Abstract

Numerical difficulty arises when DAEs are solved with inconsistent initial values of dependent variables. It can cause the solution failures of many popular DAEs solvers. An initialization subroutine, DAEIS (Differential Algebraic Equations Initialization Subroutine), has been developed to handle this issue. In DAEIS, consistent initial values of dependent variables for index-1 DAEs are obtained through solving a set of nonlinear equations based on the initialization settings. To achieve good efficiency and robustness of the initialization computation, a nonlinear equation solver, GNES, has been specially built for DAEIS. The usage of DAEIS is demonstrated through handling inconsistent initializations for combined continuous/discrete simulations. © 2001 Elsevier Science Ltd. All rights reserved.

Keywords: Consistent initialization; Differential algebraic equations; DAEs solver; Nonlinear equation solver; Hybrid simulation; Combined continuous/discrete simulation

1. Introduction

Differential and algebraic equations (DAEs) are normally obtained when modeling chemical engineering systems. They are more difficult to handle than ODEs (ordinary differential equations) due to the existence of algebraic equations. A system of DAEs is characterized by its index, which is the number of differentiations required to convert it into a system of ODEs. DAEs with index > 1 are generally hard to solve and are still under active research. DAEs with index-1 are usually encountered. They have intrinsic relationships with stiff ODEs and can be solved with the similar numerical algorithms. Several popular numerical solvers for index-1 DAEs are available at present, e.g. DASSL (Brenan, Campbell, & Petzold, 1989), RADAU5 (Hairer & Wanner, 1991), and LIMEX (Deufhld, Hairer, & Zugck, 1987). However, solution failures are frequently encountered with these solvers when inconsistent initial values of dependent variables for DAEs are presented (Kröner, Marquardt, & Gilles, 1992).

For ODEs, the consistent initialization is a trivial issue, e.g. giving reasonable initial values for all dependent variables. However for DAEs, due to the constraints of algebraic equations, the degree of freedom of the initial setting is less than the number of dependent variables, and setting consistent initial values for dependent variables becomes a nontrivial task (Kröner et al., 1992; Kröner, Marquardt, & Gilles, 1997; Leimukhler, Petzold, & Gear, 1991). A simple example is given below

\[ \frac{\rho V}{W} y_1' = \frac{j_1}{F} \quad (1a) \]
\[ j_1 + j_2 - i_{app} = 0, \quad (1b) \]

where

\[ j_1 = i_{eq} \left[ 2(1 - y_1) \exp \left( \frac{0.5F}{RT} (y_2 - \phi_{eq,1}) \right) - 2y_1 \exp \left( -\frac{0.5F}{RT} (y_2 - \phi_{eq,1}) \right) \right]. \quad (2a) \]

*Corresponding author. Tel.: +1-803-7773270; fax: +1-803-7778265. E-mail address: white@engr.sc.edu (R.E. White).
The values of parameters in Eq. (2) are \( F = 96487, R = 8.314, T = 298.15, \phi_{eq,1} = 0.420, \phi_{eq,2} = 0.303, \rho = 3.4, W = 92.7, V = 1 \times 10^{-2}, i_01 = 1 \times 10^{-4}, \) and \( i_02 = 1 \times 10^{-10}, \) and \( t_{app} = 1 \times 10^{-5}. \) The units of parameters and dependent variables are omitted here for simplicity. The above DAEs were obtained from the modeling of galvanostatic charge/open-circuit/discharge processes of a thin film nickel hydroxide electrode. The dependent variable \( y_1 \) is the mole fraction of NiOOH and \( y_2 \) the potential difference at the solid-liquid interface.

Consistent initial values for the dependent variables in Eq. (1) are not easy to choose. For a discharged nickel electrode, the initial values of dependent variables can be estimated as

\[
y_1(0) = 0.05, \quad (3a) \\
y_2(0) = 0.38. \quad (3b)
\]

However, the values in Eq. (3) are not consistent with respect to Eq. (1), i.e. they are not solutions of Eq. (1).

