ELSEVIER

Journal of Power Sources 92 (2001) 177–186

JOURNAL OF POWER SOURCES

www.elsevier.com/locate/jpowsour

A procedure for serial simulation of electrochemical processes: cycling of electrodes and batteries

B. Wu, R.E. White[*]

Center for Electrochemical Engineering, Department of Chemical Engineering, University of South Carolina, Columbia, SC 29208, USA

Received 15 May 2000; accepted 5 June 2000

Abstract

Serial simulation is required to predict the behavior of an electrochemical system undergoing many processes. This is demonstrated through the simulation of charge/open-circuit/discharge processes of a thin film nickel hydroxide electrode. The numerical issues involved in this kind of simulation are discussed. An efficient and robust procedure is presented. It can be easily used to achieve the serial simulation of electrochemical processes, e.g., battery cycling, cyclic voltammetry etc. © 2001 Elsevier Science B.V. All rights reserved.

Keywords: Serial simulation; Electrochemical process; Cycling of electrodes; Cycling of batteries

1. Introduction

Modeling of electrochemical systems usually yields large sets of differential and algebraic equations (DAEs) or partial differential and algebraic equations (PDAEs). Since PDAEs can always be converted to DAEs by some approximation methods, e.g., the method of lines (MOL) [1], only DAEs will be discussed here. DAEs can be generally expressed in an implicit form:

$$\boldsymbol{F}(t,\boldsymbol{y},\boldsymbol{y}') = 0 \tag{1}$$

where $\boldsymbol{F},\boldsymbol{y},\boldsymbol{y}' \in R^n$. A system of DAEs is characterized by its index, which is defined as the minimum number of differentiations to convert the DAEs into equivalent ODEs [2–4]. ODEs can be regarded as special DAEs with index equal to 0. DAEs with index higher than 1 are usually difficult to solve and are under active research at present. DAEs with index equal to 1 are normally encountered in the modeling of electrochemical systems. They behave similarly to stiff ODEs, and are generally solved with similar implicit methods. One popular approach to solve index-1 DAEs is the backward differentiation formulae (BDF) method, which has been implemented in many DAEs solvers [2,3].

Simulating a single process of a physical system (i.e., solving one set of ODEs/DAEs) is straightforward. However, an electrochemical system (e.g., a battery) rarely operates for one process only; instead, it usually undergoes several different processes in series. To predict the corresponding behavior, simulation of many processes consecutively is needed, which has been researched in recent years as the simulation of combined continuous/discrete processes [4] and the simulation of hybrid systems [5]. This kind of simulation is demonstrated here for a nickel hydroxide electrode.

Fig. 1 presents a schematic diagram of a thin film nickel hydroxide electrode. Modeling of the electrode behavior yields the following governing equations:

$$\frac{\partial(1-y_1)}{\partial t} = D_{\mathrm{H}^+}\frac{\partial^2(1-y_1)}{\partial x^2} \tag{2}$$

with boundary conditions

$$D_{\mathrm{H}^+}\frac{\partial(1-y_1)}{\partial x}\bigg|_{x=0} = 0 \tag{3a}$$

$$-D_{\mathrm{H}^+}\frac{\rho}{W}\frac{\partial(1-y_1)}{\partial x}\bigg|_{x=l} = \frac{j_1}{F} \tag{3b}$$

and

$$j_1 + j_2 - i_{\mathrm{app}} = 0 \tag{3c}$$

[*] Corresponding author. Tel.: +1-803-777-3270; fax: +1-803-777-8265.
E-mail address: white@engr.sc.edu (R.E. White).

0378-7753/01/$ – see front matter © 2001 Elsevier Science B.V. All rights reserved.
PII: S0378-7753(00)00519-X

Nomenclature

D_{H^+}	proton diffusion coefficient in the nickel active material (cm^2/s)
F	Faraday's constant (96 487 C/eq.)
$\boldsymbol{F}$	differential algebraic equations
$\boldsymbol{F}^{(k)}$	differential algebraic equations for Process k
$\boldsymbol{g}$	equations for discrete events
$\boldsymbol{g}^{(k)}$	equations for discrete events for Process k
i	index for discrete nodes
i_{app}	applied current density on the nickel electrode (A/cm^2)
i_{01}	exchange current density of the nickel reaction (A/cm^2)
i_{02}	exchange current density of the oxygen reaction (A/cm^2)
j_1	current density of the nickel reaction (A/cm^2)
j_2	current density of the oxygen reaction (A/cm^2)
l	thickness of the nickel active material (cm)
N	number of nodes used in the spatial discretization
p_i	interpolation polynomial for dependent variable y_i
R	ideal gas constant (8.3143 J/mol/K)
R^n	n-dimensional vector in real domain
t	independent time variable (s)
$t^{(k)}$	starting time for Process $k+1$ (s)
T	temperature (K)
W	molecular weight of Ni(OH)$_2$ (g/mol)
x	spatial coordinate across the film of the nickel active material (cm)
$\boldsymbol{y}$	dependent variables in DAEs
$\boldsymbol{y}'$	time derivatives of dependent variables in DAEs
y_1	mole fraction of NiOOH
y_2	potential difference at the solid–liquid interface (V)

