Journal of The Electrochemical Society, **149** (10) A1342-A1348 (2002)
0013-4651/2002/149(10)/A1342/7/$7.00 © The Electrochemical Society, Inc.

Newton-Krylov-Multigrid Algorithms for Battery Simulation

J. Wu,[a] Venkat Srinivasan,[b,*] J. Xu,[a] and C. Y. Wang[b,,z]**

*[a]Department of Mathematics, [b]Electrochemical Engine Center and Department of Mechanical and Nuclear Engineering,
The Pennsylvania State University, University Park, Pennsylvania 16802*

Numerical solutions to partial differential equations form the backbone of mathematical models that simulate the behavior of various electrochemical systems, specifically, batteries and fuel cells. In this paper, we present a set of numerical algorithms applied to efficiently solve this system of equations. These fast algorithms are identified by fully understanding the physics of the problem and recognizing the strength of the coupling between the governing equations. We illustrate this coupling, specifically in the two potential equations, and demonstrate the need for their simultaneous solution using the Newton method. We take a 2D thermal and electrochemical coupled Li-ion model and extend the familiar Band(J) subroutine by utilizing a Krylov iterative solver, a generalized minimal residual subroutine (GMRES), instead of the direct solver (Gauss elimination), to improve the solution efficiency of the large, nonsymmetric Jacobian system. In addition, we use a nonlinear Gauss-Seidel method to provide the initial guess for the Newton iteration, and precondition the GMRES solver with a block Gauss-Seidel and multigrid algorithm with a smoother based on the tridiagonal matrix algorithm. Every stage in this process has been seen to add to the efficiency of the resulting computer simulation with the final result being a substantial improvement in computation speed, namely, simulating complete discharge of the cell in less than 10 min for grid size of 45×32.
© 2002 The Electrochemical Society. [DOI: 10.1149/1.1505635] All rights reserved.

Manuscript submitted November 30, 2001; revised manuscript received April 2, 2002. Available electronically August 21, 2002.

The recent need for alternatives to the previously used combustion engine for transportation has led to the development of electric and hybrid electric vehicles. This need has spurred research into advanced batteries used in these applications, with emphasis on operation under various conditions and on materials and cell construction to enhance cycle-life and performance. This research effort has been undertaken by extensive experimental testing of cells and by using computer simulations based on various techniques, including those that are trained to data (*e.g.*, neural network[1]) and those developed based on the physical and chemical laws of the processes occurring in the cell (first principles[2]). These models are expected to play a critical role in cell design for a specific application, in predicting behavior under various conditions (*e.g.*, dynamic stress test) and in the integration into system models in order to predict the behavior of the whole vehicle. This latter feature adds a new level of complexity into these models, as computational speed becomes an important criterion.

In addition, as the models developed are made more comprehensive with inclusion of the thermal behavior in addition to the electrochemistry,[3] detailed transport and electrochemical mechanisms (*e.g.*, solid phase diffusion),[4,5] and multidimensional effects, significant numerical challenges arise, thereby requiring robust numerical techniques. It is also desirable that the numerical methods developed be both optimal in computational efficiency and scalable to be readily amenable to parallel computing that will become routine in the near future. An algorithm is optimal if the computational complexity is of the order of the number of unknowns, $O(n)$, where n denotes the number of unknowns. Gaussian elimination, for example, requires $O(n^3)$ operations and hence is not an optimal algorithm.[6] However, some multigrid methods feature computational complexities of $O(n)$. When implemented on a parallel computer with multiple processors, such optimal and scalable algorithms would permit solutions of problems in multidimensions with a large number of computational nodes and multiple physicochemical processes, like in fuel cell systems, without dramatically increasing computation time.

