ERROR CONTROL FOR DESCRIPTOR SYSTEMS

by

GEORGE ROBERT MANN

B. S., Pittsburg State University, 1975

A MASTERS REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1981

Approved by:

[illegible]

Major Professor

TABLE OF CONTENTS

INTRODUCTION

This report examines some techniques which are useful in solving descriptor systems. A descriptor system is an implicit system of ordinary differential equations (ODEs) coupled to a system of algebraic equations. Sophisticated programs for solving systems of ODEs have been available for many years. But the presence of algebraic equations in a descriptor system causes problems for programs which are designed to solve systems of ODEs. One of the major problems is in error control. I will discuss an error filtering mechanism proposed by Sincovec (1) which enables a program to perform seperate error control on the algebraic and ODEs. A discussion of the implementation of the error filtering mechanism and an analysis of the results for some sample problems will also be given. Some other topics which are relevant to numerical methods for solving descriptor systems will also be discussed. These topics include

1. Recovery from inconsistent initial conditions
2. Interpolation problems
3. Nonlinear problems

SCOPE OF RESEARCH

Descriptor Systems

A descriptor system is an implicit system of algebraic equations coupled to a system of ODEs. Thus, a linear descriptor system can be written in the form $Ey'=Ay+g$ where E and A may be singular. This report examines some techniques which are useful for solving descriptor systems.

Linearity

In order to see whether Sincovec's error filtering mechanism has any beneficial effects for nonlinear problems, I ran one nonlinear problem with and without the error filter. Since the solutions were accurate in both cases, I was unable to draw any conclusions from this experiment. So the remainder of my research effort on error control was devoted to linear problems. I also used the same nonlinear problem to test a recovery technique for problems with inconsistent initial conditions. The nonlinear problem and the results from this experiment are given in the section on initial conditions.

Initial Conditions

In the formulation of a descriptor system, it may be impossible or impractical to derive a set of exact initial conditions. If the given set of initial conditions do not satisfy a solution to the descriptor system then the initial conditions are inconsistent with the descriptor system. In the section on initial conditions, I will discuss a theorem proved by Sincovec (1) which describes a way to recover from inconsistent initial conditions. I will also give results from a problem which I ran that will illustrate the recovery from inconsistent initial conditions.

Error Control

The algebraic equations in a descriptor system cause problems for an error control procedure which was designed for systems of ODEs. In the section on error control, I will discuss and illustrate the nature of these problems and an error filter which will enable a program to perform seperate error control on the algebraic and differential equations. The implementation of the error filtering mechanism will be discussed and the results from a sample problem will be given and explained.

Implementation

I implemented the error filter as a modification to a program writtem by Alan Hindmarsh (2). The program uses Gear's k step method to solve descriptor systems of the form $Ey' = Ay + g$. Gear's k step method is defined in (3).

In addition to the modification of the error control procedure of Hindmarsh's program, I wrote two subroutines for the program and two additional procedures for each problem I ran to define the problem and its parameters and to print out the results.

INITIAL CONDITIONS

Sincovec proves a theorem in (1) which will allow a program to recover from inconsistent initial conditions. The theorem states that in solving a problem with inconsistent initial conditions, Gear's method will converge to the analytic solution of a descriptor system after n steps where n is greater than or equal to the nilpotency of the system. (The nilpotency of a descriptor system will be defined later.) So, the recovery technique for inconsistent initial conditions is to simply take n >= nilpotency steps. In (1), Sincovec also describes a way to calculate a consistent set of initial conditions for a problem so that the validity of the solution can be assesed.

I ran two problems with inconsistent initial conditions. The first was a linear problem of the form

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} y' = \begin{bmatrix} 1 & 1 \\ 2 & 0 \end{bmatrix} y + \begin{bmatrix} 0 \\ 5 \end{bmatrix}$$

Given the inconsistent initial conditions y1(1) = 2 and y2(1) = 1, the program produced accurate results on the first step. (The nilpotency of this system is zero.)

