COLUMBIA UNIVERSITY

SYSTEMS RESEARCH GROUP

DEPARTMENT OF
ELECTRICAL ENGINEERING
SCHOOL OF ENGINEERING AND
APPLIED SCIENCE
NEW YORK, N.Y. 10027

Technical Report No. 140

Numerical Solution of Stiff Systems of Ordinary Differential Equations with Applications to Electronic Circuits

Jerrold Stephen Rosenbaum

1971

Vincent R. Lalli
Project Manager
Lewis Research Center
Office of Reliability
& Quality Assurance
21000 Brookpark Rd
Cleveland, OH 44135

This research was partially supported by the National Science Foundation under grant GK - 2283 and the National Aeronautics and Space Administration under grant NGR 33-008-090.

ABSTRACT

Numerical Solution of Stiff Systems of Ordinary Differential Equations with Applications to Electronic Circuits

Jerrold S. Rosenbaum

Systems of ordinary differential equations in which the magnitudes of the eigenvalues (or time constants) vary greatly are commonly called stiff. Such systems of equations arise in nuclear reactor kinetics, the flow of chemically reacting gas, dynamics, control theory, circuit analysis and other fields.

It is often the case that the solution is smooth outside one or more almost-discontinuous segments. However, during an almost-discontinuous segment, there is a rapid (sometimes almost-discontinuous) variation in the solution.

The research reported herein develops a new A-stable numerical integration technique for solving stiff systems of ordinary differential equations. The method, which is called the generalized trapezoidal rule, is a modification of the trapezoidal rule. However, the new method is computationally more efficient than the trapezoidal rule when the solution of the almost-discontinuous segments is being calculated.

The basic aim of the new numerical integration technique is to transform the original system of differential equations to a new system of equations such that the eigenvalues of

the transformed system are smaller in magnitude than the eigenvalues of the original system. Also, the ratio of the real parts of the largest to the smallest eigenvalue of the transformed system is smaller than the same ratio for the original system. A consequence of shifting the eigenvalues is that for the same accuracy, one can integrate the new system of equations with a larger time step than is possible for the original system.

Particular attention has been focused on numerically integrating the differential equations for a high frequency model of a semiconductor network. It is shown how the generalized trapexoidal rule can be used to integrate the circuit equations more efficiently than the trapezoidal rule. Also, because one has an a priori knowledge of the structure of the circuit equations and the nature of their solution, one can obtain additional computational economies when integrating the circuit equations be the generalized trapezoidal rule.

In the appendix, there is a computer program for the generalized trapexoidal rule written in PL/I.

ACKNOWLEDGEMENTS

The author wishes to express his sincere gratitude to Professor Thomas E. Stern for his valuable guidance and encouragement during the course of this research.

The aughor also wishes to express his gratitude to Professor Henry E. Meadows for his valuable guidance and inspiration during the absence of Professor T. E. Stern.

The author wishes to thank his wife, Eileen, for her patience during the course of the research, and for her typing.

This research was partially supported by the National Science Foundation under grant GK - 2283 and the National Aeronautics and Space Administration under grant NGR 33-008-009.

Table of Contents

I. Introduction

Systems of ordinary differential equations in which the magnitudes of the eigenvalues (or time constants) vary greatly are commonly called stiff. Such systems of equations arise in nuclear reactor kinetics, the flow of chemically reacting gas, dynamics, control theory, circuit analysis and other fields. [1,12,14,18,20,21]*

It is often the case that the solution is smooth outside of one or more almost-discontinuous segments [21] (or transient phases or boundary layers). However, during an almost-discontinuous phase, there is a rapid (sometimes almost-discontinuous) variation in the solution. In addition, the system of differential equations is usually asymptotically stable --i.e., all the eigenvalues of the system of equations are in the left half plane (LHP).

A similar stiffness problem arises when a partial differential equation is approximated by a large system of ordinary differential equations. By differencing one of the variables (usually a space variable), the initial value problem for the resulting system of ordinary differential equations generally has a wide spread in time constants (see section II.I).

---

* Numbers in brackets indicate references listed in Chapter VI.

For most numerical integration schemes, in order to prevent the numerical solution from becoming unstable, the maximum step size that can be used to integrate a system of equations is on the order of the smallest time constant of the system. The step size limitation necessitates taking an excessively large number of steps to obtain the solution in both the smooth and almost discontinuous segments. In particular, during the smooth regions, one would like to use step sizes that are on the order of the largest time constant since the local variation is small. But, one must still take small time steps (on the order of the smallest time constant) to prevent the numerical solution from becoming unstable.

In the case of a semiconductor switching circuit, the solution is slowly varying except at the switching "instants". However, if one uses the usual integration schemes, which are generally step length limited, the largest step size allowable throughout the entire solution is on the order of the switching time even though the solution may be almost constant between switching instants.

The purpose of the research reported herein is to develop a new numerical integration technique for stiff systems of ordinary differential equations. The method, which will be called the generalized trapezoidal rule, is a modification of the trapezoidal rule. However, the new method is computationally more efficient than the trapezoidal rule when the solution of an almost-discontinuous segment is being computed.

Many different approaches have been suggested for obtaining the numerical solution of stiff systems of ordinary differential equations. Almost all of the suggested methods require the solution of a system of implicit (usually non-linear) equations at each time step. In chapter II, there is a discussion of some of the methods used to overcome the usual step length limitations when integrating stiff systems. In addition, there is a brief discussion of methods for solving the implicit equations that are generated by the various integration schemes. We also attempt to show how the integration methods are interrelated.

Chapter III is concerned with a new method for numerical integration of stiff systems of ordinary differential equations. The method is a modification of the trapezoidal rule. The basic approach of the method is: (1) to modify the given differential equations so that they are less stiff and consequently "easier" to integrate; and (2) to use differing approaches to obtain the numerical solution during the almost-discontinuous and smooth sections of the solution. The objectives of the new method are to allow the user to take larger time steps during the almost-discontinuous segments of the solution and to do fewer iterations per step in order to solve the implicit integrating equations while, at the same time, maintaining the same or improved accuracy as compared with the trapezoidal rule.

In the research reported herein particular attention

has been focused on numerically integrating the differential equations for a high frequency model of a semiconductor network [9,18,20,21]. In Chapter IV, it is shown how the integration method of the previous chapter can be used to integrate the circuit equations and how, because of an a priori knowledge of the structure of the circuit equations and the nature of their solution, one can obtain additional economies when integrating the circuit equations.

Chapter V presents a summary of the results of the previous chapters and suggestions for further research.

II. General Problem of Stiff Systems

1. Stability

In the general theory of numerical solution of ordinary differential equations, a major concern is the stability of the numerical solution. Roughly speaking, the stability of a numerical method refers to the behavior of the difference between the actual and calculated solution as the number of steps becomes large [14]. The values of the step size, h, for which a particular integration method is stable, are a function of the characteristic roots, u_i, of the integration method and the eigenvalues, λ_i, of the Jacobian of the system of ordinary differential equations being integrated [4,10,11,12,14].

The region of the $h\lambda$-plane for which all the characteristic roots of the integration scheme are on or in the unit circle is called the region of stability. If the stability region is bounded, then the integration method is called step length limited.

To demonstrate the general type of step length limitation that can be encountered with stiff systems and the problems that the step length limitation can cause, it is useful to examine a linear system of ordinary differential equations:

$$\dot{x} = Ax, \quad x(0) = x_0 \tag{2.1}$$

where the eigenvalues of A are distinct and in the LHP.

If the real parts of some of the eigenvalues of A are very much larger in magnitude than the real parts of others, the terms corresponding to the "large" eigenvalues become negligible very quickly. This type of behavior can also arise with nonlinear systems of equations or even with a single equation (see section III.3). All that is required for a system to be stiff is that any transient be damped out quickly in relation to the steady-state solution.

The difficulty in trying to use many of the common numerical methods to integrate stiff equations is well known. A numerical method can be affected by the transient components of the solution even after the effects of those components have become negligible in the true solution. Analytically, this behavior of the numerical solution leads to a restriction on the allowable step size. When many numerical methods are applied to the linear equations (2.1), the step size restriction takes the form:

$$(2.2) \qquad |h\lambda_i| < d, \quad i = 1,2,\ldots,m$$

where h is the step size, and d is some constant which is typically about 1 to 6. If some of the $|\lambda_i|$ are large, equation (2.2) forces the numerical method to use a very small step size in order to maintain stability, even though the corresponding contributions to the true solution are negligible.

For the linear system of equations (2.1), the analytic solution is

$$x(nh) = e^{nhA}x_o \quad (2.3)$$

As a consequence, any numerical procedure must, in some way, approximate e^{hA}. Also, because all the eigenvalues of A are in the LHP, the least one should require is that

$$\lim_{n \to \infty} x_n = 0, \quad (2.4)$$

where x_n is the value of the solution at the n^{th} point in the calculation.

When one applies the forward Euler, Heun or traditional fourth-order Runge-Kutta schemes to equation (2.1), it can be shown that

$$x_n = [M(hA)]^n x_o \quad (2.5)$$

where

$$M(Z) = \begin{cases} I + Z & \text{Forward Euler} \\ I + Z + Z^2/2! & \text{Heun} \\ I + Z + Z^2/2! + Z^3/3! + Z^4/4! & \text{Runge-Kutta} \end{cases} \quad (2.6)$$

Therefore, in order to satisfy equation (2.4), one must require that

$$\|M(hA)\| < 1. \quad (2.7)$$

Since the eigenvalues of A are distinct, there exists a matrix P such that

$$PAP^{-1} = \text{diag}(\lambda_1, \ldots, \lambda_n) = \lambda \quad (2.8)$$

Consequently, equation (2.5) can be rewritten as

$$(2.9) \qquad x_n = [M(hP\lambda P^{-1})]^n x_o$$
$$= P[M(h\lambda)] P^{-1} x_o$$

and equation (2.7) can be reduces to

$$(2.10) \qquad |M(h\lambda_i)|<1 \text{ for } i=1,\ldots m.$$

One can think of equation (2.10) as requiring that the integration schemes be contraction operators on the left half $h\lambda$-plane (or have spectral radii less than 1).

In particular, for the forward Euler method, one has that

$$(2.11) \qquad x_n = x_{n-1} + hf(t_{n-1},x_{n-1})$$
$$= P(1+h\lambda)P^{-1}x_{n-1}$$

Hence,

$$(2.12) \qquad |1+h\lambda_i|<1 \text{ for } i = 1,\ldots m$$

or, if all the eigenvalues are real,

$$|h\lambda_i|<2 \ .$$

For the Heun method, the stability criterion is also $|h\lambda|<2$ and for the fourth-order Runge-Kutta method, the criterion is $|h\lambda|<2.78$ [6].

However, one is generally interested in obtaining the solution over the interval $t = [0,\mathcal{O}(\max|\lambda_j|^{-1})]$. If $\min|\lambda_i| \ll \max|\lambda_j|$, then stability considerations dictate a very small step size, $h = \mathcal{O}(\min|\lambda_j|^{-1})$, over the entire interval even though the effects of the maximal eigenvalue on the solution are negligible after the first few steps.

To demonstrate explicitly the problems caused by the

wide spread in the eigenvalues, two systems of ordinary differential equations (one of which is stiff) which possess "almost" the same solution, will be considered.

The first system is

$$(2.13)\qquad \frac{dV}{dt} = DV + C\ ,\ V(0) = \begin{bmatrix}0\\0\end{bmatrix},\ D = \begin{bmatrix}-1 & 0\\ 0 & -1\end{bmatrix} \text{ and } C = \begin{bmatrix}2\\2\end{bmatrix}$$

The solution to equations (2.13) are

$$V_1(t) = V_2(t) = 2(1-e^{-t})$$

and the stability conditions are $h < 2$ for the Euler and Heun methods and $h < 2.78$ for the Runge-Kutta methods.

If, on the other hand, one considers the system

$$(2.14)\qquad \frac{dW}{dt} = AW + C,\ W(0) = \begin{bmatrix}-0.1\\0\end{bmatrix},\ A = \begin{bmatrix}-500.5 & 499.5\\ 499.5 & -500.5\end{bmatrix},$$
$$C = \begin{bmatrix}2\\2\end{bmatrix}$$

then

$$W_1(t) = 2(1-e^{-t}) - 0.1e^{-1000t}$$
$$W_2(t) = 2(1-e^{-t}) + 0.1e^{-1000t}$$

But, for $t > 0.02$, one has that $0.1e^{-1000t} < 2.5 * 10^{-10}$ or $V \approx W$ for $t > 0.02$. However, the eigenvalues of A are -1 and -1000, which dictates that $h < 0.002$ for the Euler and Heun methods and $h < 0.00278$ for the Runge-Kutta method [6].

Now, in both systems, one wishes to determine the solution over the interval $t = [0, \mathcal{O}(1)]$. For the V equations, one would need less than 10 steps to obtain the solution

over the desired interval. But, for the W equations, one would require 1000 times as many steps to obtain the solution, and increased precision might be needed for the calculation because of the increased round-off error introduced by the very large number of steps.

From the example, one can see that the step size, h, cannot be chosen to represent the information carried by modes corresponding to the smaller eigenvalues. The step size must be chosen to avoid any spurious amplification of the modes corresponding to the larger eigenvalues of the system of equations. Thus, if an integration scheme has a bounded region of stability, one is forced to take exceedingly small time steps throughout the time interval for which a solution is desired in order to prevent instability. The instability is a result of an amplification of the modes corresponding to the larger eigenvalues.

In the numerical solution of partial differential equations, the same type of problem is encountered when one approximates a partial differential equation by a system of ordinary differential equations. For example, consider the heat equation:

$$(2.15) \qquad K \frac{\partial^2 u}{\partial x^2} = \frac{\partial u}{\partial t}, \quad u(0,t) = u(L,t) = 0.$$

One of the standard first steps in the numerical solution of equation (2.15) is to difference it in the x (or space) direction. This given the following system of ordinary differential equations:

$$(2.16) \qquad \frac{du_i}{dt} = K(\Delta x)^2(u_{i-1}-2u_i+u_{i+1}).$$

In matrix notation, equation (2.16) becomes

$$(2.17) \qquad \frac{du}{dt} = Au \text{ where } A = \begin{bmatrix} -2 & 1 & & & \\ 1 & -2 & 1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & 0 & 1 & -2 & 1 \\ & & & 1 & -2 \end{bmatrix}$$

and the eigenvalues of A are

$$(2.18) \qquad \lambda_i = -4K\,(\Delta x)^{-2}\sin\frac{i\pi}{2(K+1)}\,.$$

Hence, for Euler's of Heun's methods, the stability criterion is

$$h < (\Delta x)^2(2K)^{-1}$$

But Δx must be sufficiently small that equation (2.16) is a good approximation to the original equation (2.15). Consequently, we are restricted to using a small h with all of the above methods, or any other step length limited method, even though the modes corresponding to eigenvalues closest to the origin soon become the dominant components in the solution.

2. A-Stability

For stiff systems, the concept of stability of a numerical scheme, which was discussed in the previous section, is not adequate because stability considerations generally lead to schemes whose maximum step size is very small.

Ideally, the step size should be a function only of the rate of variation of the solution during the particular interval being calculated and the desired accuracy (rather than being a function of the global properties of the solution). Also, for stiff systems, it is not enough that the transient solutions be bounded; one needs a numerical method that insures one that all transient solutions will eventually die out. Towards these ends, the concept of A-stability was proposed by G. Dahlquist [3].

Definition: A numerical method for solving differential equations is called A-stable if all solutions tend toward zero as $n \to \infty$ when the method is applied with fixed positive h to any differential equation of the form:

$$\dot{x} = qx$$

where q is a complex constant with negative real part.

In effect, the definition requires that for all eigenvalues in the LHP, the numerical solution corresponding to those eigenvalues eventually die out regardless of the step size. As pointed out earlier, an A-stable method may be regarded as one which acts as a contraction operator for equations with eigenvalues in the LHP, although this concept is not found in the literature on numerical integration techniques.

The numerical integration of stiff systems has been

considered by many authors (see references). It is known that all explicit schemes of the linear multistep and Runge-Kutta types are step length limited and consequently not A-stable. Therefore, attention must be focused on other classes of integration schemes (usually implicit).