Greek letters

$\phi_{eq,1}$	equilibrium potential of nickel reaction (V)
$\phi_{eq,2}$	equilibrium potential of oxygen reaction (V)
ρ	density of the nickel active material (g/cm^3)

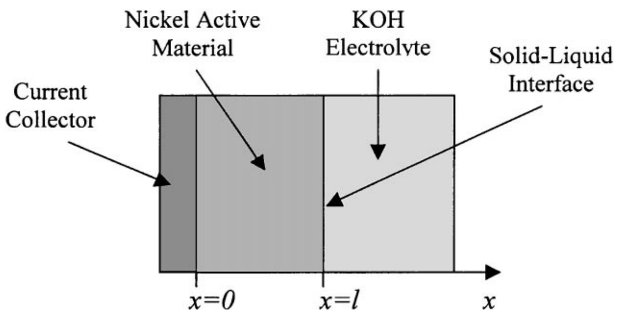

Fig. 1. Schematic of a film nickel hydroxide electrode.

where

$$j_1 = i_{01}\left[2(1-y_1)\exp\left(\frac{0.5F}{RT}(y_2-\phi_{eq,1})\right) - 2y_1\exp\left(-\frac{0.5F}{RT}(y_2-\phi_{eq,1})\right)\right] \quad (4a)$$

$$j_2 = i_{02}\left[\exp\left(\frac{F}{RT}(y_2-\phi_{eq,2})\right) - \exp\left(-\frac{F}{RT}(y_2-\phi_{eq,2})\right)\right] \quad (4b)$$

The initial conditions are

$$y_1(0 \le x \le l, t=0) = 0.05 \quad (5a)$$

$$y_2(t=0) = 0.40\,\text{V} \quad (5b)$$

In the above equations, y_1 is the mole fraction of NiOOH and y_2 is the potential difference at the solid–liquid interface. Parameter values are listed in Table 1. The value of i_{app} varies for different processes, i.e., it has positive values, zero, and negative values for charge, open-circuit, and discharge processes, respectively. To solve the above model equations, the MOL is used first to convert PDAEs to DAEs, i.e., the spatial derivatives in Eqs. (2) and (3) are approximated with three-point finite difference formulae on N uniformly spaced nodes:

$$\frac{\partial(1-y_1[i])}{\partial t} = D_{H^+}\frac{-y_1[i-1]-y_1[i+1]+2y_1[i]}{\Delta x^2} \quad \text{for } 1<i<N \quad (6a)$$

$$-D_{H^+}\frac{3y_1[1]-4y_1[2]-3y_1[3]}{\Delta x} = 0 \quad \text{for } i=0 \quad (6b)$$

$$D_{H^+}\frac{\rho}{W}\frac{-y_1[N-2]+4y_1[N-1]-y_1[N]}{\Delta x} = \frac{j_1}{F} \quad \text{for } i=N \quad (6c)$$

$$j_1 + j_2 - i_{app} = 0 \quad \text{for } i=N \quad (6d)$$

where

$$\Delta x = \frac{l}{N-1} \quad (7)$$

The DAEs in Eq. (6) have N+1 time-dependent variables where N−2 of them ($y_1[i]$, 1<i<N) are differential variables

Table 1
Parameter values in the model of the nickel hydroxide electrode

Parameter	Value
T	298.15 K
$\phi_{eq,1}$	0.420 V
$\phi_{eq,2}$	0.303 V
ρ	3.4 g/cm^3
W	92.7 g/mol
l	1×10^{-4} cm
D_{H^+}	5×10^{-12} cm^2/s
i_{01}	1×10^{-4} A/cm^2
i_{02}	1×10^{-10} A/cm^2

and three of them ($y_1[1]$, $y_1[N]$ and y_2) are algebraic variables.

For the above model, the following processes need to be simulated:

- Process 1: charge at $i_{app}=1\times10^{-4}$ A/cm^2 for 3600 s or to the cutoff voltage 0.60 V;
- Process 2: open-circuit ($i_{app}=0$) for 1800 s or to the cutoff voltage 0.30 V;
- Process 3: discharge at $i_{app}=-1\times10^{-4}$ A/cm^2 for 3600 s or to the cutoff voltage 0.20 V;
- Process 4: open-circuit ($i_{app}=0$) for 1800 s or to the cutoff voltage 0.30 V.

In addition to solving the governing equations given above for the continuous processes, the simulation has to handle the specified termination conditions for each process, which can be expressed as follows:

Process 1:

$$t - t^{(0)} - 3600 = 0 \tag{8a}$$

$$y_2 - 0.6 = 0 \tag{8b}$$

Process 2:

$$t - t^{(1)} - 1800 = 0 \tag{9a}$$

$$y_2 - 0.3 = 0 \tag{9b}$$

Process 3:

$$t - t^{(2)} - 3600 = 0 \tag{10a}$$

$$y_2 - 0.2 = 0 \tag{10b}$$

Process 4:

$$t - t^{(3)} - 1800 = 0 \tag{11a}$$

$$y_2 - 0.3 = 0 \tag{11b}$$

The termination specifications of a continuous process have been called discrete events [4,5]. The equations of the discrete events are independent of the governing equations of continuous processes. In the above example, the discrete events are given by the time specification and also by the cutoff value of potential y_2 for each process. The occurrence of a discrete event terminates the current process and starts the next process.

