Solving Stiff DAE Systems as "Near" Index Problems

Yonsoo Chung, Arthur W. Westerberg

EDRC 06-105-91

Solving Stiff DAE Systems as "Near" Index Problems

May, 1991

Yonsoo Chung
Arthur W. Westerberg

**Engineering Design Research Center
and Department of Chemical Engineering
Carnegie Mellon University
Pittsburgh. PA 15213**

ABSTRACT

This paper extends a numerical algorithm for solving nonlinear index problems of Chung and Westeiberg *[Ind. Eng. Chem. Res.*, 29 (1990), pp. 1234-1239] to include stiff DAE systems. Stiff DAE systems are shown to be *near* index problems. Near high index problems cannot be solved by existing stiff ODE/DAE solvers, such as LSODI. A stiff problem should be classified by index. Solutions to these problems can be expressed and controlled in terms of polynomials in the small coefficients which are responsible for the near singularity at the solution point With the numerical algorithm suggested in this paper, stiff DAE systems can be solved accurately and stably even with explicit integration methods.

Introduction

For a long time, there has been exhaustive research on the numerical solution of ordinary differential equations. Nowadays, lots of solution algorithms and computer codes are available even for stiff ODEs. When ODEs are constrained by algebraic equations, there occur severe problems in numerical solution. Differential-algebraic equations possessing numerical difficulties in solving have been classified as index problems. Gear [8], Pantelides [19], Bachmann et al. [1,2], and Chung and Westerberg [4] propose solution algorithms for index problems. There are many parallels between the theory for the numerical solution of DAEs and that for stiff differential equations. We shall show that an index problem can be interpreted as an infinitely stiff system. We shall also introduce a new class of DAEs - *near* index problems, and study its relation to stiff differential equations. This paper presents a numerical algorithm for solving near index problems.

This paper begins with review of previous work on index problems to help one understand the background for this paper.

Previous Work on Index Problems

For the last decade, there has been an exciting literature building on the solution of differential-algebraic equation systems. Brenan et al. [3] review more than two hundred papers on solution of differential-algebraic equations. In solving differential-algebraic equations, two major numerical difficulties have been recognized: consistent initialization and stable propagation. If systems have these problems, they **have** an "index problem.[1]* Chung and Westerberg [4] review papers [1,7,8,12,13f 14,19,20,23,24,27] presenting solution methods for index problems. In addition the following papers are relevant

Yip and Sincovec [30] investigate the properties of DAEs. They relate classical theories of matrix pencils to the solvability of DAEs. They also extend the concepts of reachability, controllability, and observability of state variable systems to DAEs.

Leimkuhler et al. [17,18] present an approximation method for the consistent initialization of DAEs. They characterize the consistency requirement with a system of equations. The consistency equations consist of the problem itself, user-controlled specifications, and derivative equations depending on the index of the system. They approximate derivative equations with one-sided finite differences and analyze the numerical solution of the resulting system for certain classes of DAEs.

Chung and Westerberg [4] propose a numerical algorithm for detecting and solving nonlinear index problems. They detect the rank of the Jacobian matrix of the original formulation to find whether the system has an index problem or not. If it has an index problem, they analyze the Jacobian matrix to find the equations responsible for the singularity of the Jacobian matrix which are then symbolically differentiated, similar to the fist step in Gear and Petzold [12], Gear [8], and Bachmann et al. [1,2]. However, they do not require a tedious symbolic elimination required in these earlier works, because they prove a theorem which shows that only the new variables derived with differentiation have nonunique solutions. Independently, Leimkuhler et al. [17,18] prove the same theorem to develop a scheme for consistent initialization. Chung and Westerberg [4] show that initialization and propagation problems are the same issue and can be treated with a common approach. With their method, numerical as well as structural singularities arising in index problems are accommodated. They show that, in principle, their algorithm can be used with any integration scheme including explicit integrators.

Definitions of "Index"

The notion of *index* has been used to characterize systems which possess difficulties in consistent initialization and stable propagation. Three different definitions of indices have been used: the index of nilpotency, the differential index, and the perturbation index.

The earliest form of index is the nilpotency of the canonical form of linear constant coefficient DAEs.

Yip and Sincovec [30], Gear and Petzold [10], **and Wilkinson [29] show how to determine the nilpotency of the canonical form.**

A linear constant coefficient system is

$$ETC = Ax + g(t) \qquad 0)$$

where E **and** A **are** n **by** n matrices, and they can be singular. **There exist nonsingular matrices** P **and** Q **which reduce the matrix** pencil A-\E to a canonical **form. When** P **and** Q **are applied to equation (1), we obtain**

$$PEQQ\text{-}^{X}J(\; = \; PAQQT^{x}x+ \; Pg(t) \qquad (2)$$

System (2) is composed of five types of uncoupled canonical **subsystems. Three of the types correspond to cases where** no solutions exist or infinitely many solutions exist **The** remaining two types of canonical subsystems correspond to the case of solvable systems. **For solvable systems, equation (2) is equivalent to**

$$y\{(f) = E_1y_1(t) + g_1(t) \qquad (3a)$$

$$E_2y_2'(t) = y_2(t) + g_2(t) \qquad (3b)$$

where

$$Q^{-1}x(t) = \begin{vmatrix} y_1(t) \\ y_2(t) \end{vmatrix}, \qquad Pg(t) = \begin{vmatrix} g_1(t) \\ g_2(t) \end{vmatrix}$$

and matrix E_2 **in equation** (3b) has the property that there exists an integer m such **that** $E_2^m = 0$. $E_2^{m\sim J}$ * **0. The value** of m is defined to be the nilpotency of the system. **The matrix** E_2 is always composed of Jordan blocks of which one subdiagonal has elements of Ts, and others have 0's. m is the size of the largest of these blocks.

Earlier studies [10,11,12, 22, 27, **30]** on DAEs use the nilpotency of the canonical form because the structure and the properties of the solution of a linear constant coefficient system is easily understood by transforming the system to the canonical form. Gear and Petzold [10] define the local and the global

nilpotencies for nonconstant coefficient linear DAEs.

A completely **nilpotcnt** system of order 3 would be

$$\begin{aligned} 0 \ &- y\backslash \ + .\ 81 \\ y\{ \ &= yi + 82 \\ y_2' &= y_3 + g_3 \end{aligned} \tag{4}$$

which when rearranged would give

$$\begin{aligned} y_1 &= -g_1 \\ y_2 &= \sim *\backslash' - \mathbf{ft} \\ y_3 &= -g_1'' - g_2' - g_3 \end{aligned} \tag{5}$$

To solve we need to differentiate first y_x and then y_2- This requirement to differentiate is what causes an index problem. The index reflects the number of differentiations required plus one.

This approach to discovering and removing the index problem requires one to perform a reduction to the canonical form, which requires one to find eigenvalues and eigenvectors. That effort is more work than solving the equations, particularly for nonlinear problems where it must be repeatedly performed.

Consider the following example to understand how the index of nilpotency is determined.

Example 1

(6a)

$$x_l\text{-}x_2\wedge S = 0 \tag{6b}$$

In this example, the corresponding matrices £ and *A* of equation (1) are

$$E = \begin{vmatrix} 1 & 1 \\ 0 & 0 \end{vmatrix}, \qquad A = \begin{vmatrix} 1 & 1 \\ 1 & -1 \end{vmatrix} \tag{7}$$

If we use nonsingular matrices *P* and *Q* as follows:

$$p = \begin{vmatrix} 0.5 & 0 \\ 0 & 0.5 \end{vmatrix}, \qquad Q = \begin{vmatrix} 1 & 1 \\ 1 & -1 \end{vmatrix}, \tag{8}$$

we obtain

$$PEQ = \begin{vmatrix} 1 & 0 \\ 0 & 0 \end{vmatrix}, \qquad PAQ = \begin{vmatrix} 1 & 0 \\ 0 & 1 \end{vmatrix} \tag{9}$$

with

$$y_1 = \frac{x_1 + x_2}{2}, \qquad y_2 = \frac{x_1 - x_2}{2} \tag{10}$$

In equation (9), $E_2 = [0]$, and its nilpotency is one.

The differential index is the extension of the index of nilpotency. Many authors [1,2,8,19,21] have used the differential index in their studies on index problems. It is determined by the number of differentiations of the DAEs that are required to generate a set of ordinary differential equations satisfied by the solutioa Gear [8,9] defines the differential index as follows:

A general DAE takes the form

$$f(*,x,t) = 0 \tag{H}$$

where/and x are in R". The differential index of the system is the minimum integer m, such that the system of equations (11) and

$$\frac{dftf,x,t)}{dt} = 0 \tag{12}$$

$$\dots$$

$$\frac{d^mf(x',x,t)}{df^1} = 0$$

can **be solved for**^ $= x'(x)$.

The differential index detennines some major characteristics of the underiying structure of the DAEs.

Let us consider Example 1 again to determine its differential index. Example 1 cannot be solved for derivatives *x{* and *x£* in terms of state variables x_1 and *x^*. Differentiating equations (6a) and (6b) produces

$$x_1'' + x_2'' = x_1' + x_2' \qquad (13a)$$

*\-'*i = 0 (136)

From the four equations (6a), (6b)$_t$ (13a), and (13b), we can obtain solutions for derivatives *x{* and *x^* in terms of state variables x_x and ^ (Note equation (13a) is not needed). We just do one differentiation to obtain the solutions for the derivatives. By definition, the differential index is one for this example, the same as its index of nilpotency.

The perturbation index is a recent form of index. The perturbation index is defined by considering the effect of perturbations of the equations on the solution. The perturbation index is one more than the number of derivatives of the perturbation that must appear in any estimate of the bound of the change in the solution. Hairer et al. [15] define the perturbation index as follows:

The general DAE (11) has the perturbation index *m* along a solution JC, if *m* is the smallest integer such that the difference between the solution of equation (11) and the solution of the perturbed equation

/(z'.z.f) = 6(r) (14)

can be bounded by an expression of the form

max||z(0-*(0ll ?S AT{||z(0) - JC(O)|| + max||5(0II (15)
+ *max* || S'(0 || + … + *max*| δ mf l \t)||}

over a finite interval *t e* [0, *T]*.

The perturbation index is clearly an important factor in the numerical solution of DAEs since it will

play a major role in determining the impact of round off errors.

Let us consider Example 1 again to determine its perturbation index. The exact solution of Example 1 is

$$x_1 = [x_1(0) + 2.5]e^{x} - 2.5 \qquad (16)$$

$$x_2 = [x_1(0) + 2.5]e^{t} + 2.5$$

where $x_x(0)$ **is the initial value for the variable** x_v **If we use the following perturbed equations,**

$$z\{ + z\{ = z_x + z_2 + 5_t(0 \qquad (17a)$$

$$z_1 - z_2 + 5 = \delta_2(t) \qquad (17b)$$

its solution is expressed as follows;

$$z_x = x_x + 0.552 + R \qquad (18)$$

$$z_2 = *2 - 0.58_2 + R \qquad (19)$$

where

$$R = 0.5e^{t}\backslash 'e\text{-}*8_l(9)dQ \qquad (20)$$

From **equations** (18) **and (19),**

$$||z(t) - x(t)|| \leq 0.5||\delta_2(t)|| + ||R|| \qquad (21)$$

Introducing a new variable A, equation (20) is converted to

$$R = 0.5\,A_1, + 0.5\,A_{Jo}^{t} e{\sim}^{Q} l\Delta_1(\theta)d\theta \qquad (22)$$

where $A\{ = 8\wedge$ This means that R is not a function of 5_t. We can know that

$$\|z(t) - x(t)\| = O(\delta_2) \tag{23}$$

By definition, the perturbation index of Example 1 is one.