No explicit Runge-Kutta scheme is A-stable, because the recurrence relation it produces when applied to the test equation $\dot{x} = qx$ is $x_{n+1} = C(hq)\, x_n$ where $C(hq)$ is a truncated exponential series for e^{hq}. If q is in the LHP, then the sequence $\{x_n\}$ does not converge to zero for arbitrary positive h. In fact, for almost every initial value, x_0 , $x_n \to \pm \infty$ for almost all values of q and h.

In a paper by Ehle [6], it is shown that Butcher's fourth-order implicit Runge-Kutta scheme is A stable; but, the non-linear functional equations that must be solved at each time step are considerably more complex than those for linear multistep methods.

An indirect approach to the integration problem for stiff systems is to transform the system of differential equations into a new system (suitably modified) that is not stiff and to solve the latter system by a conventional method which is step length limited. Lawson [13] proposed what he called a "Generalized Runge-Kutta" scheme involving the Jacobian matrix and exponential shifts in the variables (see section II.3).

The basic theory of stiff linear multistep methods was developed by G. Dahlquist [3]. He showed that no explicit linear multistep method is A-stable, and he proved the remarkable theorem that for fixed h, the most accurate linear multistep method is of order two, with the trapezoidal rule having the minimum truncation error of all second order A-stable schemes. Dahlquist also pointed out that an iterative technique like Newton-Raphson iteration must be used to solve the implicit integrating equations because the method of successive substitutions is inherently step length limited (see section II.5).

More recently, Widlund [24] and Gear [8] have developed higher order multistep schemes which are, for practical purposes, not step length limited. Widlund developed third and fourth order schemes and Gear developed second through sixth order schemes. But, both authors use a "milder" form of Dahlquist's concept of A-stability. The approaches of these two authors are discussed in section II.7.

3. Generalized Runge-Kutta Processes [13]

The basic problem inherent in all implicit integration schemes is that one must solve a system of implicit non-linear equations at each step. In order to avoid some of these problems, Lawson [13] proposed to alter the stiff

system of equations so that an explicit Runge-Kutta scheme will work efficiently on the altered set of equations.

Let us consider the stiff system of equations

$$\dot{x} = f(t,x) \ , \ x(0) = x_0 \tag{2.19}$$

with a Jacobian matrix, $(\partial f/\partial x)$, and eigenvalues in the LHP. Using the transformation

$$z(t) = e^{-tA}x(t) \tag{2.20}$$

where A is a real square matrix, it follows that

$$\dot{z} = g(t,z) = e^{-tA}\{f[t,e^{tA}z(t)] - Ae^{tA}z(t)\} \ , \ z(0) = x_0 \tag{2.21}$$

where the Jacobian matrix of the new system of differential equations (2.21) is the matrix

$$\begin{aligned}\left(\frac{\partial g}{\partial z}\right) &= e^{-tA}\left(\frac{\partial f}{\partial x}\right)e^{tA} - A \\ &= e^{-tA}\left[\left(\frac{\partial f}{\partial x}\right) - A\right]e^{tA}\end{aligned} \tag{2.22}$$

If $[(\partial f/\partial x) - A]$ has small eigenvalues, one can apply one of the classical explicit Runge-Kutta schemes (which are step length limited) to the z equations and be able to use reasonable step sizes. Substituting the z equations, (2.21), into a classical Runge-Kutta scheme, it follows that

$$\begin{aligned} z_{n+1} &= z_n + h\sum_{i=1}^{m} b_i k_i \\ k_1 &= e^{-t_nA}\{f[t_n,e^{t_nA}z_n] - Ae^{t_nA}z_n\} \\ p_i &= e^{-(t_n+c_ih)A}\left[z_n + h\sum_{j=1}^{i-1} a_{ij}k_j\right] \\ k_i &= e^{-(t_n + c_ih)A}[f(t_n+c_ih,p_i) - Ap_i] \quad i = 2,\ldots,m \end{aligned} \tag{2.23}$$

where a_{ij}, b_i and c_i are the standard Runge-Kutta parameters [13].

If one sets $y_n = e^{t_n A}$ and $k_i^* = e^{(t_n+c_i h)A} k_i$, then one may rewrite equations (2.23) as

$$
\begin{aligned}
k_1^* &= f(t_n, x_n) - A x_n \\
p_i^* &= e^{c_i hA} x_n + h \sum_{j=1}^{i-1} a_{ij} e^{(c_i - c_j)hA} k_j^* \\
k_i^* &= f(t_n + c_i h, p_i^*) - A p_i^* \\
x_{n+1} &= e^{hA} x_n + h \sum_{i=1}^{m} b_i e^{(1-c_i)hA} k_i^*
\end{aligned}
\tag{2.24}
$$

Now, if the c_i are equally spaced on the interval [0,1], as is the case with several Runge-Kutta schemes, the computation of the exponentials is greatly simplified. Also, one chooses A to be the Jacobian at some particular point(s) and uses the diagonal Pade approximations to calculate the exponential functions in order to maintain stability.

Using a generalized Runge-Kutta scheme based on the usual fourth order Runge-Kutta scheme, Lawson was able to increase the step size by a factor of between 8 and 32 (depending upon the particular test example)as compared to the usual Runge-Kutta scheme, without increasing the error in the numerical solution.

The author did not clearly specify how often to change

A and how to select h.

4. Implicit Runge-Kutta Processes [6]

Rather than altering the equations as does Lawson, Ehle [6] proposed that the implicit Runge-Kutta processes that were developed by Butcher [2] should be applied to solving a stiff system. Butcher has proved that for any positive integer n, there exists an n stage implicit Runge-Kutta process of order 2n of the form

$$\begin{aligned} x_{n+1} &= x_n + h \sum_{i=1}^{n} b_i k_i \\ k_i &= f(x_n + h \sum_{i=1}^{n} \beta_{ij} k_j) \quad i=1,\ldots,n \end{aligned} \tag{2.25}$$

If Butcher's processes are applied to the test equation $\dot{x} = qx$, then one gets a recurrence relation of the form

$$x_{n+1} = E_{n,n}(qh)x_n$$

where $E_{n,n}(qh)$ is the n^{th} diagonal Pade approximation, which is A-stable for $Re(qh) \leq 0$ (see section III.3). Consequently the implicit Runge-Kutta methods of Butcher are A-stable.

For n = 2, the coefficients of the fourth order implicit Runge-Kutta process are

$$\begin{aligned} \beta_{11} &= 1/4 & \beta_{12} &= 1/4 - \sqrt{3}/6 \\ \beta_{21} &= 1/4 + \sqrt{3}/6 & \beta_{22} &= 1/4 \\ b_1 &= 1/2 & b_2 &= 1/2 \end{aligned} \tag{2.26}$$

Ehle developed some initial approximations to k_1 and k_2 which enable an iterative process for solving the implicit

equations to converge rapidly at each step. Although he did not clearly specify the iterative procedure employed, it is probably successive substitutions. In addition, it is not obvious why his initial approximations to k_1 and k_2 work as well as claimed, and why he has been able to avoid the step length limitations inherent in successive substitutions.

5. Trapezoidal Rule, Backward Euler and Implicit Midpoint Methods

In 1963, G. Dahlquist [3] proved that no explicit linear multistep method can be A-stable and that the maximum order of an A-stable linear multistep method is 2. Moreover, for fixed h, the method with the minimum truncation error is the trapezoidal rule.

However, there are difficulties which can arise when the trapezoidal rule is used. Applying it to a stiff linear system $\dot{x} = Ax$, where all the eigenvalues of A are in the LHP, one obtains the equation:

$$x_{n+1} - x_n - \frac{h}{2}\{Ax_{n+1} + Ax_n\} = 0 \quad . \tag{2.27}$$

To solve* equation (2.27), one can approximate x_{n+1} by x_n and apply a Newton-Raphson iteration to obtain a better approximation to x_{n+1}. For a general system of equations,

---

* Equation (2.27) can be solved exactly; but, for a general system, an exact solution is not possible.

the Newton-Raphson iteration is used repeatedly to get increasingly better approximations to x_{n+1} until two successive approximations to x_{n+1} differ by some specified small amount. The number of iterations that is necessary for convergence is often used to adjust the step size which, in turn, controls the truncation error of the entire procedure.

However, in the case of a linear system, the first iteration yields the result#

$$(2.28) \qquad x_{n+1} = C(Ah)x_n \quad \text{where } C(Ah) = \frac{I + (1/2)Ah}{I - (1/2)Ah}$$

which is the exact solution to equation (2.27). Except for rounding errors, further Newton-Raphson iterations do not alter the value of x_{n+1} that is given in equation (2.28). Consequently, if one is not careful, counting the number of iterations for the approximations to x_{n+1} to converge can be very misleading when one uses the number of iterations to control the step size.

Also, even though $\|C(Ah)\| \leq 1$ for $Re(\lambda) \leq 0$ (i.e. the method is A-stable), as h becomes large, $C(\lambda h) \to -1$. That is, the numerical solution has a tendency towards slowly damped oscillations which can be very troublesome during the calculation of the transient phases of the solution.

To avoid the oscillatory problem, one can use small

In a matrix equation, whenever a fraction of the form N/D appears, we are using the notation to mean the matrix $D^{-1}N$.

steps during the transient phases when rapidly changing components affect the solution. But, once the amplitude of the transients has become sufficiently small, one can start using larger time steps that are adjusted to the rate of change of the solution.

A second way to cope with the oscillations for a general equation is to use a smoothing process proposed by B. Lindberg [15]. He suggested that one calculate the function values at the points t_n, t_{n+1}, and t_{n+2}. Then one sets

$$\hat{x}_{n+1} = (1/4)(x_n + 2x_{n+1} + x_{n+2}) \tag{2.29}$$

and continues the integration from t_{n+1} (that is, backtracking one step) using the smoothed value, $\hat{x}_{n+1}$, as the function value.

A third way of coping with the oscillations is to use the backwards Euler scheme. For a linear system, one obtains the relation

$$x_{n+1} = C'(Ah)x_n \quad \text{where} \quad C'(Ah) = \frac{I}{I - Ah} \tag{2.30}$$

which is also A-stable for systems with eigenvalues in the LHP. However, one has that $C'(\lambda h) \to 0$ as $Re(\lambda h) \to -\infty$, which is desirable when the contributions to the solution corresponsing to the eigenvalues with large negative real parts is still significant. It must be remembered that the backwards Euler scheme is only a first order scheme and that care must be taken because the damping factor $C'(Ah)$ may be too strong and consequently produce an underestimate of the exact

solution.

It is interesting to note that in both cases the multiplicative quotients, C(Ah) and C'(Ah), are Pade approximations to the exponential function [19].

H. J. Stetter [22] has pointed out that for fixed h, the implicit midpoint method

$$(2.31) \qquad x_{n+1} - x_n = f(t_{n+1/2}, x_{n+1/2}) \quad \text{where } x_{n+1/2} = \frac{x_{n+1} - x_n}{2}$$

is equivalent to the trapezoidal rule. Presently, it is not known whether the implicit midpoint rule or the trapezoidal rule is more accurate for a variable h. The last question is important because a variable step size is usually used when one applies any integration scheme. At the moment, G. Dahlquist [5] feels that the implicit midpoint method may be better for variable h, but he does not have a conclusive proof.

6. Exponential Fitting

The main concept behind the work of W. Liniger and R. Willoughby [16] is the use of families of schemes where one selects the parameters based on some judgement about the solution. Liniger and Willoughby consider two basic families of integration schemes:

$$(2.32) \qquad x_{n+1} - x_n = \frac{h}{2}[(1+a)\dot{x}_{n+1} + (1-a)\dot{x}_n] - \frac{h^2}{4}[(b+a)\ddot{x}_{n+1} + (b-a)\ddot{x}_n] + (h^3)$$

and

$$(2.33) \qquad x_{n+1} - x_n = h[(1-\lambda)\dot{x}_{n+1} + \lambda\dot{x}_n] + \mathcal{O}(h^2)$$

These schemes are A-stable in the range:

$$0 \le b-a \le 1/3 \text{ and } 1/3 \le a+b \le 2$$

for the first family of schemes and

$$0 \le \lambda \le 1/2$$

for the second family of schemes.

It should be noted that in the second scheme $\lambda = 0$ gives the backwards Euler method and $\lambda = 1/2$ gives the trapezoidal rule. Thus, the choice of λ allows one to select either of these two extremes or an "intermediate" scheme at any point during the integration.

If one lets $x_{n+1} = r^{(\nu)}(q)x$, then a, b, and λ can be selected so that $r^{(\nu)}(q) = e^{-q} + \mathcal{O}(h^p)$ for appropriate values of p and q. This approach is called exponential fitting. In the case of equation (2.32), there is fitting at two points, but for equation (2.33) there is fitting at only one point.

For the linear equation $\dot{x} = Ax$, the application equations (2.32) and (2.33) yields

$$(2.34) \qquad x_{n+1} = \frac{I + h(1-a)A + \frac{h^2}{4}(b-a)A^2}{I + h(1+a)A + \frac{h^2}{4}(b+a)A^2} x_n$$

and

$$(2.35) \qquad x_{n+1} = \frac{I + h\lambda A}{I - h(1-\lambda)A} x_n$$

respectively.

Equations (2.34) and (2.35) indicate the structure of the characteristic roots of (2.32) and (2.33) respectively and point up the strong roll of the choice of a, b, and λ in determining the degree of damping of the higher modes in the numerical solution.

The authors [16] point out that during the transient phases, one would like λ small (equivalent to exponential fitting for large q). But during the asymptotic (or smooth) phase, one would like to benefit from the increased accuracy of the trapezoidal rule ($\lambda = 1/2$), because the values of q closer to zero are usually more important -- unless the transient solution still affects the solution, in which case a λ less than 1/2 is desirable to inhibit spurious oscillations. This strategy has allowed them to use approximately the same step sizes during the transient and the smooth phases. For the case of the semiconductor equations (see chapter IV), T. E. Stern [21], found that a choice of λ opposite to that suggested by Liniger and Willoughby was more efficient.

We should again emphasize the point that in order to solve the implicit equations for each time step, we must use a scheme like Newton-Raphson iteration and not successive substitutions. For the integration scheme (2.33), successive substitutions converges only if

$$h\|J\| \leq (1-\lambda)^{-1} < 2 \quad \text{for } 0 \leq \lambda \leq 1/2$$

and $\|J\|$, where J is the Jacobian matrix, is large if the

system is stiff. That is, successive substitutions imposes a step length limitation which was not inherent in the A-stable scheme and the limitation is as severe as that imposed by a typical explicit integration scheme. Interestingly, the first step in successive substitutions is

$$x_{n+1}^{(1)} = x_n + hf(t_n, x_n)$$

which is simply the forward Euler formula and not A-stable. $x_{n+1}^{(1)}$ is the first correction to the initial guess of $x_{n+1} = x_n$.

The reasons for using the first integration scheme, (2.32), instead of the second, (2.33), are that the additional terms, which involve the second derivative, gain one some additional accuracy and a second degree of freedom at the cost of evaluating the derivative of the Jacobian matrix.

7. Global Extrapolation [14]

One approach to increasing the accuracy of any integration scheme is to use a local or global extrapolation procedure. Expanding the error term for an integration scheme into an h^p term and higher order terms, one has that

(2.36) $$x_n(h) = x(t_n) + \varepsilon_n(t_n)h^p + \mathcal{O}(h^{p+1})$$

where $x_n(h)$ is the computed approximation to $x(t_n)$ using a step size h. One also has that

(2.37) $$x_n(h/2) = x(t_n) + \varepsilon_n(t_n)\left(\frac{h}{2}\right)^p + \mathcal{O}(h^{p+1}).$$

Combining equations (2.36) and (2.37) to eliminate the

h^p term, it follows that

$$(2.38) \qquad x_n^{(1)}(h) = \frac{2^p x_n(\frac{h}{2}) - x_n(h)}{2^p - 1}$$

which differs from $x(t_n)$ by $\mathcal{O}(h^{p+1})$ - that is

$$(2.39) \qquad x_n^{(1)}(h) = x(t_n) + \mathcal{O}(h^{p+1}).$$

The extrapolation procedure can be continued using step sizes h/4, h/8, ... to eliminate successively one power of h in the error term at each repetition. If the procedure is used at each step before going on to the next step, it is called local extrapolation.