The mathematical representation of the various phenomena in a battery results in the generation of a number of coupled, nonlinear partial differential equations (PDEs), that are time and space dependent.[2,7] The solution of these equations for a given set of independent variables, initial and boundary conditions, results in the estimation of the dependent variables. One way to achieve this is to discretize the PDEs using a finite difference or finite volume method and expressing them as a set of nonlinear algebraic equations, which are then solved. The discretization process is common to the various methods used in the electrochemical literature. Subsequently, the algebraic equations are solved either sequentially using an iterative procedure or simultaneously using Newton's method.[2] The former method is attractive due to its simplicity but is inefficient and not robust when the equations are strongly coupled with each other. The latter is the procedure in the Band(J) subroutine where the Jacobain matrix resulting from the Newton's procedure is inverted using LU factorization in each Newton iteration.[2] A similar procedure is used in the standard DASSL subroutine,[8] which has also been applied to battery simulation. However, the direct solution of the Jacobian matrix is computationally inefficient, especially when dealing with two- or three-dimensional problems.[6] This inadequacy has been partly overcome in the DASPK software where the Jacobian matrix is solved iteratively using an incomplete generalized minimal residual (GMRES) subroutine[9] and sometimes coupled with an incomplete LU factorization preconditioner.[6,10] Here the Jacobian matrix is not explicitly required and is evaluated in an approximate manner. However, DASPK is a general-purpose solver used in a wide variety of applications and therefore is not tailored for battery simulation. Hence, considerable improvement in efficiency can still be gained by understanding the unique physics of the battery problem and choosing mathematical algorithms accordingly. For example, a system where the equations are decoupled from each other could be easily solved using a sequentially iterative technique, whereas use of the Newton method would involve undue complexity (evaluating Jacobians) while providing little, if any, improvement in computation speed.

The purpose of this paper is twofold: (*i*) to gain insight into the physics of the battery problem by studying the coupling between the various governing equations, and (*ii*) to develop and apply a set of advanced solution algorithms specifically tailored for strongly coupled equations governing battery behavior. To achieve the former, we use a Newton method code and investigate the effect of decoupling the equations from each other on the efficiency. The latter is achieved by using the Newton's procedure to linearize the nonlinear system, wherein the initial conditions are generated using the nonlinear Gauss-Seidel method. The resulting linear set of equations are then solved using an iterative procedure, the GMRES subroutine, along with block Gauss-Seidel and multigrid preconditioning. This procedure for the solution of the problem, as opposed to

* Electrochemical Society Student Member.
** Electrochemical Society Active Member.
z E-mail: cxw31@psu.edu

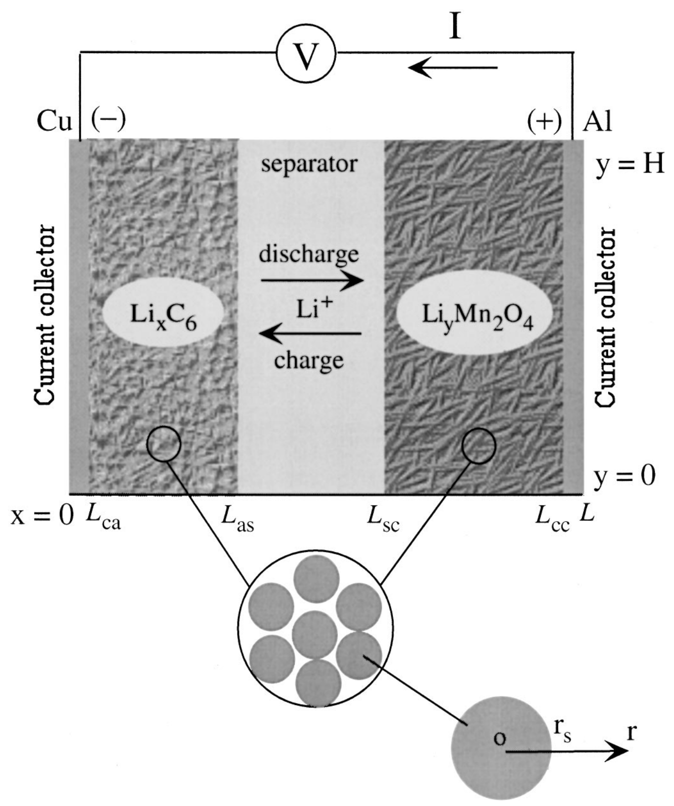

Figure 1. Schematic of the Li-ion cell modeled in this study. The cell consists of a carbon negative and manganese oxide positive electrode with a Li salt electrolyte. The active material is assumed to consist of many spherical particles.

the direct solution used in Band(J), provides considerable improvement in computation speed in this system, as the 2D nature of the problem results in a large sparse Jacobian matrix. The methods outlined here and the insight gained are expected to be useful in the development of efficient algorithms for all electrochemical systems, which have a large computational domain with multidimensional effects, like fuel-cell systems.