2. Serial simulation of different processes

To simulate the processes given above in series, a coupled system of DAEs and discrete events must be handled [4], as shown in Fig. 2 and listed below:

$$\boldsymbol{F}^{(1)}(t,\boldsymbol{y},\boldsymbol{y}') = 0, \quad [t^{(0)}, t^{(1)}] \tag{12a}$$

$$\boldsymbol{g}^{(1)}(t,\boldsymbol{y}) = 0, \quad [t^{(0)}, t^{(1)}] \tag{12b}$$

$$\boldsymbol{F}^{(2)}(t,\boldsymbol{y},\boldsymbol{y}') = 0, \quad [t^{(1)}, t^{(2)}] \tag{12c}$$

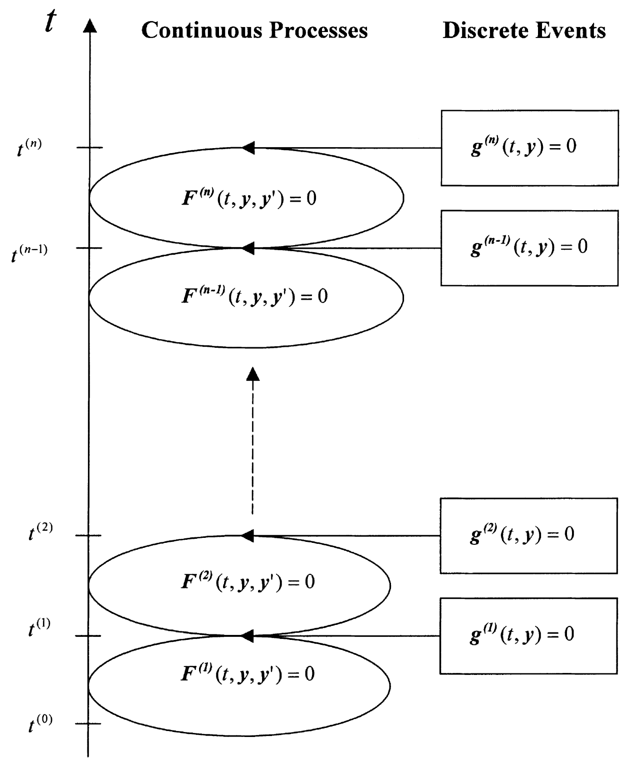

Fig. 2. Schematic of the simulation of a series of processes.

$$\boldsymbol{g}^{(2)}(t,\boldsymbol{y}) = 0, \quad [t^{(1)}, t^{(2)}] \tag{12d}$$

$$\vdots$$

$$\boldsymbol{F}^{(n)}(t,\boldsymbol{y},\boldsymbol{y}') = 0, \quad [t^{(n-1)}, t^{(n)}] \tag{12e}$$

$$\boldsymbol{g}^{(n)}(t,\boldsymbol{y}) = 0, \quad [t^{(n-1)}, t^{(n)}] \tag{12f}$$

Several numerical issues need to be addressed to solve the above system of equations [4]. The first issue is the consistent initialization of a continuous process. For ODEs, the initialization setting is trivial, i.e., specifying the values of dependent variables. However, for DAEs, the freedom of initial settings is less than the number of equations, and specifying a consistent initialization is usually a nontrivial task. If inconsistent initialization values are provided, many DAEs solvers will fail on the first integration step [6,7]. To achieve consistent initialization for DAEs, a widely used ad hoc approach is to take an implicit Euler integration for a very small time step without continuity constraint on the algebraic variables. With this approach, the continuity of differential variables is maintained, while algebraic variables may have jumps in their values, which is undesirable when algebraic variables are more accurately known. In this work, a flexible and robust initialization solver DAEIS [7] has been utilized.

DAEIS is designed for index-1 DAEs with differential and algebraic variables explicitly identified, which is encountered frequently in the modeling of electrochemical systems, and can be expressed as

$$\boldsymbol{F}(t, \boldsymbol{y}_1, \boldsymbol{y}_1', \boldsymbol{y}_2) = 0 \tag{13}$$

where $\boldsymbol{y}_1$ and $\boldsymbol{y}_2$ are vectors of differential and algebraic variables, respectively. Supposing the DAEs in Eq. (13) consist of n equations, p differential variables, and $n-p$ algebraic variables, to determine uniquely all initial values of the dependent variables and time derivatives, p degrees of freedom exist. In DAEIS, the calculation of a consistent set of initializations for Eq. (13) is treated as specifying p variables of $\boldsymbol{y}_1$, $\boldsymbol{y}_2$ and $\boldsymbol{y}_1'$ and solving Eq. (13) for the remaining n variables of $\boldsymbol{y}_1$, $\boldsymbol{y}_2$ and $\boldsymbol{y}_1'$, which is essentially a nonlinear equation solving problem. DAEIS allows the initial values of accurately known variables (including algebraic variables) to remain unchanged and the initial values of other variables to be determined during the initialization process.

Consistent initialization is not only needed for the first process. 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 subsequent process. However, if there is no outside disturbance that changes the differential variables (conservative quantities), the consistent initial values at the beginning of the succeeding process can be determined from the ending state of the previous process by assuming the continuity of the differential variables [4]. This can also be easily handled by DAEIS [7].