Global extrapolation differs from local extrapolation in that one first computes the x_n over the entire interval desired using a basic step size $h = h_0$. Then one recomputes the values of the solution over the same interval using the step sizes h/2, h/4, ... Finally, one uses the various values of x_n that were independently computed - namely $x_n(h)$, $x_n(h/2)$, ... to extrapolate at each point. The big disadvantage of global extrapolation is that it requires considerably more computer storage than does local extrapolation.

In the use of extrapolation, it is important not to introduce instabilities into the numerical solution. In the case of the trapezoidal rule, local extrapolation destroys the A-stability of the scheme. But, one can still apply global extrapolation, which does not affect the A-stability of an integration scheme because the extrapolation is done

after the entire integration is performed and not during the integration [3,14].

In the case of the trapezoidal rule, at the i^{th} point, the error expansion is of the form

$$x_i(h) = x(t_i) + \tau_i(t_i)h^2 + \tau_2(t_i)h^4 + \dots . \tag{2.40}$$

Consequently, equation (2.38) becomes

$$x_n^{(1)} = \frac{4^m x_n(\frac{h}{2}) - x_n(h)}{4^m - 1} \tag{2.41}$$

and, in addition, each stage of the extrapolation increases the accuracy by h^2.

In general, the global extrapolation procedure can be visualized as computing the following table

$$\begin{array}{lll} x_i(h) & \longrightarrow & x_i^{(1)}(h) \longrightarrow x_i^{(2)}(h) \\ x_i(h/2) & \nearrow & x_1^{(1)}(h/2) \nearrow \\ x_i(h/4) & \nearrow & \end{array} \tag{2.42}$$

where the x_i are the calculated values using the step size specified and

$$x_i^{(m)}\left(\frac{h}{2^k}\right) = \frac{4^m x_i^{(m-1)}\left(\frac{h}{2^{k+1}}\right) - x_i^{(m-1)}\left(\frac{h}{2^k}\right)}{4^m - 1} \tag{2.43}$$

It is important to remember that in order for the extrapolation to work well, one must use a sufficient number of Newton-Raphson iterations so that the error in solving the implicit equations at each point is less than the error desired through extrapolation.

8. Alternatives to A-Stability

As pointed out earlier, A-stability imposes a very severe restriction on the types of linear multistep methods which are acceptable. Namely, methods can be of order two at most and must be implicit. Consequently, in order to maintain accuracy for long time calculations, the step size may have to be limited (even though there is not any problem of stability) or global extrapolation used. But these limitations are not as severe as those posed by stability. Several authors have proposed alternate stability criteria for stiff systems that have allowed them to develop higher order linear multistep methods that satisfy their alternate criteria.

Olof Widlund [24] has proposed A(α)-stability.

Definition: A linear difference method is A(α)-stable for $\alpha \in (0,\Pi/2)$ if all solutions of the linear difference method tend towards zero as $t \to \infty$ when the method is applied with fixed h to $\dot{x} = qx$, where q is a complex constant and lies in the set

$$S_\alpha = \{ z : |\arg(-z)| < \alpha, z \neq 0 \} .$$

Widlund's definition requires that all the eigenvalues are in a wedge shaped sector in the LHP (see figure 2.1b). We should note that A($\Pi/2$)-stability corresponds precisely to Dahlquist's A-stability.

Widlund was able to show that for $\alpha \in [0,\Pi/2)$ there are

$A(\alpha)$-stable linear multistep methods of orders 3 and 4 (of degrees 3 and 4 respectively). The methods are useful for many problems, but if α is close to $\Pi/2$ some of the parameters in the linear multistep methods become very large, thereby making the methods unsuitable for practical uses in such cases. There is also the usual difficulty in changing step sizes because the degrees of the methods are greater than 1.

A second alternative to A-stability was proposed by C.W. Gear [8], who developed the notion of "stiffly stable" schemes. Such a scheme would be stable and accurate for eigenvalues in a rectangular region of the $h\lambda$-plane which includes the origin and stable in all parts of the LHP to the left of the rectangle. (See figure 2.1c.) Gear's criterion has enabled him to produce up to sixth order linear multistep methods. He also developed automatic procedures for selecting the step size and order of the scheme during the calculation. However, the dimensions of the region of stability that were given by Gear are $D \approx -6.1$, $\theta \approx 0.5$ and $\alpha \approx 0.1$ (see figure 2.1c), which prohibits us from having eigenvalues lying in the important regions in the LHP above and below the rectangle.

If there is a single eigenvalue, λ, in the unstable region of the LHP, then one can either increase or decrease h so that $h\lambda$ falls within the region of stability. However, if there are several eigenvalues along a ray in the LHP that passes through the unstable region, it may be very difficult to adjust h so

that all the eigenvalues along the ray fall into the region of stability without making h unduly large or small [27].

Also, because the stiffly stable scheme is variable order and variable step size, it is exceedingly difficult to get bounds on the error terms. In addition, there are the usual difficulties that are associated with any linear multistep method of degree (number of previous function values needed) greater than one. Namely, starting values must be computed for the integration scheme by a special procedure and changing the step size can be difficult.

C. W. Gear [27] has pointed out that the coefficients used in the stiffly stable scheme, particularly the sixth order scheme are not optimal. However, an attempt is being made (in the United Kingdom) to calculate better coefficients. F. H. Branin [26] has found that Gear's scheme works quite well on the whole. During the almost-discontinuous segments of the solution, the automatic order selection procedure usually selects the second and third order schemes, and during the smooth segments, a third through fifth order scheme is usually selected.

It should be noted that the trapezoidal rule with global extrapolation can also produce truncation errors of the sixth or higher orders. It has been found that the trapezoidal rule with global extrapolation is the most accurate of the integration schemes for stiff systems that has been discussed but it is also one of the most time consuming [14].

Figure 2.1a
A-stability

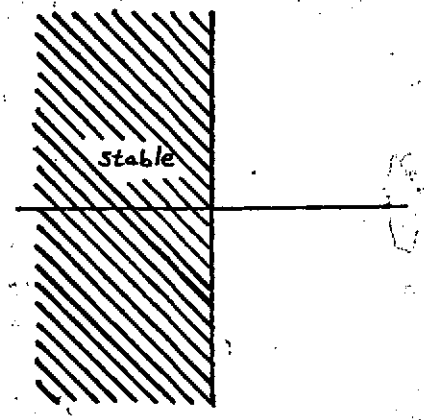

Figure 2.1b
A(α)-stability

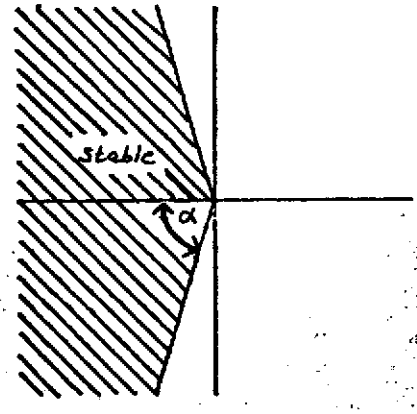

Figure 2.1c
Stiffly Stable

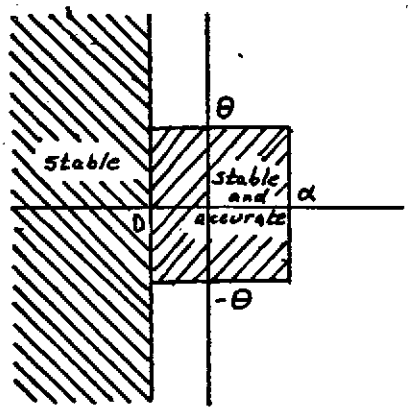

Figure 2.1 : Regions corresponding to various notions of stability for stiff systems of ordinary differential equations.

III. Generalized Trapezoidal Rule

1. Description of the Integration Scheme

The new numerical integration scheme for stiff systems or ordinary differential equations, that is presented in this chapter, is a modification of the trapezoidal rule. The scheme will be called the generalized trapezoidal rule. The two major aspects of the integration scheme, that differ from the trapezoidal rule, are:

(1) the use of different numerical integration techniques for the smooth and the almost-discontinuous segments of the solution;

and (2) the transformation of the original system of equations, during the almost-discontinuous segments of the solution, to a new system of differential equations that is less stiff, and, consequently, "easier" to integrate.

The transformation employed involves an exponential time shift, related to the Jacobian of the original system of equations. The resulting transformed system of equations will have eigenvalues closer to the origin, and have a lesser spread between the largest and the smallest eigenvalue than the original system.

The objectives of the new method are:

(1) to allow the use of larger time steps during the

integration of the almost-discontinuous segments of the solution and still maintain the same or improved accuracy as compared to the trapezoidal rule,
and (2) to "lessen" the oscillatory problem that is inherent with the trapezoidal rule (see section II.5).

Generally speaking, the approach of the proposed integration scheme is to use the trapezoidal rule with Newton-Raphson iterations to solve the implicit equations for calculating the smooth segments of the solution. During the almost-discontinuous segments, the equations are transformed to a "smoother" set of equations by means of the transformation suggested by Lawson [13]. The transformed set of equations is integrated by means of the trapezoidal rule and both Newton-Raphson and modified Newton-Raphson iterations are employed to solve the implicit equations.

The approach described herein differs from Lawson's generalized Runge-Kutta methods [13] is several respects. The transformation is applied to an A-stable integration scheme and is only applied during the almost-discontinuous segments in conjunction with a linear time shift, in order to reduce the norms of the matrices involved in the exponential function; and Newton-Raphson and modified Newton-Raphson iterations are used to solve the implicit equations in order to reduce the amount of work per iteration.

A detailed description of the proposed scheme for the smooth segments is found in section III.1a and for the

almost-discontinuous segments in section III.1b. Step size control and techniques for differentiating between the smooth and almost-discontinuous segments will be discussed in section III.4.

The integration scheme for the smooth segments is a relatively standard version of the trapezoidal rule; whereas, the integration scheme for the almost-discontinuous segments is a new approach.

1a. Calculation of the Smooth Segments

Consider the stiff system of differential equations:

$$\dot{x} = f(t,x),\ x(0) = x_0. \tag{3.1}$$

If one applies the trapezoidal rule:

$$x_{n+1} = x_n + \frac{h}{2}[\dot{x}_n + \dot{x}_{n+1}] + e(t_n,h), \tag{3.2a}$$

where

$$e(t_n h) = \frac{h^3}{4}\int_0^1 2\theta(\theta-1)x^{(3)}(t_n+\theta h)d\theta \tag{3.2b}$$

is the error term [17], and one Newton-Raphson iteration to integrate equations (3.1), it can be shown that

$$x_{n+1}^{(1)} = x_{n+1}^{(0)} - \frac{(x_{n+1}^{(0)} - x_n) - \frac{h}{2}[f(t_n,x_n) + f(t_{n+1},x_{n+1}^{(0)})]}{I - \frac{h}{2}\left(\frac{\partial f}{\partial x}\right)(t_{n+1},x_{n+1}^{(0)})} \tag{3.3}$$

where $x_{n+1}^{(0)}$ is an initial approximation to x_{n+1} and $x_{n+1}^{(1)}$ is the first corrected approximation to x_{n+1}. Usually, the initial approximation to x_{n+1} is

$$x_{n+1}^{(0)} = x_n . \tag{3.4}$$

For additional corrected approximations to x_{n+1}, the superscripts in equation (3.3) are simply increased by one for each succeeding iteration, and equation (3.3) is repeatedly iterated until two sucessive approximations to x_{n+1} differ by less than some prescribed small amount. When the difference is small enough, it is said that the sequence of approximations, $\{x_{n+1}^{(i)}\}$, has converged to x_{n+1}.

Equation (3.3) can be rewritten as

$$x_{n+1}^{(1)} = x_{n+1}^{(0)} - \left[I - \frac{h}{2}\left(\frac{\partial f}{\partial x}\right)(t_{n+1}, x_{n+1}^{(0)})\right]^{-1} v_{n+1}^{(0)} \tag{3.5a}$$

where

$$v_{n+1}^{(0)} = (x_{n+1}^{(0)} - x_n) - \frac{h}{2}[f(t_n, x_n) + f(t_{n+1}, x_{n+1}^{(0)})]. \tag{3.5b}$$

From equations (3.5), it can be seen that the work necessary for the first Newton-Raphson iteration consists of:

2 function evaluations
1 Jacobian evaluation
1 matrix inversion
1 matrix vector product.

In order to calculate the amount of work necessary to do the second and later iterations, one must decide whether Newton-Raphson (NR) or modified Newton-Raphson (MNR) iterations will be used. For the first iteration, equations (3.5), both types of iterations are identical. For the second and later iterations, both iterative schemes use equation (3.5), with superscripts suitably modified. However, in the case of MNR iterations, the Jacobian matrix is only evaluated during the first iteration and never reevaluated for succeeding iterations - that is, the same Jacobian is used for each

iteration.

Hence, in the case of MNR iterations, the second and later iterations require:

1 function evaluation
1 matrix vector product.

In the case of NR iterations, the Jacobian matrix is reevaluated for each iteration. Thus, the second and later iterations require:

1 function evaluation
1 Jacobian evaluation
1 matrix inversion
1 matrix vector product.

W. Liniger [16] pointed out that if the initial approximation, $x_{n+1}^{(0)}$, to x_{n+1} differed from x_{n+1} by $\mathcal{O}(h^g)$, $g \geq 1$, then the first two MNR iterations will differ from x_{n+1} by $\mathcal{O}(h^{2g+1})$ and $\mathcal{O}(h^{3g+2})$ respectively; whereas, the first two NR iterations will differ from x_{n+1} by $\mathcal{O}(h^{2g+1})$ and $\mathcal{O}(h^{4g+3})$ respectively. If, as suggested, one used equation (3.4) for the initial approximation to x_{n+1}, then $x_{n+1}^{(0)}$ differs from x_{n+1} by $\mathcal{O}(h)$ --i.e. $g = 1$. This is true because the trapezoidal rule is a consistant integration scheme (i.e. its error term is of order 1 or greater).

One should note that the matrix inverse called for in equation (3.5a) does not have to be performed. Equation (3.5a) can be changed so as to allow one to solve a linear system of equations instead. (It requires fewer operations to solve a linear system than to invert a matrix.)

If one is solving a system of m equations, then the work necessary for the first NR iteration is:

2 function evaluations
1 Jacobian evaluation
$\frac{m^3 + 3m^2 + 2m}{3}$ multiplications
$\frac{m^2 + 3m}{2}$ divisions.

The second and later NR iterations require

1 function evaluation
1 Jacobian evaluation
$\frac{m^3 + 3m^2 + 2m}{3}$ multiplications
$\frac{m^2 + 3m}{2}$ divisions,

and the second and later MNR iterations require

1 function evaluation
$m^2 + m$ multiplications.

In later sections, the above tabulations will be used to compare the computational efficiency of the trapezoidal rule and the generalized trapezoidal rule.

1b. Calculation of the Almost-Discontinuous Segments

During the almost-discontinuous segments of the solution, the original differential equations will be altered by means of the exponential transformation suggested by Lawson [13] (see section II.3). The effect of the transformation will be to create a new system of equations such that all the eigenvalues (those in the LHP and RHP) of the transformed

system are closer to the origin than the eigenvalues of the original system. In addition to Lawson's transformation, time shifts will be performed in order to prevent the norms of the matrices, that appear in the exponential transformation, from becoming too large.

Consider the stiff system of differential equations

(3.6) $\dot{x} = f(t,x),\ x(0) = x_0.$

If x_n, the numerical approximation to $x(t_n)$, has already been computed, a time shift,

(3.7a) $\tau = t - t_n$

is performed. From equation (3.7a), it follows that

(3.7b) $x(t) = x(t_n + \tau)$

and the differential equations in the shifted variables are

(3.8) $\frac{dx}{d\tau} = f(\tau,x),\ x(0) = x_n.$

Applying the transformation

(3.9) $z(\tau) = e^{-\tau K}x(\tau)$

to equation (3.8) yields

(3.10a) $\frac{dz}{d\tau} = g(\tau,z) = e^{-\tau K}f(\tau,e^{\tau K}z) - Kz,\ z(0) = x_n.$

and

(3.10b) $\left(\frac{\partial g}{\partial z}\right) = e^{-\tau K}\left[\left(\frac{\partial f}{\partial x}\right) - K\right]e^{\tau K}.$

It can be seen from equations (3.9) and (3.10) that the time shift, equation (3.7), was needed to prevent excessively large exponents in $e^{-\tau K}$ and $e^{\tau K}$. Large exponents can cause precision problems during the actual computation.