The second problem I ran was a nonlinear problem of the form

$$\begin{bmatrix}1 & 0 & 0\\ 0 & 0 & 0\\ 0 & 0 & 1\end{bmatrix} y' = \begin{bmatrix} y2*y3 \\ y1 - t**10 \\ 10t**9 \end{bmatrix}$$

The program produced good results for this problem with consistent initial conditions, but it could not achieve corrector convergence with inconsistent initial conditions. Thus, this indicates that Sincovec's theory on initial conditions only holds for linear problems.

ERROR CONTROL

Structure of Descriptor Systems

In order to facilitate the discussion of error control, we need to examine the structure of descriptor systems. A linear descriptor system can be written in the form

Ey' = Ay + g (4.1)

where the matrix E is singular. The singularity of E implies that the system 4.1 contains an algebraic subsystem. For example, the system might contain an algebraic equation like y1 = t**10. In a general descriptor system, the algebraic equations are not easy to identify since they may be expressed as a linear combination of the other equations. But the system 4.1 can be transformed into canonical form where the algebraic equations are seperated out into an independent subsystem.

In general, the transformationn of a descriptor system into canonical form can be accomplished by the equation

PEQ INVERSE(Q) y'(t) = PAQ INVERSE(Q) y(t) + P g(t)

where P and Q are appropriate nonsingular matrices (1). The canonical representation of Ey' = Ay + g contains two (solvable)independent subsystems:

STATE SUBSYSTEM:

$$x1'(t) = E1\ x1(t) + f1(t) \tag{4.2.a}$$

NONSTATE SUBSYSTEM:

$$E2\ x2'(t) = x2(t) + f2(t) \tag{4.2.b}$$

In this representation, x1, x2, f1, and f2 are vectors which are related to the orignial system by the equations

$$\text{INVERSE}(Q)\ y(T) = \begin{bmatrix} x1(T) \\ x2(T) \end{bmatrix}$$

$$P\ g(T) = \begin{bmatrix} f1(T) \\ F2(T) \end{bmatrix}$$

Since x1, x2, f1, and f2 are vectors, the notation xk,i (k = 1,2) will represent component i of xk and fk,i (k = 1,2) will represent component i of fk.

The matrix E2 of the non-state subsystem is composed of Jordan blocks which are of the form

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

The nilpotency (m) of the system is defined to be the size of the largest Jordan block in E2.

Because of the special structure of E2, we can solve the system 4.2.b by backward substitution and differentiation. For example, the nonstate subsystem

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} x2' = \begin{bmatrix} x2,1 + f2,1 \\ x2,2 + f2,2 \\ x2,3 + f2,3 \end{bmatrix}$$

has a solution of

$$x2 = \begin{bmatrix} -f2,3'' - f2,2' - f2,1 \\ -f2,3' - f2,2 \\ -f2,3 \end{bmatrix}$$

Thus, the non-state subsystem is like an algebraic system.

Truncation Error in Descriptor System Solutions

The error control components in most ODE solvers assume that the truncation error at a given step is propagated to later steps. An error control procedure (ECP) which is based on this assumption must place a much tighter error tolerance on each step than the global error tolerance which is expected after the final step. This strategy is reasonable for the control of error in a typical ODE system, but it is not appropriate for descriptor systems, where some of the variables may not contain propagated truncation errors. This can be illustrated by applying a backward Euler step to the system

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} y' = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} y + \begin{bmatrix} -t{*}{*}10 \\ 2t \\ -t{*}{*}10 + 2t \end{bmatrix} \qquad (4.3)$$

A backward Euler step on the descriptor system $Ey' = Ay + g$ is defined by the equation

$$(E-hA)\, y(n+1) = E\, y(n) + h\, g(n+1) \qquad (4.4)$$

After n backward Euler steps on the system 4.3 we obtain

$$\begin{bmatrix} 1-h & -h & 0 \\ 1 & -h & 1 \\ -h & 0 & 1 \end{bmatrix} y(n+1) = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} y(n) + h \begin{bmatrix} -t{*}{*}10 \\ 2t \\ -t{*}{*}10 + 2t \end{bmatrix}$$

From this equation, it appears as though the truncation errors from y1(n) will be propagated to y1(n+1) and y2(n+1) and the truncation errors from y3(n) will be propagated to y2(n+1) and y3(n+1). But the solution is given by y(n+1) = INVERSE(E - hA) * (E y(n) + h g)

So

$$y(n+1) = \begin{bmatrix} -1/(2h) & 1/(2h) & -1/(2h) \\ (-1-h)/(2h**2) & (1-h)/(2h**2) & (-1+h)/(2h**2) \\ -1/2 & 1/2 & 1/2 \end{bmatrix}$$

* (Ey(n) + hg).