The next issue of serial simulation is discrete event detection during the integration of a continuous process. For serial simulation, the ending of each continuous process is due to the occurrence of one of two kinds of discrete events: either explicit or implicit [4], as shown in Fig. 3. An explicit event is specified by the independent time variable only, which is actually the normal working mode of DAEs solvers (i.e., specifying the ending time). However, an implicit event is based on some dependent variables, which can only be determined during the integration process. The detection of discrete events requires special handling in solving DAEs, e.g., checking discrete event equations at the end of each integration step. A sign change of the residual evaluation of a discrete event equation represents the occurrence of a discrete event. Normally, the event location is determined by finding the root of interpolation polynomials [2,4,8], i.e., for the dependent variables, polynomials can be constructed on an interval $[t_{k-1}, t_k]$ based on the past solutions:

$$p_i = f(t, y_i^{t_k}, y_i^{t_{k-1}}, y_i^{t_{k-2}}, \ldots) \tag{14}$$

The polynomial equations are then substituted into the event definition equations (e.g., Eq. (8)). The earliest event can then be located by combining the bisection method (determining the time interval of the event) and Newton's algorithm (finding the accurate time location of the event) [2]. With the determined time location of the event, the values of dependent variables at that moment can be easily obtained from Eq. (14).

Few DAE solvers are capable of discrete event detection. DASRT, an extension of the popular DAEs solver DASSL [2], is used in this work. In DASRT, the continuous DAEs and discrete events are defined in separate subroutines. When an event is detected during the integration process, DASRT will return to the calling program, indicate which event occurred, and provide the state of dependent variables when the event happened.

The last issue of serial simulation is efficiency and robustness of numerical techniques. The modeling of elec-

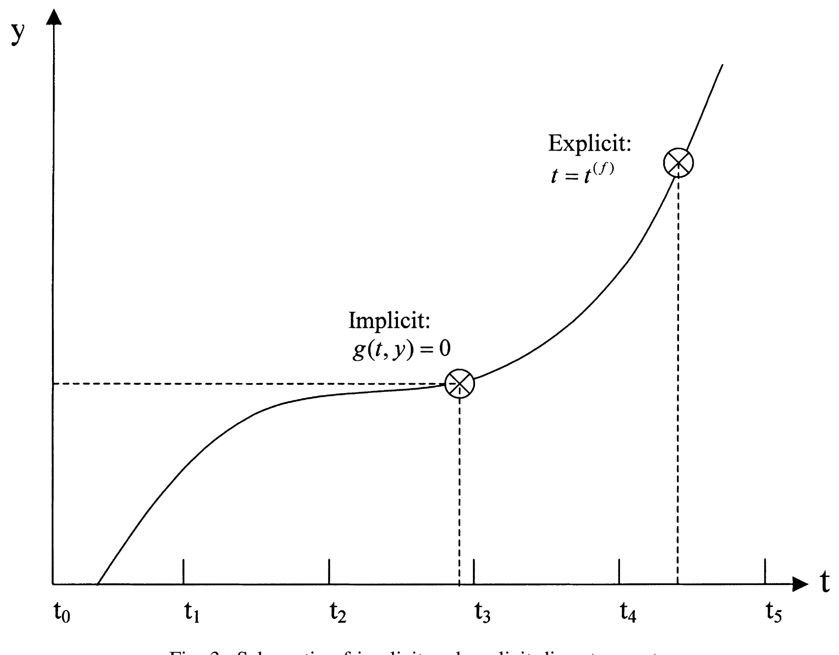

Fig. 3. Schematic of implicit and explicit discrete events.

trochemical systems usually produces hundreds or thousands DAEs (e.g., after spatial discretization of PDAEs). Solving these DAEs normally demands significant computational power. In addition, an electrochemical system may repeat the same operations for many times, e.g., a secondary battery undergoes thousands of cycles of charge/open-circuit/discharge processes. It can take days or weeks to simulate if the solving efficiency is low, which is just too time-consuming to be useful. To improve the simulation efficiency, using fast hardware is a simple choice. However, a more effective approach is using efficient numerical techniques. It is common that a simulation taking hours to finish by one technique (e.g., using an in-house built implicit Euler integration code) only takes minutes by another technique (e.g., using a variable order variable step size solver such as DASSL) with essentially the same results. Besides the efficiency, the robustness of the numerical techniques is another critical factor in the serial simulation. Coding a simulation program for large DAEs is usually a nontrivial task and it is even harder for serial simulations. Debugging the numerical code can be terrible if the robustness of the numerical techniques utilized is poor. Actually, simulation failures due to numerical algorithms or solvers are usually impossible to detect and correct. Therefore, a simple numerical code that lacks sophisticated error controls and diagnoses must be avoided, and high-quality professionally built numerical packages should be used instead. There are many efficient and robust solvers for DAEs, some of them are available free from NETLIB (http://www.netlib.org). The algorithm of the DAEs solver used in this work, DASRT, is based on the BDF method up to the fifth order. At each integration step, DASRT uses the predictor polynomial to predict the solution first and then uses the corrector polynomial to calculate the final results. The low-order BDF method is used at the start of integration with a small integration step size. After enough solution points have been obtained, DASRT will adjust the integration step size and BDF order based on the error estimations. There are many details involved, which can be found in [2].