If one uses the trapezoidal rule with a step size h on

equation (3.10), with a NR (or MNR) iteration and an initial approximation of $z_1^{(0)}$ to z_1, it follows that

$$(3.11a)\quad z_1^{(1)} = z_1^{(0)} - \frac{(z_1^{(0)} - z_0) - \frac{h}{2}[g(0,z_0) + g(h,z_1^{(0)})]}{I - \frac{h}{2}\left(\frac{\partial g}{\partial z}\right)(h,z_1^{(0)})}$$

or

$$(3.11b)\quad z_1^{(1)}=z_1^{(0)} - \frac{(z_1^{(0)}-z_0)-\frac{h}{2}[e^{-0K}f(0,e^{0K}z_0)-Kz_0+e^{-hK}f(h,e^{hK}z_1^{(0)})-Kz_1^{(0)}]}{I - \frac{h}{2}e^{-hK}[\left(\frac{\partial f}{\partial x}\right) - K]e^{hK}}$$

$$(3.11c)\quad z_1^{(1)}=z_1^{(0)} - \frac{(z_1^{(0)}-z_0)-\frac{h}{2}[f(0,z_0)+e^{-hK}f(h,e^{hK}z_1^{(0)})-K(z_0+z_1^{(0)})]}{e^{-hK}[I - \frac{h}{2}[\left(\frac{\partial f}{\partial x}\right) - K]]e^{hK}}.$$

Equation (3.11c) can be reqritten as

$$(3.12a)\quad z_1^{(1)} = z_1^{(0)} - e^{-hK}[I - \frac{h}{2}[\left(\frac{\partial f}{\partial x}\right) - K]]^{-1}e^{hK}v_1^{(0)}$$

where

$$(3.12b)\quad v_1^{(0)}=(z_1^{(0)}-z_0)-\frac{h}{2}[f(0,z_0)+e^{-hK}f(h,e^{hK}z_1^{(0)})-K(z_0+z_1^{(0)})].$$

To calculate the amount of work per iteration, the manner in which the matrix K is selected must be considered. Also, the method for evaluating matrix exponentials must be discussed.

From equation (3.10b), it can be seen that the magnitude of the eigenvalues of the Jacobian, $(\partial g/\partial z)$, depend solely on the difference $[(\partial f/\partial x) - K]$. The other two factors, e^{-hK} and e^{hK}, act as a similarity transformation. Consequently, the matrix K will always be chosen to be equal to the Jacobian matrix of the original system, $(\partial f/\partial x)$, at one of the previously calculated grid points.

If K is changed at the point (t_n,x_n), which corresponds to the time shifted point $(0,z_0)$ with time shift t_n, then the new choice for K will be the value of the Jacobian, $(\partial f/\partial x)$, of the original system of equations at the initial point (t_n,x_n). A new value for K should be chosen when the solution being calculated has just entered an almost-discontinuous segment or when the solution is already in an almost-discontinuous segment and the iterations at the previous point converged too slowly.

This particular choice of K at (t_n,x_n) results in a significant reduction in the work involved in calculating x_{n+1}. For the first iteration for the calculation of x_{n+1}, equations (3.12) can be simplified to

$$z_1^{(1)} = z_1^{(0)} - v_1^{(0)} \tag{3.13a}$$

where

$$v_1^{(0)}=(z_1^{(0)}-z_0)-\frac{h}{2}[f(0,z_0)+e^{-hK}f(h,e^{hK}z_1^{(0)})-K(z_0+z_1^{(0)})] \tag{3.13b}$$

because

$$e^{-hK}[I - [\left(\frac{\partial f}{\partial x}\right) - K]]e^{hK} = I. \tag{3.13c}$$

For the second and further iterations, if an MNR iteration is used, then equations (3.13) can be applied again. However, if a NR iteration is used, one must revert to equations (3.12) because equation (3.13c) is no longer satisfied for a general system of equations (see section III.3a for a discussion of a linear system).

For the calculation of the exponential functions, the

diagonal Pade approximations will be used because they are A-stable and make optimal use of the powers of the matrix, hK, that will be computed. In particular, either the E_{11} or E_{22} approximations,

$$(3.14) \qquad E_{11}(hK) = [I - \tfrac{1}{2}hK]^{-1}[I + \tfrac{1}{2}hK] + \mathcal{O}(\|hK\|^3)$$

$$(3.15) \qquad E_{22}(hK) = [I - \tfrac{1}{2}hK + \tfrac{1}{12}(hK)^2]^{-1}[I + \tfrac{1}{2}hK + \tfrac{1}{12}(hK)^2] + \mathcal{O}(\|hK\|^5)$$

will be used because their error terms are of the same or slightly better order than the generalized trapezoidal rule. It is important to note that if global extrapolation is to be used, a higher order Pade approximation may be necessary (see section II.7).

In order to get increased accuracy in the Pade approximations, the argument reduction scheme,

$$(3.16) \qquad [e^{(2^{-s}hK)}]^{2^s} = e^{hK}$$

is used. The bracketed expression is calculated using E_{11} or E_{22} and then squared s times. Equation (3.16) effectively decreases the norm of the argument of the Pade approximation and thereby increases the accuracy of the approximation. In practice, s = 4 or 5 is usually sufficient [19]. In section III.3b, there is a more detailed discussion of the error in the Pade approximation.

To calculate the amount of work per iteration, two pairs of cases must be considered: using a new value for K, and using an old value for K. For each

possibility for K, either an NR or an MNR iteration can be used for the second and further iterations. In addition to the work per iteration, there is one extra matrix-vector product needed to obtain x_{n+1} from z_1.

When selecting a new value for K, the amount of work for the first NR (or MNR) iteration, using equations (3.13), is

2 function evaluations
1 Jacobian evaluation (for K)
2 Pade approximations
3 matrix-vector products
1 matrix-scalar product.

For the second and later MNR iterations, equations (3.13) can still be used. The work per iteration is

1 function evaluation
3 matrix-vector products
1 matrix-scalar product.

However, for the second and later NR iterations, one requires

1 function evaluation
1 Jacobian evaluation
1 matrix inversion
6 matrix-vector products
1 matrix-scalar product.

Clearly, it is preferable to use MNR iterations since a NR iteration requires much more work per iteration than a MNR iteration.

When using an old value of K, all iterations must make use of equations (3.12). The first iteration requires

2 function evaluations
1 Jacobian evaluation
1 matrix inverse
6 matrix-vector products
1 matrix-scalar product.

For the second and later iterations, an MNR iteration requires

1 function evaluation
6 matrix-vector products,

and a NR iteration requires

1 function evaluation
1 Jacobian evaluation
1 matrix inverse
6 matrix-vector products
1 scalar-matrix product.

Again one should note that the matrix inverse called for in equation (3.12a) and indicated above does not have to be performed. Equation (3.12a) can be changed so as to allow one to solve a linear system of equations instead.

When one is solving a system of m equations, and has selected a new value for K, the amount of work for the first iteration is

2 function evaluations
1 Jacobian evaluation
2 Pade approximations
$4m^2 + m$ multiplications.

The second and later MNR iterations require

1 function evaluation
$4m^2 + m$ multiplications,

and the second and later NR iterations require

1 function evaluation
1 Jacobian evaluation
$\frac{m^3 + 21m^2 + 2m}{3}$ multiplications
$\frac{m^2 + 3m}{2}$ divisions.

However, when an old value of K is being used, then the first NR iteration requires

2 function evaluations
1 Jacobian evaluation
$\frac{m^3 + 2lm^2 + 2m}{3}$ multiplications
$\frac{m^2 + 3m}{2}$ divisions.

The second and later MNR iterations need

1 function evaluation
$6\,m^2 + m$ multiplications,

and the second and later NR iterations need

1 function evaluation
1 Jacobian evaluation
$\frac{m^3 + 2lm^2 + 2m}{3}$ multiplications
$\frac{m^2 + 3m}{2}$ divisions.

In the later sections of this chapter, it will be shown that the generalized trapezoidal rule is computationally more efficient than the trapezoidal rule for computing almost-discontinuous segments. The theoretical error comparisons and illustrative computer results presented later indicate that the generalized trapezoidal rule requires more work per iteration but the overall amount of work needed to compute the almost-discontinuous segment is less than that required by the trapezoidal rule.

2. Rationale of the Integration Scheme

When an A-stable integration scheme is being chosen to solve a stiff system of ordinary differential equations, a

decision must be made as to how much detail one wishes to see in the various segments of the solution. This decision is particularly crucial for the calculation of the almost-discontinuous segments of the solution.

If one wishes to obtain extremely fine details of the structure of the solution during an almost-discontinuous segment, then extremely small step sizes of the order of the smallest time constant must be used. In that case, the present author and others [5] have suggested that high order explicit schemes, such as fourth and fifth order Runge-Kutta schemes, be used for the almost-discontinuous segments of the solution. The small step size (or perhaps a reasonable fraction of it such as 1/2 or 1/4) satisfies the stability criterion for explicit schemes and the explicit schemes are much easier to implement (there are no implicit equations to solve).

However, during the smooth portions of the solution, an A-stable scheme will probably be needed because the desired step sizes will probably be outside the region of stability for a step length limited scheme. It remains to be proved that the use of an explicit scheme within it's region of stability does not cause stability problems when one switches to an A-stable scheme to calculate the smooth segments of the solution, although it is probably true.

The generalized trapezoidal rule proposed herein is not meant for obtaining very fine detailed solutions during the

almost-discontinuous sections. The basic aim of the scheme is to transform the original equations to a new system of equations such that the eigenvalues of the transformed system are smaller in magnitude than the eigenvalues of the original system, and the ratio of the real parts of the largest to the smallest eigenvalue of the transformed system is smaller than the same ratio for the original system. A consequence of shifting the eigenvalues is that for the same accuracy, one can integrate the new system of equations with a larger time step than is possible for the original system. However, a larger time step means that one cannot expect to see as much fine detail as for a smaller time step.

Therefore, the proposed scheme is primarily suggested for the integration of systems of equations where one wishes to see a moderate to coarse degree of detail, but with a relatively high degree of accuracy, for the almost-discontinuous segments, and is not interested in very fine detail. The proposed scheme allows one to solve a smoother set of equations at the expense of having to calculate a difficult transformation and having to do more work per step than the trapezoidal rule. However, in the almost-discontinuous segments, the extra work is more than compensated for by allowing one to use larger time steps when solving the transformed equations as compared with solving the original equations.

The other consideration in the proposed scheme is that

the smooth and almost-discontinuous segments are calculated differently. The reasoning behind this decision is that simple A-stable schemes, such as the trapezoidal rule, work very will when the solution, over the particular region being integrated, is smooth. A transformation whose objective is to smooth out the solution in a smooth region cannot help very much, since the solution of the original system is already smooth. Therefore, the extra work necessary for computations using the generalized trapezoidal rule, as compared with the trapezoidal rule, probably cannot be favorable compensated for by an increase in the step size (to be illustrated in section III.3a).

3. Theoretical Error Calculations

The trapezoidal rule and the generalized trapezoidal rule discussed in section III.1 are both second order schemes. However, it is constructive to compare the errors produced by each scheme when each is applied to several examples where the exact error can be calculated. In section III.3a, three examples are considered and the errors in the numerical solution using each scheme are calculated and tabulated for various step sizes.

In section III.3b, the errors incurred when one uses the Pade approximation, E_{11} and E_{22}, are discussed and tabulated. The reduction in the error as a result of using the argument scheme (3.16) is also considered. Errors for other Pade

approximations can be found in [19].

3a. Theoretical Errors for Various Ordinary Differential Equations *

In each example, it will be assumed that all exponential functions can be calculated exactly. The errors incurred in calculating the exponentials will be discussed in the next section. Also, all calculations will be based on a fixed step size.

Example 1:

The first example that will be considered is the stiff linear time invariant system

$$\dot{x} = Ax, \quad x(0) = x_0 . \tag{3.17}$$

When the exponential transformation,

$$z = e^{-tC}x, \tag{3.18}$$

is applied to (3.17), it follows that

$$\frac{dz}{dt} = e^{-tC}Ae^{tC}z - Cz, \quad z(0) = x_0 \tag{3.19}$$

If $C = A$, then equation (3.19) reduces to

$$z' = 0, \; z(0) = x_0 \tag{3.20}$$

which can be solved exactly --i.e. no error is incurred in the numerical solution.

However, if $C \neq A$, but $C \approx A$ and C and A commute, then

---

* The second and third examples in this section were suggested by [5].

it follows that

$$(3.21) \qquad \frac{dz}{dt} = (A-C)z, \quad z(0) = x_0 .$$

If we have that

$$\rho(A-C) \ll \rho(A)$$

where $\rho(A)$ is the spectral radius of A, then the trapezoidal rule will give more accurate results for equation (3.21) than for equation (3.17). Equivalently, for the same accuracy, one can take larger steps when integrating equation (3.21) than when integrating equation (3.17). Also, since the trapezoidal rule applied to a linear system is equivalent to using the E_{11} Pade approximation (without using the argument reduction scheme), the errors in the computation can be found on tables (3.3) and (3.4) in section III.3b.

Example 2:

For the second example, the inhomogeneous scalar equation,

$$(3.22) \qquad \dot{x} = qx + e^{st}, \quad x(0) = 1,$$

will be considered. The exact solution to equation (3.22) is

$$(3.23) \qquad x = e^{qt} + \frac{1}{s-q}[e^{st}-e^{qt}].$$

Applying the trapezoidal rule with one NR iteration, $x_{n+1}^{(0)} = x_n$ and step size h results in

$$(3.24a) \qquad \bar{x}_n = \frac{1+\frac{h}{2}q}{1-\frac{h}{2}q}\bar{x}_{n-1} + \left[\frac{\frac{h}{2}(1+e^{sh})}{1-\frac{h}{2}q}\right]e^{(n-1)sh}$$

$$(3.24b) \qquad = E\bar{x}_{n-1} + Fe^{(n-1)sh}$$

where $\bar{x}_n$ is our numerical approximation to $x(t_n)$.

From equation (3.24b) it follows that

$$(3.25a)\quad \bar{x}_n = E^n x_0 + [e^{(n-1)sh} + Ee^{(n-2)sh} + E^2 e^{(n-3)sh} + \ldots + E^{n-2}e^{sh} + E^{n-1}]F$$

or

$$(3.35b)\quad \bar{x}_n = E^n x_0 + \left[\frac{e^{nsh}-E^n}{1-Ee^{-sh}}\right] e^{-sh} F$$

because (3.25a) is a geometric progression.

Applying the transformation

$$(3.26)\quad z = e^{-tq} x$$

to equation (3.22) yields

$$(3.27a)\quad z' = e^{\alpha t},\ z(0) = x_0$$

where

$$(3.27b)\quad \alpha = s-q$$

Use of the trapezoidal rule with one NR iteration, $z^{(0)}_{n+1} = z_n$ and step size h, in the transformed equation (3.27) leads to

$$(3.28a)\quad \bar{z}_n = \bar{z}_{n-1} + \frac{h}{2}[e^{(n-1)h\alpha} + e^{nh\alpha}]$$

$$(3.28b)\Rightarrow\quad \bar{z}_n = z_0 + \frac{h}{2}[1+e^{h\alpha}][1+e^{h\alpha}+e^{2h\alpha}+\ldots+e^{(n-1)h\alpha}]$$

$$(3.28c)\Rightarrow\quad \bar{z}_n = z_0 + \frac{h}{2}\left[\frac{1+e^{\alpha h}}{1-e^{\alpha h}}\right]\left[1-e^{nh\alpha}\right]$$

or

$$(3.29)\quad \bar{\bar{x}}_n = x_0 e^{nhq} + \frac{h}{2}\left[\frac{1+e^{\alpha h}}{1-e^{\alpha h}}\right]\left[e^{nhq}-e^{nhs}\right]$$

where $\bar{\bar{x}}_n$ is our numerical approximation to $x(t_n)$ using the generalized trapezoidal rule.