After simplifying,

$$y(n+1) = \begin{bmatrix} t**10 \\ (t**10-y1(n))/h = (y1(n+1)-y1(n))/h \\ y3(n) + 2ht \end{bmatrix}$$

So, we can see that there is no truncation error on y1(n+1) since it is calculated exactly as t**10. The truncation error on y2 will not be propagated since y2(n) is not present as a term in the solution at time t = n+1. But the truncation error on y3(n) will be propagated to y3(n+1).

An equivalent canonical representation for the system 4.3 is given by

$$\begin{bmatrix}0&0&0\\1&0&0\\0&0&1\end{bmatrix} y' = \begin{bmatrix}1&0&0\\0&1&0\\0&0&0\end{bmatrix} y + \begin{bmatrix}\mathtt{-t**10}\\0\\2t\end{bmatrix}$$

In this form, we can see that the independent algebraic subsystem contains the equations

y1 = t**10

y2 = y1'

Therefore, this example shows that

1. Truncation errors from the algebraic subsystem are not propagated to the system of ODEs.

2. Some of the solutions to the algebraic subsystem may not contain propagated truncation errors. (In this particular example, neither of the two solutions contained propagated truncation errors.)

A typical error control procedure (ECP), such as the one used in Hindmarsh's program (2) accepts a step if the sum of a function of the error estimates for all variables in the system is less than a local error bound (LB). The value of this local error bound is dependent on factors such as the desired accuracy of the solutions, the size of the system, and the order of the method.

If the ECP assumes that all of the solution components contain propagated truncation errors, then the LB must be tight enough so that the propagated truncation errors will not cause a large global error. In other words, the LB must be much more stringent than the desired global accuracy. But, as I have shown in the previous section, some of the solutions to the algebraic subsystem may not contain propagated truncation errors. So an ECP which assumes that all components of a solution contain propagated truncation errors will attempt to place a much tighter LB on some components of a descriptor system solution than the specified global error tolerance. In fact, the components of a solution which do not contain propagated truncation errors need to be no more accurate at a given step than the global error tolerance.

This effect is even more significant when you consider the size of the truncation errors. If we start with exact initial values and calculate the truncation errors on a system of the form Ey' = Ay + g then the error after one backward Euler step is

(INVERSE(E-hbA))E(h**2/2) y"(T) (4.5)

where to <= T <= t1 (4)

For a system like 4.3, the truncation error is

$$\begin{bmatrix} 0 \\ -h/2 * y1''(T) \\ -h{*}{*}2/2 * y3''(T) \end{bmatrix} \qquad (4.6)$$

So, we can see that the truncation error in y2 is O(h) while the truncation error on y3 is O(h**2).

This observation has important consequences for an ECP which assumes that all variables have propagated truncation errors since an ECP of this type will be placing a very stringent LB on the variables which do not propagate truncation errors. And if the errors in these variables are larger than the errors in the other variables (This difference is potentially several orders of magnitude.) then the error test will only pass with a very large LB. So the accuracy of the results for variables which do contain propagated truncation errors will be as weak as the LB.

It should be noted here that if the ECP calculates error estimates which are based on h**2, then the ECP will underestimate the actual errors in the algebraic equations. This underestimation could partially negate the effect of the other invalid assumption that non-state variables will contain propagated truncation errors. But in any case, such an error estimate would be inaccurate for the non-state variables. In fact, any ECP which assumes that the size and/or propagation of errors is the same for algebraic equations as it is for ODEs will be inappropriate for error control in a descriptor system.

In order to illustrate these effects, I used Hindmarsh's program to run problem 4.3. Then, I added Sincovec's error filter and ran the same problem. The results are summarized in Table 1.