Hairer et al. [IS] show that the differential and the perturbation indices are equal if the solution derivative appears multiplied by a constant matrix, in particular, for semi-explicit DAE systems. Example 1 belongs to this case where the differential and the perturbation indices are the same. They also show an example of which the differential index is one whereas its perturbation index is two in general. A detailed relationship between the differential and the perturbation indices can be found in Gear [9]. Gear shows that

differential index £ *perturbation index* £ *differential index* + 1 (24)

and that both indices are the same if the derivative components of the DAE are total differentials. Gear also shows that for (the differential) index zero problems the perturbation index is one more than the differential index. In Gear [9], the notion of the perturbation index is extended to systems of integral algebraic equations.

Studies on index problems [8,10,12,15,19,21,22] have reported that systems exhibit different characteristics in solution according to their index. Gear and Petzold [12] demonstrate that a DAE can be solved in general without any difficulties in initialization and propagation, only if the index does not exceed one. Pantelides [19] also show that the solution of an index one system is no more difficult than the solution of ODEs of which the differential index is zero by the definition. However, some index one problems have difficulty in consistent initialization.

Let us consider equation (16): the exact solution of Example 1. It has only one integration constant *x^O)* for two variables x_1 and x^. It is apparent that arbitrary initial values cannot be assigned to variables x_l and *2 because of the algebraic constraint •- equation (6b). In solving Example 1, we have difficulty in consistent initialization. We note that the differential and the perturbation indices of Example 1 are one.

If index problems are defined as systems which have difficulties in consistent initialization and/or in stable propagation, index problems must include index two or higher and some index one problems. The differential and the perturbation indices are not indicators as to the existence of an index problem. This ambiguity **resides** in the **purpose** of the index. The notion of index has been used as a standard to measure the distance of **DAEs** from ODEs. We note that the differential index is a measure of difficulties in converting **DAEs** to **ODEs.**

Consider the following form of the general DAEs:

$$\boldsymbol{f(x,xf_9y,t) \ = \ 0} \qquad \textbf{(25)}$$

where *x, ii* are n-dimensional vectors of differential variable and its time derivative, respectively, *y* is an m-dimensional vector of algebraic variable, / i s an n+m-dimensional vector function which is assumed to be sufficiently differentiable. According to Chung and Westerberg [4]$_f$ equation (25) has an index problem if it has a singular Jacobian matrix with respect to variables *J£* and *y*. Their algorithm terminates when the Jacobian matrix becomes nonsingular after adding a new set of equations derived using symbolic differentiation. Investigating the singularity of the Jacobian matrix can give us a way to make a clear division between index and nonindex problems. The following is a new definition of index.

Definition of **Index**

> The index of equation (25) is one plus the number of differentiations of DAEs required to make the Jacobian matrix with respect to variables *x'* and *y* nonsingular using the algorithm of Chung and Westerberg [4].

The new index measures difficulties in converting index problems to nonindex ones. It leads a new definition of index problems.

Definition of an **Index Problem**

> If the index of a system is larger than one, in other words, if a Jacobian matrix of the original equations with respect to variables *li* and *y* is singular, the system has an index problem.

Let us consider Example 1 again to show how these definitions work for it. It has two differential variables x_x and x^ and no algebraic variables. We need to examine the following Jacobian matrix wiih **respect to variables *x{* and *x{*\**

$$\begin{vmatrix} 1 & 1 \\ 0 & 0 \end{vmatrix} \qquad \textbf{(26)}$$

This matrix is singular. According to Chung and Westerberg [4], we need to differentiate equation (6b) of Example 1 symbolically to obtain

$$V-x{\wedge}O \qquad (27)$$

With equations (6a) and (27), the Jacobian matrix with respect to variables $x\{$ and $x\{$ becomes

$$\begin{vmatrix} 1 & 1 \\ 1 & -1 \end{vmatrix} \tag{28}$$

and this Jacobian matrix is nonsingular. To get this nonsingular Jacobian matrix, we needed one differentiation of equation (6b). By the definitions of index and index problem, Example 1 has an index problem, and its index is two. We note that Example 1 has initialization problem even though its differential and perturbation indices are one. We can see clearly that Example 1 will have initialization problem, because it is detected to have an index problem.

The above definitions of index and index problem can be easily implemented with the algorithm of Chung and Westerberg [4]. When a system is detected to have an index problem, their algorithm converts the index problem to the nonindex problem - index one system, and automatically determines its index. The new definition of index can be used as a clear divider between index and nonindex problems. From this point on, the index for an example will be this index.

Stiff Systems vs. Near Index Problems

Problems distinguished from others by a term of "stiff^1 are very important because they occur in many physically important systems and have severe numerical difficulties in solving. The magnitudes of eigenvalues of a stiff system vary greatly. Although the component of the solution corresponding to the largest eigenvalue is of no practical interest, the criterion of absolute stability forces one to use an extremely small step size according to the largest eigenvalue over the entire range of integration. As a result, the computation time necessary to integrate a highly stiff system can become excessively large. The major issue in solving stiff systems is how to maintain stability using as large a step size as possible. Seinfeld et al. [25] review and present several numerical integration algorithms for stiff equations. Shampine and Gear [26] discuss numerical algorithms for stiff problems. It is known that explicit integration methods are not suitable to solve stiff systems because they must use very small step size to avoid a stability problem. Implicit integration methods are currently required to solve stiff systems.

The stiffness ratio *SR* [6] is a measure of the stiffness of ordinary differential equations. For ordinary differential equations

$$y' = f(y) \qquad (29)$$

the stiffiiess ratio, *SR*, is defined as follows;

max\Re\t\

SR ' 1S^\\ O0 >

where Xr is an eigenvalue of the Jacobian matrix [|£]. Typically *SR* around 20 is not stiff, *SR* around 10^3 is stiff, and *SR* around 10^6 is very stiff.

In the numerical solution of ordinary differential equations, the matrix [%] characterizes the behavior of the solution, which is why the Jacobian matrix [^] is used for calculating the stiffiiess ratio of ordinary differential equations. For the numerical solution of DAEs (25), the matrix [^, %] plays the most important role in characterizing the solution behavior. This matrix is the same as that used in the above definitions of index and index problems. We can use this matrix to check the stiffness of DAEs.

By the definitions in the previous section, an index problem is a system of which the Jacobian matrix is singular. In a singular Jacobian matrix, at least one eigenvalue has value of zero. If the Jacobian matrix has at least one nonzero eigenvalue, the stiffness ratio of the index problem is infinite, meaning that an index problem can be interpreted as an infinitely stiff system.

Consider the following linear ODEs

$$/ = Ay \qquad (31)$$

where *n* by *n* matrix *A* is assumed to have *n* linearly independent eigenvectors and these vectors arc chosen to be the column of a matrix S, then

$$/ = SAS^{-1}y \qquad (32)$$

where A is a diagonal matrix with the eigenvalues of *A* along its diagonal [28]. Premultiplication by S"'1 produces

$$S^{-1}y' = \Lambda S^{-1}y \qquad \text{03)}$$

If we introduce a new variable z for *S~^{l}y* then z' =S"V and

$$z' = A\,z \qquad (34)$$

Premultiplying equation (34) by A"^{1} gives

$$A\sim^{*}\ /\ =\ z \qquad (35)$$

where A*^{1} is a diagonal matrix with the reciprocals of eigenvalues of A along its diagonal.

If λ_{max} and λ_{m}*in* ^$^{\text{ave}}$ *$^{\text{c}}$ maximum and the minimum absolute values of eigenvalues *of A* respectively, the stiffness ratio *SR* of equation (34) is

$$SR = !\hat{}\,i! \qquad (36)$$

If we assume that | A,,^! is a very large number, for example, 10^{6}, and other eigenvalues have values of order of 1, then *SR* is of order 10^{6}. This indicates that equation (31) is very stiff. The coefficient matrix of equation (35), A"^{1}, has *n*-\ elements of order of 1 and one element of order of KT^{6}. It is very close to being singular. We will call this kind of matrix a near singular matrix. At the extreme, when | X^^l is infinite, A""^{1} is exactly a singular matrix and equation (35) has an index problem. We use again that an index problem can be interpreted as an infinitely stiff system. We note that equation (35) has the form of the DAE - equation (25), and the coefficients matrix A"^{1} is its Jacobian matrix. When the Jacobian matrix of DAEs is nearly singular, we will call that class of DAEs a near index problem. We define the *near* index as the index of the system when terms responsible for the near singularity vanish to make the Jacobian matrix singular. From the above discussion, we can say that stiff systems are near index problems. These ideas can be extended to nonlinear ODEs and DAEs without any loss of generality.

The Near Index Problem

Consider **the following** example problem.

Example 2

$f_1 = x_1' - x_2 = 0$ **(37)**

$f_2 = x_2' - y = 0$

h =*i-e;y- $g(t) = 0$

where

0 £ e « 1

To check whether this example has an index problem or not, we need to examine the Jacobian matrix with respect to variables *x{, x{* and *y:*

$$\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & -e \end{vmatrix} \tag{38}$$

Eigenvalues of this matrix are 1, 1 and -e. The matrix (38) is *nearly* singular because £ is assumed to be a very small number. If we assume that e is exactly zero, this matrix is singular. It can be easily shown that index of equation (37) with e = 0 is three. The differential and the perturbation indices are also three. So, the **near** index of equations (37) is three.

Index three systems have difficulties in consistent initialization and stable propagation. Do near index three systems have the same problems? The answer is yes. From equation (37), we can obtain the following relations.

$x_1 = g(t) + ey$ (39*a*)

(3%)

$x_2 = g'(t) + \varepsilon y'$

$$y = g''(t) + \varepsilon y'' \qquad (39c)$$

Substituting equation (39c) into equation (39a) to eliminate variable y generates

$$x_1 = g(t) + \varepsilon g''(t) + e^2{:}y'' \qquad (40)$$

Repeated symbolic differentiation of equation (39c) and substitution into equation (40) provides the following solution for variable x_x\

$$*i = g(t) + eg''(t) + e\,V^{4)} + \;\ldots \qquad (41)$$

Equation (41) includes no integration constant; an arbitrary set of initial values cannot be assigned to this example. Near index three systems also have difficulties in consistent initialization.

Let us consider the linear DAE system (1) again to look at its error behavior.