Using equations (3.23), (3.25b) and (3.29) with the values:

$q = -1000$
$s = -1$
$n = 12$

one can see from table (3.1) that in order to obtain a solution with the accuracy of ϵ_{12} after 12 steps, one can take steps that are 10 to 25 times as large with the generalized trapezoidal rule as with the trapezoidal rule.

Theoretical Errors for Example 2

Table (3.1)

ϵ_{12}	h-trap	h-Gen.Trap
10^{-4}	6×10^{-5}	1.5×10^{-3}
10^{-5}	3×10^{-5}	4×10^{-4}
10^{-6}	1.25×10^{-5}	1.5×10^{-4}
10^{-7}	5×10^{-6}	5.5×10^{-5}
10^{-8}	2.5×10^{-6}	2.5×10^{-5}

where

ϵ_{12} = desired error after 12 steps.
h_{trap} = step size for trapezoidal rule which achieves the desired error.
h-gen.trap. = step size for the generalized trapezoidal rule which achieves the desired error.

The above table of errors was calculated using approximately 17 digit arithmetic.

Example 3:

The third example to be considered is the second order non-linear system

(3.30) $\dot{x} = qx \qquad x(0) = 1$

$\dot{y} = x^2 + sy \qquad y(0) = 1$

whose exact solution is

$$(3.31)\qquad x = x_0 e^{qt}$$

$$y = y_0 e^{st} + \frac{x_0^2}{2q-s}[e^{2qt} - e^{st}]$$

By applying the transformation

$$(3.32)\qquad \begin{pmatrix} u \\ v \end{pmatrix} = e^{-Jt}\begin{pmatrix} x \\ y \end{pmatrix}$$

where J is the Jacobian of equations (3.30), it follows that

$$(3.33)\qquad \begin{aligned} \dot{u} &= 0 && ,u(0) = x_0 \\ \dot{v} &= e^{(2q-s)t} u^2 - 2x_0 e^{(q-s)t} u && ,v(0) = y_0 \end{aligned}$$

If one applies the trapezoidal rule to equations (3.30), one has that

$$(3.34a)\qquad \begin{aligned} \bar{x}_n &= \left[\frac{1+\frac{h}{2}q}{1-\frac{h}{2}q}\right]^n x_0 \\ \bar{y}_n &= E_1^{\,n} + K\left[\frac{E_1^{\,n} - E_2^{\,2n}}{E_1 - E_2^{\,2}}\right] \end{aligned}$$

where

$$(3.34b)\qquad E_1 = \left[\frac{1+\frac{h}{2}s}{1-\frac{h}{2}s}\right] \quad E_2 = \left[\frac{1+\frac{h}{2}q}{1-\frac{h}{2}q}\right] \quad K = \left[\frac{h^2 q}{(1-\frac{h}{2}s)(1-\frac{h}{2}q)} + \frac{h}{1-\frac{h}{2}s}\right] x_0^{\,2}$$

The use of the generalized trapezoidal rule to equations (3.30) yields

$$(3.35a)\qquad \begin{aligned} \bar{\bar{x}}_n &= x_0 e^{qnh} \\ \bar{\bar{y}}_n &= \frac{2x_0^{\,2}}{q-s}(e^{qnh} - e^{snh}) + y_0 e^{snh} \\ &\quad + e^{snh}(\tfrac{h}{2})x_0^{\,2}[1+e^{\alpha h}]\left[\frac{1-e^{n\alpha h}}{1-e^{\alpha h}}\right] \\ &\quad + e^{snh} h x_0^{\,2}[1+e^{\beta h}]\left[\frac{1-e^{n\beta h}}{1-e^{\beta h}}\right] \end{aligned}$$

where

(3.35b) $\alpha = 2q - s$

$\beta = q - s$.

Using equations (3.31), (3.34), and (3.35) with the values:

$q = -1000$
$s = -1$
$n = 12$

one can see from table (3.2) that in order to obtain a solution with an accuracy of ϵ_{12} in both x and y after 12 steps, one can take steps that are 8 to 50 times as large with the generalized trapezoidal rule as with the trapezoidal rule.

Theoretical Errors for Example 3

Table (3.2)

ϵ_{12}	h-Trap(x)	h-Trap(y)	h-Gen.Trap(y)
10^{-4}	6×10^{-5}	9×10^{-4}	2×10^{-3}
10^{-5}	3×10^{-5}	3×10^{-4}	1.5×10^{-3}
10^{-6}	1.25×10^{-5}	1×10^{-4}	6.5×10^{-4}
10^{-7}	5×10^{-6}	4×10^{-5}	5×10^{-5}
10^{-8}	2.5×10^{-6}	2×10^{-5}	2×10^{-5}

where

ϵ_{12} = desired error after 12 steps.
h-trap = Step size for trapezoidal rule for variable specified.
h-Gen.Trap. = Step size for generalized trapezoidal rule for y variable.

The column h-Gen.Trap.(x) has been omitted because the generalized trapezoidal rule produces an exact answer for x_n.

When looking at the results of the calculation for the third set of equations (3.30), one must remember that in the solution, x is rapidly changing, but y is slowly changing. From equations (3.31), one can see that y goes as e^{-st} with a slight perturbation due to the second term. For s = -1 and q = -1000, the perturbation is less than 1/2000 --i.e. it shows up in the third or fourth decimal places at most. Thus, when using the trapezoidal rule to integrate equation (3.30), it is the x equation that determines the step size at the beginning of the numerical computation.

If one compares the step sizes for y in table (3.2), one can see that there is not that much difference between them. But, the solution to the y equations is smooth and we should not expect a smoothing process to signigicantly decrease the required step size. For this reason the transformation should be applied only during the almost-discontinuous segments and not during the smooth segments.

3b. Accuracy of the Pade Approximations [19]

The Pade approximations are useful for computing e^{tM}, where M is an n x n matrix, when the stability of the approximation is an important criterion. For the Pade approximations, one sets

$$(3.36) \qquad e^{tM} \approx Ep,q(tM) = \frac{N_{p,q}(tM)}{D_{p,q}(tM)}$$

where N_{pq} and D_{pq} are polynomials with real coefficients and of orders q and p respectively. The coefficients of N and D are chosen so that E_{pq} agrees with the Taylor series expansion of e^{tM} through and including terms of order (p+q). This requirement leads to the equations [25]

$$N_{p,q}(tM) = \sum_{k=0}^{q} \frac{(p+q-k)!q!}{(p+q)!k!(q-k)!}(tM)^k$$

(3.37)

$$D_{p,q}(tM) = \sum_{k=0}^{p} \frac{(p+q-k)!p!}{(p+q)!k!(p-k)!}(tM)^k$$

In particular,

$$E_{11}(tM) = \frac{I + \frac{1}{2}tM}{I - \frac{1}{2}tM} \qquad (3.38a)$$

and

$$E_{22}(tM) = \frac{I+\frac{1}{2}tM+\frac{1}{12}t^2M^2}{I-\frac{1}{2}tM+\frac{1}{12}t^2M^2} \qquad (3.38b)$$

In equation (3.36), if $p \geq q$, then the Pade approximation is A-stable for all $t \geq 0$ when M has eigenvalues in the LHP. That is

$$\|E_{p,q}(tM)\| \leq 1$$

for arguments with eigenvalues in the LHP.

As pointed out earlier, to increase the accuracy of the Pade approximations, the argument reduction scheme,

$$[e^{(2^{-s}tM)}]^{2^s} = e^{tM} \qquad (3.39)$$

is used (see[19] for a proof).

For the E_{11} and E_{22} approximations, the errors for various t for selected values of s are tabulated below in table (3.3) for E_{11} and table (3.4) for E_{22}.

Error in the E_{11} Pade Approximation ($E_{11} - e^{-t}$)

Table (3.3)

t	e^t	s=0	s=4	s=5	s=6
-5	$6.7*10^{-3}$	$-4.4*10^{-1}$	$-2.7*10^{-4}$	$-6.8*10^{-5}$	$-1.7*10^{-5}$
-10	$4.5*10^{-5}$	$-6.7*10^{-1}$	$-1.3*10^{-5}$	$-3.6*10^{-6}$	$-9.2*10^{-7}$
-50	$1.9*10^{-22}$	$-9.2*10^{-1}$	$2.9*10^{-11}$	$-1.9*10^{-22}$	$-1.8*10^{-22}$
-100	$3.7*10^{-44}$	$-9.6*10^{-1}$	$2.5*10^{-5}$	$8.4*10^{-22}$	$-3.7*10^{-44}$
-200	$1.4*10^{-89}$	$-9.8*10^{-1}$	$5.7*10^{-2}$	$6.1*10^{-22}$	$7.1*10^{-43}$
-250	$2.6*10^{-178}$	$-9.8*10^{-1}$	$1.6*10^{-2}$	$5.3*10^{-8}$	$3.7*10^{-32}$

Error in the E_{22} Pade Approximation ($E_{22} - e^{-t}$)

Table (3.4)

t	e^t	s=0	s=4	s=5	s=6
-5	$6.7*10^{-3}$	$9.8*10^{-2}$	$4.5*10^{-7}$	$3.1*10^{-8}$	$5.0*10^{-9}$
-10	$4.5*10^{-5}$	$3.0*10^{-1}$	$9.9*10^{-8}$	$6.1*10^{-9}$	$3.8*10^{-10}$
-50	$1.9*10^{-22}$	$7.9*10^{-1}$	$8.9*10^{-19}$	$1.2*10^{-22}$	$5.2*10^{-24}$
-100	$3.7*10^{-44}$	$8.9*10^{-1}$	$9.2*10^{-14}$	$7.9*10^{-37}$	$5.8*10^{-44}$
-200	$1.4*10^{-89}$	$9.4*10^{-1}$	$2.2*10^{-7}$	$8.4*10^{-27}$	$6.2*10^{-73}$
-250	$2.6*10^{-178}$	$9.5*10^{-1}$	$4.6*10^{-6}$	$6.9*10^{-22}$	$1.4*10^{-72}$

From tables (3.3) and (3.4), one can see that depending upon the accuracy desired, the E_{11} or E_{22} approximations with s=5 or 6 will be sufficiently accurate for calculating the various exponential functions needed for the exponential transformation used in the generalized trapezoidal rule. One should note that the amount of work needed to compute E_{22} with q=m is the same as the work needed to compute E_{11}

with $s = m+1$. However, the E_{22} approximation with $s = m$ is more accurate than the E_{11} approximation with $q = m+1$. Therefore, the E_{22} approximations will be used for computations.

4. Step Size Control and Detection of Almost-Discontinuous Segments

For a particular problem, the user always specifies the maximum step size, h_{max}. For example, h_{max} is at most the sample period for the solution or the points at which the user wishes to see the value of the solution over the range of integration. To begin calculating the solution, it will be assumed that one will calculate an almost-discontinuous segment (unless told otherwise) and begin with a step size $h = h_{max}/16$. The reason for the 1/16 is that the user may have specified a large h_{max}, and, if h is too large, too many iterations may have to be done before the approximations to the next point converge.

During the smooth segments, the step size control that will be used is a standard method based on counting the number of iterations necessary to solve the implicit trapezoidal rule equations (3.2). If it takes one or two iterations for the approximations to x_{n+1} to converge, then the step size, h, will be doubled for the calculation of the next point. However, if the approximations have not converged after four iterations, then the results of the iterations will be discarded and the calculation will

be repeated with half as large a step size as the one that failed.

For the almost-discontinuous segments, the amount of work needed for changing the step size --except for doubling the step size, which requires two matrix multiplications, or else two matrix-vector procucts and some bookkeeping-- is considerable because e^{-hK} and e^{hK} must be recomputed. Therefore, the rule of thumb used for changing step sizes will be that if the approximation to z_1 converges in two or fewer iterations, h will be doubled, but if it takes more than five iterations then h will be halved and K reselected. The reselection of K whenever h is halved is done in order to reduce significantly the work for the first iteration with the new h --i.e. one uses equations (3.13) instead of (3.12). The reason for changing h after five iterations, instead of four as in the smooth segments, is that changing h requires alot of work.

The detection of whether one is calculating a smooth segment or an almost-discontinuous segment is at best a difficult task. If one does not have a priori information about the nature of the solution or the location of the smooth and almost-discontinuous segments then there are two available alternatives. One can calculate finite difference approximations to the derivative of each variable or one can count the number of NR (or MNR) iterations needed to obtain the

solution to the implicit integrating equations. The latter approach will be chosen.

The procedure that will be used is as follows: (1) if one is calculating an almost-discontinuous segment and the step size for the next step will exceed $h_{max}/8$, then one will say that one is in the smooth region and change to the integration of a smooth segment of the solution; but (2) if one is calculating a smooth segment and the step size for the next step will be less than $h_{max}/32$, then one will say that one is in the almost-discontinuous region and change to the integration of an almost-discontinuous step.

The cross-over point between the two types of segments is not the same in both directions. This is intentional and is meant to prevent the numerical technique from oscillating back and forth between the two phases of the integration scheme. If rapid oscillations between the two phases were permitted, then the value of the approach in the almost-discontinuous segments might be nullified due to the initial overhead involved.

5. Illustrative Computer Results

In this section, the results of actual numerical computations using both the generalized trapezoidal rule and the trapezoidal rule will be presented. The examples chosen are the same as those analysed in section III.3a (Theoretical

Error Calculations).

For the linear equations, the exponential functions needed for the generalized trapezoidal rule were evaluated by means of the E_{11} Pade Approximation with s = 5. However, for the inhomogenous equations and the non-linear equations, the E_{22} Pade approximation with s = 5 was used. In each example, the step size was controlled by means of the method suggested in section III.4. All computations were done in PL1 to approximately 17 decimal places on the IBM 360/91.

Example 1:

For the linear equation

(3.40) $\dot{x} = qx, \quad x(0) = 1.$

the solutions were computed over the interval t = (0,.25) for q = -10, -100, and -1000. Maximum step sizes of h = .1 and .01 were used.

In the tables below, "error" means the error in the numerical solution at t = .25. "Oscillates" means that the numerical solution was oscillating because the step size was too large, and that the indicated error is a poor measure of the solution over the entire interval t = (0,.25). "Iter" refers to the number of Newton-Raphson iterations needed to obtain the solution. Also, an error of 10^{-8} means that the error occurred in a decimal position beyond the last decimal digit printed out.

Computational Error for Example 1

Table (3.5a)

q = -10	Trap.Rule		Gen.Trap.Rule	
h_{max}	error	iter	error	iter
.1	$1x10^{-2}$	12	$7x10^{-6}$	6
.01	$1x10^{-4}$	58	$3x10^{-5}$	29

Table (3.5b)

q = -100	Trap. Rule		Gen.Trap.Rule	
h_{max}	error	iter	error	iter
.1	$1.8x10^{-3}$ oscillates	12	10^{-8}	6
.01	10^{-8}	43	10^{-8}	29

Table (3.5c)

q = -1000	Trap. Rule		Gen. Trap. Rule	
h_{max}	error	iter	error	iter
.1	$2.6x10^{-1}$ oscillates	12	10^{-8}	6
.01	$2x10^{-7}$ oscillates	50	10^{-8}	29

From the three tables above, one can see that for comparable accuracy, the generalized trapezoidal rule allowed one to take step sizes that were 10 times as large as the step sizes needed for the trapezoidal rule. Also, the generalized trapezoidal rule eliminated the problem of oscillations that was evident in the computational results for the trapezoidal rule.

Example 2:

For the inhomogenous equation

(3.41) $\dot{x} = qx + e^{-t}, \quad x(0) = 1.,$

the solutions were computed over the interval $t = (0,1/q)$ for $q = -100$ and -1000 and for $s = -1$. The maximum step size for the integrations was 1 q but for the trapezoidal rule, the integration was also performed for maximum step sizes of 1/2q, 1/4q, 1/8q and 1/16q.