Many DAEs solvers failed when solving Eq. (1) with the initial values given by Eq. (3). Three popular DAEs solvers: DASSL, RADAU5, and LIMEX, were used in this work to demonstrate the inconsistent initialization problem. These three solvers have been built upon different algorithms: DASSL used the backward difference formulae (BDF) method (Brenan et al., 1989); RADAU5 used the implicit Runge–Kutta method (Hairer and Wanner, 1991); and LIMEX used the extrapolation method based on the implicit Euler algorithm (Dieuflhard et al., 1987). To achieve good integration efficiency, all three solvers are able to adapt the integration step size based on the previous solutions. However, it was found that these solvers had troubles to solve DAEs (especially nonlinear DAEs) with inconsistent initial values of dependent variables. The failure messages from these solvers when solving Eq. (1) with initial values given by Eq. (3) are listed below:

**DASSL:** THE CORRECTOR FAILED TO CONVERGE REPEATEDLY OR WITH ABS(H) = HMIN.

**RADAU5:** MATRIX IS REPEATEDLY SINGULAR.

**LIMEX:** MORE THAN JRMAX = 20 STEP-SIZE REDUCTIONS DUE TO EXTRAPOLATION TABLEAU.

The initialization failure of DAEs solvers is normally caused by two numerical reasons: (1) the convergence problem of the Newton iterations due to poor initial guesses; and (2) the singular Jacobian matrix resulted from extremely small integration step sizes. Inside a DAEs solver, the Newton iteration algorithm is usually fine-tuned to provide good efficiency, e.g. reusing the Jacobian matrix for several iterations. However, this also causes robust problems when handling nonlinear equations with poor initial guesses. When the convergence rate of the Newton iterations is poor, DAEs solvers usually adjust the integration step size to a smaller value to improve the guessing accuracy. This is a valid approach for DAEs with consistent initial values, but it is not useful at all for DAEs with inconsistent initial values. Then the integration step size will be reduced with each failure of the Newton iterations. Eventually the integration step size is so small that the Jacobian matrix becomes singular and the solution failure is resulted.

By adjusting the initial values of dependent variables, it is found that when the initial values of dependent variables are sufficiently near a consistent setting, these DAEs solvers are able to start and finish the integration without problems. Table 1 shows the convergence range of one dependent variable while the other dependent variable is held constant when solving Eq. (1) with different DAEs solvers. It is observed that LIMEX is more tolerable with inconsistent initial values than DASSL, which in turn is more tolerable with inconsistent initial values than RADAU5. Since Eq. (1) is only a simple set of DAEs, it can be expected that the convergence range of dependent variables for large size nonlinear DAEs will be very limited with these DAEs solvers.

<table>
<thead>
<tr>
<th>Solver</th>
<th>Convergence range when ( y_1(0) = 0.05 )</th>
<th>Convergence range when ( y_2(0) = 0.38 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>DASSL</td>
<td>( 0.321 \leq y_1(0) \leq 0.370 ) ( 0.071 \leq y_1(0) \leq 0.352 )</td>
<td>( 0.071 \leq y_1(0) \leq 0.352 )</td>
</tr>
<tr>
<td>LIMEX</td>
<td>( 0.318 \leq y_1(0) \leq 0.377 ) ( 0.055 \leq y_1(0) \leq 0.418 )</td>
<td>( 0.055 \leq y_1(0) \leq 0.418 )</td>
</tr>
<tr>
<td>RADAU5</td>
<td>( 0.348 \leq y_1(0) \leq 0.352 ) ( 0.143 \leq y_1(0) \leq 0.190 )</td>
<td>( 0.143 \leq y_1(0) \leq 0.190 )</td>
</tr>
<tr>
<td>DAEIS</td>
<td>( -0.974 \leq y_1(0) \leq 1.663 ) ( 0.0 \leq y_1(0) \leq 1.0 )</td>
<td>( 0.0 \leq y_1(0) \leq 1.0 )</td>
</tr>
<tr>
<td>Consistent value</td>
<td>( y_2(0) = 0.35024 ) ( y_2(0) = 0.15513 )</td>
<td>( y_2(0) = 0.15513 )</td>
</tr>
</tbody>
</table>

For Eq. (1), comparing the integration results from inconsistent initial values of dependent variables with those from consistent initial values of dependent variables showed that the integration results from inconsistent values are not influenced by the initial value of
the algebraic variable \( y_2(0) \) when the integration is successful. This is actually an expected result for many implicit methods. For example, if the implicit Euler method is used to solve the following semi-explicit index-1 DAEs,

\[
\begin{align*}
y'_1 &= f(t, y_1, y_2), \\
g(t, y_1, y_2) &= 0,
\end{align*}
\]  

