE(6,3) (J, Y(K,J),R(K,J),J=31,60)
300   CONTINUE
C
      WRITE(6,1) K, (A(1,K), I = 1,3), (B(I,K), I =1,3)
C
1      FORMAT('1',' ',/,' ',/,' ',/,' ',/,' ',/,' ',/,
     *16X,'TEST FUNCTION ',12,/,' ',/,16X,'A1=',F10.7,
     *2X,'A2=',F10.7,2X,'A3=',F10.7,/,' ',/,16X
     *'B1=',F10.7,2X,'B2=',F10.7,2X,'B3=',F10.7,/)
2      FORMAT(21X,'REAL(Y)',8X,'IMAG(Y)',7X,'MAG(Y)',/)
3      FORMAT(1P,11X,I3,3F15.4,/)
4      FORMAT(' ',/,' ',/)
       STOP
       END
```

APPENDIX B

ROOT ERROR ANALYZER PROGRAM

```
      REAL RI,RX
      COMPLEX EX0,Y0,XC0,YL0
      COMPLEX AF0,CF1,CF2,CF3,CF4,Y1,Y2,Y3
      COMPLEX CK11,CK12,CK13,CK14,CK15,CK16,CD,CD1,CD2
      COMPLEX CD3,CD4,CD5,CD6,CK21,CK22,CK23,CK24,CK25
      COMPLEX CK26,CK31,CK32,CK33,CK34,CK35,CK36,C(14)
      COMPLEX D(14),G(14),P(14),Q(14),R(14),S1,S2,S3,S4
      COMPLEX S5,S6,S7,S8,SUM1
C
      COMPLEX S9,S10,S11,S12,S13,SUM2,S14,S15,S16,S17
      COMPLEX CP,CQ,CR,CA,CB,CAA,BA,BB,CMAG,CPH,RTA1
      COMPLEX RTA2,RTA3,RTB1,RTB2,RTB3,ROOT1,ROOT2,ROOT3
      COMPLEX ZZ1,ZZ2,ZZ3,CNK1,CDK1,CK1,CNK2,CDK2,CK2
      COMPLEX CNK3,CDK3,CK3,CNK4,CDK4,CK4,CO,CMM,CN,CCP,
      COMPLEX CCQ,CCR
C
      REAL XX,E,EE,XINC,XNEW,YNEW,RSSFLG
      INTEGERS IYY,IXX,IX,X,I,J,K,TA,TY,T,U
      COMPLEX FAT, ALFAT,NAT,XX0,XX1,A(3,4), B(3,4),Z,Z2
      COMPLEX Z3,W1,W2,Y,CMPLX,W,RZ,Z1,X0,X1,X2,X3,F1
      COMPLEX F2,F3,EX,TMP,XC,AT,F4
C
C
C NUMERICAL INTEGRATION METHODS
C
C
C MOD GURK - O33
C
      A(1,1)=(1.146208,0.0)
      A(2,1)=(-0.201087,0.0)
      A(3,1)=(0.054879,0.0)
      B(1,1)=(1.641586,0.0)
      B(2,1)=(-1.008013,0.0)
      B(3,1)=(0.275097,0.0)
C
C
C MTH .01-.9
C
      A(1,2) =(.614318, 0.0)
      A(2,2) =(.0097546, 0.0)
      A(3,2) =(.375927, 0.0)
      B(1,2) =(1.707964, 0.0)
```

```
      B(2,2) =(-.161177, 0.0)
      B(3,2) =(.214822, 0.0)
C
C
C SECOND ORDER ADAMS - O12
C
      A(1,3) = (1.0, 0.0)
      A(2,3) =(0.0, 0.0)
      A(3,3) =(0.0, 0.0)
      B(1,3) =(1.5, 0.0)
      B(2,3) =(-.5, 0.0)
      B(3,3) =(0.0, 0.0)
C
C
C MTH .01-1
C
      A(1,4) =(1.006667, 0.0)
      A(2,4) =(-.006667, 0.0)
      A(3,4) =(0.0 , 0.0)
      B(1,4) =(1.488056, 0.0)
      B(2,4) =(-.494722, 0.0)
      B(3,4) =(0.0 , 0.0)
C
C
C CONSTANT ROOT ERROR PROGRAM
C
C
C THE BASIC METHOD TO THIS APPROACH IS TO PICK XNEW,
C (REAL(AT), AND VARY YNEW, (W), WHICH IS THE FREQUENCY
C TERM OR IMAG(AT) OF THE DESIRED SOLUTION UNTIL THE
C RELATIVE ERROR APPROACHES THE ACCURACY BEING SOUGHT.