Computational Errors for Example 2

Table (3.6a)

q = -100	Trap. Rule		Gen. Trap. Rule	
h_{max}	error	iter	error	iter
0.01	5×10^{-3}	10	1×10^{-4}	10
0.005	5×10^{-3}	12	-----	
0.0025	2.5×10^{-3}	16	-----	
0.00125	4.3×10^{-4}	24	-----	
0.000625	1.1×10^{-4}	40	-----	

Table (3.6b)

q = -1000	Trap. Rule		Gen. Trap. Rule	
h_{max}	error	iter	error	iter
0.001	4.5×10^{-3}	10	2.4×10^{-4}	10
0.0005	4.5×10^{-3}	12	-----	
0.00025	1.6×10^{-3}	16	-----	
0.000125	4.3×10^{-4}	20	-----	
0.0000625	1.2×10^{-4}	24	-----	

From the tables above, one can see that for comparable errors, the generalized trapezoidal rule allowed an increase in step size by a factor of 8 or more as compared to the trapezoidal rule.

It is difficult to compute the amount of work per iteration for both integration schemes because it is not known how much work was needed for the function evaluations. It should be noted, however, that the Pade approximations were not very expensive to compute for equation (3.41) because the Jacobian of the system was constant. Consequently, one needed to use the Pade approximations to obtain the exponentials only for the first integration step. For the remaining steps, the exponentials could be generated by simply squaring the previous values since the step size was doubled.

Example 3:

For the nonlinear system of equations

(3.42) $\dot{x} = qx$

$\dot{y} = x^2 - sy, \quad x(0) = y(0) = 1.,$

the solutions were computed over the interval $t = (0, 1/q)$ for $q = -100$, and -1000 and $s = -1$. The step sizes used were the same as for example 2.

Computational Errors for Example 3

Table (3.7a)

$q = -100$	Trap. Rule			Gen. Trap. Rule		
h_{max}	error(x)	error(y)	iter	error(x)	error(y)	iter
0.01	5×10^{-3}	5.2×10^{-5}	15	1.6×10^{-5}	3.4×10^{-5}	10
0.005	1.5×10^{-3}	5.1×10^{-5}	17	---	---	---
0.0025	1.5×10^{-3}	5.2×10^{-5}	19	---	---	--
0.00125	4.2×10^{-4}	1.5×10^{-5}	24	---	---	--
0.000625	1.1×10^{-4}	4.3×10^{-6}	40	---	---	--

Table (3.7b)

q=-1000	Trap. Rule			Gen.Trap. Rule		
h_{max}	error(x)	error(y)	iter	error(x)	error(y)	iter
0.001	4.5×10^{-3}	1.3×10^{-5}	11	2.4×10^{-5}	1×10^{-5}	9
0.0005	4.5×10^{-3}	1.1×10^{-5}	13	---	---	--
0.00025	1.5×10^{-3}	4.8×10^{-6}	16	---	---	--
0.000125	3.2×10^{-4}	1.5×10^{-6}	24	---	---	--
0.0000625	1.1×10^{-4}	5×10^{-7}	40	---	---	--

The tables again show that the maximum step size can be increased by a factor of over 16 when using the generalized trapezoidal rule instead of the trapezoidal rule in order to get errors of about 10^{-5} in both x and y. There was no oscillatory problem when using the trapezoidal rule because the step sizes were small, that is, the step sizes were less than 2/q.

From the three examples considered in this section, it can be seen that during the transient, the generalized trapezoidal rule has two advantages over the trapezoidal rule:

(1) the generalized trapezoidal rule allows one either to take larger step sizes while maintaining the same accuracy as the trapezoidal rule, or to take the same step sizes and increase the accuracy of the numerical solution, and

(2) the generalized trapezoidal rule minimized the problem of oscillations during the transient. Such oscillations are common to the trapezoidal rule.

IV. Applications to Semiconductor Networks

1. Semiconductor Network Equations [21]

In this section, the structure of the differential equations for a semiconductor network will be briefly discussed. We will also point out some ways in which an a priori knowledge of the particular structure of the network equations can be exploited to reduce the computational effort needed to integrate the equations.

The network equations are a special canse of the general equation

(4.1a) $$\dot{z} = f(t,z)$$

where

(4.1b) $$z = (x,u)^*$$

and x and u are vectors. The dynamic behavior of u can usually be considered "fast" compared with that of x. As a result, the system of equations will be stiff.

Consider a semiconductor network composed of the following types of circuit elements: independent voltage sources, linear positive time-invariant capacitors, linear positive time-invariant inductors, linear positive time-invariant

---

*$z = (x,u)$ means that z is the vector whose components are $z_1 = x_1, z_2 = x_2 \ldots z_m, z_{m+1} = u, z_{m+2} = u_2 \ldots z_{m+n} = u_n$.

resistors, and semiconductor branches (edges). The terminal characteristics of the semiconductor branches are assumed to be of the form:

(4.2) $\quad i_T = C(u)\dot{u} + M\rho(u)$

where

i_T is an n-vector of semiconductor branch currents

u is an n-vector of semiconductor branch voltages

$\rho(u) = [\rho_1(u_1),\ldots,\rho_n(u_n)]^t$ is an n-vector of carrier densities

$C(u) = \text{diag}[C_1(u_1),\ldots,C_n(u_n)]$ is a matrix of incremental capacitances (positive for all u)

M is a constant symmetric non-singular n x n matrix.

The model of the transistor branches that is being employed is the high frequency model which allows for the close approximation of high frequency effects such as rise time. [20]

Any network consisting of the above classes of elements can be described by a set of equations of the form:

(4.3a) $\quad P\dot{x} = Ax + Nu + y(t)$

(4.3b) $\quad C(u)\dot{u} = N^t x - Gu - M\rho(u) + w(t)$

where

x is an m vector of state variables associated with linear reactive elements,

P and A are constant symmetric non-singular m x m matrices derived respectively from linear reactive and resistive element values.

N is a constant m x n matrix

G is a constant symmetric positive semi-definite n x n matrix derived from linear resistive element values

y(t) and w(t) are respectively m and n vectors derived from the independent voltage sources.

It is important to note that the non-linearities occur only in equations (4.3b) --the equations associated with the semiconductor elements. In addition, because the time constants associated with the semiconductor equation (4.3b) are usually much smaller than those associated with the linear equations (4.3a), the dynamics associated with u are generally much faster than that associated with x, resulting in a stiffness problem.

Because of the wide variation in time constants, it is useful to think of the solution as composed of two types of segments:

(1) smooth segments where x and u vary slowly at about the same rate,

and (2) almost discontinuous segments characterized by a very rapid change in u while x remains almost constant.

Behavior of type (1) is typical of digital wave forms between switching instants, while type (2) is characteristic of transients that occur at switching times.

From equations (4.3), it follows that the Jacobian of the system of equations is

$$(4.4a) \qquad \begin{bmatrix} P^{-1}A & P^{-1}N \\ SN^t & \Delta - SG + SM\rho'(u) \end{bmatrix}$$

where

$$(4.4b) \qquad S = S(u) = C^{-1}(u) \quad \text{and}$$

$$(4.4c) \qquad \Delta = S'\{\text{diag}[N^t x] + \text{diag}[M\rho(u)] - G\}$$

The matrices $\text{diag}[N^t x]$ and $\text{diag}[M\rho(u)]$ are diagonal matrices whose i^{th} diagonal element is the i^{th} element of the vectors $N^t x$ and $M\rho(u)$ respectively and

$$(4.4d) \qquad \rho'(u) = \left[\text{diag} \frac{\partial \rho_1(u)}{\partial u_1}, \cdots, \frac{\partial \rho_n(u)}{\partial u_n}\right]$$

is also a diagonal matrix.

Furthermore, it has been pointed out by Hachtel and Rohrer [9] that it is often a very good approximation to neglect S' and, consequently, Δ. Hence, whenever the numerical integration technique described in Chapter III calls for the Jacobian of the original system, the matrix

$$(4.5) \qquad J = \begin{bmatrix} P^{-1}A & P^{-1}N \\ SN^t & S[-G+M\rho'(u)] \end{bmatrix} = \begin{bmatrix} J_{11} & J_{12} \\ J_{21} & J_{22} \end{bmatrix}$$

will be used instead of the exact Jacobian matrix.

It should be noted that J_{11} and J_{12} are constant and the only submatrices of J which vary are J_{21} and J_{22}. Furthermore, the only parts of J_{21} and J_{22} which vary are $S(u)$ and $\rho'(u)$. The foregoing observations permit a reduction in computational effort during both the smooth and the almost-discontinuous segments of the calculation.

During the smooth portions of the solution, one uses the trapezoidal rule, equation (3.5), to obtain the numerical solution. Substituting equation (4.3) into equation (3.5a), one has that

$$(4.6) \qquad \begin{bmatrix} x^{(1)} \\ u \end{bmatrix}_{n+1} = \begin{bmatrix} x^{(0)} \\ u \end{bmatrix}_{n+1} - \left[I - \frac{h}{2}J\right]^{-1} V_{n+1}^{(0)}$$

Substituting equation (4.5) in to equation (4.6), it can be seen that

$$(4.7) \qquad L = \left[I - \frac{h}{2}J\right]^{-1} = \begin{bmatrix} I_m - \frac{h}{2}J_{11} & -\frac{h}{2}J_{12} \\ -\frac{h}{2}J_{21} & I_n - \frac{h}{2}J_{22} \end{bmatrix} = \begin{bmatrix} L_{11} & L_{12} \\ L_{21} & L_{22} \end{bmatrix}$$

For a fixed h, L_{11} and L_{12} are constant. Therefore, one can perform a partial Gaussian elimination of the L matrix and store the result back in L (see[23]).

Now, if h is varied, then one can multiply the elements in the upper rows of the partially reduced form of L, that was generated for a previous value of h, by the appropriate ratio of the step sizes to get the partially reduced form of L for the new h.

The savings in computational effort due to partially reducing L and storing the result back in L depend upon the number of components in the x and u vectors (m and n respectively.)

Another reduction in computational effort can be obtained by noting that S(u) and $\rho'(u)$ are diagonal matrices. Therefore, when calculating J_{21} and J_{22} (and consequently L_{21} and L_{22}) several of the matrix multiplications, which require n^3

operations can be reduced to the multiplication of the rows by the corresponding element in the diagonal matrix, which requires only n^2 multiplications.

During the almost-discontinuous portions of the solution, savings due to the a priori knowledge of the structure of the equations are more dramatic than during the smooth portions of the solution.

As pointed out in section III.1b, the matrix K, that is used in equation (3.9) to transform the original system of equations, is always chosen to be equal to the Jacobian matrix at some previously calculated grid point. Let

$$K = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} \tag{4.8}$$

where the dimensions of the K submatrices are the same as the corresponding submatrices of J. Since J_{11} and J_{12} are constant, it follows that

$$K_{11} = J_{11} \text{ and } K_{12} = J_{12}. \tag{4.9}$$

Therefore, one has that

$$[J-K] = \begin{bmatrix} 0 & 0 \\ J_{21}-K_{21} & J_{22}-K_{22} \end{bmatrix} \tag{4.10}$$

Substituting equation (4.1) and (4.10) into equation (3.12a), it can be seen that

$$L = \left[I - \frac{h}{2}[J-K]\right] = \begin{bmatrix} 0 & 0 \\ -\frac{h}{2}[J_{21}-K_{21}] & I_n - \frac{h}{2}[J_{22}-K_{22}] \end{bmatrix} \tag{4.12}$$

Consequently, the use of the transformation, equation (3.9), effectively yields a system where the first m rows of L are already reduced --i.e., the first m rows do not require further Gaussian eliminations.

2. Detection of Smooth and Almost-Discontinuous Segments

As indicated in section III.4, an a priori knowledge of the nature of the solution can be useful for detecting whether a smooth or almost-discontinuous segment is being calculated. As indicated earlier (section IV.1), the smooth segments are characterized by x and u both changing at about the same rate and the almost-discontinuous segments are characterized by a very rapid change in u while x remains almost constant.

If we let

$$\|\Delta x\| = \frac{1}{m}\sum_{i=1}^{m}|\Delta x_i| \tag{4.13a}$$

and

$$\|\Delta u\| = \frac{1}{n}\sum_{i=1}^{n}|\Delta u_i| \tag{4.13b}$$

then a possible criterion for deciding which segment of the solution is being calculated can be based on the ratio

$$R = \frac{\|\Delta x\|}{\|\Delta u\|} \tag{4.14a}$$

as follows:

(1) if the smooth portion is being calculated, then

it will be assumed that the smooth portion is still being calculated if

(4.14b) $R > 1/16$;

otherwise, it will be assumed that the almost-discontinuous portion has been entered;

and (2) if the almost-discontinuous portion is being calculated, then it will be assumed that the almost-discontinuous portion is still being calculated if

(4.14c) $R < 1/8$;

otherwise, it will be assumed that the smooth portion has been entered.

We should note that, as before, the cross-over point is not the same in both directions. This is intentional and is meant to prevent the numerical technique from oscillating back and forth between the two phases of the integration scheme. This criterion is also related to the criterion for separating the two portions that was suggested by [18].

As before, it will be assumed that the initial point is in the almost-discontinuous section unless otherwise noted. Also, the initial step size will be set at 1/16th the maximum step size in order to prevent too many iterations from being done to solve the implicit equations.

It is suggested by the present author that the new technique for integrating the almost-discontinuous segments be used if

the criterion for being in the almost-discontinuous segment of the solution described in this section or in section III.4 is satisfied. Otherwise, the usual trapezoidal rule, which was described in section III.1a, will be used.

3. Application of the Generalized Trapezoidal Rule to an Astable Multivibrator Circuit

In this section, the numerical solution of the differential equations for an astable multivibrator will be used to illustrate the computational efficiencies of both the trapezoidal rule and the generalized trapezoidal rule. The particular multivibrator that will be analyzed is shown in figure (4.1). By using the high frequency model of a transistor, the original circuit (fig. 4.1) is replaced by the circuit shown in figure (4.2) where the incremental capacitances are shown in dotted lines.