(4a) (4b)

where \( y_1 \) is the vector of differential variables and \( y_2 \) is the vector of algebraic variables. Approximating the time derivatives of dependent variables with the first order backward difference formula gives

\[
\begin{align*}
y_1(t_0 + \Delta t) - y_1(t_0) &= f(t, y_1(t_0 + \Delta t), y_2(t_0 + \Delta t)), \\
g(t, y_1(t_0 + \Delta t), y_2(t_0 + \Delta t)) &= 0.
\end{align*}
\]  

(5a) (5b)

It is obvious that \( y_2(t_0) \) does not show up in Eq. (5) at all, thus \( y_2(t_0) \) has no effects on the final solution of the implicit Euler integration.

As \( \Delta t \rightarrow 0 \), the following equations are obtained

\[
\begin{align*}
y_1(t_0) - y_1(t_0) &= 0, \\
g(t, y_1(t_0), y_2(t_0)) &= 0.
\end{align*}
\]  

(6a) (6b)

It is obvious that the continuity of differential variables is maintained, but the converged values of algebraic variables may have value jumps if the initial value \( y_2(t_0) \) does not satisfy Eq. (3), i.e. \( y_2(t_0) \neq y_2(t_0^+) \), where \( y_2(t_0^+) \) is consistent with \( y_1(t_0) \).

It can be easily verified that when applying the implicit Euler method on 1 for one integration step, the value of \( y_2 \) after the first integration step approaches a constant value, i.e. \( y_2(0^+) \), as the integration step size gets smaller.

Since there are only two dependent variables in Eq. (1), it is still possible to find a near consistent set of initial values by the trial and error method to start the DAEs solvers. For general nonlinear DAEs (especially large-size DAEs), it is usually infeasible to take such an unreliable and time-consuming approach. An initialization subroutine for index-1 DAEs, DAEIS, was thus developed in this work to provide consistent initializations for DAEs solvers.

2. Consistent initialization for DAEs solvers

Some approaches have been proposed for the consistent initialization of DAEs (Leimkuhler et al., 1991; Kröner et al., 1992; Kröner et al., 1997; Brown, Hindmarsh, & Petzold, 1998; Shampine, Reichelt, & Kierzenka, 1999). Among them, the implicit Euler method without the error control on the continuity of algebraic variables has been widely used. Some ad hoc approaches are also available, e.g. the dummy variable method in Appendix A is one effective approach to get around the initialization problem of DASSL. Since index-1 DAEs with differential and algebraic variables explicitly identified are the most encountered DAEs when modeling physical systems (in fact most DAEs solvers are designed for solving this kind of DAEs only), the consistent initialization calculation in this work focuses on the index-1 DAEs in the form

\[
\begin{align*}
F(t, y_1, y_1', y_2) &= 0, \\
g(t, y_1, y_2) &= 0,
\end{align*}
\]  

(7a) (7b)

where \( y_1 \) and \( y_2 \) are the vector of differential variables and the vector of algebraic variables, respectively. Semi-explicit DAEs Eq. (4) is a special form of Eq. (7).

Supposing the DAEs in Eq. (7) have \( n \) equations, \( p \) differential variables, and \( n-p \) algebraic variables, to uniquely determine all the initial values of dependent variables and time derivatives of dependent variables, \( p \) degrees of freedom exist. The finding of a consistent set of initializations for Eq. (7) can actually be treated as specifying \( p \) variables of \( y_1, y_2 \) and \( y_1' \) and solving the left \( n \) variables (Pantelides, 1988; Brown et al., 1998). For example, a consistent set of initial values for Eq. (1) can be obtained by setting the value of one variable of \( y_1(0), y_2(0), \) and \( y_1'(0) \), and the other two variables can be determined from Eq. (1), i.e. based on Eq. (1) the following equations exist at the initial time

\[
\frac{\rho V}{W} y_1'(0)
\]

\[
= \frac{i_{01}}{F} 
\left[ 2(1 - y_1(0)) \exp \left( \frac{0.5F}{RT} (y_2(0) - \phi_{eq,1}) \right) + \frac{F}{RT} \left( y_2(0) - \phi_{eq,1} \right) \right] 
\]

(8a)

\[
= \frac{i_{01}}{F} 
\left[ 2(1 - y_1(0)) \exp \left( \frac{0.5F}{RT} (y_2(0) - \phi_{eq,1}) \right) 
- 2y_1(0) \exp \left( \frac{0.5F}{RT} (y_2(0) - \phi_{eq,1}) \right) \right] 
\]