Problem Formulation and Model Equations

Figure 1 schematizes the dual lithium ion insertion cell modeled in this study, consisting of a carbon negative electrode, a separator, and a positive manganese dioxide electrode. Both active materials, consisting of a large number of particles, assumed to be spherical in this study, are pasted on conductive grids forming a porous electrode. The whole cell is filled with electrolyte, which is a solution of lithium salt in a nonaqueous solvent. The reactions during charge-discharge in the two electrodes can be represented as

$$Li_{y-x}Mn_2O_4 + xLi^+ + xe^- \underset{\text{charge}}{\overset{\text{discharge}}{\rightleftarrows}} Li_yMn_2O_4 \qquad [1]$$

$$Li_xC_6 \underset{\text{charge}}{\overset{\text{discharge}}{\rightleftarrows}} LiC_6 + xLi^+ + xe^- \qquad [2]$$

with the reaction occurring at the electrode/electrolyte interface and sustained due to lithium diffusion from/into the bulk of the solid phase. The present model is the same as the one described in Ref. 3 and thus is not elaborated here. Instead, we only provide the basic governing equations for species, charge, and energy balance, as details of the values of the parameters and their interdependence (*e.g.*, equilibrium potential as a function of concentration) are largely independent of solution procedure and the insight outlined here. See Ref. 3 for these parameters.

Mass balance.—Using the volume-averaging method to represent the concentration of species, the mass balance in the solution and solid phase can be represented using

$$\frac{\partial(\varepsilon_e c_e^{Li})}{\partial t} = \nabla \cdot (D^{eff}\nabla c_e^{Li}) + \frac{1 - t_+^0}{F} J^{Li} \qquad [3]$$

$$\frac{\partial(\varepsilon_e c_s^{Li})}{\partial t} = -\frac{j^{Li}}{F} \qquad [4]$$

The reaction current j^{Li} is related to the surface overpotential through the Butler-Volmer equation, namely

$$J^{Li} = a i_{oj}\left[\exp\left(\frac{\alpha_a F}{RT}\eta_j\right) - \exp\left(-\frac{\alpha_c F}{RT}\eta_j\right)\right] \qquad [5]$$

for both the electrodes, with the parameters being different in each. The exchange current density is a function of concentration of lithium in both the electrolyte and the solid phase and is given by

$$i_{oj} = k(c_e^{Li})^{\alpha_a}(c_{s\,max}^{Li} - \bar{c}_{se}^{Li})^{\alpha_a}(\bar{c}_{se}^{Li})^{\alpha_c} \qquad [6]$$

where k is determined from the initial exchange current density and species concentration. In addition, the overpotential is related to the surface concentration through the equilibrium potential

$$\eta_j = \phi_s - \phi_e - U_j(\bar{c}_{se}^{Li}) \qquad [7]$$

The area-averaged concentration of lithium at the electrode/electrolyte interface is determined by

$$\frac{D_s}{l_{se}}(\bar{c}_{se}^{Li} - c_s^{Li}) = -\frac{j^{Li}}{aF} \qquad [8]$$

Charge balance.—Charge conservation is expressed through Ohm's law in the matrix and a modified Ohm's law in solution, respectively, given by

$$\nabla \cdot (\sigma^{eff}\nabla\phi_s) - j^{Li} = 0 \qquad [9]$$

$$\nabla \cdot (\kappa^{eff}\nabla\phi_e) + \nabla \cdot (\kappa_D^{eff}\nabla \ln c_e^{Li}) + j^{Li} = 0 \qquad [10]$$

where the diffusion conductivity, κ_D^{eff}, is given by

$$\kappa_D^{eff} = -\frac{2RT\kappa^{\varepsilon ff}}{F}(1 - t_+^0)\left(1 + \frac{d \ln f_\pm}{d \ln c_e}\right) \qquad [11]$$