TABLE 1

	STANDARD ECP			SINCOVEC'S ECP		
TIME	EY1	EY2	EY3	EY1	EY2	EY3
1.2	0	.002	.127	0	.002	.000001
1.4	0	.000003	.228	0	.005	.000001
1.6	0	.000001	.262	0	.004	.000001
1.8	0	.000002	.248	0	.003	.000001
2.0	0	.000002	.210	0	.009	.000001
LB:		.1E-2			.1E-6	
STEPS:		3654			80	

(Eyi indicates relative error in component i of y)

Without the error filter, the program had to use a very large local bound (.1e-2) and a very small stepsize in order to pass the error test. (The error test fails if a smaller LB is used.)

As a result, the accuracy on variables y1 and y2 is much better than LB was meant to require and the accuracy on y3 is poor. In contrast, by using Sincovec's error filter, the errors in y1, y2, and y3 satisfy a much tighter LB (.1E-6). And, as the number of steps indicates, the amount of computation with Sincovec's error filter is a small fraction of that which is required without it.

It should be noted here that Sincovec's error filter (hereafter refered to as the EF) does not eliminate error control problems for descriptor systems. The EF will enable a program to partition the solutions of a descriptor system into two sets. One set contains the solutions to a standard system of ODEs in which there is not a large difference in the size of the errors, and all of the variables contain propagated truncation errors. Thus, a standard ECP will work well for this set. The second set contains solutions to a system of algebraic equations where the solutions are defined by derivatives. If this subsystem has $n > 2$ equations, then the approximation of derivatives can generate truncation errors on some of the solutions which are propagated to later steps. So, the general problem with propagation of truncation errors is still present in this subsystem. But, at least the error filter will enable a program to calculate accurate solutions for the system of ODEs. And even though there are some difficult problems in solving the algebraic subsystem, this subsystem should be easier to solve if we know which of the solutions it defines. This is discussed furthur in the next section.

An Error Filter for Non-state Components

Traditional ODE solvers have had little success in solving descriptor systems because the behavior of the algebraic subsystem is so much diferent from that of the ODEs. One area where this behavior differs radically is in the size and propagation of truncation errors. As I have shown previously, some of the variables in the algebraic subsystem may not contain propagated truncation errors. And the errors in these variables can be much larger than the errors in variables which do contain propagated truncation errors.

Until recently, there has been no practical way to determine whether a given variable in the solution to a descriptor system belongs to the ODEs or the algebraic equations. In theory, a descriptor system can be transformed into canonical form, where the algebraic equations are independent of the ODEs, but this transformation is much too expensive to be practical. (The amount of computation required to convert a general descriptor system with n equations to canonical form in on the order of n**4.) But the error filter proposed by Sincovec (1) will enable a program to determine which components of a solution correspond to the ODEs and which belong to the algebraic equations. Thus, this error filter will facilitate the implementation of separate error control procedures for the ODEs and the algebraic equations.

The EF is a matrix

```
   M = INVERSE(E - hbA)E)**k
```

where k is greater than or equal to the nilpotency m, and b is a constant. When m is multiplied by a vector of error estimates for the descriptor system 4.1, the error estimates which correspond to solutions for the nonstate subsystem are filtered out (They are set to zero.)

The matrix M would be difficult to calculate if it were not for the fact that Gear's k step method (3) uses the matrix INVERSE(E - hbA). This matrix is available with no additional calculations. So the calculation of M is relatively easy.

Since the error filter M is multiplied by a vector e0 of error estimates for the current step, I have implemented the error filtering mechanism with the following procedure:

M*e0 = INV(E - hbA) ** k * e0

STEP1:

M*e0 = (INV(E-hbA)E)**(k-1)*(INV(E-hbA)E*e0)

Let e1 = (INV(E - hbA) E e0

Then, calculate e1 by solving the system

(E - hbA) e1 = E e0.

(Note that if INV(E - hbA) has been factored into a pair of lower and upper triangular matrices - as it Hindmarsh's program - then the calculation of e1 can be accomplished with a forward and backward substitution.)