When we use the Backward Euler method to integrate equation (1), we use the relation:

$$E\frac{y_{n+1} - y_n}{{}^{n}n+l} = A y_{n+1} + g(t_{n+1}) \qquad (42)$$

The exact solution at the n+l-th step can be calculated by using a Taylor series expansion and the mean value theorem as follows:

$$y(t_{n+1}) = y(t_n) + \;\ldots\; \ldots\; \frac{h^2}{\angle} \qquad (43)$$

where

$$t_n \le \xi \le t_{n+1}$$

Rearranging equation (43) for the derivative $/(f_{n+1})$ and substituting it into equation (1) generates

$$E\frac{y(t_{n+1}) - y(t_n) - \frac{h^2}{2}y(\xi)}{h_{n+1}} = A\,y(t_{n+1}) + g(t_{n+1}) \qquad (44)$$

The right hand side terms are calculated **at time** *t^*. **With the local error as follows,**

$$e_{n+1} = y_{n+1} - y(t_{n+1}) \qquad (45)$$

subtracting equation (44) from equation (42) generates

$$E\frac{e_{n+1} - e_n + \frac{h^2}{2}y''(\xi)}{h_{n+1}} = A\,e_{n+1} \qquad (46)$$

Yip and Sincovec[30] show that if equation (1) is a solvable system, **the matrix** *[E-h*$_{rH\cdot l}$*A]* is nonsingular. Equation (46) can be rearranged as follows;

$$e_{n+1} = (E-hA)^{-1}E\,e_n - (E-hA)^{-1}E\frac{h^2}{2}y''(\xi) \qquad (47)$$

Equation (47) is **the** error equation for the linear DAE system, equation (1). This equation shows the **error at the n+l-th** step as a function of that at *n-th* step. If we start with the **exact** solution **at** the n-ih **step, the error at the next** step is

$$e_{n+1} = -(E-hA)^{-1}E\frac{h^2}{2}y''(\xi) \qquad (48)$$

Equation (48) shows the local error after one step propagation. Let us calculate the local error ot Example 2. In equation (37), the corresponding matrices *E* and *A* of equation (1) are

$$E = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{vmatrix}, \qquad A = \begin{vmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & -e \end{vmatrix} \qquad (49)$$

If we insert these matrices into equation (48) and calculate the local error for each variable, we obtain

$$e_{n+1}(x_1) = \frac{h^2}{2(h^2-\varepsilon)}(\varepsilon x_1'' + h\varepsilon x_2'') \qquad (50)$$

$$e_{n+1}(x_2) = \frac{h^2}{2(h^2-\varepsilon)}(hx_1'' + \varepsilon x_2'')$$

$$e_{n+1}(y) = \frac{h^2}{2(h^2-\varepsilon)}(x_1'' + hx_2'')$$

The largest error comes from the first error term for variable *y*. The error of the example can be represented as follows;

$$e_{n+1} = \tilde{O}(\frac{h^2}{h^2-\varepsilon}x) \qquad (51)$$

The local error cannot be converged to zero for very small values of e. Example 2 cannot be solved by existing stiff ODE/DAE solvers, such as LSODI [16], for small e.

We note that the above example is a near index three system. We also note that a stiff system is the same as a near index problem. From these two facts, we make the observation: ***a*** *stiff problem should be* ***classified as a near index problem, in other words, stiffness should be classified by index.***

A Numerical Algorithm for Near Index Problems

We shall discuss the detection of near index problems and then present an algorithm for solving them.

When we have a set of DAEs to solve,

$$/(x.x'.y.r) = 0 \qquad (52)$$

we must check first whether they describe a well posed problem or not. To be well posed there must be no way for the equations to provide inconsistent definitions for the variables in the problem. Two cases

are possible: (1) the equations contain inconsistent definitions for a variable and (2) the equations contain inconsistent definitions for a variable x_x and its time derivatives *x{*.

The first instance is detected if the equations are always singular in all the variables x, *xf* and *y* in the problem. Then one of the equations must be dependent or inconsistent with those remaining. Case 1 arises if it is inconsistent

The second case for the equations to be pooriy specified occurs when both variables x_r and its time derivative *x(* are forced to be computed by the equations. The following example illustrates.

$$*1 = {}^{z}i - {}^{2}2 \; {}^{+} \; {}^{2} \qquad (53)$$

$$x\{ \;» \; ij - z_2 - 3$$

$$x_1 = 4t$$

There are four variables in this problem: x_v x_x\ z_x and z_2. As there are three equations, this problem has one degree of freedom. z_x and z_2 appear only as their difference here; one of them has to be specified elsewhere and must be the variable which consumes the degree of freedom. Therefore, x_x and *x{* must both be computed. It is unlikely that the equations will give consistent definitions for both* Here for example the results are that

$$x\{ = 41 - 5 \qquad (54)$$

$$Xj = At$$

which are inconsistent definitions forxj and its time derivative.

Assuming a well posed problem, we have to know that it has an index or a near index problem. Chung and Westerberg [4] show us the way to detect and solve an index problem. They detect an index problem by discovering that the Jacobian matrix of equation (52) with respect to variables x' and *y* is singular at a solution point. We note that the Jacobian matrix is square, and its dimension is n+m. To detect a near index problem, we do the same. A near index problem is a system for which the Jacobian matrix is nearly singular. The key operation in detecting an index or a near index problem is to check the singularity of the Jacobian matrix with respect to variables *xf* and *y*.

Assuming a matrix which is well scaled to start with, numerical singularity can be detected by Gaussian

elimination. The final form of a matrix with Gaussian elimination is the echelon form. If there are some zero elements on the main diagonal of the echelon form, the matrix is singular. If all pivots of the echelon form are nonzero, and some pivots have very small values compared with others, the matrix is nearly singular. It should be noted that detecting numerical singularity for really large sets of equations can be difficult Partitioning the equations into smaller sets where possible and using high precision arithmetic can aid in proper detection. Duff et al. [5] discuss Gaussian elimination and its implementation for sparse matrices.

Consider the Newton formula for equation (52).

$$\frac{Z_j^*f}{dx}\Delta x + \frac{\partial J}{dxf}\Delta x' + \frac{\Delta f}{dy}\Delta v + \frac{df}{dt} zi\text{-}At = -/Ux\backslash y.O \qquad (55)$$

In the above equation, Ax' and *Ay* terms are unknowns, and others **are known from previous steps. Therefore,** ***Ax*** **and** ***At*** **are** zero, and equation (55) reduces to:

$$\frac{\partial f}{\partial x'}\Delta x' + \frac{\wedge Ay}{By} = -/U\wedge.y.O \qquad (56)$$

We note that the coefficients in the left hand side **terms of equation (56) are exactly the same as elements of the Jacobian matrix we use to detect the existence of an index or a near index problem. If we introduce a new variable** ***z*** **for variables / and** *y*, equation (56) becomes

$$\frac{\wedge Az}{dz} \gg RHS \qquad \mathbf{(57)}$$

where the dimensions of equation (57) and variable Az are both **n+m. Let** ***U*** **be the echelon** form of the **Jacobian and** ***b*** **be the vector of** *RHS* after Gaussian elimination. Then,

$$UAz = b \qquad \mathbf{(58)}$$

With **a** nonsingular *U,* there is no problem in calculating Az, which is the case **of** nonindex problems, i.e., index one problems. When *U* has some zero pivots, we cannot calculate some **of the** Az. Index

problems belong to this case because some zero pivots **of the echelon form mean that the Jacobian is singular. When some pivots of** *U* have very small values compared with other pivots, the situation is not simple. **Premultiplying by** $U\sim^{l}$ or back substitution is unstable, and sometimes practically impossible. **Near index problems belong to** this case.

Once we have detected the existence of an index or a near index problem, we need to convert the problem to one we can solve. Chung and Westerberg [4] convert an index problem to a nonindex one, i.e., convert an infinitely stiff problem to a nonstiff one. Their algorithm can be **a** basis for solving a near index problem.

To use their algorithm for solving near index problems, we need first to find the equations responsible for the near singularity of the Jacobian. We can discover the linear combination of the equations responsible for the singularity of the Jacobian. If we allow some tolerance in finding the linear combination, we can treat a near singular Jacobian as if it were a singular one. For example, consider matrix (38) in Example 2. If we assume that e is of order lCT^{4} and specify the tolerance of $lO"^{3}$, we can treat the element e as if it were zero. The matrix (38) can be treated as **a** singular matrix. With this tolerance in a linear combination, we can find the equations responsible for the near singularity. Equation $/_3$ alone in Example 2 is responsible for the near singularity of the Jacobian matrix (38).

The set of equations responsible for the singularity is differentiated. Following differentiation of the equations responsible for the singularity, we have a new set of equations to augment the original set of equations. If we can calculate variables *ii* and *y* from the expanded set of equations, the algorithm for solving index problems terminates. The expanded set of equations obtained after algorithm termination will be called an augmented equation set. If the expanded set of equations is still singular for variables / and y, detecting and differentiating equations responsible for the singularity is repeated until the nonsingularity is guaranteed for variables *li* and y.

The augmented **set** of equations includes variables which have very small elements in the Jacobian or in the echelon form which resulted from performing a Gaussian elimination on the Jacobian matrix. In the above procedure, these variables were not chosen as pivots because they were treated as if they were zero elements. The actual Jacobian matrix of the augmented set of equations is not square, but rectangular. It includes nonpivot variables whose elements are very small in the Jacobian or in its echelon form. Nonpivot variables in the augmented set will be called augmented nonpivot variables. We note that the Jacobian of the augmented set has always full rank for pivot variables after termination of the above procedure.

Figure 1 shows the augmented set of equations and the rectangular Jacobian matrix for a near index two problem. The 0 marks are used for zero elements resulting from performing a forward Gaussian elimination. Variable *z* is used for variables *ii* and *y*. *f* is generated by differentiating equations responsible for the singularity of the original equations/. We note that nonzero elements *a* and *b* do not have small values, and they are used as pivots.

Because the Jacobian of the augmented set has full rank for the pivot variables, we can solve the augmented set of equations for variable z with given values for augmented nonpivot variables. In Figure l_v the solution for variable z_2 is

$$*2 \; = \; b\sim Hg_2'-u^Tx) \; - \; eJT^1z_2' \qquad (59)$$

The solution consists of two parts: one for the solution of a corresponding pure index problem and the other for the product of augmented nonpivot variables and their small coefficients. Assuming e * 0, we obtain the solution of the corresponding index problem. With nonzero but small coefficients, the term which is the product of the augmented nonpivot variables and their coefficients should be added to the solution of the corresponding index problem. If e is small enough, say KT^6, we might ignore this correction to z_2; otherwise we will need to proceed as we shall describe in the next section. Once variable z_2 is estimated, z_x is simply computed by back substitution.

Figure 2 shows how to obtain the augmented set of equations for a near index three problem. If element *b* has a very small value, variable z cannot be calculated with the original and first set of augmenting equations. A new set of equations is needed to cure the near singularity for variable z. A second set of augmenting equations,/', is generated by differentiating the equations responsible for the singularity of the original equations/and the first set of augmenting equations/". For a near index three problem, we need two sets of augmenting equations to obtain the augmented set Nonzero elements a, *c* and *d* will not have small values, and they are used as pivots.