C
C
C RSSFLG=0.0  COMPUTE ROOT ERROR ONLY
C RSSFLG=1.0  COMPUTE RSS ERROR ONLY
      RSSFLG=1.0
07    CONTINUE
C
C 'EE' REPRESENTS THE % ACCURACY DESIRED
       EE=10.0
C
C
C
C
C
C 'K' REPRESENTS THE NUMBER OF THE INTEGRATION FORMULA
C TO BE EVALUATED.
24    K=1
      IF(K,EQ.4) GO TO 05
```

```
      IF(K.EQ.3) GO TO 03
      IF(K.EQ.2) GO TO 01
      WRITE(6,21) EE
      WRITE(6,29)
C
C EXACT STARTING POINT FOR MOD GURK
      START= 1.66185D-03
      GO TO 18
C
01    WRITE(6,02) EE
      WRITE(6,29)
C EXACT STARTING POINT FOR MTH .01-.9
      START= 4.222919587D-03
      GO TO 18
C
03    WRITE(6,04) EE
      WRITE(6,29)
C EXACT STARTING POINT FOR SECOND ORDER ADAMS
      START=1.41843997D-02
      GO TO 18
C
05    WRITE(6,06) EE
      WRITE(6,29)
C EXACT STARTING POINT FOR MTH .01-1
      START= 3.389544845D-03
C
C 'E' REPRESENTS THE % ROOT ERROR BEING INVESTIGATED
18    E=EE/100.0
      IF(RSSFLG.EQ. 0.0) GO TO 08
      GO TO 09
08    WRITE(6,80)
09    CONTINUE
C
C
C SET UP REAL AXIS INCREMENT
      XINC=EXP(ALOG(5.0E+04)/40.0)
      XDOUB=XINC
      XX2 = SQRT(XDOUB)
C
C DESIRED ROOTS
C SET UP FIRST VALUES OF XNEW AND YNEW
      XNEW = (-1.0E-05)/XINC
      YNEW=(START)*SQRT(EE)/XX2
C
C
C % ERROR ITERATION
      DO 210 I=1,41
C
C ITERATES THE STARTING POINT
```

```
C
      XNEW=XNEW*XINC
      YNEW=YNEW*XX2
C
C
C NUMBER OF TRIES COUNTER
C
      IYY=0
      IX=0
C
C
C
C
C
C REAL PART OF DESIRED ROOT
C
      RZ=CMPLX(EXP(XNEW),0.0)
C
C IMAGINARY PART (FREQUENCY TERM) OF DESIRED ROOT
C
34    W=CMPLX(COS(YNEW), SIN(YNEW))
      Z=RZ*W
      Z2=Z*Z
      Z3=Z2*Z
      W1=Z3-A(1,K)*Z2-A(2,K)*Z-A(3-K)
      W2=B(1,K)*Z2+B(2,K)*Z+B(3,K)
C
C ITERATION FORMULA
C
C Y = (1-(A1*Z**(-1)+A2*Z**(-2)+A3*Z**(-3)))/
C (B1*Z**(-1)+B2*Z**(-2)+B3*Z**(-3))
      Y=(W1)/(W2)
C Y=A', ITERATED OR EXACT ROOT
C THE PERCENT OF ROOT CHANGE
C CALCULATE THE PERCENT INCREASE OR DECREASE FROM THE
C EXACT ROOT TO THE DESIRED ROOT
C
      PCRY=((REAL(Y)-XNEW)/XNEW)*100.0
      PCIY=((AIMAG(Y)-YNEW)/YNEW)*100.0
C
C MAGNITUDE OF THE PERCENT ROOT CHANGE
C PERCENT CHANGE OF Y BOUNDS
C (.94)*(E) < %CHY < (1.06)*(E)
C
      PCY = SQRT(PCRY*PCRY+PCIY*PCIY)
C
C FRACTION OF ERROR
C 'ER' REFERS TO THE DEVIATION OF THE IMAGINARY PART
C OF THE CALCULATED ROOT FROM YNEW THE IMAGINARY PART
```

```
C OF THE DESIRED ROOT.
C A VALUE GREATER THAN 1.06 INDICATES YNEW, IS ABOVE
C THE ORIGINAL BOUND SET FOR THAT ROOT.