Now M e0 = (INV(E - hbA))**(k-1) * e1

STEPi: (1 <= i <= k)

M e0 = (INV(E-hbA))**(k-i)*INV(E-hbA)E e(i-1)

Let ei = INV(E - hbA)E e(i-1)

Solve (E - hbA) e1 = E e(i-1)

Then M e0 = (INV(E - hbA) ** (k-i)) ei

In order to illustrate the error filtering mechanism, I will calculate the EF for problem 4.3:

$$\text{INV}(E - hbA) = \begin{bmatrix} -1/h & 0 & 0 \\ -1/h**2 & -1/h & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Since the nonstate subsystem has nilpotency 2,

$$M = (\text{INV}(E - hbA)E)**2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

If the error estimates at the current step are

$$e = \begin{bmatrix} e1 \\ e2 \\ e3 \end{bmatrix}$$

$$\text{Then } M\,e = \begin{bmatrix} 0 \\ 0 \\ e3 \end{bmatrix}$$

Thus, the error estimates for the non-state subsystem variables are set to zero.

<u>Error Control for the Nonstate Subsystem</u>

The error filtering mechanism will enable a program to perform separate error tests on the algebraic equations and the differential equations. If the error estimates for the descriptor system are contained in a vector e then the error on the ODEs is e(ODE) = M*e and the error on the algebraic equations is e(alg) = e - M*e. The vector e(ODE) can be used by a conventional error control procedure to control the error on the ODEs. But some problems remain in controlling error on the algebraic subsystem. These problems include

1. Propagation of truncation errors
2. Special problems for systems with nilpotency >= 3.

3. Determination of the nilpotency.

4. Interpolation.

A discussion of these problems follows.

Propagation of truncation errors: One of the problems in error control for the nonstate subsystem is that some components of the solution may contain propagated truncation errors while others do not. For example, if we use the backward Euler formula to calculate a solution to the system

$$\begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} y' = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} y + g$$

Then $y(n+1) = \begin{bmatrix} (y2(n+1) - y2(n))/h - g1(n+1) \\ (y3(n+1) - y3(n))/h - g2(n+1) \\ -g3(n+1) \end{bmatrix}$

So y1(n+1) will contain propagated truncation errors while y2(n+1) and y3(n+1) will not.

Note however, that if the nilpotency of the nonstate subsystem is less than or equal to two then none of the nonstate variables will contain propagated truncation errors. In this case, an error control procedure for the nonstate subsystem could use a global error tolerance for the nonstate variables instead of a more stringent local error tolerance. In the examples I ran with nilpotency equal to two, the error control on the ODEs was stringent enough so that the program produced good results for the nonstate subsystem with no error control on this subsystem at all (see table 1).

But, when the nilpotency is greater than or equal to three, some of the variables in the nonstate subsystem will contain propagated truncation errors, and these errors will be larger than the errors in the state subsystem (as an example, see 4.6). So there must be some error control on the algebraic subsystem when the nilpotency is greater than or equal to three.

Error control is difficult on a system where some of the variables contain propagated truncation errors and some of the variables do not. But this problem should be easier to deal with in the algebraic subsystem than it is within the coupled descriptor system for two reasons:

1. Due to the special structure of the nonstate subsystem, it may be possible to determine which of the variables do not contain propagated truncation errors. If this can be accomplished then the error control procedure for the nonstate subsystem could use a global error tolerance for these variables and a local error tolerance for the variables which do contain propagated truncation errors.

2. In the coupled descriptor system, the errors on variables which do not contain propagated truncation errors are O(h) for the algebraic equations and O(h**2) for the ODEs. But within the nonstate subsystem, all of the variables which do not contain propagated truncation errors have errors on the order of h. Thus, the accuracy of the results should be more consistent within the nonstate subsystem than it is in the coupled descriptor system.

Special problems for systems with nilpotency >= 3 : Linda Petzold (4) discusses two problems which complicate error control for nonstate subsystems. One problem for nonlinear nonstate subsystems with nilpotency >= 3 is that Gear's k step method produces solutions on the first step which do not become arbitrarily small as the stepsize is decreased. These errors can be relatively large, and for nonlinear problems, they may be present throughout the entire interval of integration. Another problem is that "...large errors are introduced into some components of the solution whenever the stepsize is changed." These problems must be solved before a good error control procedure can be designed for nonstate subsystems with nilpotency >= 3.