If we know the exact values for augmented nonpivot variables, the solution obtained in the above can be the exact solution for the problem. The augmented nonpivot variables are always derivatives of the algebraic variables or higher order derivatives of the state variables. This situation does not allow us to guess the right values for the augmented nonpivot variables. We need a means to update the augmented

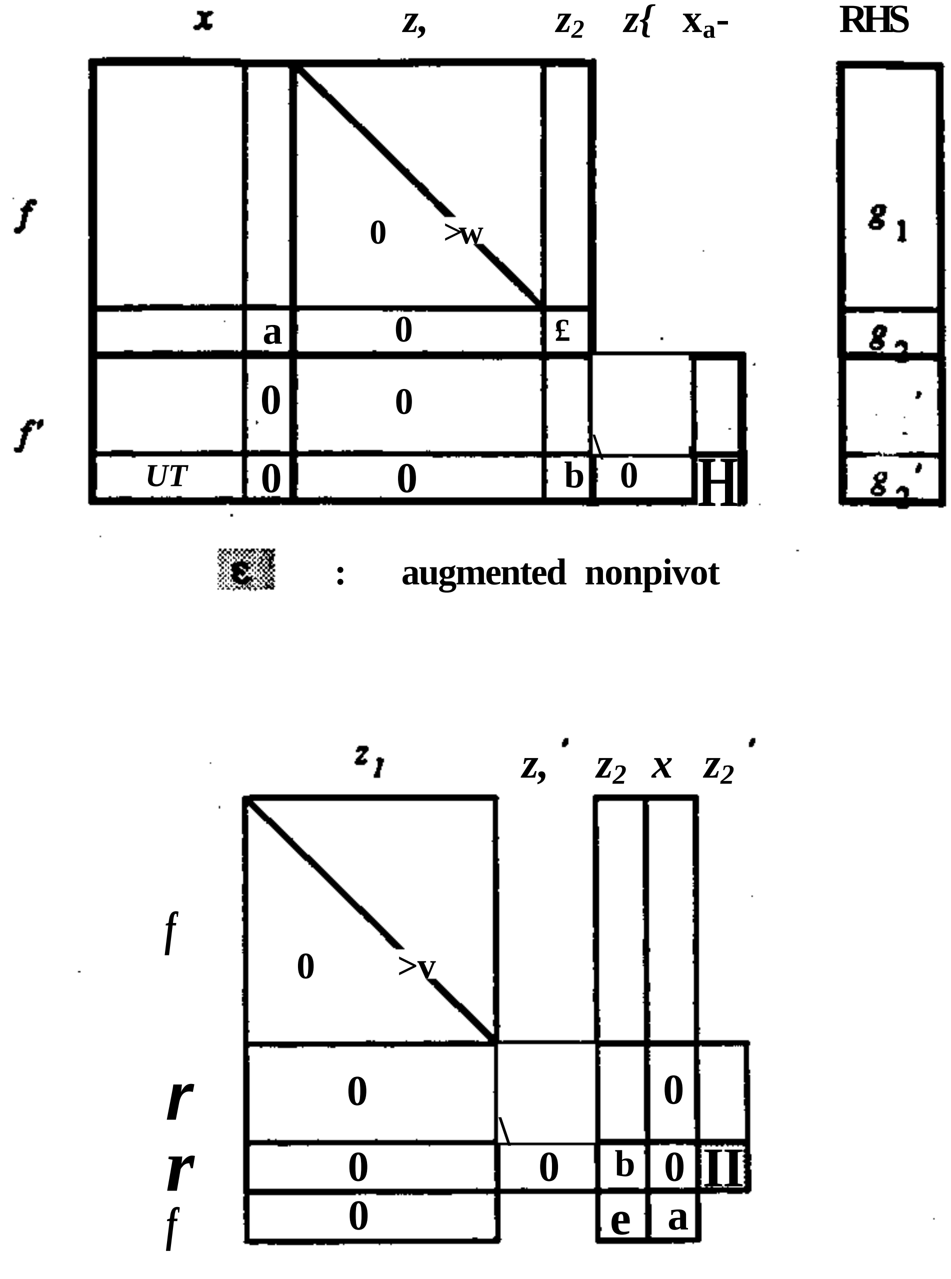


Figure 1: The Augmented Set of a Near Index Two Problem

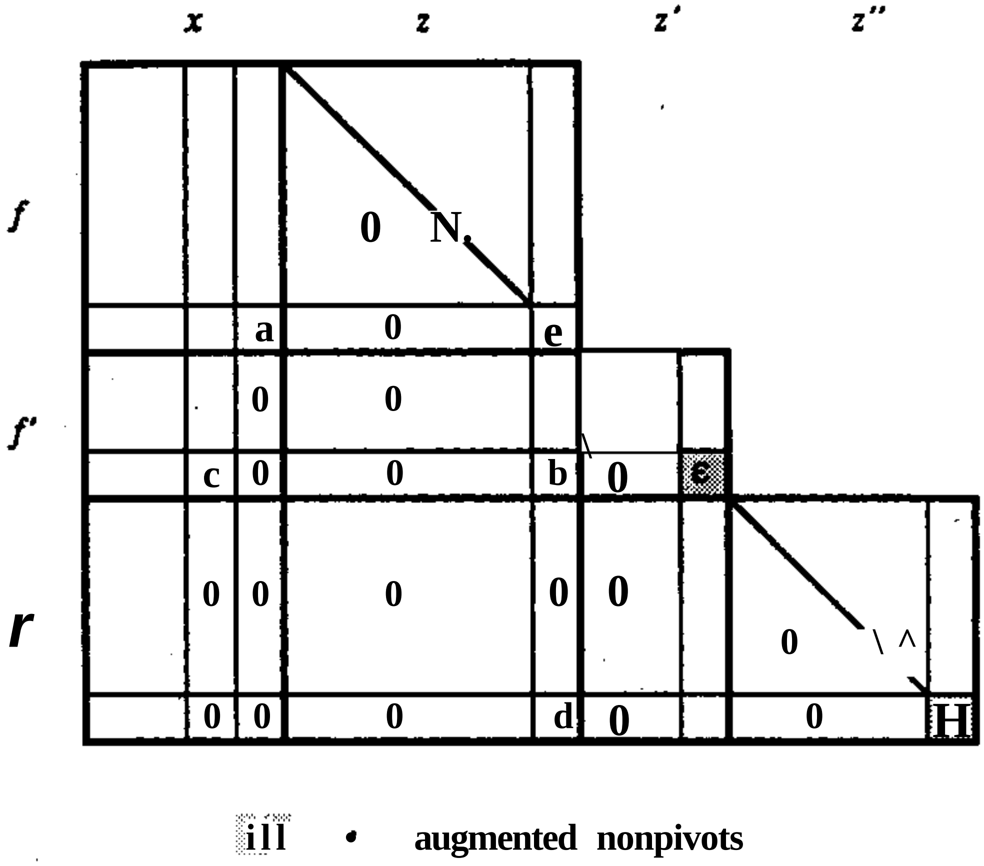


Figure 2: The Augmented Set of a Near Index Three Problem

nonpivot variables to enhance the solution accuracy.

A naive approach to update augmented nonpivot variables is to approximate and iterate them using information from their ancestors. For example, if a derivative of the algebraic variable, /, is an augmented nonpivot variable and other variables including its ancestor *y* are expressed as functions of it, the augmented nonpivot variable / can be estimated from values of *y* at previous steps. The estimated value for / is used as a new guessed value for the augmented nonpivot / to calculate the other variables. / is approximated again using the new value of *y*. This procedure continues until a specified error tolerance is satisfied. If the second derivative x" of the state variable *x* is an augmented nonpivot, it can be approximated by using previous values of the state variable *x* and its first derivative x'. This method includes approximation errors in estimating the nonpivots from their ancestors. Error accumulation will occur.

As mentioned in Chung and Westerberg [4], differentiating a set of equations introduces derivatives one order higher with respect to time for all of the variables, and the Jacobian of the original equations becomes precisely the coefficient matrix for these higher derivatives. From the augmented set of equations, variables x' and *y* can be solved for in terms of the product of augmented nonpivot variables x" and / and their coefficients. Differentiating the augmented set of equations generates a new set of equations from which variables x" and / can be solved in terms of the product of new nonpivot variables x"' and *y"* and their coefficients. If the new set of equations is added to the existing augmented set of equations, original variables x^7 and *y* can be solved in terms of the product of new nonpivot variables x"' and y" and *square* of their coefficients.

Figure 3 shows addition of the second set of augmenting equations to the augmented set of a near index two problem. The second set of augmenting equations is generated by differentiating the augmented set. Note, the first set of augmenting equations already exists in the augmented set We can solve the second set of augmenting equations for the variable z_2'.

$$z_2' = c^{-1} \wedge)'' - v^T x \qquad ec\text{-}^{x} z_2'' \tag{60}$$

which gives

$$z_2 = trHg_2'\text{-}u^T x) - eb\text{-}^{x}c\text{-}Hg_2''\text{-}fx) + e^2 b\text{-}^{l}c\text{-}^{l}z_2'' \tag{61}$$

Solution (61) is more accurate than solution (59) by one order of e. Adding more sets of augmenting

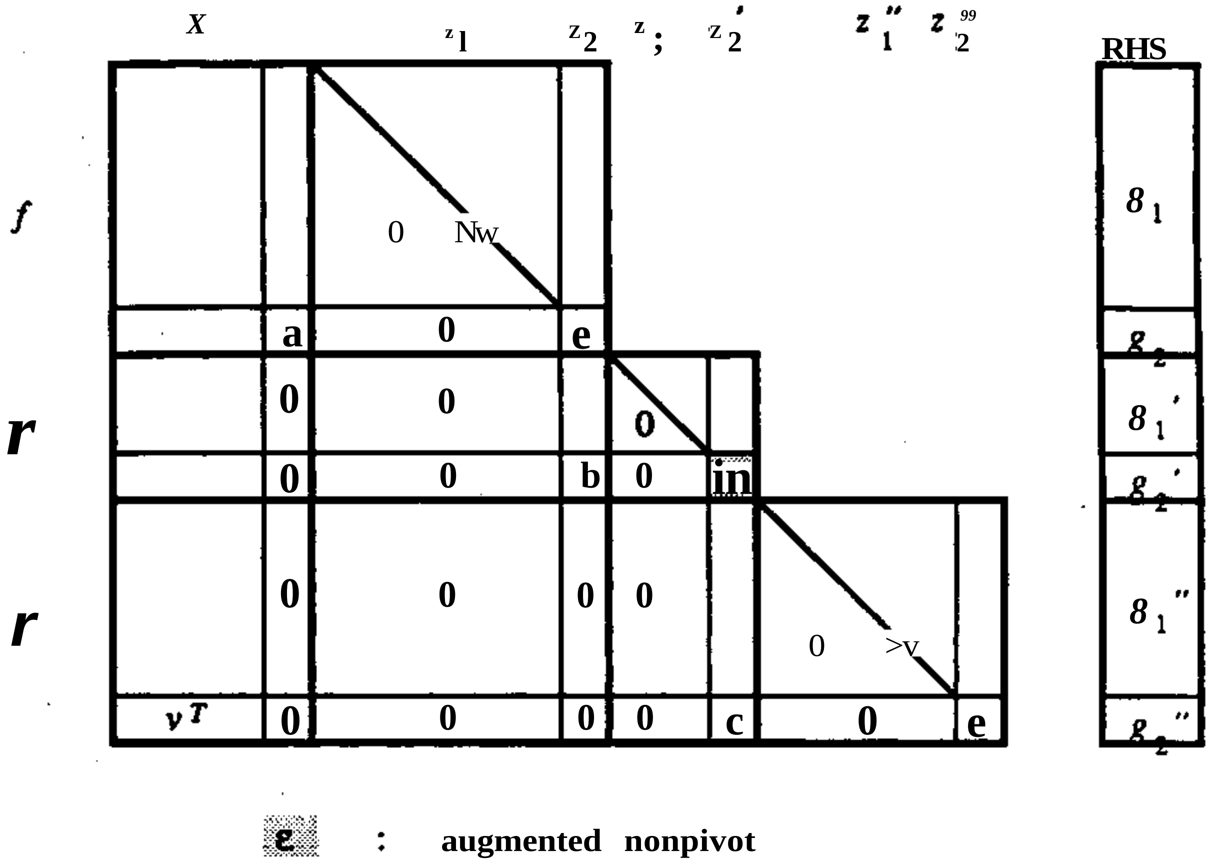


Figure 3: Adding a Set of Augmenting Equations for a Near Index Two Problem

equations generates more accurate solution.