Energy balance.—The energy balance is developed based on the local heat generation method. Here, in each control volume, the energy equation is expressed as

$$\frac{\partial(\rho C_p T)}{\partial t} = \nabla \cdot (\lambda\nabla T) + q \qquad [12]$$

where the heat generation rate is expressed as

$$q = j^{Li}\left(\phi_s - \phi_e - U_j + T\frac{\partial U_j}{\partial T}\right) + \sigma^{eff}\nabla\phi_s \cdot \nabla\phi_s + \kappa^{eff}\nabla\phi_e \cdot \nabla\phi_e + \kappa_D^{eff}\nabla \ln c_e^{Li} \cdot \nabla\phi_e \qquad [13]$$

While the first term represents heat effects due to electrode reactions, the other terms represent Joule heating.

Initial/boundary conditions.—Uniform initial concentrations are assumed everywhere, leading to

$$c_e^{Li} = c_{e,0}^{Li} \quad c_s^{Li} = c_{s,0}^{Li} \quad T = T_o \qquad [14]$$

The computational domain is confined between two current collectors with internal boundaries being represented by continuity of the various phases. As no reaction occurs at the current collector surfaces, the boundary conditions are expressed as

$$\frac{\partial c_e^{\mathrm{Li}}}{\partial n} = 0 \quad \frac{\partial c_s^{\mathrm{Li}}}{\partial n} = 0 \quad \frac{\partial \phi_e}{\partial n} = 0 \qquad [15]$$

Current is applied at the top of the cell (through the tabs) and heat is dissipated either through the tabs or from the sides as well. These are expressed as
At $y = H$

$$-\sigma^{\mathrm{eff}} \frac{\partial \phi_s}{\partial y} = I \qquad [16a]$$

$$-\lambda \frac{\partial T}{\partial y} = h(T - T_\infty) \quad \text{if } x < L_{\mathrm{ca}} \text{ or } x > L_{\mathrm{cc}} \qquad [16b]$$

At the other boundaries

$$\frac{\partial \phi_s}{\partial n} = 0 \qquad [17a]$$

$$-\lambda \frac{\partial T}{\partial n} = h(T - T_\infty) \qquad [17b]$$

Discretization

The transient terms are discretized by a fully implicit scheme making use of the backward Euler method. For example, Eq. 4 is expressed as

$$(\varepsilon_s c_s^{\mathrm{Li}})_p - (\varepsilon_s c_s^{\mathrm{Li}})_p^0 = -\Delta t \frac{j^{\mathrm{Li}}}{F} \qquad [18]$$

where the subscript p denotes the nodal point where the discretization is performed and the term with the superscript 0 represents the value at the previous time step.

In order to discretize the spatial terms, we choose the finite volume formulation as detailed by Patankar.[11] Here the computational domain is divided into a number of control volumes with each control volume surrounding a grid point. Each differential equation is integrated over the control volume and the resulting first derivatives expressed using a two-point difference, assuming a linear profile, to yield an algebraic equation, which depends on the values of the dependent variables at the nodal points adjacent to the point under consideration. As the methodology involves integrating the governing equation over the computational volume, conservation of the quantities is always ensured, even when a small number of nodal points are used. In contrast, equations obtained using finite-difference and finite-element methods are truly conserved only when the number of grid points is infinitely large. In addition, as interfaces are chosen at the faces of the control volume, continuity at the interfaces is ensured automatically.

Newton's Method

The resulting set of coupled algebraic equations are solved using the Newton's method at each time step. As explored later in the paper, the equations for the potential in the matrix and the solution are strongly coupled to each other, necessitating the use of this algorithm. Let us represent the set of algebraic equations in vector notation as

$$f(s) = 0 \qquad [19]$$

where $f\colon \mathbf{R}^{4n} \to \mathbf{R}^{4n}$, where n is the number of vertices in the grid. Let

$$J(s) = \frac{\partial f}{\partial s} \qquad [20]$$

be the $4n \times 4n$ Jacobian matrix of the system $f(s)$.