C A VALUE LESS THAN 0.94 MEANS YNEW IS BELOW THE
C ORIGINAL
C BOUND SET FOR THAT ROOT.
C
C    RELATIVE ERROR TOLERANCE
C LOWER AND UPPER BOUNDS OF 'ER'
C 0.94 < ER < 1.06
C
     ER=EE/PCY
C
C    ABSOLUTE ERROR TOLERANCE
     ERR = ABS(ER - 1.0)
C
C ITERATION COUNTER
     IYY=IYY+1
C
C RESTRICT ABSOLUTE ERROR TOLERANCE TO BE LESS THAN 0.06
C TO AVOID LIMITING PRECISION DIFFICULTIES ARISING FROM
C IMPOSSIBLE ACCURACY REQUESTS
C ERR=(ER - 1) ≦ 0.06
     IF(ERR.LT.0.06) GO TO 180
C
C ABSOLUTE ERROR TOLERANCE TOO LARGE
     IF(IX.EQ.1)GO TO 20
     YMAX=YNEW
     YMIN=YNEW
C
C MAKE YNEW SMALLER
66   YNEW=ER*YNEW
     IYY=IYY+1
     IX=1
     IF(ER.GE.1.0) GO TO 31
          YMIN=YNEW
100  CONTINUE
C
C
C
C
C
C TO AVOID PREMATURE UNDERFLOW IN THE IMAGINARY PART OF Y
C RESCALE THE ITERATION COUNTER IYY
     IF(IYY.EQ.7) GO TO 66
     IF(IYY.GT.14) GO TO 210
     IF((ER.GT.1.06).OR.(ER.LT.0.94)) GO TO 33
C
C GET LARGER VALUE FOR YNEW
```

```
          YNEW=(YMAX+YMIN)/2
      GO TO 34
20    IF(PCY.GT.EE)GO TO 31
C
C MAKE YNEW THE LOWER BOUND
          YMIN=YNEW
      GO TO 100
C MAKE YNEW THE UPPER BOUND
31    YMAX=YNEW
      GO TO 100
C
C GET SMALLER VALUE FOR YNEW
33    YNEW = SQRT(YMIN*YMAX)
      GO TO 34
C
200   CONTINUE
      WRITE(6,29)
      WRITE(6,27)IYY
      WRITE(6,28)XNEW,YNEW,Y,PCY,YMAX,YMIN,ER,ERR,IYY
      WRITE(6,29)
      GO TO 210
C
180   CONTINUE
      RI = FLOAT(I)
      IF(I.LT.16) GO TO TO 210
C
C
      GO TO 190
C
C
190   CONTINUE
C
C TRANSIENT ROOT ERROR
C NUMERICAL SOLUTION
C RSS CASE
C
C PROGRAM TO GENERATE LARGEST ERROR AND WHERE IT OCCURS
C LARGEST RSS ERROR AND WHEN IT OCCURS
C TO CALCULATE THE RSS ERROR FIND THE QUADRATIC MEAN VALUE
C OF THE ABSOLUTE ERROR,
C WHERE THE ABSOLUTE ERROR= INTEGRATION FORMULA - EXACT
  SOLUTION
C
C INITIALIZATION FOR RSS ERROR
C EXACT ROOTS
C
      X0 = CEXP(Y)
C ASSIGN INITIAL CONDITIONS TO VARIABLES
      F3 = CMPLX(1.0, 0.0)
```

```
      F2 = F3/X0
      F1 = F2/X0
      EX= CMPLX(1.0, 0.0)
      AMX=0.0
      YMAX=0.0
      TA=0.0
      TY=0.0
      RSS2=0.0
C
C X IS THE NUMBER OF AT'S, I.E.,  X=N(AT)
C WHERE 'AT=1.0' IMPLIES ONE PERIOD
C UPPER LIMIT OF INTEGRATION
      X=0.0
C
C FOR IMPROVED RSS ACCURACY TO ELIMINATE
C ROUNDOFF ERROR OF ADDING SMALL NUMBER TO
C LARGER NUMBER
      SS1=0.0
      SS2=0.0
      SS3=0.0
      ISS1=0.0
      ISS2=0.0
      ISS3=0.0
   99 X=X+1.0
C
      EX=EX*X0
C
C INTEGRATION FORMULA
C F4 REPRESENTS THE RESULT OF ESTIMATING THE EXACT
C ROOT, A'T= YT
C VIA F4, THE INTEGRATION FORMULA.