The state variable equations for the high frequency model of the astable multivibrator are:

$$\dot{x}_1 = f_1(x,u) = \frac{1}{R}(u_3-u_5-\frac{x_1}{c_1}+E) + \frac{1}{R_B}(u_3-u_5-\frac{x_1}{c_1}+u_6)$$

$$\dot{x}_2 = f_2(x,u) = \frac{1}{R}(u_4-u_6-\frac{x_2}{c_2}+E) + \frac{1}{R_B}(u_4-u_6-\frac{x_2}{c_2}+u_5) \qquad (4.15)$$

$$C_c(u_3)\dot{u}_3 = g_3(x,u) = -f_1(x,u) - F_c(u_3,u_5) - \frac{1}{R_2}(u_3-u_5+E)$$

$$C_c(u_4)\dot{u}_4 = g_4(x,u) = -f_2(x,u) - F_c(u_4,u_6) - \frac{1}{R_2}(u_4-u_6+E)$$

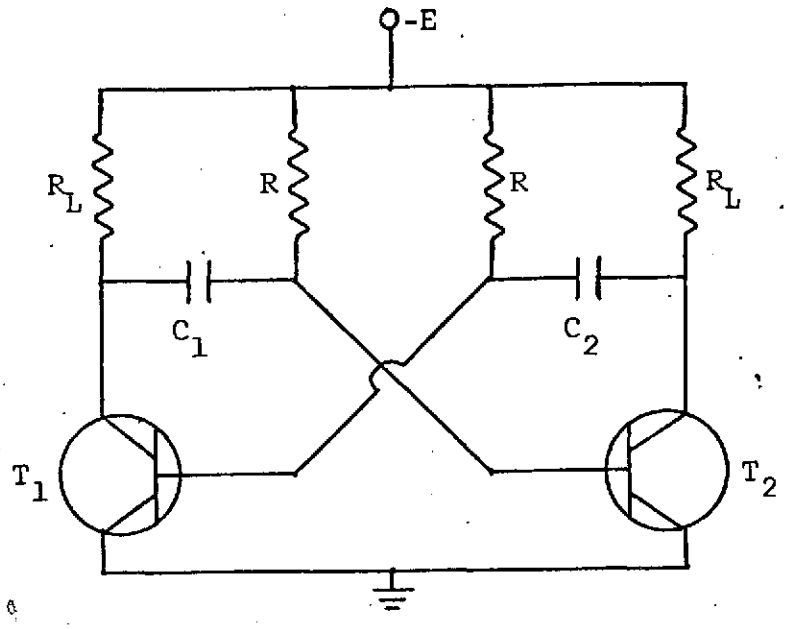

Figure 4.1 - Transistor Multivibrator

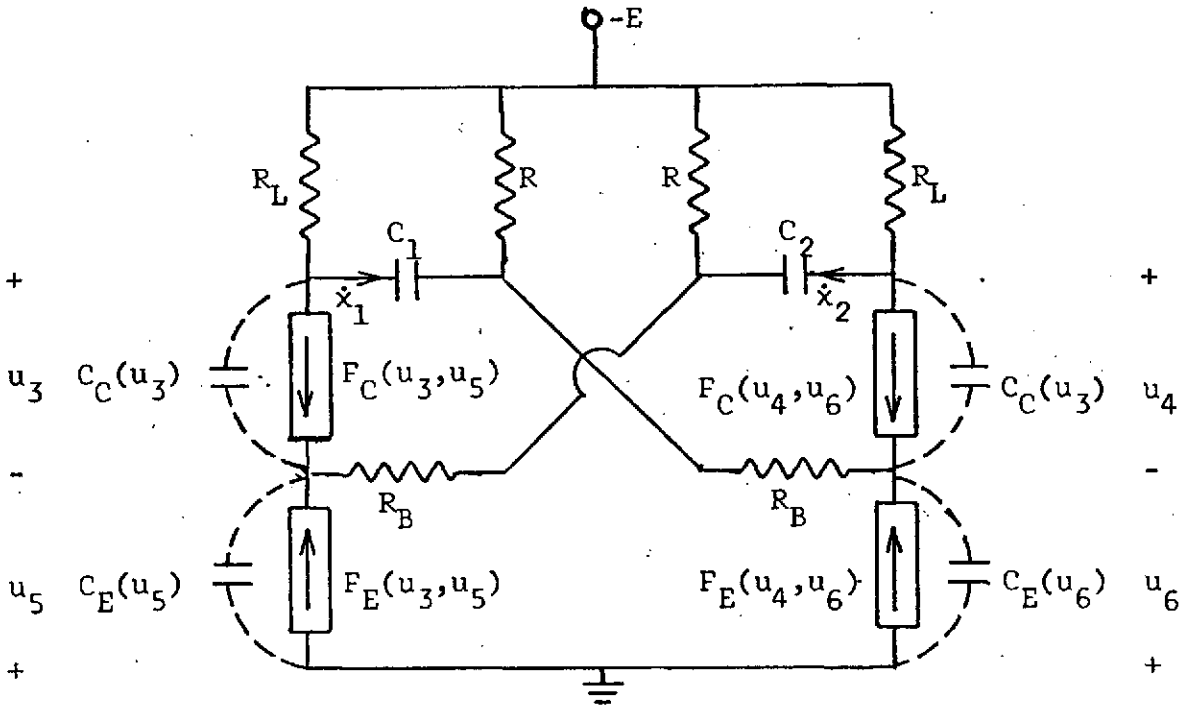

Figure 4.2 - High Frequency Model of Multivibrator in fig 4.1

$$C_E(u_5)\dot{u}_5 = g_5(x,u) = f_1(x,u) - F_E(u_3,u_5) + \frac{1}{R_B}\left(\frac{x_2}{c_2} - u_4 - u_5 + u_6\right) + \frac{1}{R_2}(u_3 - u_5 + E)$$

(4.15) con't.

$$C_E(u_6)\dot{u}_6 = g_6(x,u) = f_2(x,u) - F_E(u_4,u_6) + \frac{1}{R_B}\left(\frac{x_1}{c_1} - u_3 - u_6 + u_5\right) + \frac{1}{R_2}(u_4 - u_6 + E)$$

where

$$C_c(u) = C_E(u) = 4\times10^{-4}e^{40u} + 2 \text{ pF}$$

$$F_c(V_c,V_E) = 10^{-5}[2(e^{40V_c}-1) - 0.98(e^{40V_E}-1)]\text{mA}$$

$$F_E(V_c,V_E) = 10^{-5}[0.98(e^{40V_c}-1) + (e^{40V_E}-1)]\text{ mA}$$

(4.16)

$$R_B = 0.2 \text{ K}\Omega$$

$$R_L = 0.6 \text{ K}\Omega$$

(Units: volts, mA, KΩ, pF)

$$R = 6. \text{ K}\Omega$$

$$E = 10 \text{ volts}$$

$$C_1 = C_2 = C \text{ (varied) pF}$$

The variables x_1 and x_2 are the charges on the capacitors C_1 and C_2 respectively, and u_3, u_4, u_5, and u_6 represent voltages within the transistor model (see figure 4.2).

If one writes equation (4.3) as

$$\begin{bmatrix}\dot{x}\\ \dot{u}\end{bmatrix} = Q\begin{bmatrix}x\\ u\end{bmatrix} + R(u) + T(t) \tag{4.17}$$

where

$$Q = \begin{bmatrix}P^{-1}A & P^{-1}N\\ C^{-1}N^t & -C^{-1}G\end{bmatrix} \tag{4.18}$$

(4.18) con't

$$R = \begin{bmatrix} 0 \\ C^{-1}Mp(u) \end{bmatrix} \quad \text{and}$$

$$T = \begin{bmatrix} P^{-1}y(t) \\ C^{-1}w(t) \end{bmatrix}$$

then, for the multivibrator equations, (4.15), the matrices in equations (4.17) and (4.18) become

$$R = \begin{bmatrix} 0 \\ 0 \\ -C_c^{-1}(u_3)F_c(u_3u_5) \\ -C_c^{-1}(u_4)F_c(u_4,u_6) \\ -C_E^{-1}(u_5)F_E(u_3,u_5) \\ -C_E^{-1}(u_6)F_E(u_4,u_6) \end{bmatrix} \qquad P = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

(4.19)

$$T = \begin{bmatrix} \frac{E}{R} \\ \frac{E}{R} \\ -C_c^{-1}(u_3)\left(\frac{1}{R} + \frac{1}{R_L}\right)E \\ -C_c^{-1}(u_4)\left(\frac{1}{R} + \frac{1}{R_L}\right)E \\ -C_E^{-1}(u_5)\left(\frac{1}{R} + \frac{1}{R_L}\right)E \\ -C_E^{-1}(u_6)\left(\frac{1}{R} + \frac{1}{R_L}\right)E \end{bmatrix}$$

(4.19)
continued

$$
Q=\left[\begin{array}{c|c|c|c|c|c}
-\frac{1}{C_1}\left(\frac{1}{R}+\frac{1}{R_B}\right) & 0 & \left(\frac{1}{R}+\frac{1}{R_B}\right) & 0 & -\left(\frac{1}{R}+\frac{1}{R_B}\right) & \frac{1}{R_B} \\
0 & -\frac{1}{C_2}\left(\frac{1}{R}+\frac{1}{R_B}\right) & 0 & \left(\frac{1}{R}+\frac{1}{R_B}\right) & \frac{1}{R_B} & -\left(\frac{1}{R}+\frac{1}{R_B}\right) \\
\hline
\frac{C_C^{-1}(u_3)}{C_1}\left(\frac{1}{R}+\frac{1}{R_B}\right) & 0 & -C_C^{-1}(u_3)\left(\frac{1}{R}+\frac{1}{R_B}+\frac{1}{R_L}\right) & 0 & C_C^{-1}(u_3)\left(\frac{1}{R}+\frac{1}{R_B}+\frac{1}{R_L}\right) & -\frac{C_C^{-1}(u_3)}{R_B} \\
0 & \frac{C_C^{-1}(u_4)}{C_2}\left(\frac{1}{R}+\frac{1}{R_B}\right) & 0 & -C_C^{-1}(u_4)\left(\frac{1}{R}+\frac{1}{R_B}+\frac{1}{R_L}\right) & -\frac{C_C^{-1}(u_3)}{R_B} & C_C^{-1}(u_3)\left(\frac{1}{R}+\frac{1}{R_B}+\frac{1}{R_L}\right) \\
-\frac{C_E^{-1}(u_5)}{C_1}\left(\frac{1}{R}+\frac{1}{R_B}\right) & \frac{C_E^{-1}(u_5)}{R_B C_2} & C_E^{-1}(u_5)\left(\frac{1}{R}+\frac{1}{R_B}+\frac{1}{R_L}\right) & -\frac{C_E^{-1}(u_5)}{R_B} & -C_E^{-1}(u_5)\left(\frac{1}{R}+\frac{2}{R_B}+\frac{1}{R_L}\right) & \frac{2C_E^{-1}(u_5)}{R_B} \\
\frac{C_E^{-1}(u_6)}{R_B C_1} & -\frac{C_E^{-1}(u_6)}{C_2}\left(\frac{1}{R}+\frac{1}{R_B}\right) & -\frac{C_E^{-1}(u_6)}{R_B} & C_E^{-1}(u_6)\left(\frac{1}{R}+\frac{1}{R_B}+\frac{1}{R_L}\right) & \frac{2C_E^{-1}(u_6)}{R_B} & -C_E^{-1}(u_6)\left(\frac{1}{R}+\frac{2}{R_B}+\frac{1}{R_L}\right)
\end{array}\right]
$$

(4.19) continued

$$C = \begin{bmatrix} C_c(u_3) & 0 & 0 & 0 \\ 0 & C_c(u_4) & 0 & 0 \\ 0 & 0 & C_E(u_5) & 0 \\ 0 & 0 & 0 & C_E(u_6) \end{bmatrix}$$

$$M\rho(u) = \begin{bmatrix} F_c(u_3,u_5) \\ F_c(u_4,u_6) \\ F_E(u_3,u_5) \\ F_E(u_4,u_6) \end{bmatrix}$$

One also has that

(4.20)
$$\frac{\partial M\rho(u)}{\partial u} = 10^{-5} \begin{bmatrix} 80e^{40u_3} & 0 & -39.2e^{40u_5} & 0 \\ 0 & 80e^{40u_4} & 0 & -39.2e^{40u_6} \\ -39.2e^{40u_3} & 0 & 40e^{40u_5} & 0 \\ 0 & -39.2e^{40u_3} & 0 & 40e^{40u_6} \end{bmatrix}$$

The matrix $\frac{\partial}{\partial u} M\rho(u)$ is needed for the evaluation of the Jacobian of the system of equations (4.15).

For C = 200 picofarads and initial values

$x_1 = 0.$ coulombs

$x_2 = -500.$ coulombs

$u_3 = 0.$ volts

$u_4 = -10.$ volts

$u_5 = 0.$ volts

$u_6 = 0.$ volts

the first almost-discontinuous segment occurs between

approximately t = 0 nanoseconds and t = 3 ns. The "correct" solution at t = 3 ns was obtained by applying global extrapolation to the solutions that were calculated by means of the trapezoidal rule with step sizes of 1/512 ns and 1/1024 ns.

In the tables below, the tabulated relative error in each variable (difference between the calculated and "correct" solution divided by the "correct" solution for each variable) in the solution were calculated by means of the generalized trapezoidal rule and the trapezoidal rule for various maximum step sizes. The exponential functions needed for the generalized trapezoidal rule were evaluated by means of the E_{22} Pade approximation with s = 4.

Table 4.1a -- Error in x_1 (charge on capacitor C_1)

Method	h_{max}	Relative error	Number of iterations
Trap.rule	1/128 ns	2.0×10^{-3}	1664
Trap.rule	1/256 ns	2.6×10^{-3}	1734
Trap.rule	1/512 ns	1.6×10^{-3}	3223
Trap.rule	1/1024 ns	4.2×10^{-4}	6310
Gen.trap.rule	1/32 ns	4.6×10^{-3}	737

Table 4.1b -- Error in x_2 (charge in capacitor C_2)

Method	h_{max}	Relative error	Number of iterations
Trap.rule	1/128 ns	6.3×10^{-4}	1664
Trap.rule	1/256 ns	6.1×10^{-4}	1734
Trap.rule	1/512 ns	4.3×10^{-4}	3223
Trap.rule	1/1024 ns	1.8×10^{-4}	6310
Gen.Trap.rule	1/32 ns	6.9×10^{-4}	737

Table 4.1c -- Error in u_3 (collector voltage on transistor T_1)

Method	h_{max}	Relative error	Number of iterations
Trap.rule	1/128 ns	1.2×10^{-3}	1664
Trap.rule	1/256 ns	1.2×10^{-3}	1734
Trap.rule	1/512 ns	5.5×10^{-3}	3223
Trap.rule	1/1024 ns	1.4×10^{-3}	6310
Gen.Trap.rule	1/32 ns	1.9×10^{-4}	737

Table 4.1d -- Error in u_4 (Collector voltage on transistor T_2)

Method	h_{max}	Relative error	Number of iterations
Trap.rule	1/128 ns	1.1×10^{-3}	1664
Trap.rule	1/256 ns	1.0×10^{-3}	1734
Trap.rule	1/512 ns	7.8×10^{-4}	3223
Trap.rule	1/1024 ns	1.4×10^{-4}	6310
Gen.trap.rule	1/32 ns	5.5×10^{-4}	737

Table 4.1e -- Error in u_5 (Emitter voltage on transistor T_1)

Method	h_{max}	Relative error	Number of iterations
Trap.rule	1/128 ns	4.0×10^{-5}	1664
Trap.rule	1/256 ns	2.2×10^{-5}	1734
Trap.rule	1/512 ns	2.4×10^{-5}	3223
Trap.rule	1/1024 ns	6.7×10^{-6}	6310
Gen.trap.rule	1/32 ns	2.3×10^{-5}	737

Table 4.1f -- Error in u_6 (Emitter voltage on transistor T_2)

Method	h_{max}	Relative error	Number of iterations
Trap.rule	1/128 ns	5.9×10^{-4}	1664
Trap.rule	1/256 ns	3.5×10^{-5}	1734
Trap.rule	1/512 ns	3.6×10^{-5}	3223
Trap.rule	1/1024 ns	9.0×10^{-6}	6310
Gen.trap.rule	1/32 ns	3.4×10^{-5}	737

From the above tables, it can be seen that the generalized trapezoidal rule with a maximum step size of 1/32 ns produced a solution whose error was less than the error for the trapezoidal rule with a step size of 1/512 ns but not as accurate as the latter with a step size of 1/1024 ns. Moreover, the generalized trapezoidal rule required approximately 1/5 as many iterations over the interval t = 0 to t = 3 ns as did the trapezoidal rule with h_{max} = 1/512 ns. As was the case with the third example in section III.5, for a given h_{max} the generalized trapezoidal rule and the trapezoidal rule produced solutions with approximately the same accuracy for the "slow" variables (x_1 and x_2), but the generalized trapezoidal rule produces more accurate solutions for the "fast" variables (u_3, u_4, u_5, and u_6).

V. Conclusions and Suggestions for Further Research

1. Conclusions

The research reported herein developed a new numerical integration technique for solving stiff systems of ordinary differential equations. The method, which is called the generalized trapezoidal rule, is a modification of the trapezoidal rule. However, the new method is computationally more efficient than the trapezoidal rule when the solution of the almost-discontinuous segments is being computed.

The computation of the solution of the system of ordinary differential equations is divided into two parts. For the computation of the smooth segments of the solution, the trapezoidal rule with Newton-Raphson iterations to solve the implicit equations is used. However, during the almost-discontinuous segments, the generalized trapezoidal rule with both Newton-Raphson and modified Newton-Raphson iterations to solve the implicit equations is used to calculate the solution. The details of the computation for both types of segments is found in sections III.1a and III.1b.

As pointed out in section III.4, the detection of whether one is calculating a smooth or an almost-discontinuous segment is at best a difficult task. The detection method based on the step size chosen by the integration scheme that was used

in the research presented herein can be hazardous if the user has specified a "very large" maximum step size, h_{max}. It is hazardous in the sense that the integration procedure might choose the generalized trapezoidal rule instead of the trapezoidal rule during the smooth segments. However, during the smooth segments, the generalized trapezoidal rule is not as computationally efficient as the trapezoidal rule. For a given step size, both the trapezoidal rule and the generalized trapezoidal rule produce solutions with approximately the same accuracy. Since the generalized trapezoidal rule requires more work per step than the trapezoidal rule, the latter should be used for the calculation of smooth segments of the solution.