(8b)

\[
\frac{i_{01}}{F} 
\left[ 2(1 - y_1(0)) \exp \left( \frac{0.5F}{RT} (y_2(0) - \phi_{eq,1}) \right) 
- 2y_1(0) \exp \left( \frac{0.5F}{RT} (y_2(0) - \phi_{eq,1}) \right) \right] 
+ i_{02} \left[ \exp \left( \frac{F}{RT} (y_2(0) - \phi_{eq,2}) \right) 
- \exp \left( \frac{F}{RT} (y_2(0) - \phi_{eq,2}) \right) \right] - i_{app} = 0
\]
than differential variables, e.g. for Eq. (1), the potential of a charge process at the starting time point is usually easier to estimate than the mole fraction of NiOOH. Thus, it is desirable to be able to set the initialization values with mixed algebraic variables and differential variables, i.e. the values of accurately known variables should not be changed during the initialization process. This cannot be achieved by DASPK and the implicit Euler integration method. In this work, DAEIS is designed to provide more flexible control of the initialization process, and to provide an independent subroutine that can be used for any DAEs solvers.

The kernel of DAEIS is constructed around the conversion from DAEs to NEs (nonlinear equations) and a nonlinear equation solver, GNES (General Nonlinear Equation Solver). The conversion from DAEs to NEs is through a special designed procedure that prepares the nonlinear equations from the DAEs based on user specifications, as summarized below:

1. Map the initial values of the undetermined variables of $Y$ and $YPRIME$ to one vector of dependent variables $X$ based on the initialization specifications, and call nonlinear equation solver GNES to solve $X$.

2. In the residual evaluation subroutine for GNES, define two vectors $Y_1$ and $YPRIME_1$. Set each variable in $Y_1$ and $YPRIME_1$ equal to the corresponding initial value in $Y$ and $YPRIME$ if it is supposed to remain constant, otherwise set it equal to the corresponding iteration value in $X$ (using the same variable mapping as in step 1) provided by GNES.

3. Call the residual evaluation subroutine of DAEs with $Y_1$ and $YPRIME_1$ to get the residue evaluation for nonlinear equations needed by GNES.

4. $X$ will be solved out by GNES as long as the initialization setting is correct and the corresponding nonlinear equations are solvable.

5. The consistent set of values of $Y$ and $YPRIME$ can be constructed based on the final solution of $X$ from GNES.

The implementation of the above procedure involves some programming tricks, the details of which can be found in the source code of DAEIS (available from authors). After calling DAEIS, the consistent values for the dependent variables are available in vectors $Y$ and $YPRIME$. Then the DAEs solvers can be called to start the integration of DAEs. Since DAEIS has similar calling protocol to that of DASSL family solvers, it can share many parameters and working space with these solvers. The application of DAEIS and DASSL for Eq. (3) with initial value given by Eq. (10a) is listed in Appendix B. In this

If set

$$y_1(0) = 0.05,$$  \hspace{1cm} (9a)

solving Eq. (8) yields

$$y_1'(0) = 2.8259 \times 10^{-4},$$  \hspace{1cm} (9b)

$$y_2(0) = 0.35024.$$  \hspace{1cm} (9c)

However, if set

$$y_2(0) = 0.38,$$  \hspace{1cm} (10a)

solving Eq. (8) gives

$$y_1(0) = 0.15513,$$  \hspace{1cm} (10b)

$$y_1'(0) = 2.8255 \times 10^{-4}. $$  \hspace{1cm} (10c)

Setting $y_1'(0)$ to determine $y_1(0)$ and $y_2(0)$ makes little sense for Eq. (1) and thus is not presented here. The integration results from above two settings are given in Fig. 1. It is noted from Fig. 1 that different choices of initialization strategy produce quite different results. Thus care should be taken in the setting and calculation of initial values for DAEs, otherwise, undesired results could be obtained.