One widely used ad hoc procedure for the serial simulation is to use a fixed time step integration of DAEs, and to check if some conditions (discrete events) are triggered after each integration step. If an event is detected, the current process is terminated and the final state of the process is used to start the next process. However, it is not a rigorous approach, i.e., numerical errors due to the inaccurate termination of the previous processes will accumulate and cause erroneous simulation results for the later processes. Certainly, if the integration time step is chosen to be sufficiently small, e.g., 10^{-5}, the error due to discrete event detection of this approach will be insignificant, but the simulation efficiency will be greatly affected.

With the consideration of above issues, a procedure for the serial simulation of electrochemical processes is provided as follows:

1. Construct the governing equations for continuous processes. If PDAEs are obtained, the MOL can be used to transform the PDAEs to DAEs, e.g., using finite difference approximations for spatial derivatives. For each process, identify the corresponding discrete events and put them in equation form. The physical constraints at the switching between processes also need to be determined, e.g., which conservation law will apply.
2. Use the initialization subroutine DAEIS to provide the consistent initialization for the DAEs of a continuous process. For the first process, initial values of more accurately known variables are maintained and initial values of other variables are determined to be consistent with those variables. When switching between processes without outside disturbances, values of differential variables need to be maintained and values of algebraic variables are determined to be consistent with differential variables.
3. Apply the DAEs solver with discrete event detection DASRT to solve the DAEs. The continuous processes are defined in one subroutine and the discrete events are specified in another subroutine. When a discrete event is detected by DASRT, the current process is terminated. The state of the previous process at the discrete event is used to provide the starting state for the subsequent process.

Although the above procedure seems to be simple, the complexity of the relevant numerical algorithms is hidden inside the DAEIS and DASRT solvers, e.g., providing a numerically determined Jacobian matrix. In fact, each solver has thousands of lines of carefully constructed code to guarantee the efficiency and robustness of involved algorithms. To use the above procedure effectively, a user only needs to become familiar with the calling protocols of the DAEIS and DASRT solvers. No knowledge of the algorithms used in these two solvers is required. Normally, it takes several hundreds lines of code to solve a medium size model in the combined continuous/discrete domain.

For the given example problem, less than 200 lines of FORTRAN code were used, which is given in Appendix A. The DAEs for continuous processes are given in the subroutine DRES1. The discrete events are defined in the subroutine GR1. The residue form of equations are used in both DRES1 and GR1. The working space, solver parameters, and equation definition subroutines are shared by DAEIS and DASRT. The main subroutine iterates over processes and cycles. It is obvious from the code that the details of solving algorithms are hidden from the users by the two solvers, which makes the usage of the solving procedure much easier. In fact, the source code for the example problem can be easily adapted to handle other similar simulations.

The results of the simulation are shown in Figs. 4–8. It is obvious from Fig. 4 that Processes 2 and 4 are terminated by the explicit discrete events of time specifications, and Processes 1 and 3 are ended by the implicit discrete events of

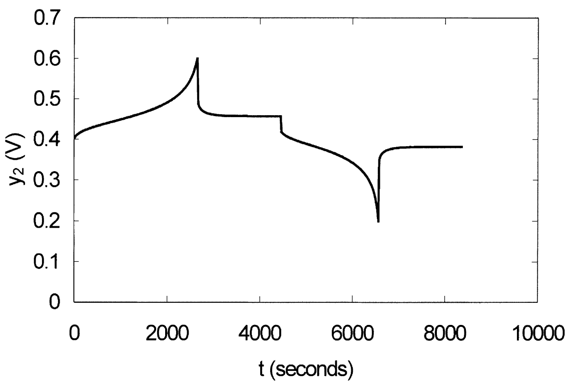

Fig. 4. Simulated potential behavior of the example problem.

cutoff voltage specifications. Due to diffusion limitations, there are nonuniform concentration distributions of NiOOH during each process as shown in Figs. 5–8. Simulation of the cycling like that shown for one cycle of the example processes in Fig. 4 can be done for thousands of cycles.

From the results of the example problem, it is clear that much more information can be obtained from the serial simulation of multiple processes than from the simulation of a single process. The advantage of the serial simulation is also obvious: investigations that are impossible to conduct with the simulation of a single process (e.g., the open-circuit relaxation behavior after a charge process) can be easily achieved. The DAEs resulting from the modeling of electrochemical systems are usually more complex than the given example; however, the same numerical solution procedure applies. In fact, the procedure has been utilized in the simulation of several complex models of battery systems, including a nickel-hydrogen cell model, a nickel-metal hydride cell model, and a lithium-ion cell model, with satisfactory results. One potential application of serial simulation is to investigate failure mechanisms by simulating the long time cycling behavior of an electrochemical system with a model including degradation processes.

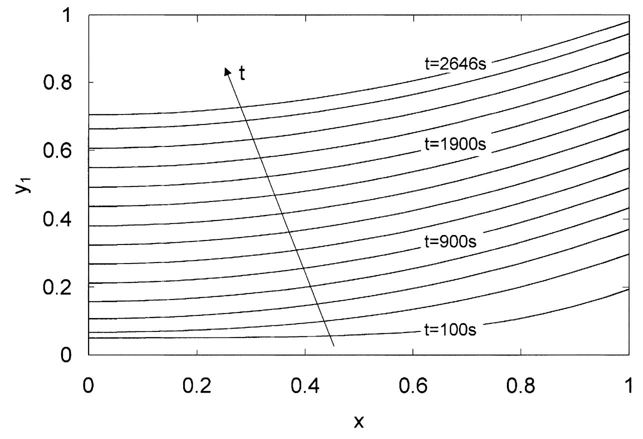

Fig. 5. Simulated concentration profile of the example problem (x is a dimensionless coordinate) for Process 1 (charge process).