Table I. The maximum norm of the residual with iteration number during the Newton iteration showing the speed of convergence of the Newton's method.

m	$\|\eta^m\|_\infty$
1	1.0495364×10^{-2}
2	3.4196594×10^{-4}
3	7.4618763×10^{-7}
4	$7.2593409 \times 10^{-10}$
5	$5.1082980 \times 10^{-13}$
6	$3.3319922 \times 10^{-16}$
7	$2.1239212 \times 10^{-19}$

Using a previous iteration value, s^m, we first solve the linear system

$$J(s^m)\eta^m = -f(s^m) \qquad [21]$$

to evaluate η^m, from which the value of s in the next iteration is calculated by

$$s^{m+1} = s^m + \eta^m \qquad [22]$$

In other words, Eq. 21 defines the direction vector $\eta \in \mathbf{R}^{4n}$, called the Newton direction, which is followed in Eq. 22 by a unit step in this direction in order to move from the current point s^m to the next point s^{m+1}. The convergence of Newton method is locally quadratic[12] and has been found to be particularly fast for the present model problem. Table I lists the maximum norm of η^m at each time step and shows that the solution to Eq. 19 can be obtained in a few iterations using this method.

A critical step in the Newton iteration is the choice of the initial guess. As choosing an initial guess at the beginning of the simulation for the global problem is difficult, we make use of the nonlinear Gauss-Seidel method[13] to solve the two potential equations (Eq. 9 and 10), which is then used as the initial guess for the Newton iteration. After the first time step, the previous converged solution is seen to be sufficient for providing adequate initial guess for convergence of the solution at the current time step. However, it is expected that in more complex profiles, like current-interrupt and pulse operation, where the solution at the previous time step is considerably different from the converged solution, the nonlinear Gauss-Seidel initial guess solver would be crucially needed at each time step.

Note that j^{Li} is an implicit function of c_e^{Li}, c_s^{Li}, ϕ_e, and ϕ_s, which can be seen by substituting Eq. 6 into 5, where η_j, $\overline{c_{se}^{\mathrm{Li}}}$, and c_s^{Li} are all functions of j^{Li}. Therefore, the accuracy with which j^{Li} is calculated in each Newton iteration would have a significant effect on the convergence rate. In order to increase the accuracy of this calculation, the Newton method is also used to obtain the value of j^{Li} at each vertex, which is a (local) one-dimensional problem. Since both α_a and α_c are less than 1, there are singular points for the function expressed in Eq. 6 (*i.e.*, the Jacobian derivative has negative exponents in the variable, $\overline{c_{se}^{\mathrm{Li}}}$). If the solution is near the singular point, then Newton's method is inefficient even with a good initial guess. For example, solving the simple equation $g(x) = \sqrt{x} - 0.1 = 0$ by Newton's method, starting from an initial guess $x_0 = 0.1$, results in the value going less than zero in the next iteration, defined by $x_1 = x_0 - g(x_0)/g'(x_0)$, where the prime denotes differentials. Keeping the iterations in the definition domain is an important factor for the efficiency and robustness of Newton iteration. This difficulty is circumvented by choosing the initial guess close to the singular point.

GMRES Method

The use of the Newton method, as detailed above, results in the conversion of the nonlinear algebraic equations into a linear system, with a large and sparse Jacobian matrix. Solution of such a system using direct solvers such as Gaussian elimination is inefficient. For example, the computation work in Band(J) is $O(J^2n)$, where n is the number of unknowns and J the bandwidth of the matrix.[6] For a 2D problem, such as the one in this paper, J is large (for example, $J \approx m = \sqrt{n}$ for an $m \times m$ grid), suggesting that the banded direct-solver would be inefficient. On the other hand, iterative techniques provide considerable improvement in performance. For example, a preconditioned GMRES method would result in the computation being $O[n\log(n)]$; a significant improvement for a large system.