In DASPK (Brown et al., 1998), an updated version of DASSL, two types of initialization controls are available: (i) set the initial values of differential variables; (ii) set the initial values of the time derivatives of differential variables. For the first case, the initial values of algebraic variables and time derivatives of differential variables will be determined. For the second case, the initial values of differential variables and algebraic variables will be determined. In both cases, the values of algebraic variables have no influence on the integration results. However, sometimes, algebraic variables are known more accurately
case, the algebraic variable is more accurately known initially than the differential variable, and both the differential variable and its time derivative are determined to be consistent with the algebraic variable. This cannot be handled by both the implicit Euler integration and the DASPK initialization routine. As can be observed in Appendix B, the usage of DAEIS for Eq. (1) is quite straightforward and only tens of lines of code are needed. The convergence ranges of DAEIS for Eq. (1) are shown in Table 1. If choosing $y_1(0) = 0.38$ as the initial condition, DAEIS can converge for the whole range of $y_1$ that makes physical sense. If choosing $y_1(0) = 0.05$ as the initial condition, DAEIS can converge for a much wider range of $y_1$ than the listed DAEs solvers. Table 1 clearly shows that DAEIS is much more effective in handling inconsistent initializations than DAEs solvers and thus should be used whenever initialization difficulty arises with DAEs solvers.

3. Nonlinear equation solving in DAEIS

Solving nonlinear equations is a fundamental problem in numerical simulations (Rice & Do, 1995; Ascher & Petzold, 1998; Shacham, Brauner, & Pozin, 1998). There are many FORTRAN solvers for nonlinear equations available in the public domain, e.g. HYBRD, BRENT, and DSOS (http://www.netlib.org). However, testing of these solvers showed that they could not be used as an embedded solver in DAEIS due to their poor efficiency. It would be awkward that with the same amount of time that an initialization routine costs, a good DAEs solver could integrate hundreds of steps. For a DAEs problem with frequent restarts, the low efficiency of the initialization routine will be intolerable. A nonlinear equation solver, GNES, has been developed to achieve high efficiency as well as good robustness of DAEs.

For nonlinear equations, many solving methods have been invented, including the bisection method, the successive substitution method, and the Newton–Raphson method (Rice & Do, 1995), etc. Among them, the Newton–Raphson (also called Newton iterations) method and its variations are widely used due to their simplicity and good convergence rate. A variation of the Newton–Raphson method is also used in GNES. The efficient and robust linear algebraic routines in LINPACK have been used to provide the LU decomposition and the solving of linear systems as did in DASSL and LIMEX.

The following convergence criterion has been used to determine the convergence of the Newton iterations in GNES

$$ \| x^{k+1} - x^k \| < 0.5. \quad (11) $$

Error tolerances are implicitly considered in the above convergence test with the usage of the following weighted norm

$$ DNENRM = \frac{1}{N} \sum_{j=1}^{N} \left| \frac{x_j}{WT_j} \right|, \quad (12) $$

where the weight vector is calculated by using the tolerance settings

$$ WT_j = RTOL (j) ABS(x_j) + ATOL (j). \quad (13) $$

In Eq. (13), the relative tolerance becomes more important when the dependent variable is large, and absolute tolerance becomes more important when the dependent variable is small. The weights of dependent variables are updated during the Newton iterations in GNES.

Since analytical derivatives are usually difficult to obtain, they are approximated by the forward finite difference formula in GNES, i.e. the following equation is used

$$ \frac{\partial F_i}{\partial x_j} = \frac{F_i(x + e_j h_j) - F_i(x)}{h_j} \quad i = 1 \cdots n, \quad j = 1 \cdots n, \quad (14) $$

where $e_j$ is the $j$th component of an $n$-dimension unit vector and $h_j$ is a small perturbation value of the $j$th component of dependent variables. The following perturbation value has been used

$$ h_j = U^{0.5} \max(|x_j|, |WT_j|, 1.0), \quad (15) $$

where the unit round-off of the machine is the smallest positive number $U$ that satisfies

$$ 1.0D0 + U \neq 1.0D0 \quad (16) $$

Since the Jacobian evaluation is very costly, the Jacobian matrix is reused during the iterations to achieve high efficiency. Damped update has been used to improve the robustness of the Newton iterations, i.e. the following equation is used

$$ x^{k+1} = x^k - \delta J^{-1} (x^k) F(x^k), \quad (17) $$

where $\delta$ is the damping coefficient, which will be decreased with each failed trial. The corresponding convergence criterion is

$$ \| x^{k+1} - x^k \| < 0.5\delta. \quad (18) $$

The steps taken by GNES are summarized below.