C
C INTEGRATION FORMULA
C
C Z-TRANSFORM TRANSFER FUNCTION OF
C THE DIFFERENCE EQUATION
      CF1 = (A(1,K) + B(1,K)*Y)*F3
      CF2 = (A(2,K) + B(2,K)*Y)*F2
      CF3 = (A(3,K) + B(3,K)*Y)*F1
      F4 = CF1 + CF2 + CF3
C
C USED OLD VALUE FOR NEXT ITERATION
C SET UP STORAGE FOR VARIABLES
      F1 = F2
      F2 = F3
      F3 = F4
C
C
C THE ERROR DEFINED BY XC=EX-F4, REPRESENTS THE
```

```
C  DIFFERENCE
C BETWEEN EXACT SOLUTION AND FORMULA
      XC = EX - F4
C
C YL IS THE SQUARE OF THE DIFFERENCE
      YL=REAL(XC)**2+AIMAG(XC)**2
C
C UNDERFLOW LIMIT
      IF(YL . LT.0.0) GO TO 38
C
C OVERFLOW LIMIT
      IF(YL.GT.1.0D+20) GO TO 38
C
C SUMMATION OF ERRORS
      SS1=SS1+YL
      ISS1=ISS1+1
      IF(ISS1.NE.100)GO TO 43
      ISS1=0.0
      ISS2=ISS2+1
      SS2=SS2+SS1
      SS1=0.0
      IF(ISS2.NE.100)GO TO 43
      ISS2=0.0
      ISS3=ISS3+1
      SS3=SS3+SS2
      SS2=0.0
43    CONTINUE
C
C TEST LARGEST TRUE MAXIMUM ABSOLUTE ERROR
      IF(YMAX.LT.YL)GO TO 11
C
C RSS2 IS THE ARITHMETIC MEAN OF THE SQUARE OF THE
C DIFFERENCE, YL
C X IS THE UPPER LIMIT OF THE INTEGRATION FORMULA
C
 12   RSS2=(SS1+SS2+SS3)/X
C
C TEST LARGEST ARITHMETIC MEAN
      IF(AMX.LT.RSS2)GO TO 13
C
C ARITHMETIC MEAN AND UPPER LIMIT TEST
14    IF(RSS2.GT.(0.5*AMX).AND.((-X)*REAL(Y).LT.1))GO TO
     99
C RSS DEFINES THE MEAN SQUARE ERROR OF THE ARITHMETIC
C MEAN OF THE
C SQUARE OF THE DIFF., YL
C RSS IS THE MEAN SQUARE DEVIATION
C FROM THE BOUNDS TAKEN OVER THE
C DIFFERENCES, YL, IN THE
```

```
C INTEGRATION FORMULA.
17    RSS = SQRT(RSS2)
C YYY IS THE SQUARE ROOT OF THE
C TRUE MAXIMUM ERROR TAKEN OVER THE DIFFERENCES, YL, IN.
C THE INT. FORMULA.
      YYY = SQRT(YMAX)
      GO TO 16
C MAXIMUM VALUE FOR X COMPONENT
C YMAX IS THE OLD YL VALUE
C TRUE MAXIMUM ABSOLUTE ERROR (LARGEST)
C 'TY' IS THE MAXIMUM VALUE FOR THE X COMPONENT
C WHEN 'YL' HAS REACHED ITS MAXIMUM VALUE, YMAX
 11   YMAX=YL
      TY=X
      GO TO 12
C LARGEST ARITHMETIC MEAN
C AMX IS THE OLD RSS2 VALUE
C 'TA' IS THE MAXIMUM VALUE FOR THE X COMPONENT
C WHEN RSS2 HAS REACHED ITS MAXIMUM VALUE , AMX
 13   AMX=RSS2
      TA=X
      GO TO 99
C
C
38    WRITE(6,39)
      WRITE(6,40)YL
      WRITE(6,41)
      GO TO 17
16    CONTINUE
C
C TRANSIENT ROOT ERROR
C ANALYTIC SOLUTION
C RSSE CASE
C COMPLEX CUBIC EQUATION
C Z**3+CPZ**2+CQZ+CR=0.0
C X**3+CAX+CB=0.0
C Z=X-CP/3
C
C THE ZEROS OR ROOTS OF THIS
C POLYNOMIAL ARE COMPUTED
C ONE AT A TIME IN ROUGHTLY
C INCREASING ORDER OF MODULUS.