If we want to update augmented nonpivot variables for a near index three problem by one higher order, we need another two sets of augmenting equations. In this way, we can update the augmented nonpivot variables up to any order in the small parameters. The solution of a near index problem can be expressed and controlled in terms of polynomials in these small parameters.

We note here that the rank deficiency of the Jacobian matrix indicates the existence of an index problem, but it does not decide the index of the system. When a subset of original equations is differentiated and added as a set of augmenting equations to the original equations and the expanded set of equations cures the singularity of the original Jacobian matrix with respect to variables x' and *y*, the expanded set of equations is the augmented set. If the expanded set of equations cannot cure the singularity of the original Jacobian, we need another set of augmenting equations to cure the singularity of the original Jacobian. This procedure continues until the singularity for variables *li* and *y* is cured. The index is affected not by the rank deficiency, but by the capability of the sets of augmenting equations to cure the singularity of the original Jacobian.

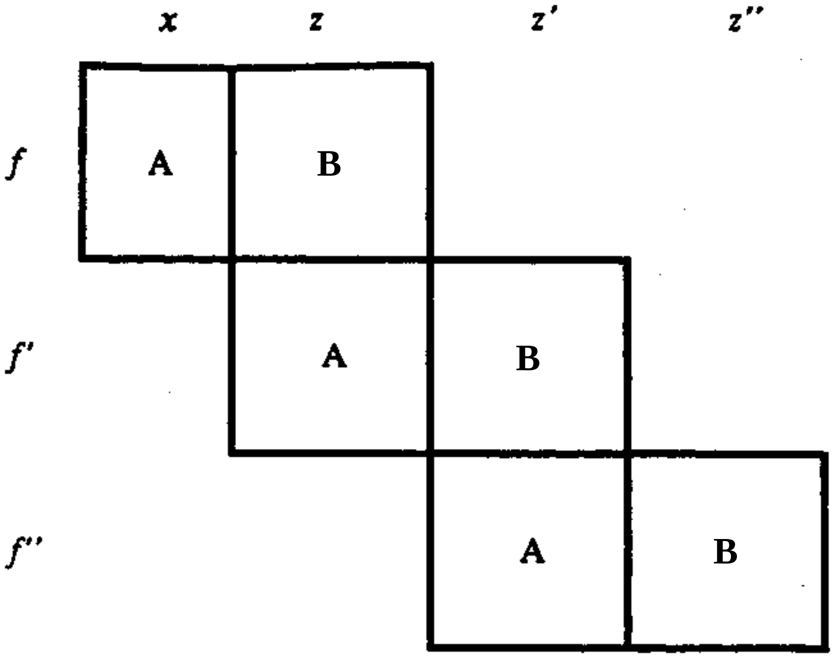
The situation of a near index problem is the same. If a set of augmenting equations cannot cure the near singularity of the original Jacobian, we need another set of augmenting equations. At this stage, there exist two sets of nonpivot variables; one is a set of nonpivot variables in the original set and the other is a new set of nonpivot variables introduced by the set of augmenting equations. Even when the set of augmenting equations cures the near singularity of the original Jacobian, it introduces a new set of nonpivot variables. After curing the near singularity of the original Jacobian for a near index *m* problem, we have m-1 sets of nonpivot variables remaining because we need to add m-1 sets of augmenting equations to estimate the set of nonpivot variables of the original equations. So, an augmented set of equations for a near index m problem includes m-1 sets of augmented nonpivot variables. We note that the number of nonpivots in a set of nonpivot variables is determined by the rank deficiency of the Jacobian. A set of augmenting equations is used to update one set of augmented nonpivot variables by one order of small coefficients. We need m-l sets of augmenting equations to update all sets of augmented nonpivot variables for a near index *m* problem. All sets of augmented nonpivot variables have the same coefficients because differentiation generates the same Jacobian for newly introduced highest order derivatives of the variables.

The equations responsible for the singularity of the Jacobian matrix are detected by finding the linear combination of equations causing the Jacobian to be singular, and the equations in this linear combination are differentiated to cure the singularity. Consider a linear DAE system which has an index problem. The Jacobian matrix with respect to variable *x* is assumed to be *A* and the Jacobian with respect to variables *it* and *y* is assumed to be *B*. With differentiating the original equations, the Jacobian *A* becomes the coefficient matrix for *x** and the Jacobian *B* becomes the coefficient matrix for *x"* and /. With one more differentiation of the original equations, the Jacobian *A* becomes the coefficient matrix for *x"* and the Jacobian *B* becomes the coefficient matrix for x"' and /'. Figure 4 shows propagation of the Jacobian matrix with differentiation in the case of a linear DAE system. Differentiating the equations involved in the linear combination of equations causing the original singularity maintains the singularity even though the size of the Jacobian is reduced. Theorem of Chung and Westerberg [4] is valid for the reduced singular matrix. We can obtain unique solutions for variables *xf* and *y*. Differentiating the equations in this linear combination is sufficient to solve an index or a near index problem.

For a nonlinear DAE system which has an index problem, the situation is more complex. Figure S shows the propagation of the Jacobian matrix with differentiation in case of a nonlinear DAE system. In Figure 5, matrices C, *E* and *F* contain only newly filled nonzero elements resulting from the differentiation. Matrices *D* and *G* will generally have nonzeroes where matrix *A* has (but not necessarily with the same values) plus newly filled elements. Using the linear combination to find the equations to differentiate is generally a sufficient condition for the nonlinear case.

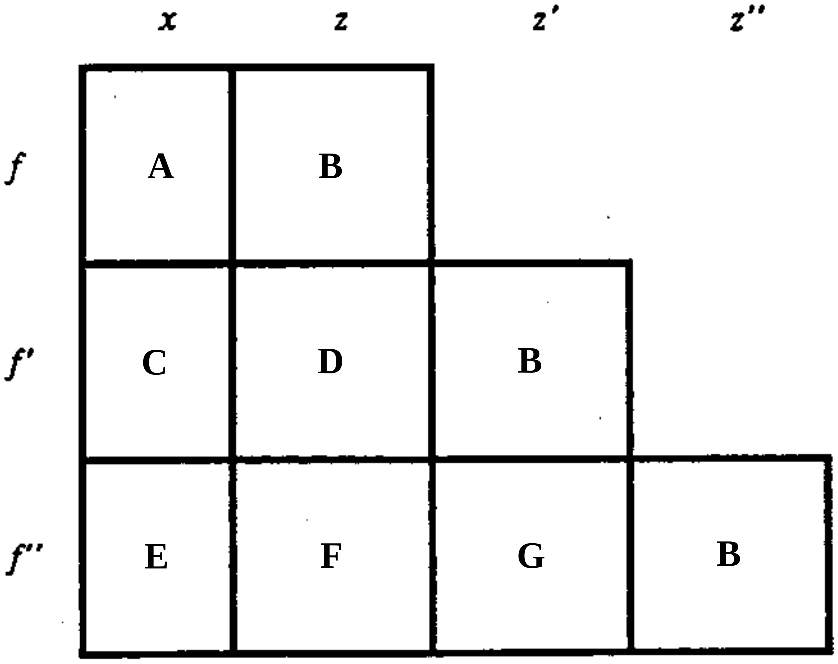
The following is the summary of an algorithm for identifying variables to be integrated numerically and obtaining their time derivatives at any time.

1. Given known or guessed values JC, X' and *y*, make the Jacobian matrix of the original equations.
2. Detect the rank of the Jacobian matrix with respect to all the variables. Detect this rank where no variable x_i and its time derivative x_i' are simultaneously pivoted. If rank deficient, the problem is not well posed. Quit with a warning. (One has only guessed the solution point for the equations and has not yet converged them. Actual rank deficiency can only be detected at a solution point so this test is ambiguous at first.)
3. Detect the rank of the original equations with respect only to variables x' and *y*.
4. If rank deficient with respect to *xf* and y, the equations have an apparent index problem. Continue with the next step. If not, the equations can have a near index problem. Follow step 15.
5. Identify the equations responsible for the rank deficiency of the Jacobian.
6. Differentiate the detected equations.
7. Pivot through the Jacobian matrix of the.differentiated equations only in the columns under



$$\begin{array}{c|cccc} & x & z & z' & z'' \\ \hline f & A & B & & \\ f' & & A & B & \\ f'' & & & A & B \end{array}$$

Figure 4: Propagation of the Jacobian Matrix of a Linear DAE System



$$\begin{array}{c|cccc} & x & z & z' & z'' \\ \hline f & A & B & & \\ f' & C & D & B & \\ f'' & E & F & G & B \end{array}$$

Figure 5: Propagation of the Jacobian Matrix of **a Nonlinear DAE System**

newly generated variables until it can no longer be pivoted using only these variables.

£. Pivot through the Jacobian matrix of all the equations with respea to variables x' and *y* plus the pivoted variables found in the previous step.

9. If all these variables cannot be pivoted, repeat steps 5 to 8. Else, continue with the next step.

10. Assign arbitrary but convenient fixed values to those variables in the set of the newly generated variables not pivoted but appearing in the new equations.

11. For equations not yet pivoted in step 8, allow pivoting among the state variables, *x*.

12. Solve the expanded set of equations.

13. Repeat steps 1 to 12 until the equations are converged at a solution.

14. Return with identity of state variables not solved in step 1 to 13 and with their time derivatives.

15. Check whether the Jacobian with respect to variables *it* and *y* is nearly singular. If nearly singular, continue with the next step. If not, solve the equations for variables / and *y*. Return with all state variables identified to be integrated and with their time derivatives, *xf*.