1. Set maximum iteration number to 500, maximum Jacobian reuse number to 3, and damping coefficient to 0.5;
2. Compute the unit round-off of the machine. Save initial guesses of dependent variables;
Table 2  
Convergence performance of GNES compared to some popular mathematical software tools when solving problem C-1 and C-2*  

<table>
<thead>
<tr>
<th>Initial Guess</th>
<th>GNES</th>
<th>Maple</th>
<th>Matlab</th>
<th>Polymath</th>
</tr>
</thead>
<tbody>
<tr>
<td>( C_{D} ), ( C_{X} ), ( C_{Z} )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(a) Results for Problem C-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.61, 0.2, 0.4</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>0.59, 0.2, 0.4</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>0.7, 0.28, 0.4</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>0.7, 0.31, 0.4</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>0.7, 0.2, 0.48</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>0.7, 0.2, 0.51</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>(b) Results for Problem C-2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Initial Guess ( x_{11}, x_{12}, x_{21}, x_{22} )</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0, 1, 1, 0</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>0.05, 1, 0.95, 0</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
</tr>
<tr>
<td>0.1, 1, 0.9, 0</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>0.2, 1, 0.8, 0</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>0, 0.5, 1, 0.5</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>0, 0.3, 1, 0.7</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
</tbody>
</table>

* 'Y', converged to solution; 'N', did not converge.

The comparisons of the convergence results of GNES and those of some popular software tools (Shacham et al., 1998) for problems C-1 and C-2 are shown in Table 2. The comparisons of the computing time of GNES and those of several other nonlinear equation solvers for problem C-3 are given in Table 3. The significantly good efficiency of GNES is mainly due to the second order convergence rate of the Newton iterations (other solvers used more conservative methods) and the usage of highly efficient LINPACK routines for matrix manipulations (other solvers used less efficient routines). As shown in Tables 2 and 3, satisfactory robustness and efficiency results are demonstrated by GNES compared to other solvers on these nonlinear problems.

4. Application of DAEIS in combined continuous/discrete simulation

Simulation of one process of a physical system is straightforward, i.e. providing appropriate initial conditions and using ODEs/DAEs solvers to obtain the results. However, a system with combined continuous processes and discrete events are frequently encountered (Branicky & Mattsson, 1997; Barton & Pantelides, 1994). To simulate such systems, a sequence of DAEs needs to be solved (Barton & Pantelides, 1994)

\[ F^{(1)}(t, y, y') = 0, \quad [t^{(0)} , t^{(1)}], \]  
\[ F^{(2)}(t, y, y') = 0, \quad [t^{(1)} , t^{(2)}], \]  
\[ \ldots \]  
\[ F^{(n)}(t, y, y') = 0, \quad [t^{(n-1)} , t^{(n)}]. \]  

There are two important issues related to the solving of the combined continuous/discrete processes. One is the discrete event detection during the integration of a continuous process and the other is the consistent initialization for the subsequent process (Barton & Pantelides, 1994).

Table 3  
Computation time used by GNES and other nonlinear equation solvers when solving problem C-3*  

<table>
<thead>
<tr>
<th>Name</th>
<th>Size of solver (KByte)</th>
<th>Error tolerance specification</th>
<th>Time taken (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNES</td>
<td>40</td>
<td>RTOL = 10^{-8}</td>
<td>2.09</td>
</tr>
<tr>
<td>HYBRD</td>
<td>58</td>
<td>ATOL = 10^{-8}</td>
<td>23.12</td>
</tr>
<tr>
<td>BRENT</td>
<td>16</td>
<td>RTOL = 10^{-8}</td>
<td>90.52</td>
</tr>
<tr>
<td>DSOS</td>
<td>244</td>
<td>RTOL = 10^{-8}</td>
<td>507.25</td>
</tr>
</tbody>
</table>

* On an IBM compatible PC with Pentium II 266 MHz and 128 M RAM.
For the combined continuous/discrete simulation, the ending of each continuous process is due to the occurrence of a discrete event, either explicit or implicit (Barton & Pantelides, 1994; Park & Barton, 1996). An explicit event is specified by the independent time variable only. It is actually the normal working mode of DAEs solvers (i.e. ending time specification). However, an implicit event is based on some dependent variables. It can only be determined during the solving process of the continuous process. The detection of discrete events usually requires special handling in the DAEs solving algorithm. A widely used approach is to check the status of the discrete events (described in the residue function form) at the end of each successful integration step. If one discrete event is detected, the earliest time that event occurred is then determined by interpolations (Brenan et al., 1989; Barton & Pantelides, 1994). Few DAE solvers with the event detection function are available today. DASRT, an extension of DASSL (Brenan et al., 1989), is used in this work. In DASRT, discrete events need to be defined in a separate subroutine in the following form

\[ g(t, y) = 0. \tag{20} \]

The sign change in any function of Eq. (20) indicates the occurrence of a discrete event. If a discrete event occurs during the integration process, DASRT will return to the calling program and provide the indication of which event has been detected.