C AS EACH ROOT OR ZERO IS FOUND,
C THE POLYNOMIAL IS DEFLATED
C TO ONE OF LOWER DEGREE.
C     COMPLEX POLYNOMIAL COEFFICIENTS
      AT = CMPLX(XNEW,YNEW)
      FAT = CEXP(AT)
      CP = -Y*B(1,K)-A(1,K)
```

```
      CQ = -Y*B(2,K)-A(2,K)
      CR = -Y*B(3,K)-A(3,K)
      CA = CQ-((CP)**2)/(3.0,0.0)
      CB = (CP**3)*(2.9,0.0)/(27.0,0.0)-(CP*CQ)/(3.0,0.0)
           +CR
      CAA = CSQRT((CB**2)/(4.0,0.0) + (CA**3)/(27.0,0.0))
      BA = (-CB/(2.0, 0.0) + CAA)
      NDEG = 3
      SAV = 2*(3.141592654/NDEG)
C
C A ROOTS
C RECTANGULAR TO POLAR
      CM = CABS(BA)
      CT = ATAN2(AIMAG(BA),REAL(BA))
      CM = CM**(1.0/NDEG)
      CMAG = CMPLX(CM, 0.0)
      CT = CT/NDEG
      CPH = CMPLX(COS(CT), SIN(CT))
C POLAR TO RECTANGULAR
      RTA1 = CMAG*CPH
      CM = CABS(RTA1)
      CT = ATAN2(AIMAG(RTA1),REAL(RTA1))
      CT = CT + SAV
      CMAG = CMPLX(CM,0.0)
      CPH = CMPLX(COS(CT),SIN(CT))
      RTA2 = CMAG*CPH
      CM = CABS(RTA2)
      CT = ATAN2(AIMAG(RTA2),REAL(RTA2))
      CT = CT + SAV
      CMAG = CMPLX(CM,0.0)
      CPH = CMPLX(COS(CT),SIN(CT))
      RTA3 = CMAG*CPH
C
C B ROOTS
      BB = (-CB/(2.0, 0.0) - CAA)
      CM = CABS(BB)
      CT = ATAN2(AIMAG(BB),REAL(BB))
      CM = CM**(1.0/NDEG)
      CT = CT/NDEG
      CMAG = CMPLX(CM,0.0)
      CPH = CMPLX(COS(CT),SIN(CT))
      RTB1 = CMAG*CPH
      CM = CABS(RTB1)
      CT= ATAN2(AIMAG(RTB1),REAL(RTB1))
      CT = CT + SAV
      CMAG = CMPLX(CM,0.0)
      CPH = CMPLX(COS(CT),SIN(CT))
      RTB2 = CMAG*CPH
      CM = CABS(RTB2)
```

```
      CT = ATAN2(AIMAG(RTB2),REAL(RTB2))
      CT = CT + SAV
      CMAG = CMPLX(CM,0.0)
      CPH = CMPLX(COS(CT),SIN(CT))
      RTB3 = CMAG*CPH
C TEST1
      TSTFL = 0
      ROOT1 = RTA1 + RTB1
      ROOT2 = RTA2 + RTB3
      ROOT3 = RTA3 + RTB2
60    CONTINUE
C (Z-ZZ1)(Z-ZZ2)(Z-ZZ3)=0.0
      ZZ1 = ROOT1 -CP/(3,0,0.0)
      ZZ2 = ROOT2 - CP/(3.0,0.0)
      ZZ3 = ROOT3 - CP/(3.0, 0.0)
C CHECK
      CCP = -(ZZ1+ZZ2+ZZ3)
      CCQ = (ZZ1*ZZ2)+(ZZ1*ZZ3)+(ZZ2*ZZ3)
      CCR = -(ZZ1*ZZ2*ZZ3)
      CRR = ABS(REAL(CCR) - REAL(CR))
      IF(CRR.LE.1.0E-05) GO TO 62
       IF(TSTFL.EQ.1) GO TO 61
      IF(TSTFL.EQ.2) GO TO 62
C TEST2
      ROOT1 = RTA1 + RTB3
      ROOT2 = RTA2 + RTB2
      ROOT3 = RTA3 + RTB1
      TSTFL = 1
      GO TO 60
61    CONTINUE
C TEST 3
      ROOT1 = RTA1 + RTB2
      ROOT2 - RTA2 + RTB1
      ROOT3 = RTA3 + RTB3
      TSTFL = 2
      GO TO 60
62    CONTINUE
C COEFFICIENTS
      CO = CLOG(ZZ1)
      CMM = CLOG(ZZ2)
      CN = CLOG(ZZ3)
C
C INITIAL CONDITIONS - SAME AS NUMERICAL