16. Treat the problem as if it were an index problem. Find the augmented set of equations with the algorithm used in steps 5 to 9.

17. Determine the near index of the system and locate augmented nonpivot variables.

18. To update augmented nonpivot variables, make a set of augmenting equations by differentiating the highest order equations in the augmented set

19. Differentiate the set of augmenting equations found in the previous step as many times as one less than the near index of the system. Repeat this step if higher accuracy required.

20. Solve all equations found in steps 1 to 19.

21. Return with identity of state variables not solved in step 1 to 20 and with their time derivatives.

Example Problems

Example 2

Let us consider Example 2 again to show how the above procedure works for it

To check whether Example 2 has an index problem or not, we need to form the Jacobian matrix with respect to only variables x_x\ x^ and *y:*

$$\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & -e \end{vmatrix} \qquad (62)$$

In the above section, we used eigenvalues to check the singularity of the matrix. We note here that calculation of eigenvalues is typically very expensive. The Jacobian matrix (62) has an echelon form. The first and the second pivots have 1 's, while the third pivot, - e, has a small value. The Jacobian matrix of Example 2 with respect to x_x\ *x{* and *y* is nearly singular. Therefore, Example 2 has a near index problem.

We have to differentiate equation/$_3$. We obtain

$$f_3' = \text{x!'-£/-£'(0} = 0 \tag{63}$$

Equation (63) has only one new variable /. Because / has a coefficient of e which is treated as zero, there appears to be no pivot variable for this new equation. We have four equations/i»/2t/$_3$ and/$_3$'. The Jacobian of the four equations/$_{\text{If}}$/$_2$,/$_3$ and/$_3$' with respect to *x^ x{*$_\%$ *x^*$_\%$ *y* and / is

$$\begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & -e & 0 \\ 1 & 0 & 0 & e \end{vmatrix} \tag{64}$$

Assuming e - 0, this matrix is only rank 2. We need to differentiate the set of equations which is responsible for this singularity. Three equations,/$_{\text{If}}$/$_3$ and /$_3$', are involved in the linear combination leading to the rank deficiency. One of them,/$_3$, has been already differentiated, so differentiating it does not add a new equation. We differentiate the other two equations/^ and/$_3$' to get

$$f_1' = x_1'' - x_2' = 0 \tag{65}$$
$$f_3'' = x_1'' - \varepsilon y'' - g''(t) = 0$$

These two equations introduce two new variables: *x{'* and y". Again, *y"* has a coefficient of e which is treated as zero. We can pivot only one of these new variables, *x{* Now, we have six equations: *f^f^fy* f_x\\f_3*'* and/$_3$". The Jacobian matrix of these six equations for the eight variables x_v *x^* x_x\ *x{, y, x{* / and y" is

$$\left|\begin{array}{ccccccccc} 0 & - & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & - & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & - & e & O & O & O \\ 0 & 0 & 1 & 0 & 0 & 0 & - & e & O \\ 0 & 0 & 0 & - & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & - & e \end{array}\right| \qquad (66)$$

Note the last two columns are for nonpivot variables / and y". This Jacobian matrix has a rank of six. The bold figures are output variables. We now have an augmented set of equations which consists of six equatk>ns:/$_{1t}$/$_2$,/3,/i',/$_3$' and/$_3$". The augmented set includes two sets of augmenting equations: the first set consists of/$_3$' and the second set consists *off{* and/$_3$". To obtain the augmented set of equations, we differentiated/^ twice. The index of Example 2 is three, as we said before.

The Jacobian matrix (66) has a rank of six for the six pivot variables x_v x^ $x\{_{\%}$ x^, y and x_x". With given values for two augmented nonpivot variables / and y", we can obtain solutions for the pivot variables in terms of augmented nonpivot variables / and y". Solutions for variables x_v x^ and y are

$$x_1 = git) + \varepsilon g''(t) + \varepsilon^2 y'' \qquad (67)$$

$$x_2 = gV) + \varepsilon y'$$

$$y = g''(t) + \varepsilon y''$$

The first term of each solution is the solution of the corresponding index problem in which e is zero. The remaining terms include a known term *eg"(t)* and products of augmented nonpivot variables and their small coefficients. If we assume that functions *g(t)*$_9$ *g'(t), g"(t)* and variables y' and y" have the same order of magnitudes, we can say that the solution accuracy is order of e here.

Examining the equations whose linear combination gave rise to the small terms indicates all were involved Only three equations $f_vf\{$ and/$_3$" have never been differentiated Differentiating these three equations generates the third set of augmenting equations:/^',//' and/$_3$'". We can solve the third set of augmenting equations for the original nonpivot variable *y'* in terms of the new nonpivot variable y"'. The solution for / is

/ = *'"(;) +ey'" (68)

We note that solutions (67) from the augmented set of equations included augmented nonpivot variables / and y". We have the solution for variable /. If we substitute this into equation (67), we obtain

$$x_1 = g(0 \quad + \varepsilon\, g''(t) + e^{2} /' \tag{69}$$

$$x_2 = g'(t) \; + \varepsilon\, g'''(t) + \varepsilon^2 y'''$$

$$y = g''(t) + \varepsilon\, y''$$

The solution for the variable ^ now has a term involving the product of a new nonpivot /" and the *square* of the coefficient which was there for the original nonpivot variable, /. Again if we assume that functions *gf(t)* and *g"\t)* and the variable /" are of the same order of magnitude, we can say that the solution accuracy for the variable *X2* is now of the order of e^2, while it was order of the order of e in the solution of the previous augmented set. The solution accuracy for the variable *x^* has been enhanced by one order of e.

There still remains another augmented nonpivot variable y" which has been never updated. The value for this variable can be updated by another set of augmenting equations. The fourth set of augmenting equations can be generated by differentiating the third set of augmenting equations. This fourth set of augmenting equations is $/_2''$, f_1''' and $/_3^{(4)}$. We can solve the fourth set of augmenting equations for variable y" in terms of the new nonpivot variable *y^* The solution for variable y" is

$$y'' = g^{(4)}(t) + e/^{4}> \tag{70}$$

Substituting this solution into equation (69) generates

$$*i = g(t) \quad + e\, g''(t) + e^2 g^{<4>}(r) + e^3 y^{<4>} \tag{71}$$

$$*2 = \text{«'('>} \; + e\, *'''(') + e^{2} / ''$$

$$y = g''(0 + \varepsilon\, g W(t) + \varepsilon^2 y^{(4)}$$

If we assume again that functions g(r), *g'(f), g"(t),* $g'''(t)_9$ $g^{(4)}$(r) and variables /" and $y^{(4)}$ are of the same order of magnitude, we can say that the solution accuracy is of the order of e^2. Solution (71) is one order more accurate than solution (67) for all the variables x_v ^ and *y.*

To enhance the solution accuracy from the order of e to the order of e^2, we used two sets of augmenting

equations. Two more sets of augmenting **equations** will **update the** solution accuracy **up to** order **e^{3}. Repetition of the above** procedure will generate the following solution.

x = g(t) + e«"(0 +e^{2}*<$^{4)}$(0 + e V$^{6)}$ « + • • • (72)
y = g\t) + egT(t) + e^{2}s<$^{5)}$(0 + £g^{0}Xt) + ...
z = g'Xt) + eg<%) + e^{2}g<$^{6)}$(') + e^{3}g<$^{8)}$(0 + ...

This solution is exactly the same as that from the regular perturbation method. We note that we have generated polynomials in e. Solution accuracy can be controlled to be of O(e") for any n. In solving Example 2, we never integrate any differential equations because the solution of a set of resulting algebraic equations determines values for all the state and algebraic variables at any time. Figure 6 shows how this updating procedure works for Example 2.

The forcing function *sin t* was selected because its higher derivatives never become zero. If the higher derivatives disappear for g(r), those terms disappear in the expansions (72). Higher accuracies will then cease to occur with more augmenting equations, and the test is less interesting. We solved Example 2 using both LSODI [16] and our procedure for *g(t) = sin t.* We specified the error tolerance of KT^{4} for both test cases. LSODI could not solve Example 2 with values of e smaller than KT^{4}. The solutions for e larger than KT^{4} were far from the exact solutions. Tables 1 and 2 show the numerical solutions of variable *y* from LSODI for the cases e = 0.1 and 0.01. As we can see in Table 1 and 2, the solutions from LSODI are wrong. They diverge to infinity, while the exact solutions are bounded by 1. LSODI is unaware it is failing and gives no message to that effect The accuracy requested on LSODI (parameters ATOL and RTOL) were set from 10~^{2} to 10~^{7}. LSODI failed more quickly the smaller the tolerance were set The reason is, of course, that the higher the index of the near index problem, the more incorrect is LSODI's error control, as the discussion with Example 2 illustrates.

Tables 3 and 4 show the numerical solutions for variable *y* using our procedure for e = 0.1 and **0.01** respectively. The first columns are for time r, the second columns are for the solutions from the augmented sets of equations, the third columns are for the solutions from the augmented set plus the third and the fourth sets of augmenting equations, the fourth columns are for the solutions from the augmented set plus the third through the sixth sets of augmenting equations .and the fifth columns are for the exact solutions which are the same as those used in Table 1 and 2.

In Tables 3 and 4, the solutions become more accurate as more augmenting equation sets are used. In