Therefore, care must be taken not to specify a value for h_{max} that will be much larger than the step size chosen by the step size control procedure used with the trapezoidal rule. If the user does not have experience in choosing a proper maximum step size for a particular problem, he should do some preliminary calculations with the trapezoidal rule and observe the step sizes selected.

The results discussed in sections III.3 (Theoretical Error Calculations), III.5 (Illustrative Computer Results), and IV.3 (Computational Results for a Multivibrator Circuit) indicate that the generalized trapezoidal rule is computationally more efficient than the trapezoidal rule for computing the solution during an almost-discontinuous segment of the

solution. That is, the generalized trapezoidal rule enables one to use larger step sizes and to do fewer iterations to solve the implicit integrating equations during the almost-discontinuous segments than the trapezoidal rule while maintaining the same or improved accuracy as compared to the trapezoidal rule. Also, the numerical solution produced by the generalized trapezoidal rule did not have an oscillatory behavior at step sizes for which the trapezoidal rule produced an oscillating solution (even though the actual solution was decaying exponentially).

The generalized trapezoidal rule is not meant for obtaining very fine detailed solutions during the almost-discontinuous segments. The basic aim of the scheme is to take larger time steps than is possible with the trapezoidal rule while maintaining a predetermined accuracy in the solution. Consequently, if one needs very fine details of the solution during the almost-discontinuous segments, the trapezoidal rule, or perhaps step length limited schemes such as explicit Runge-Kutta methods should be used instead of the generalized trapezoidal rule.

From the foregoing remarks and the results of the previous chapters, it may be concluded that for stiff systems of ordinary differential equations even though the generalized trapezoidal rule requires more work per iteration than the trapezoidal rule, the overall computational effort needed to compute the almost-discontinuous segment of the solution

is less for the generalized trapezoidal rule.

2. Suggestions for Further Research

The research presented herein suggests several important questions that should be pursued further. The applicability of an exponential shift in variables to other A-stable schemes should considered. Of particular interest would be the effects of the transformation on the concept of exponential fitting [17], and on the implicit midpoint scheme [22].

One would like to know whether the exponential transformation would improve the computational efficiency of exponential fitting. The exponential transformation has a tendency to compress the eigenvalues, especially the larger ones; whereas, the exponential fitting method tries to approximate the decay of the larger eigenvalues. It is not clear how a combination of the two ideas would perform in terms of computational efficiency.

The implicit midpoint scheme is closely related to the trapezoidal rule and it is quite likely that exponential time shifts would improve the computational efficiency in about the same way that the transformation improves the efficiency of the trapezoidal rule. However, this remains to be demonstrated.

The application of the exponential transformation to the stiffly stable multi-order scheme of Gear [8] is also of considerable interest. The compressing effect on the eigenvalues of the exponential transformation might alleviate the problem of poorly situated eigenvalues (see section II.8). Also, it would be interesting to note the effect of the transformation on the order of the scheme selected by the automatic procedure.

A third direction for further research is to find the effect of the transformation on the h^2 nature of the error expansion for the trapezoidal rule. The fact that the error terms for the trapezoidal rule can be expressed in powers of h^2 allows for a great increase in the computational efficiency of global extrapolation. Thus, the question to be examined is how the exact exponential transformation and the various Pade approximations (inconjunction with the usual argument reduction scheme) change the basic form of the error expansions for the trapezoidal rule.

VI. References

[1] G. Bjurel, G. Dahlquist, B. Lindberg, S. Linde & L. Oden, "Survey of Stiff Ordinary Differential Equations," Report NA 70.11 Department of Computer Science, Royal Institute of Technology, Stockholm, Sweden

[2] J.C. Butcher, "Implicit Runge-Kutta Processes," Mathematics of Computation, 18, (1964),#85, pp50-64

[3] G. Dahlquist, "A Special Stability Problem for Linear Multistep Methods," BIT (Nordisk Tidskrift for Information-behandling) 3, (1963), pp 27-43

[4] G. Dahlquist, "Stability Questions for Some Numerical Methods for Ordinary Differential Equations," Proceedings of the Symposium of Applied Mathematics, 15 , (1963)

[5] G. Dahlquist and B. Lindberg, Private communications (1971)

[6] B. Ehle, "The Application of Implicit Runge-Kutta Processes to Stiff Systems," Research Report CSRR 2007, University of Waterloo, Waterloo, Ontario, Canada

[7] M. Fowler & R. Warten, "A Numerical Integration Technique for Ordinary Differential Equations with Widely Separated Eigenvalues," IBM Journal of Research and Development, September 1967, pp 537-543

[8] C.Gear, "The Automatic Integration of Stiff Ordinary Differential Equations," IFIPS Conference 1968 (Edinburgh, Scotland)

[9] G. D. Hachtel & R. A. Rohrer, "Techniques for Optimal Design and Synthesis of Switching Circuits," Proceedings of the IEEE, 55, #11, (November 1967), pp 1864-1877

[10] P. Henrici, Discrete Variable Methods in Ordinary Differential Equations, J. Wiley & Sons, (1962)

[11] P. Henrici, Error Propagation for Difference Methods, J. Wiley & Sons, (1964)

[12] T. E. Hull, "The Numerical Integration of Ordinary Differential Equations," IFIPS Congress 1968 (Edinburgh, Scotland)

[13] J. D. Lawson, "Generalized Runge-Kutta Processes for Stable Systems with Large Lipschitz Constants," SIAM Journal of Numerical Analysis, 4, (1967), #3, pp 372-380

[14] L. Lapidus & J. Seinfeld, Numerical Solution of Ordinary Differential Equations, Academic Press (1971)

[15] B. Lindberg, "On Smoothing and Extrapolation for the Trapezoidal Rule," BIT, 11, (1971), pp29-52

[16] W. Liniger, "A Stopping Criterion for the Newton-Raphson Method in Implicit Multistep Integration Algorithms for Nonlinear Systems of Ordinary Differential Equations," Communications of the ACM, 14,#9, (September 1971), pp 600-601

[17] W. Liniger & R. Willoughby, "Efficient Numerical Integration of Stiff Systems of Ordinary Differential Equations," IBM Research Report RC 1970 (December 1970)

[18] S. J. Oh, T. E. Stern & H. E. Meadows, "On the Analysis of Non-Linear Irregular Networks," Symposium on Generalized Networks, Polytechnic Institute of Brooklyn, April 1966

[19] J. Rosenbaum, "Pade Approximations to the Exponential Functions Exp (-z) and Exp (-At), " Technical Memorandum, Department of Electrical Engineering, Columbia University, (December 1969)

[20] T. E. Stern, "Computer Aided Analysis of Non-Linear Networks," Systems Research Group, Technical Report # 107, Department of Electrical Engineering, Columbia University, (July 1968)

[21] T. Stern, "Efficient Computer Aided Analysis of Non-Linear Semiconductor Networks," IEEE Mexico Conference (January 1971)

[22] H. J. Stetter, Private communication through G. Dahlquist (1971)

[23] W. Tinney & J. Walker, "Direct Solution of Sparse Network Equations by Optimally Ordered Triangular Factorization," Proceedings of the IEEE, 55,#11, (November 1967) pp 1801-1809

[24] O. Widlund, "A Note on Unconditionally Stable Linear Multistep Methods," BIT, 7, (1967), pp 65-70

[25] R. S. Varga, Matrix Iterative Analysis, Prentice Hall (1962)

[26] F. H. Branin, Jr., Private communications (1971)

[27] C. Gear, Private communications (1971)

[28] J. Millman and H. Taub, Pulse Digital and Switching Waveforms, McGraw Hill (1965)

VII. Appendix

On the next several pages there is a listing of the PL/I program used to calculate the results reported herein. The program is set up for a maximum of 6 equations, but the declaration statements can easily be changed to accomodate any other number of equations.

```
(NOUNDERFLOW):GTRAP:PROCEDURE OPTIONS(MAIN);
/* GENERALIZED TRAPEZOIDAL RULE INTEGRATION*/
DECLARE (GI(6),GF(6),ZI(6),ZF(6),ZGS(6),X(6),INITIAL(6),
  V(6),JACOB(6,6),LJACOB(6,6),K(6,6),EKH(6,6),EMKH(6,6),
        H,T,TI,TF,TSHIFT,FAC) BINARY FLOAT;
NEWDATA: GET LIST (N,(INITIAL(I) DO I= 1 TO N));
X=INITIAL;
ZI=X;
PUT PAGE EDIT (' INITIAL VALUES=',ZI) (A,6 E(15,7));
PARTDATA:GET LIST (TZERO,TMAX,HMAX);
PUT SKIP DATA (TZERO,TMAX,HMAX);
 EPSIL=.00001;
 H=HMAX/16. ;
 ITERTOTAL=0;
 PUT SKIP DATA (EPSIL,H) ;
/*SET UP THE TIME FOR THE INITIAL POINT*/
 TI=TZERO;
/* SET UP TIME SHIFT */
 START:T=0. ;
     TSHIFT=TI;
     IF ((TI+H) > TMAX) THEN H=TMAX-TI;
     ITERSUM=0;
     ZI=X;
/*OBTAIN FUNCTION VALUES AT THE INITIAL POINT */
     CALL CALCJ(K,ZI,T);
     CALL CALCG(GI,ZI,X,T,TSHIFT,K);
/*SET UP FOR ITERATIONS*/
ITERATION: TF=TI+H;
/* GET PADE APPROXIMIONS FOR THE CURRENT STEP */
     CALL PADE(EMKH,K,H,N); /*EMKH=EXP(-KH)*/
     CALL MTXINV(EKH,EMKH,N,DET); /*EKH=EXP(+KH)*/
     ZF=ZI;
     ITER=0;
ITERLOOP:ITER=ITER+1;ITERSUM=ITERSUM+1;
     IF (ITER <= 5) THEN  GO TO CALCULATE;
     PUT SKIP DATA (H);
     H=H/2.; GO TO ITERATION;
CALCULATE:CALL CALCG(GF,ZF,X,H,TSHIFT,K);
     IF (ITER > 1) THEN GO TO WORK;
     ZGS=ZF+(H/2.)*(GI+GF);
     GO TO ERROR;
WORK:CALL CALCJ(JACOB,X,TF);
  /*IMPLIMENT TRAPEZOIDAL RULE FORMULA*/
     FAC=H/2.;
     LJACOB=-FAC*(JACOB-K);
     DO I=1 TO N;
      LJACOB(I,I)=1.+LJACOB(I,I); END;
     CALL MTXINV(LJACOB,LJACOB,N,DET);
     V=ZF-ZI-FAC*(GI+GF);
     CALL MTXVEC(ZGS,EKH,V,N);
     CALL MTXVEC(V,LJACOB,ZGS,N);
     CALL MTXVEC(ZGS,EMKH,V,N);
```

```
    ZGS=ZF-ZGS;
/*SEE IF ITERATION HAS CONVERGED YET*/
ERROR:DELTA=0.; XNORM=0.;
    DO I=1 TO N;
     DELTA=DELTA+ABS(ZGS(I)-ZF(I));
     XNORM=XNORM+ABS(ZF(I)); END;
    IF (XNORM ¬= 0.) THEN DELTA=DELTA/XNORM;
    ZF=ZGS;
    IF (DELTA <= EPSIL) THEN GO TO CONVERGED;
                        ELSE GO TO ITERLOOP;
CONVERGED: ZI=ZGS;
    ITERTOTAL=ITERTOTAL+ITERSUM;
    PUT SKIP DATA (TF,H,ITERSUM,ITERTOTAL);
    PUT SKIP EDIT (' Z=',ZI) (A,6 E(15,7));
    CALL MTXVEC(X,EKH,ZI,N);
    PUT SKIP EDIT (' X=',X) (A,6 E(15,7));
  SKIP:TI=TF;
    IF (ITERSUM > 2) THEN GO TO DONEYET;
    H=2.*H;
    IF (H > HMAX) THEN H=HMAX;
DONEYET: IF (TF >= TMAX-1.E-6*TMAX) THEN GO TO PARTDATA;
                                    ELSE GO TO START;
/*****************************************************/
/*FUNCTION VALUES FOR TRANSFORMED EQUATIONS Z'=G(Z,T)*/
/*     G(Z,T)=EXP(-KT)*F(EXP(KT)*Z,T) - K*Z      */
CALCG:PROCEDURE (ANS,Z,X,T,TSHIFT,K);
       DECLARE (ANS(6),Z(6),X(6),K(6,6),W(6))BINARY FLOAT;
       DECLARE (T,TSHIFT,TP) BINARY FLOAT;
       IF (T ¬= 0. ) THEN GO TO NONZERO;
       CALL CALCF(ANS,Z,TSHIFT);
       CALL MTXVEC(W,K,Z,N);
       ANS=ANS-W;
       RETURN;
     NONZERO: TP=T+TSHIFT;
       CALL MTXVEC(X,EKH,Z,N);
       CALL CALCF(ANS,X,TP);
       CALL MTXVEC(W,EMKH,ANS,N);
       ANS=W;
       CALL MTXVEC(W,K,Z,N);
       ANS=ANS-W;
       RETURN;
END CALCG;
/*  MATRIX OPERATIONS PACKAGE  */
MTXOPS:PROCEDURE ;
       DECLARE (A(6,6),B(6,6),C(6,6),V(6),W(6),X(6,6))
                BINARY FLOAT;
IDMTX :ENTRY (A,N);
       A=0.;
       DO I=1 TO N;
        A(I,I)=1.;END;
       RETURN;
MTXMLT:ENTRY (A,B,C,N);
```

```
        A=0.;
        DO I = 1 TO N; DO J = 1 TO N;
         A(I,J)=SUM(B(I,*)*C(*,J)); END; END;
        RETURN;
MTXPWR:ENTRY (B,C,N) ;
       DO I = 1 TO N; DO J = 1 TO N;
        X(I,J)=SUM(B(I,*)*C(*,J)); END; END;
        B=X;
        RETURN;
MTXINV:ENTRY (A,B,N,DET);
        /*A = B INVERSE */
        A=B; CON=0.;
        CALL MINV(A,N,DET,CON);
        IF (DET = 0.) THEN PUT SKIP EDIT
                          ('SINGULAR MATRIX')(A);
        RETURN;
MTXVEC:ENTRY (V,A,W,N);
        DO I = 1 TO N;
        V(I)=SUM(A(I,*)*W(*)); RETURN;
END MTXOPS;
/* 2,2 PADE APPROXIMATION TO EXP(-KT) */
PADE:PROCEDURE (EMKT,K,T,N);
        DECLARE (EMKT(6,6),K(6,6),NUM(6,6),DEN(6,6),
         W(6,6),WSQ(6,6),C1,C2,T) BINARY FLOAT;
        W=(2.**-4)*T*K;
        CALL MTXMLT(WSQ,W,W,N);
        C1=.5000000000000000E+0;
        C2=.0833333333333333E+0;
        NUM=-C1*W+C2*WSQ;
        DEN= C1*W+C2*WSQ;
        DO I=1 TO N;
         NUM(I,I)=1.+NUM(I,I);
         DEN(I,I)=1.+DEN(I,I);END;
        CALL MTXINV(DEN,DEN,N,DET);
        CALL MTXMLT(EMKT,DEN,NUM,N);
       DO I = 1 TO 4;
         CALL MTXPWR(EMKT,EMKT,N); END;
        RETURN;
END PADE;
FUNCTIONS: PROCEDURE;
    DECLARE (JACOB(6,6),ANS(6),X(6)) REAL FLOAT;
/* CALCULATE THE JACOBIAN AT THE GIVEN POINT */
    CALCJ: ENTRY(JACOB,X,T);
    /* INSERT A ROUTINE TO CALCULATE THE
       JACOBIAN HERE */
    RETURN;
    CALCF: ENTRY (ANS,X,T);
    /* INSERT A ROUTINE TO EVALUATE THE RIGHT HAND
       SIDE OF THE EQUATION HERE */
    RETURN
END FUNCTIONS;
END GTRAP;
```