Determination of the nilpotency : The filtering matrix M is defined to be (INV(E - hbA)E) ** m where m is greater than or equal to the nilpotency of the descriptor system. So we must be able to determine a lower bound on the nilpotency of a descriptor system in order to calculate the matrix M. Since the matrix M is constant for any m greater than the nilpotency of the system, then m is the smallest positive integer such that

(INV(E - hbA)) **m = (INV(E - hbA) E))**(m+1)

So a program could generate m by repeated exponentiation of the matrix (INV(E - hbA)E) until the result is a constant matrix. In my implementation of the procedure to generate M, I read in the nilpotency of the system as an input parameter. But the procedure described above would work just as well.

Interpolation : One of the difficulties which I encountered while experimenting with Sincovec's error filter was interpolation on the algebraic equations. In Hindmarsh's program, the output time (the time when the solutions to the system are printed) is specified by the user of the program as an input parameter. When the integrator passes the specified output time, it interpolates back to the ouptut time in order to print out the solutions at that time. I noticed that the solutions to the algebraic equations in the system 4.3 were in error by as much as 50% when the program used interpolation. Yet, the solutions to the differential equation were accurate. And if the integrator is able to hit the output time exactly, then the errors on the algebraic equations are reduced to .3%.

The explanation for this effect is simple: Interpolation works on the differential equations since the solution to a differential is a trajectory. But the solution to an algebraic equation is a point. So it makes no sense to try and interpolate back to a previous time. Therefore, in order to get accurate results on the algebraic equations, interpolation must be avoided.

CONCLUSIONS

This report has examined some techniques which are useful in solving descriptor systems. The major focus was on the implementation and experimentation with the error filter proposed by Sincovec. Some other topics which were investigated include recovery from inconsistent initial conditions, interpolation problems, nonlinear problems and determination of nilpotency. From this experimentation, I drew the following conclusions:

1. Sincovec's theorem on initial conditions only holds for linear problems.
2. The error filter described by Sincovec is an easy and inexpensive modification to a program which uses Gear's k step method.
3. The error filter will allow a program which uses Gear's method to perform seperate error control on the algebraic and ODEs.
4. For problems with nilpotency <= 2, a global error tolerance can be used to control error on the algebraic subsystem. In some cases the error control on the ODEs will be stringent enough to produce good answers on the algebraic subsystem.

5. Furthur research is needed for solving problems with nilpotency >= 3.

6. Interpolation yields inaccurate results on the algebraic equations. In order to avoid this problem, the output time must be "hit exactly" by the integrator. That is, interpolation must be avoided on algebraic equations.

REFERENCES

1. Richard F Sincovec, A.M. Weisman, E. Yip, M.A. Epton, "Analysis of Descriptor Systems Using Numerical Algorithms", IEEE Transactions on Automatic Control, Vol. AC - 26, No. 1, February, 1981

2. A.C. Hindmarsh, W. Gear, "Ordinary Differential Equation System Solver", UCID-30001 Rev. 3, Lawrence Livermore Laboratory, P.O. Box 808, Livermore, Ca., 94550, Dec., 1974

3. C.W. Gear, "Numerical Initial Value Problems in Ordinary Differential Equations", Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1971

4. Linda Petzold, "Differential/Algebraic Equations are not ODE's", Applied Mathematics Division, Sandia National Laboratories, Livermore, California

ERROR CONTROL FOR DESCRIPTOR SYSTEMS

by

GEORGE ROBERT MANN

B. S. Pittsburg State University, 1975

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1981

ABSTRACT

This report examines some techniques which are useful in solving descriptor systems. A descriptor system is an implicit system of ordinary differential equations coupled to a system of algebraic equations. Sophisticated programs for solving systems of ordinary differential equations have been available for many years. But the presence of algebraic equations in a descriptor system causes problems for programs which are designed to solve systems of ordinary differential equations. One of the major problems is in error control. I will discuss an error filtering mechanism proposed by Sincovec which enables a program to perform separate error control on the algebraic and differential equations. A discussion of the implementation of the error filtering mechanism and an analysis of the results for some sample problems will also be given. Some other topics which are relevant to numerical methods for solving descriptor systems will also be discussed. These topics include recovery from inconsistent initial conditions, interpolation problems, and nonlinear problems.