However, because the matrix in Eq. 21 is not symmetric, traditional iterative methods like conjugate gradient (CG) and preconditioned conjugate gradient (PCG) cannot be applied. Hence, we choose GMRES, as introduced by Saad and Schultz,[14] for solving this large, sparse, linear system of equations. While CG applies to symmetric matrices, GMRES is a Krylov subspace approximation method for general matrices. For simplicity, we rewrite Eq. 21 as

$$J\eta = b \qquad [23]$$

Given an initial guess, η_0, we set $r_o = b - J\eta$, and define the Krylov subspace

$$K(J, k, r_o) = \text{span}(r_o, Jr_o, \ldots, J^{k-1}r_o) \qquad [24]$$

the new approximation η^{k+1} satisfies

$$\|b - J\eta_{k+1}\|_2 = \min_{\nu \in K(J,k,r_0)} \|b - J(\eta_o + \nu)\|_2 \qquad [25]$$

This method of applying the GMRES every j steps, using the latest iteration as the guess for the next GMRES cycle, termed GMRES(j), is popular, as good performance is achieved with low storage requirements.[6,14]

Preconditioners for the GMRES Method

The efficiency of the GMRES method depends on many factors and most notably the condition number of the matrix J (defined by $\|J\|\|J^{-1}\|$ for a matrix norm $\|\ \|$). If J is close to identity, then the convergence of GMRES will be very fast. Therefore, a preconditioner, M, is used and the following equivalent system is solved

$$M^{-1}J\eta = M^{-1}b \qquad [26]$$

using the GMRES method. A good preconditioner possesses the following properties: (*i*) for any vector μ, $M^{-1}\mu$ is obtained easily; and (*ii*) $M^{-1}J$, in some sense, is close to identity. Among the numerous preconditioning techniques, we choose the block Gauss-Seidel and multigrid techniques in this study.

Block Gauss-Seidel method.—Write $J = D - L - U$ where D is a block diagonal matrix, and L and U are the strictly lower and upper block triangular parts of $(J - D)$, respectively. If we assume that $M = D - L$, then M is also called the block Gauss-Seidel preconditioner. The efficiency of the block Gauss-Seidel preconditioner varies depending on the arrangement of the unknowns. If we arrange the unknowns such that the nonzero parts and the relatively large elements are predominantly in the lower block triangular part, then $M^{-1}J$ will be close to identity. This is ensured by manipulating the Jacobian matrix and expressing it as

Table II. Effect of preconditioning on the efficiency of the GMRES method. A block Gauss-Seidel and multigrid preconditioning was used.

Preconditioned GMRES		GMRES without preconditioner	
Iteration	Error	Iteration	Error
0	0.1000000×10^{1}	0	0.1000000×10^{1}
1	0.4536254×10^{-1}	2	0.3444273×10^{0}
2	0.3136039×10^{-1}	4	0.1956817×10^{0}
3	0.2530226×10^{-2}	8	0.9661916×10^{-1}
4	0.1815118×10^{-3}	16	0.5184772×10^{-1}
5	0.4202251×10^{-5}	32	0.2799835×10^{-1}
6	0.9042787×10^{-7}	55	0.1548874×10^{-1}
7	0.2034770×10^{-8}	110	0.8121148×10^{-2}

$$J = \begin{pmatrix} J_{c_ec_e} & J_{c_e\phi_s} & J_{c_e\phi_e} & J_{c_eT} \\ J_{\phi_sc_e} & J_{\phi_s\phi_s} & J_{\phi_s\phi_e} & J_{\phi_sT} \\ J_{\phi_ec_e} & J_{\phi_e\phi_s} & J_{\phi_e\phi_e} & J_{\phi_eT} \\ J_{Tc_e} & J_{T\phi_s} & J_{T\phi_e} & J_{TT} \end{pmatrix} \qquad [27]$$