Consistent initialization is not only needed for the first process in the combined continuous/discrete simulation. When switching between two different processes, inconsistency of initial values naturally arises because the values of dependent variables from the previous process usually do not satisfy the governing equations of the next process. Normally, if there are no outside disturbances, the initial values of dependent variables at the beginning of the succeeding process can be simply determined from the ending state of the previous process by assuming the continuity of differential variables (conservative quantities). DASRT, like other DAEs solvers, cannot handle well the inconsistent initialization problem encountered in the combined discrete/continuous simulation. However, if combining DASRT with DAEIS, a simple approach for simulations in the continuous/discrete domain is available:

1. DAEIS is used for the consistent initialization of a process. For the consistent initialization of the first process, the initial values of more accurately known variables are maintained and the initial values of other variables are determined to be consistent with these variables. When switching between processes with no outside disturbances, the values of differential variables are maintained and the values of algebraic variables will be determined to be consistent with those of differential variables.

2. DASRT is utilized for solving the DAEs and detecting discrete events. The equations of continuous processes are defined in one subroutine and the equations of discrete events are defined in another subroutine. When a discrete event is detected by DASRT, the current process will be terminated and the subsequent process will be initiated.

The simulation of the galvanostatic charge/open-circuit/discharge processes of the thin film nickel electrode given by Eq. (1) is presented as an example. The simulation is for thirty cycles of the following processes:

1. Charge at \( i_{app} = 1 \times 10^{-5} \) for 500 s for the first cycle, with 100 s more for each additional cycle. Terminate the process if the potential reaches 0.60 V.
2. Open-circuit (\( i_{app} = 0 \)) for 500 s. Terminate the process if the potential reaches 0.25 V.
3. Discharge at \( i_{app} = 1 \times 10^{-5} \) for 1000 s. Terminate the process if the potential reaches 0.25 V.

The initial condition for the first process is given in Eq. (10a). The differential variable \( y_1 \) and its time derivative \( y'_1 \) are determined by DAEIS to be consistent with the algebraic variable \( y_2 \) for the first process. The consistent initialization values for subsequent process are calculated by DAEIS to maintain the continuity of the differential variable \( y_1 \) (then \( y'_1 \) and \( y_2 \) will be determined to be consistent with \( y'_1 \)).

The discrete events in the example are given by the time specification of each process (explicit discrete event) and also by the cut-off value of potential \( y_2 \) (implicit discrete event). If a discrete event is triggered, the simulation will move on to the next process. There are less than three hundred lines of FORTRAN code (available from authors upon request) to solve this problem with DAEIS and DASRT. The results of simulation are shown in Fig. 2. It is observed that process 1 and 2 are always ended by the explicit event of time specifications. However, process 3 is stopped by the cut-off potential at first because nickel active material is not sufficiently charged. Then with the increase of charging time, process 3 cannot consume all the inputting charge and eventually is stopped by the explicit time event.

The DAEs resulted from modeling of physical systems are usually more complex than the above example, however, the numerical solution procedure for combined continuous/discrete simulations is essentially the same. The advantage of using DAEIS to handle the
inconsistent initialization problem is obvious for this kind of simulations.

5. Conclusions

An initialization subroutine, DAEIS, has been designed for semi-explicit index-1 DAEs. It can be used to solve the frequently encountered initialization failure of many DAEs solvers. From a simple DAEs problem, it is shown that the initialization freedom of DAEs is less than the number of equations, thus care must be taken for the initialization process of DAEs. The efficient and robust solution of nonlinear equations is essential for a DAEs initialization subroutine. A nonlinear equation solver based on the Newton iterations and the LINPACK routines has been used in DAEIS to achieve good efficiency and robustness. Examples are presented to demonstrate the usage of DAEIS.
Appendix A. Get around the initialization failure of DASSL with the dummy variable approach