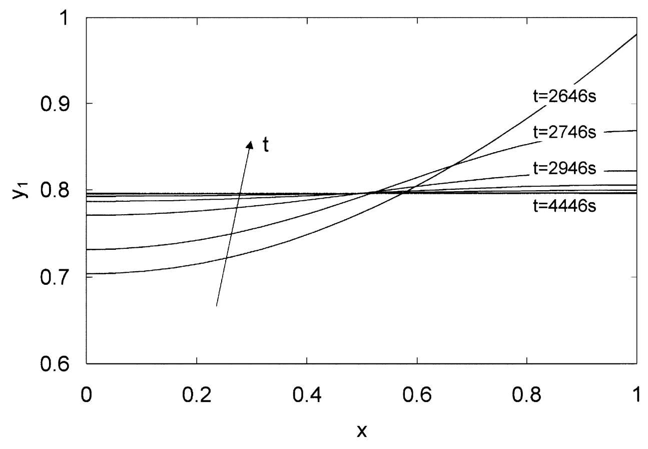

Fig. 6. Simulated concentration profile of the example problem (x is a dimensionless coordinate) for Process 2 (first open-circuit process).

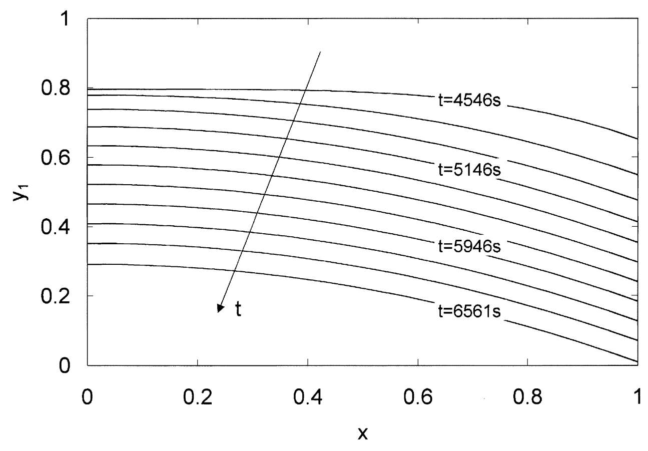

Fig. 7. Simulated concentration profile of the example problem (x is a dimensionless coordinate) for Process 3 (discharge process).

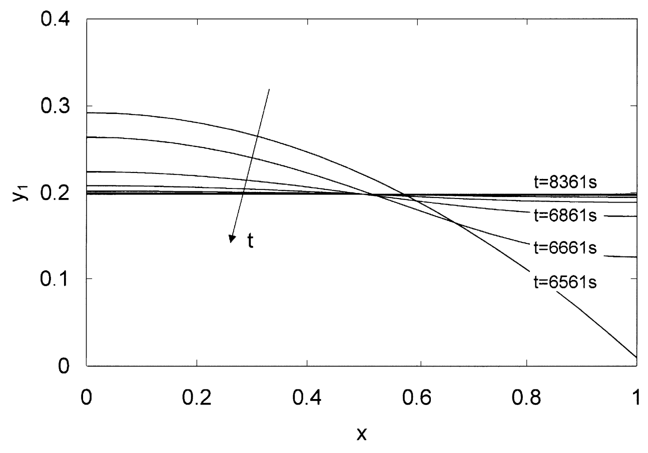

Fig. 8. Simulated concentration profile of the example problem (x is a dimensionless coordinate) for Process 4 (second open-circuit process).

3. Conclusions

An electrochemical system usually operates under different processes consecutively. To simulate the corresponding behavior, serial simulation of many processes is required, which is demonstrated with the simulation of charge/open-circuit/discharge processes of a nickel hydroxide electrode. Three primary numerical issues need to be considered in such a simulation: consistent initialization of a process, accurate termination of a process, and efficiency and robustness of numerical techniques. It is shown that, with two numerical solvers, DAEIS and DASRT, serial simulation can be easily achieved. Compared to the simulation of a single process, the serial simulation of many processes allows more challenging investigations of electrochemical systems.