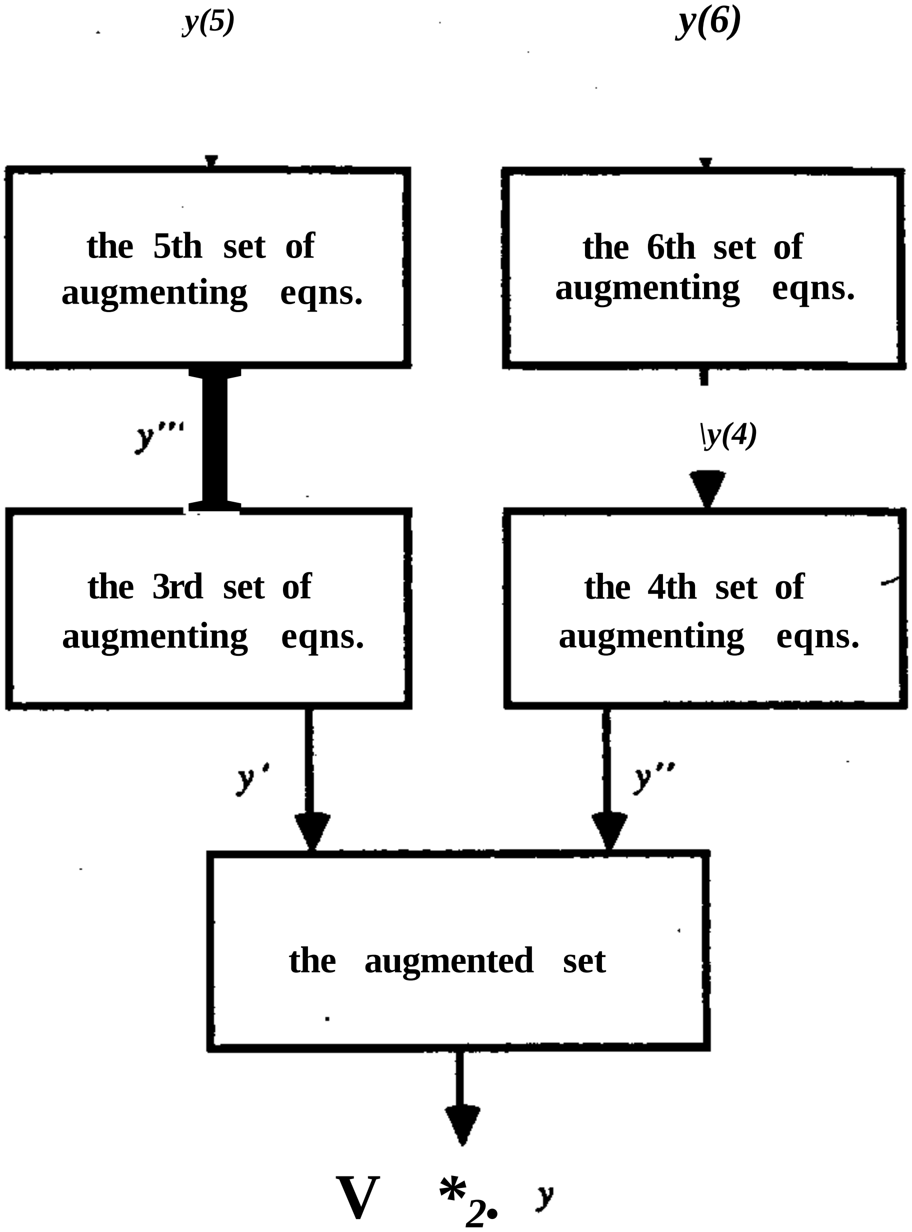

y(5)
y(6)
the 5th set of augmenting eqns.
the 6th set of augmenting eqns.
y'''
y(4)
the 3rd set of augmenting eqns.
the 4th set of augmenting eqns.
y'
y''
the augmented set
V *2. y

Figure 6: Updating Augmented Nonpivot Variables of Example 2

Table 1: Solution fory from LSODI with e = 0.1 in Example 2

time	solution from LSODI	exact solution
0.0	0.0000e+00	0.0000e+00
1.0	0.2635e+01	-0.7650e+O0
2.0	0.7970e+02	-O.8266e+OO
3.0	0.1904e+04	-O.1283e+OO
4.0	0.4502e+05	O.688Oe+OO
5.0	0.1065e+07	0.8718e+OO
6.0	0.2517e+08	0.2540e+00
7.0	0.5953e+09	-0.5973e+00
8.0	0.1408e+ll	-0.8994e+00
9.0	0.3328e+12	-0.3747e+00
10.0	0.7870e+13	0.4946e+00

Table 2: Solution for? from LSODI with e = 0.01 in Example 2

time	solution from LSODI	exact solution
0.0	0.0000e+00	0.0000e+00
1.0	0.1098e+04	-0.8331e+00
2.0	0.2429e+08	-0.9003e+00
3.0	0.5370e+12	-0.1397e+00
4.0	0.1187e+17	0.7493e+00
5.0	0.2624e+21	0.9494e+00
6.0	0.5802e+25	0.2767e+00
7.0	0.1283e+30	-0.6505e+00
8.0	0.2836e+34	-0.9796e+00
9.0	0.6269e+38	-0.4080e+00
10.0	0.1386e+43	0.5386e+00

Table 3: Solution for *y* from Our Method with e = 0.1 in Example 2

time	solution from augmented set	solution after 1st updating	solution after 2nd updating	exact solution
0.0	0.1000	0.0100	0.0010	0.0000
1.0	-0.7415	-0.7473	-0.7647	-0.7650
2.0	-0.8093	-0.8084	-0.8265	-0.8266
3.0	-0.0411	-0.1170	-0.1274	-0.1283
4.0	0.8568	0.6911	0.6897	0.6880
5.0	1.0589	0.8730	0.8736	0.8718
6.0	0.3794	0.2615	0.2553	0.2540
7.0	-0.5570	-0.5813	-0.5969	-0.5973
8.0	-0.8894	-0.8804	-0.8993	-0.8994
9.0	-0.3121	-0.3609	-0.3740	-0.3747
10.0	0.6440	0.4996	0.4961	0.4946

Table 4: Solution for *y* from Our Method with e = 0.01 in Example 2

lime	solution from augmented set	solution after 1st updating	solution after 2nd updating	exact solution
0.0	0.0100	0.0001	0.0000	0.0000
1.0	-0.8315	-0.8330	-0.8331	-0.8331
2.0	-0.8993	-0.9001	-0.9003	-0.9003
3.0	-0.1311	-0.1396	-0.1397	-0.1397
4.0	0.7668	0.7493	0.7493	0.7493
5.0	0.9689	0.9494	0.9494	0.9494
6.0	0.2894	0.2767	0.2767	0.2767
7.0	-0.6470	-0.6503	-0.6505	-0.6505
8.0	-0.9794	-0.9794	-0.9796	-0.97%
9.0	-0.4021	-0.4079	-0.4080	-0.4080
10.0	0.5540	0.5387	0.5386	0.5386

Table 4, we can see that the solution after the second updating procedure is the **same** as the exact solution. This result can **be easily expected** because the accuracy of the solution after the second updating is order of 1CT^{6} which is **more** accurate than the specified error tolerance. We note that the solution in case of the smaller **e** converges faster to the exact solution than that in case of the larger e.

Example 3

Suppose we have two tanks connected as in Figure 7. In the connecting pipe, there is **a** valve to control the flow between two tanks.

The mathematical model for this example problem is as follows.

$$^{A}\backslash^{h}\backslash\ ' = \wedge 0\ \text{"}\ {}^{F}i \qquad (73a)$$
$$A_2 h_2' = F_1 - {}^{F}2 \qquad (736)$$
$$F_1 R = f_1(h_1, A_2) \qquad \mathbf{(73c>}$$
$$F_2 = f_2(h_2) \qquad (73d)$$

where A_i and h_t are the cross-sectional area and the height of the i-th tank, respectively. F_o is the input flow, and F_i is the output flow from the ż-th tank. R is a resistance of the valve.

When the input flow F_o and the valve resistance R are given, this example is **a** well posed and solvable problem. The Jacobian matrix with respect to variables $h\{$, $\backslash t\{_y$ F_x and F_2 is structurally nonsingular. With large enough values for /?, the Jacobian matrix is numerically nonsingular. This is a nonindex problem. It can be solved by the existing ODE/DAE solvers, such as LSODI. If R has zero value, the Jacobian matrix is numerically singular. The problem then has an index problem. The solution method of Chung and Westerberg [4] can be applied to solve it With small values for R_f the Jacobian matrix is nearly singular. Example 3 now has a near index problem. With the error tolerance of 10"^{4}, LSODI failed to solve this problem when R = 10"^{5} or less. The algorithm suggested in this paper readily solves this problem.

The augmented set of equations consists of the original equations (73a) to (73d) and the derivative equation

$$F\{R \quad \wedge(A\wedge V'V) \qquad (^{74})$$

which is obtained by differentiating equation (73c) with respect to time. The augmented set has five

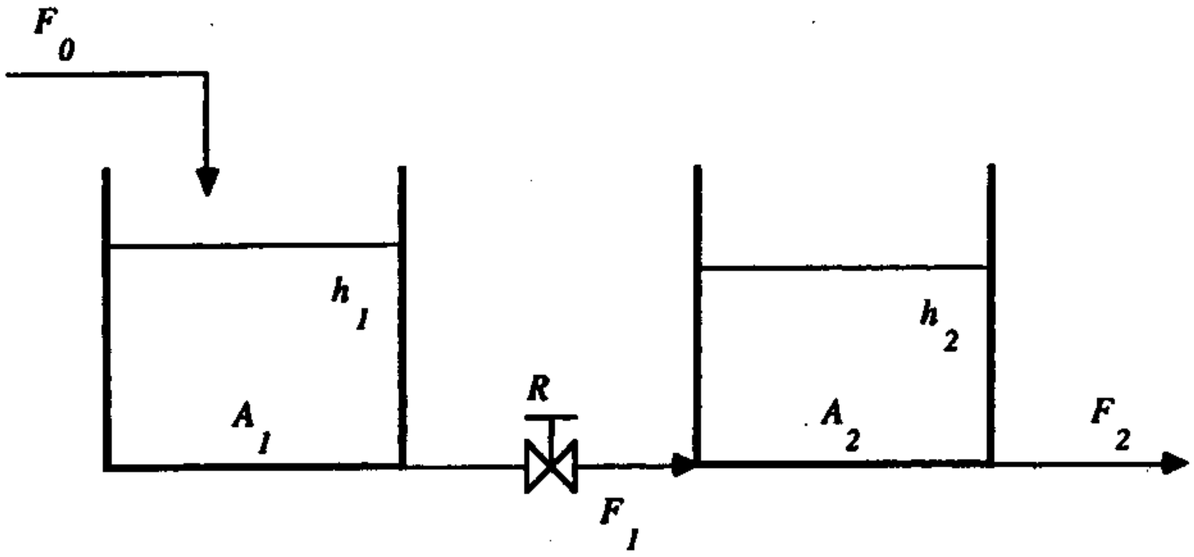


Figure 7: Configuration of Two Tanks in Example 3

equations and seven variables h_v h^ h_x\ h{, F_v F_2 and F_x'. There is one augmented nonpivot variable F_x' whose coefficient is R. Excluding the augmented nonpivot F_x\ the number of variables is still one more than the number of equations. We need to assign an initial value for either variable /i_t or /^ at the starting point and need to integrate this same variable at each propagation step. We note that an explicit integration method can be used if it is stable and accurate enough. The solution from the augmented set has the accuracy of order of R.

The second set of augmenting equations is generated by differentiating equations (73a), (73b), (73d) and (74). It consists of

$$W = {}^{F}0\,' \sim{}^{F}\backslash\,' \tag{75}$$

$${}^{A}2V\ {}^{a}\ {}^{F}l\,' - {}^{F}2\,'$$

$${}^{F}2' = f_4(h_2, h_2')$$

$$F_1''R = f_5(h_1, h_2, h_1', h_2', h_1'', h_2'')$$

The solution of the nine equations (73a) to (75) has an accuracy of order R^2. Differentiating equations (75) will generate the third set of augmenting equations. The solution of this example problem can then be expressed and controlled in terms of polynomials in R.

Solution of Differential-Algebraic Equations

Two major problems to solve for dynamic simulation arc: (1) how to solve stiff systems and (2) how to detect and solve index problems. In this paper, we have extended the notion of index problems to include near index problems. We have also demonstrated that near index problems can be thought as stiff problems. We have shown that stiffness should be classified by index. Figure 8 shows the relation of index problems to stiff problems. The vertical lines in Figure 8 show the relations between each category of these two problems.

Many authors have tried to solve index problems using extensions of stiff integration methods. Their methods are good for moderately stiff systems and some of low index problems. However, they have difficulties in solving high index problems.