C SOLUTION
      X0 = CEXP(Y)
      CF3 = CMPLX(1.0, 0.0)
      CF2 = CF3/X0
      CF1 = CF2/X0
      CD1 = ZZ1*((ZZ2)**2)*((ZZ3)**3)
```

```
      CD2 = -ZZ1*((ZZ2)**3)*((ZZ3)**2)
      CD3 = -ZZ2*((ZZ1)**2)*((ZZ3)**3)
      CD4 = ZZ2*((ZZ1)**3)*((ZZ3)**2)
      CD5 = ZZ3*((ZZ1)**2)*((ZZ2)**3)
      CD6 =-ZZ3*((ZZ1)**3)*((ZZ2)**2)
      CD = CD1+CD2+CD3+CD4+CD5+CD6
      CK11 = CF1*((ZZ2)**2)*((ZZ3)**3)
      CK12 = -CF1*((ZZ2)**3)*((ZZ3)**2)
      CK13 = -ZZ2*CF2*((ZZ3)**3)
      CK14 = ZZ2*CF3*((ZZ3)**2)
      CK15 = ZZ3*CF2*((ZZ2)**3)
      CK16 = -ZZ3*CF3*((ZZ2)**2)
      CK1 = (CK11+CK12+CK13+CK14+CK15+CK16)/CD
      CK21 = ZZ1*CF2*((ZZ3)**3)
      CK22 = -ZZ1*CF3*((ZZ3)**2)
      CK23 = -CF1*((ZZ1)**2)*((ZZ3)**3)
      CK24 - CF1*((ZZ1)**3)*((ZZ3)**2)
      CK25 = ZZ3*((ZZ1)**2)*CF3
      CK26 = -ZZ3*((ZZ1)**3)*CF2
      CK2 = (CK21+CK22+CK23+CK24+CK25+CK26)/CD
      CK31 = ZZ1*((ZZ2)**2)*CF3
      CK32 = -ZZ1*((ZZ2)**3)*CF2
      CK33 = -ZZ2*((ZZ1)**2)*CF3
      CK34 = ZZ2*((ZZ1)**3)*CF2
      CK35 = CF1*((ZZ1)**2)*((ZZ2)**3)
      CK36 = -CF1*((ZZ1)**3)*((ZZ2)**2)
      CK3 = (CK31+CK32+CK33+CK34+CK35+CK36)/CO
C
C ANALYTIC SOLUTION
C
      AT = CMPLX(XNEW, YNEW)
      FAT = CEXP(AT)
      EX0 = FAT**X
C ASSIGN INITIAL CONDITIONS TO VARIABLES
      RX = FLOAT(X)
      X0 = CEXP(Y)
      EX = CMPLX(1.0, 0.0)
      SS1 = 0.0
      NT = 1.0
      U = 4
70    CONTINUE
C Z-TRANSFORM SAMPLED-DATA TRANSFER
C FUNCTION OF THE NUMERIC TRANSFER
C FUNCTION
      Y1 = CK1*((ZZ1)**U)
      Y2 = CK2*((ZZ2)**U)
      Y3 = CK3*((ZZ3)**U)
C OUTPUT
      AF0 = Y1 + Y2+ Y3
```

```
      EX = EX*X0
      S3 = EX - AFO
C MULTIPLY BY CONJUGATE TO MAKE IT A REAL #
      AYL = REAL(S3)**2 + AIMAG(S3)**2
      SS1 = SS1 + AYL
      U = U + 1
      NT = NT + 1.0
C UPPER LIMIT TEST
      IF(NT.GT.RX) GO TO 72
      GO TO 70
72    CONTINUE
      RSSE = SQRT(SS1/X)
      WRITE(6,44) RI,RX,EX0,F4,AF0,RSS,RSSE
      WRITE(6,26)
      IF(I.EQ.30) GO TO 230
      IF(I.EQ.60) GO TO 230
      IF(I.EQ.90) GO TO 230
      IF(I.EQ.120) GO TO 230
      IF(I.EQ.150) GO TO 230
      IF(I.EQ.180) GO TO 230
      IF(I.EQ.210) GO TO 230
      IF(I.EQ.240) GO TO 230
      GO TO 210
C
230   IF(K.EQ.4) GO TO 234
      IF(K.EQ.3) GO TO 233
      IF(K.EQ.2) GO TO 232
      WRITE(6,21) EE
231   WRITE(6,29)
      WRITE(6,22)
      GO TO 210
232   WRITE(6,02) EE
      GO TO 231
233   WRITE(6,04) EE
      GO TO 231
234   WRITE(6,06) EE
      GO TO 231
 210 CONTINUE
 220 CONTINUE
C
C
C