Appendix A. Source code for solving the sample problem with DAEIS and DASRT

```
C     Serial simulation of a thin film nickel electrode.
      PROGRAM NickelElectrode

      IMPLICIT NONE
      INTEGER NEQMAX, MAXORD, LRW, LIW, MAXPROC
      PARAMETER (NEQMAX=501, MAXORD=5, MAXPROC=100)
      PARAMETER (LRW = 50+(MAXORD+5)*NEQMAX+NEQMAX**2,
     &  LIW = 20+NEQMAX+2*NEQMAX)
      DOUBLE PRECISION T, TOUT, RWORK(LRW), RPAR(100)
      DOUBLE PRECISION Y(NEQMAX), YPRIME(NEQMAX), ATOL, RTOL
      INTEGER IWORK(LIW), IPAR(2), INFO(15), IDID, NEQ, NG, JROOT(2)
      EXTERNAL DRES1, DJAC1, DDAEIS, GR1

      INTEGER I, ICOUNT, NCYCLE, NPROC, NODES, NTOTCYCLE, NTOTPROC
      DOUBLE PRECISION DTOUT, TEMP0, THICKNI, CNIOOH0, XCURR(MAXPROC),
     &  XVCUT(MAXPROC), XTCUT(MAXPROC), RATENI, RATEO2, DIFFH

      RTOL = 1.D-5         ! Relative Tolerance
      ATOL = 1.D-9         ! Absolute Tolerance

      OPEN(UNIT=10,STATUS='unknown',file='op.txt')  ! Input File
      READ(10,*) NODES     ! number of nodes
      READ(10,*) DTOUT     ! time interval of outputs (seconds)
      READ(10,*) TEMP0     ! ambient temperature (C)
      READ(10,*) THICKNI   ! thickness of nickel active material (cm)
      READ(10,*) CNIOOH0   ! initial NiOOH concentration (mole fraction)
      READ(10,*) DIFFH     ! proton diffusion coefficient (limit1 (cm2/s))
      READ(10,*) RATENI    ! exchange current density of nickel reaction (A/cm2)
      READ(10,*) RATEO2    ! exchange current density of oxygen reaction (A/cm2)
      READ(10,*) NTOTCYCLE ! total number of cycles
      READ(10,*) NTOTPROC  ! total number of processes
C      Read operating conditions per process
      DO ICOUNT = 1,NTOTPROC
           READ(10,*) XCURR(ICOUNT),XVCUT(ICOUNT),XTCUT(ICOUNT)
      END DO
      CLOSE(10)

      TEMP0 = 293.15 + TEMP0
      RPAR(51) = 96487.D0      ! constant
      RPAR(52) = 8.3141        ! constant
      RPAR(53) = TEMP0
      RPAR(54) = 3.4D0/92.71D0 ! maximum proton concentration, mol/cm3
      RPAR(55) = DIFFH
      RPAR(56) = RATENI
      RPAR(57) = RATEO2
      RPAR(58) = 0.420         ! equilibrium potential of nickel reaction, V
      RPAR(59) = 0.303         ! equilibrium potential of oxygen reaction, V
      RPAR(60) = THICKNI       ! thickness of nickel active material, cm3

      NEQ = NODES + 1
      IPAR(1) = NODES
      NG = 2

      OPEN(UNIT=12,STATUS='unknown',FILE='pot.txt')  ! Output File

      T = 0.0D0                         ! Initial time.
      DO NCYCLE = 1,NTOTCYCLE
         DO NPROC = 1,NTOTPROC

            DO I = 2, NEQ-2
            IWORK(20+NEQ+I) = 1         ! Y(I) remains initial value
            IWORK(20+2*NEQ+I) = -1      ! YPRIME(I) needs to be determined
            END DO
            IWORK(20+NEQ+1) = -1        ! Y(I) needs to be determined
            IWORK(20+2*NEQ+1) = 0       ! YPRIME(I) does not exist
            IWORK(20+NEQ+NEQ-1) = -1
            IWORK(20+2*NEQ+NEQ-1) = 0
            IWORK(20+NEQ+NEQ) = -1
            IWORK(20+2*NEQ+NEQ) = 0

            RPAR(1) = XCURR(NPROC)
            RPAR(2) = T + XTCUT(NPROC)
            RPAR(3) = XVCUT(NPROC)

            IF (NCYCLE.EQ.1.AND.NPROC.EQ.1) THEN
               DO I = 1, NEQ-1
                  Y(I) = CNIOOH0        ! Initial value.
               END DO
               Y(NEQ) = 0.40D0          ! Initial value.
            END IF
            DO I = 1, NEQ
               YPRIME(I) = -0.01        ! Initial guess values.
            END DO

            DO I = 1,15
               INFO(I) = 0
            END DO

            CALL DDAEIS(DRES1,NEQ,T,Y,YPRIME,INFO,RTOL,ATOL,IDID,
     &         RWORK,LRW,IWORK,LIW,RPAR,IPAR,DJAC1)

 150        CONTINUE
               TOUT = T + DTOUT
               CALL DDASRT(DRES1,NEQ,T,Y,YPRIME,TOUT,INFO,RTOL,ATOL,
     &            IDID,RWORK,LRW,IWORK,LIW,RPAR,IPAR,DJAC1,GR1,NG,JROOT)
               IF (IDID .LT. 0) GO TO 180
               IF (IDID .EQ. 4) GO TO 160
               CALL RESULTS(NEQ,T,Y,YPRIME)
             GOTO 150

 160        CONTINUE
            IF (IDID.EQ.4.AND.JROOT(1).EQ.1) THEN
               CALL RESULTS(NEQ,T,Y,YPRIME)
               WRITE (*,1000)
1000           FORMAT(//1X,' Cut-off time reached',/)
            ELSE IF (IDID.EQ.4.AND.JROOT(2).EQ.1) THEN
               CALL RESULTS(NEQ,T,Y,YPRIME)
               WRITE (*,1010)
1010           FORMAT(//1X,' Cut-off voltage reached',/)
            END IF
        END DO
      END DO

      WRITE (*,2000)
2000  FORMAT(//1X,' Simulation Completed',/)
      CLOSE(12)
      STOP
 180  CONTINUE
      WRITE (*,2010)
2010  FORMAT(//1X,' Simulation Failed',/)
      CLOSE(12)
      STOP
      END

      SUBROUTINE RESULTS(NEQ,T,Y,YPRIME)
C     ================================================
      IMPLICIT NONE
      INTEGER NEQ, I
      DOUBLE PRECISION T,Y(*), YPRIME(*)
        WRITE (*,1020) T,(Y(I),I=1,NEQ)
        WRITE (12,1020) T,(Y(I),I=1,NEQ)
1020    FORMAT(1X,22(E15.5))
      RETURN
      END
      SUBROUTINE DRES1(T,Y,YPRIME,DELTA,IRES,RPAR,IPAR)
C     ==================================================
      IMPLICIT NONE
      INTEGER IRES, IPAR(*), NEQ, I
      DOUBLE PRECISION T, Y(*), YPRIME(*), DELTA(*), RPAR(*)
      DOUBLE PRECISION FJ1, FJ2, DELTAH

      DELTAH = RPAR(60)/FLOAT(IPAR(1)-1)
      NEQ = IPAR(1) + 1
      FJ1 = RPAR(56) * ((2.d0*(1-Y(NEQ-1)))*exp(0.5*
     &  RPAR(51)/RPAR(52)/RPAR(53)*(Y(NEQ)-RPAR(58))) - (2.d0*Y(NEQ-1))
     &  *exp(-0.5*RPAR(51)/RPAR(52)/RPAR(53)*(Y(NEQ)-RPAR(58))))
      FJ2 = RPAR(57) *
     &  (exp(1.0*RPAR(51)/RPAR(52)/RPAR(53)*(Y(NEQ)-RPAR(59)))
     &  -exp(-1.0*RPAR(51)/RPAR(52)/RPAR(53)*(Y(NEQ)-RPAR(59))))
      DELTA(1) = (-3*Y(1)+4*Y(2)-Y(3))*RPAR(54)/DELTAH/2*RPAR(55)

      DO I = 2,NEQ-2
         DELTA(I) = YPRIME(I)
     &       - (Y(I+1)+Y(I-1)-2*Y(I))/DELTAH/DELTAH*RPAR(55)
      END DO

      DELTA(NEQ-1) = (3*Y(NEQ-1)-4*Y(NEQ-2)+Y(NEQ-3))*RPAR(54)
     &   /DELTAH/2*RPAR(55) - FJ1/RPAR(51)

      DELTA(NEQ) = FJ1 + FJ2 - RPAR(1)

      RETURN
      END

      SUBROUTINE GR1 (NEQ, T, Y, NG, GROOT, RPAR, IPAR)
      IMPLICIT NONE
      INTEGER NEQ, NG, IPAR(*)
      DOUBLE PRECISION T, Y(*), GROOT(*), RPAR(*)

      GROOT(1) = T - RPAR(2)
      GROOT(2) = Y(NEQ) - RPAR(3)

      RETURN
      END

      SUBROUTINE DJAC1(T,Y,YPRIME,PD,CJ,RPAR,IPAR)
C     ==========================================
      IMPLICIT NONE
      INTEGER IPAR(*)
      DOUBLE PRECISION T, Y(*),YPRIME(*),PD(*),CJ,RPAR(*)
C       DUMMY
      RETURN
      END
```