We have tried to solve stiff systems with an extension of our solution algorithm for index problems.

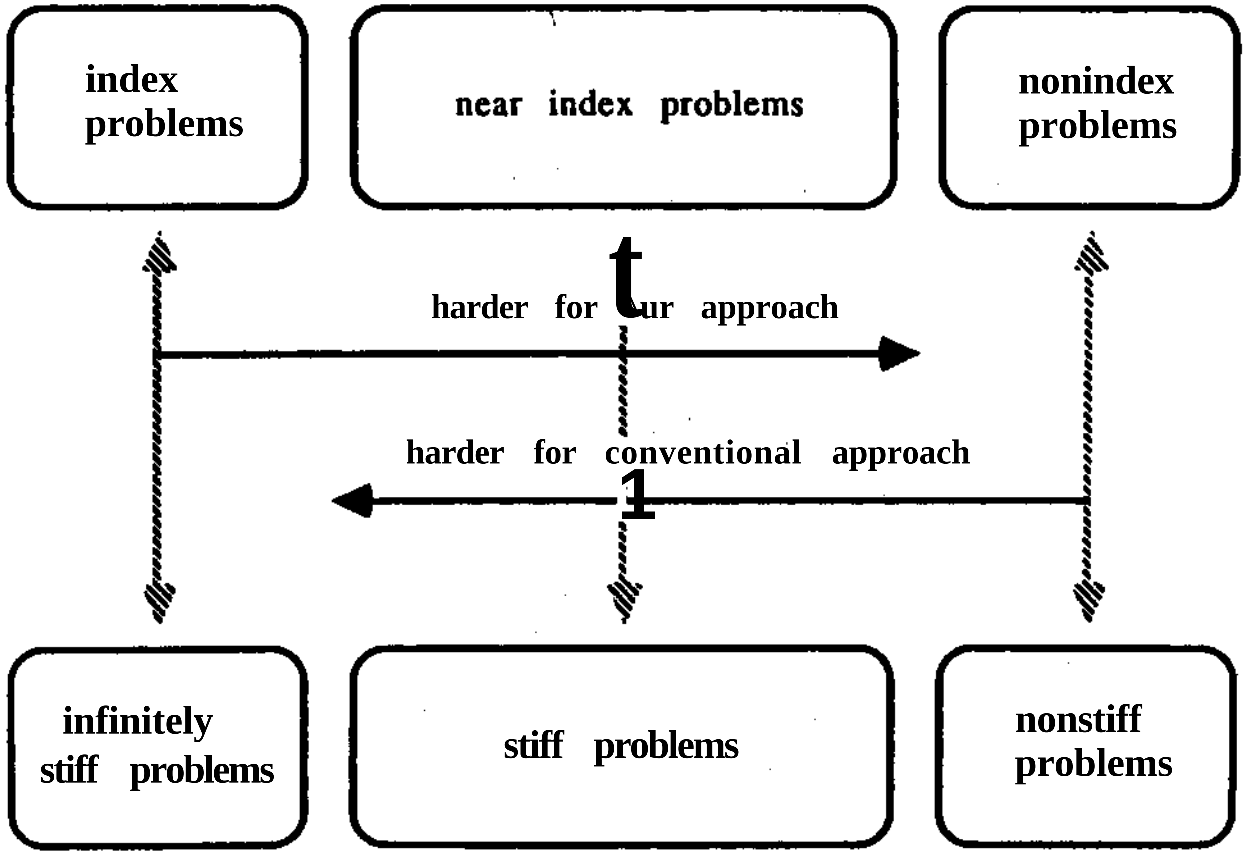


Figure 8: Relations between Index Problems and Stiff Problems

We have extended the algorithm to include near index problems. The solution of a near index problem can be expressed and controlled in terms of polynomials in the small parameters which are responsible for the near singularity at the solution point. Index problem can be solved easily by our algorithm. Near index problems with small enough coefficients are also easy to solve. When the coefficients increase, our method becomes expensive because we need more differentiations and calculations.

As seen in Figure 8, we can trade off two methods in solving differential-algebraic equations. We can use the algorithm presented in this paper to solve index problems and near index problems for which some of the coefficients are very small. When it become expensive with increasing coefficients, existing stiff ODE/DAE methods can be used.

In Conclusion

In this paper, we have shown that stiff DAE systems are the same as near index problems, and existing stiff ODE/DAE solvers, such as LSODI, cannot solve near high index problems. These facts have led us to conclude that stiffness should be classified by index. A new definition of index and a numerical algorithm for solving near index problems have been presented. Solutions of near index problems can be expressed and controlled in terms of polynomials in small coefficients which are responsible for the near singularity at the solution point Solution algorithm for nonlinear index problems has been extended to include near index problems. With our algorithm, stiff DAE systems can be solved accurately and stably even with explicit integration methods.

Acknowledgement

Support for this work comes from the National Science Foundation through its support for Design Research Center, NSF Grant EDC-8943164.

References

[1] **Bachmann, It, L,** Briill, U. Pallaske, and Th. Mrziglod.
A Contribution to The Numerical Treatment of Differential Algebraic Equations Arising in **Chemical** Engineering.
Dechema-Monographs *vol.116-VCH Verlagsgesellschaft*:343-349,1989.

[2] Bachmann, R., L. Brüll, Th. Mrziglod, and U. Pallaske.
On Methods for Reducing the Index of Differential Algebraic Equations.
Comput. Chem. Engng. 14(11):1271-1273,1990.

[3] Brenan, K.E., S.L. Campbell, and L.R.Petzold.
Numerical Solution of Initial-Value Problems in ***Differential-Algebraic Equations.***
North-Holland, N.Y., 1989.

[4] Chung, Y. and A.W. Westerberg.
A Proposed Numerical Algorithm for Solving Nonlinear Index Problems.
Ind. Eng. Chem. Res. 29(7):1234-1239,1990.

[5] Duff, I.S., A.M. Erisman, and J.K.Reid.
Direct Methods for Sparse Matrices.
Oxford, N.Y., 1986.

[6] Finlayson, B.A.
Nonlinear Analysis in Chemical Engineering.
McGraw-Hill, N.Y., 1980.

[7] Gear, C.W.
Simultaneous Numerical Solution of Differential-Algebraic Equations.
IEEE Trans, on Circuit Theory :89-95, January, 1971.

[8] Gear, C.W.
Differential-Algebraic Equation Index Transformation.
SIAMJ. SCI. STAT. COMPUT. 9(l):39-47, January, 1988.

[9] Gear, C.W.
Differential Algebraic Equations, Indices, and Integral Algebraic Equations.
SIAMJ. NUMER. ANAL. 27(6):1527-1534, December, 1990.

[10] Gear, C.W. and L.R. Petzold.
Differential/Algebraic Systems and Matrix Pencils.
Report No. UIUCDCS-R-82-I086, Dept. Comp. Sci., U. Illinois, Urbana , 1982.

[11] Gear, C.W. and L.R. Petzold.
Singular Implicit Ordinary Differential Equations and Constraints.
Report No. UIUCDCS-R-82-U10. Dept. Comp. Sci., U. Illinois, Urbana , 1982.

[12] Gear, C.W. and L.R. Petzold.
ODE Methods for The Solution of Differential/Algebraic Systems.
SIAMJ. NUMER. ANAL. 21(4):716-728, August, 1984.

[13] Gritsis, D., C.C. Pantelides, and R.W.H. Sargent
The Dynamic Simulation of Transient Systems Described by Index Two Differential-Algebraic Equations.
3rd International Symposium on Process System Engineering, Sydney, Australia, August 28-September2,1988.

[14] Gupta, GJC.C.W. Gear, and B. Leimkuhler.
Implementing Linear Multistep Fonnulas for Solving DAEs.
Report No. UlUCDCS-R-85-1205, Dept. Comp. Set, U, Illinois, Urbana, 1985.

[IS] Hairer, E., C Lubich, and M Loche.
The Numerical Solution ofDifferential-Algebraic Systems by Runge-Kutta Methods.
Lecture Notes in Mathematics, 1409, Springer- Verlag, Berlin, 1989.

[16] Hindmarsh, A.C.
Two New Initial Value Ordinary Differential Equation Solvers.
ACM-Signum Newsletters 15(4):10-tl, 1980.

[17] Leimkuhler, B.J., L.R. Petzold, and C.W. Gear.
On Obtaining a Consistent Set of Initial Values for A System of Differential-Algebraic Equations.
Report No. UIUCDCS-R-87-1344, Dept. Comp. Sci., U. Illinois, Urbana, 1987.

[18] Leimkuhler, B., L.R. Petzold, and C.W. Gear.
Approximation Methods for the Consistent Initialization of Differential-Algebraic Equations.
SIAMJ. NUMER. ANAL. 28(l):205-226, February, 1991.

[19] Pantelides, C.C.
The Consistent Initialization of Differential-Algebraic Systems.
SIAMJ. SCI. STAT. COMPUT. 9(2):213-231,1988.

[20] Pantelides, C.C.
SPEEDUP - Recent Advances in Prosess Simulation.
Comput. Chem. Engng. 12(7):745-755,1988.

[21] Pantelides, C.C., D. Gritsis, K.R. Morison, and R.W.H. Sargent
The Mathematical Modelling of Transient Systems Using Differential-Algebraic Equations.
Comput. Chem. Engng. 12(5):449-454,1988.

[22] Petzold-, L.R.
Differential/Algebraic Equations Are Not ODE's.
SIAMJ. SCI. STAT. COMPUT. 3(3):367-384, September, 1982.

[23] Petzold, L.R.
A Description of DASSL: A Differential/Algebraic System Solver.
In *10th IMACS World Congress on System Simulation and Scientific Computation,* pages 430-432. 1983.

[24] Petzold, L JL
Order Results for Implicit Runge-Kutta Methods Applied to Differential/Algebraic Systems.
SIAMJ. Numer. Anal. 23(4):837-852, August, 1986.

[25] Seinfeld, J.H., L. Lapidus, and M. Hwang.
Review of Numerical Integration Techniques for Stiff Ordinary Differential Equations.
Ind. Eng. Chem. Fundam. 9(2):266-275,1970.

[26] Shampine, L.F. and C.W. Gear.
A User's View of Solving Ordinary Differential Equations.
SIAM Review 2\{\):\-\l, 1979.

[27] Sincovec,R.F., A.M. Erisman, E.L. Yip, and M.A. Epton.
Analysis of Descriptor Systems Using Numerical Algorithms.
IEEE Trans. Automatic Control AC-26(1):139-147, February, 1981.

[28] Strang.G.
Linear Algebra and its Applications.
Academic, N.Y., 1980.

[29] Wilkinson, J.H.
Linear Differential Equations and Kronecker's Canonical Form.
Recent Advances in Numerical Analysis.
Academic, N.Y., 1978, pages 231-265.

[30] Yip, E.L. and RJ\Sincovec.
Solvability, Controllability, and Observability of Continuous Descriptor Systems.
IEEE Trans. Automatic Control AC-26(3):702-707,1981.