02    FORMAT('1',' ',/,' ',/,' ',/,' ',/,' ',/
     ,' ',/,' ',/,
     *19X,'MTH .01-.9 METHOD',10X,F5.2,'% ROOT
     ERROR',/)
04    FORMAT('1',' ',/,' ',/,' ',/,' ',/,' ',/
     ,' ',/,' ',/,
     *17X,'SECOND ORDER ADAMS METHOD', 4X,F5.2,'% ROOT
```

```
     ERROR',/)
06    FORMAT('1',' ',/,' ',/,' ',/,' ',/,' ',/
      ,' ',/,' ',/,
     *19X,'MTH .01-1 METHOD',10X,F5.2,'% ROOT ERROR',/)
21    FORMAT('1',' ',/,' ',/,' ',/,' ',/,' ',/
      ,' ',/,' ',/,
     *19X,'MOD GURK METHOD',10X,F5.2,'% ROOT ERROR',/)
22    FORMAT(18X,'REAL(AT)',5X,"IMAG(AT)',5X,'REAL(Y)',
     *6X,'IMAG(Y)',/)
23    FORMAT(9X,'X',10X,'RSS',10X,'TA',10X,'YYY',10X,
      'TY',11X,'AMX',
     &11X,'YMAX',10X,'YL',10X,'RY',10X,'IY',/)
26    FORMAT(' ',/)
27    FORMAT(3X,'ERROR',5X,'IYY=  ',I3)
28    FORMAT(11X,I3,4E13.4,/)
C
C
C
C
C
C
C
29    FORMAT(' ',/)
39    FORMAT(' ',/)
40    FORMAT(3X,'ERROR',5X,'YL=  ',1E13.4,/)
41    FORMAT(' ',/)
42    FORMAT(1P,6D13.4,/)
44    FORMAT(1P,10D13.4,/)
80    FORMAT(23X,'EXACT ROOT',13X,'CALCULATED ROOT',/)
90    FORMAT(1P,10D13.4,/)
C     DUMMY CARD
37    WRITE(6,21) EE
C
      STOP
      END
```

LIST OF REFERENCES

1. Knoop, P. A. Applications of a Simulation Analyzer Program for Deriving and Evaluating Numerical Integration Techniques. AFHRL-TR-68-9, Air Force Human Resources Laboratory, Wright-Patterson Air Force Base, Ohio, March 1969.

2. Nigro, B. J. Study of Numerical Integration Techniques for Real-Time Digital Flight Simulation. AMRL-TR-67-4, Aerospace Medical Research Laboratories, Wright-Patterson Air Force Base, Ohio, March 1967.

3. Ball, J. A. Algorithms for RPN Calculators. New York: John Wiley and Sons, 1978, 135-153.

4. Forsythe, G. E.; Malcolm, M. A.; and Moler, C. B. Computer Methods for Mathematical Computations. Englewood Cliffs, New Jersey; Prentice-Hall, Inc., 1977, 84-147.

5. Kreyszig, E. Advanced Engineering Mathematics. New York: Wiley, 1979, 757-804.

6. Linz, P. Theoretical Numerical Analysis. New York: Wiley-Interscience, 1979, 71-80.

7. Scheid, F. Theory and Problems of Numerical Analysis. Schaums Outline Series. New York: McGraw-Hill Book Company, 1968, 97-124.

8. Ferziger, J. H. Numerical Methods for Engineering Applications. New York: John Wiley and Sons, 1981, 37-85.

9. Karworski, R. J. Introduction to the z Transform and Its Derivation. El Segundo, California: TRW, 1979.

10. Design of Digital Flight Trainers. Moore School of Electrical Engineering, Research Division Report No. 54-09. Philadelphia: University of Pennsylvania. September 1953.