The input file ‘op.txt’ for the program is given below:

```
20      ! number of nodes in the nickel active layer
100.0   ! time interval of outputs (seconds)
25.0    ! ambient temperature (C)
1.d-4   ! thickness of nickel active materials (cm)
0.05    ! initial NiOOH concentration(mole fraction)
5.d-12  ! proton diffusion coefficient(cm2/s)
1.d-4   ! exchange current density of nickel reaction (A/cm2)
1.d-10  ! exchange current density of oxygen reaction (A/cm2)
1       ! number of repeating times
4       ! number of processes
1.D-4          0.60              3600.0
0.0            0.30              1800.0
-1.D-4         0.20              3600.0
0.0            0.30              1800.0
current(A)     cut-off Volt(V)   time length (sec)
```

References

[1] W.E. Schiesser, The Numerical Methods of Lines: Integration of Partial Differential Equations, Academic Press, San Diego, CA, 1991.

[2] K.E. Brenan, S.L. Campbell, L.R. Petzold, Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations, North-Holland, New York, 1989.

[3] U.M. Ascher, L.R. Petzold, Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations, SIAM, Philadelphia, PA, 1998.

[4] P.I. Barton, C.C. Pantelides, AIChE 40 (1994) 966.

[5] M.S. Branicky, S.E. Mattsson, Hybrid Systems IV, in: P. Antsaklis, W. Kohn, A. Nerode, S. Sastry (Eds.), Lecture Notes in Computer Science, Vol. 1273, Springer, New York, 1997, p. 31.

[6] A. Kröner, W. Marquardt, E.D. Gilles, Comput. Chem. Eng. 16 (1992) S131.

[7] B. Wu, R.E. White, Comput. Chem. Eng., 1999, submitted for publication.

[8] T. Park, P. Barton, ACM Trans. Modeling Comput. Simulations 6 (1996) 137.