Dummy variables can be used to decrease the index of DAEs, e.g. converting index-2 DAEs to index-1 DAEs (Ascher & Petzold, 1998). It is also an effective approach to get around the initialization failure of DASSL, e.g. for semi-explicit index-1 DAEs,

\[ y_1' = f(t, y_1, y_2), \]  
\[ g(t, y_1, y_2) = 0. \]  

Define all algebraic variables to be the time derivatives of some variables, i.e.

\[ y_2 = f'_2. \]  

Substituting Eq. (A2) into Eq. (A1) yields a system of implicit ODEs

\[ y_1' = f(t, y_1, f'_2), \]  
\[ 0 = g(t, y_1, f'_2). \]
Fig. 2. Results of the combined continuous/discrete simulation of Eq. (1).

It is obvious that \( \tilde{y}_2 \) does not appear in Eq. (A3), thus the values for \( \tilde{y}_2 \) can be arbitrarily set. The initialization setting for Eq. (A3) can then be initial values for \( y_1 \) and arbitrary initial values for \( \tilde{y}_2 \), and the initial values of \( y'_1 \) and \( \tilde{y}_2 \) can be calculated out from Eq. (A3). Since \( \tilde{y}_2 \) has finite values and the following equation holds

\[
\tilde{y}_2 = \int_{t_0}^{t} \tilde{y}_2 \, dt,
\]

the continuity of \( \tilde{y}_2 \) can be guaranteed even if \( \tilde{y}_2 \) has step discontinuities. Then the continuity constraints used by DASSL in the error control can be always met, and thus there is no initialization trouble for DASSL to solve Eq. (A3). Since the time derivatives \( \tilde{y}'_2 \) (equivalent to \( y_2 \)) will be provided by DASSL at each time step, solving Eq. (A3) with DASSL will give the desired solutions of the original DAEs Eq. (A1).

Appendix B. List of code for solving Eq. (1) with DAEIS and DASSL

Appendix C. Testing nonlinear equation problems

Problem C-1:
Example 1 in Shacham et al., 1998.

Problem C-2:
Example 3 in Shacham et al., 1998.

Problem C-3:
A boundary value problem with two dependent variables:

\[
\begin{align*}
\frac{d^3 y_1}{dz^3} - K_1 y_1^3 &= 0, \quad \text{for } z \in [0, 1], \\
\frac{d^3 y_2}{dz^3} - K_2 y_1 y_2 &= 0, \quad \text{for } z \in [0, 1],
\end{align*}
\]

with boundary conditions

\[
\begin{align*}
y_1|_{z=0} &= 0.21, \\
\frac{dy_1}{dz}|_{z=0} &= 0, \\
\frac{dy_1}{dz}|_{z=1} &= 0, \\
y_2|_{z=1} &= 0.127.
\end{align*}
\]

Applying three-point finite difference approximation on \( N \) uniform node points yields

\[
\begin{align*}
\frac{y_{1,i-1} + y_{1,i+1} - 2y_{1,i}}{\Delta z^2} - K_1 y_{1,i}^2 &= 0 \quad \text{for } 1 < i < N, \\
\frac{y_{2,i-1} + y_{2,i+1} - 2y_{2,i}}{\Delta z^2} - K_2 y_{1,i} y_{2,i} &= 0 \quad \text{for } 1 < i < N,
\end{align*}
\]

where the second subscript represents the node number and

\[
\Delta z = \frac{1}{N-1}.
\]

To solve the above problem with GNES, the dependent variables need to be converted to one vector of dependent variables \( x_1...x_N \), i.e.

\[
\begin{align*}
x_{2i-1} &= y_{1,i} \quad \text{for } 1 < i < N, \\
x_{2i} &= y_{2,i} \quad \text{for } 1 < i < N.
\end{align*}
\]

The nonlinear equations expressed in \( x_1...x_N \) are

\[
\begin{align*}
x_1 - 0.21 &= 0, \\
\frac{-3x_2 + 4x_4 - x_6}{\Delta z} &= 0, \\
\frac{x_{2i-2} + x_{2i+2} - 2x_{2i-1}}{\Delta z^2} - K_2 x_{2i-1} x_{2i} &= 0 \quad \text{for } 1 < i < N,
\end{align*}
\]

with boundary conditions

\[
\begin{align*}
x_{2N-5} &= -x_{2N-3} + 3x_{2N-1}, \\
x_{2N} &= 0.127.
\end{align*}
\]
References