11. Patz, B. W. "Real Time Digital Simulation of Continuous Systems." Math Modeling Class Notes, Naval Training Center, Orlando, Fla., 1975.

12. Gurk, H. M., and Rubinoff, M. "Numerical Solution of Differential Equations." Proceedings of the Eastern Joint Computer Conference. Philadelphia: American Institute of Electrical Engineers, December 1954, 54-63.

13. Nigro, J.: Woodward, A.; and Brucks, C. A Digital Computer Program for Deriving Optimum Numerical Integration Techniques for Real-Time Flight Simulation. AMRL-TR-68-4, Aerospace Medical Research Laboratories, Wright-Patterson Air Force Base, Ohio, May 1968.

14. Saucedo, R., and Schiring, E. E. Introduction to Continuous and Digital Control Systems. New York: Macmillan Company, 1968, 132-141.

BIBLIOGRAPHY

Ball, J. A. Algorithms for RPN Calculators. New York: John Wiley and Sons, 1978, 135-153.

Design of Digital Flight Trainers. Moore School of Electrical Engineering, Research Division Report No. 54-09. Philadelphia: University of Pennsylvania. September 1953.

Ferziger, J. H. Numerical Methods for Engineering Applications. New York: John Wiley and Sons, 1981, 37-85.

Forsythe, G. E.; Malcolm, M. A.; and Moler, C. B. Computer Methods for Mathematical Computations. Englewood Cliffs, New Jersey; Prentice-Hall, Inc., 1977, 84-147.

Fowler, M. E. "A New Numerical Method for Simulation." Simulation 4 (May 1965): 324-330.

Gilbert, E. G. "Some Critical Remarks on a New Numerical Method for Simulation of Dynamical Systems." Simulation 6 (February 1966): 90-91.

Gray, H. M., Jr. "Digital Computer Solution of Differential Equations in Real Time." Proceedings of the Western Joint Computer Conference. Los Angeles: American Institute of Electrical Engineers, May 1958, 87-91.

Gurk, H. M., and Rubinoff, M. "Numerical Solution of Differential Equations." Proceedings of the Eastern Joint Computer Conference. Philadelphia: American Institute of Electrical Engineers, December 1954, 54-63.

Hull, T. E., and Newbery, A. C. R. "Integration Procedures Which Minimize Propagated Errors." Journal of Society Industrial and Applied Mathematics 9 (March 1961): 31-47.

Karworski, R. J. Introduction to the z Transform and Its Derivation. El Segundo, California: TRW, 1979.

Knoop, P. A. Applications of a Simulation Analyzer Program for Deriving and Evaluating Numerical Integration Techniques. AFHRL-TR-68-9, Air Force Human Resources Laboratory, Wright-Patterson Air Force Base, Ohio, March 1969.

Kreyszig, E. Advanced Engineering Mathematics. New York: Wiley, 1979, 757-804.

Linz, P. Theorical Numerical Analysis. New York: Wiley-Interscience, 1979, 71-80.

Lipschutz, S., and Poe, A. Theory and Problems of Programming with Fortran. Schaums Outline Series. New York: McGraw-Hill Book Company, 1978, 264-287.

Nigro, B. J. Study of Numerical Integration Techniques for Real-Time Digital Flight Simulation. AMRL-TR-67-4, Aerospace Medical Research Laboratories, Wright-Patterson Air Force Base, Ohio, March 1967.

Nigro, J.; Woodward, A.; and Brucks, C. A Digital Computer Program for Deriving Optimum Numerical Integration Techniques for Real-Time Flight Simulation. AMRL-TR-68-4, Aerospace Medical Research Laboratories, Wright-Patterson Air Force Base, Ohio, May 1968.

Papian, L. E., and Ball, W. E. "The Spectrum of Numerical Integration Methods with Computed Variable Stepsize." Journal of Mathematical Analysis and Applications 31 (1970): 259-284.

Patz, B. W. "Real Time Digital Simulation of Continuous Systems." Math Modeling Class Notes, Naval Training Center, Orlando, Fla., 1975.

Saucedo, R., and Schiring, E. E. Introduction to Continuous and Digital Control Systems. New York: Macmillan Company, 1968, 132-141.

Scheid, F. Theory and Problems of Numerical Analysis. Schaums Outline Series. New York: McGraw-Hill Book Company, 1968, 97-124.