For a given vector $p = (p_1, p_2, p_3, p_4) \in R^{4n}$

$$\xi_1 = J^{-1}_{c_ec_e}p_1 \qquad [28a]$$

$$\xi_2 = J^{-1}_{\phi_s\phi_s}(p_2 - J\phi_sc_e\xi_1) \qquad [28b]$$

$$\xi_3 = J^{-1}_{\phi_e\phi_e}(p_3 - J_{\phi_ec_e}\xi_1 - J_{\phi_e\phi_s}\xi_2) \qquad [28c]$$

$$\xi_4 = J^{-1}_{TT}(p_4 - J_{Tc_e}\xi_1 - J_{T\phi_s}\xi_2 - J_{T\phi_e}\xi_3) \qquad [28d]$$

Then $\xi = (\xi_1, \xi_2, \xi_3, \xi_4) = (D - L)^{-1}p$ is the desired vector.

We arrange the unknowns in the order of c_e, ϕ_s, ϕ_e, and T, expecting $(D - L)$ to be close to J, then $(D - L)$ is a good preconditioner. Table II shows the typical error estimates by our preconditioned GMRES method compared to a GMRES method without preconditioner, where both the speed and the efficiency of the preconditioning are clear.

Multigrid method.—In order to perform the above-described procedure, we need to obtain $J^{-1}_{c_ec_e}\nu$ and similar terms for a given vector ν. This is equivalent to solving the system

$$J_{c_ec_e}w = \nu \qquad [29]$$

We note that the matrices $J_{c_ec_e}$, $J_{\phi_s\phi_s}$, $J_{\phi_e\phi_e}$, J_{TT} are symmetric and positive definite. Here we solve this system using the multigrid method, although any iteration method would suffice [*e.g.*, the tridiagonal matrix algorithm (TDMA), PCG].[15] The multigrid method has a very fast convergence speed which is independent of the grid size, and the number of whole arithmetic operations needed is only $O(n)$ [or $O(n \log n)$, more rigorously], where n is the number of the unknowns. This technique has been particularly useful in solving symmetric positive definite problems arising from discretizing elliptic or parabolic partial differential equations.[10,16-18] In addition, we choose TDMA, an efficient solver for problems that are essentially one-dimensional, as the smoother in each level.[15] While the multigrid method is more efficient than simple TDMA, especially for the truly 2D problem, it is comparable for a nearly 1D problem. So it is relatively easy to get the vector $x = (D - L)^{-1}y$ for any given vector **y**. As the multigrid method is applied in the preconditioner, it is not necessary to get $J^{-1}_{e_ee_e}\nu$ and similar terms very accurately for a given vector $\nu \in R^n$. In our numerical experiments on the model problem only one or two iterations were needed to maintain the same convergence speed of the GMRES method. These suggest that

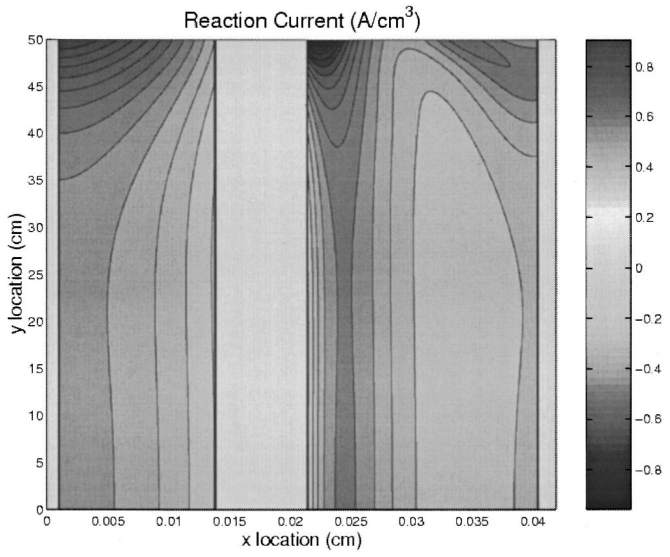

Figure 2. Reaction current contours across the cell indicating the 2D nature of the problem. The figure was generated at 842s during a 3 C discharge.

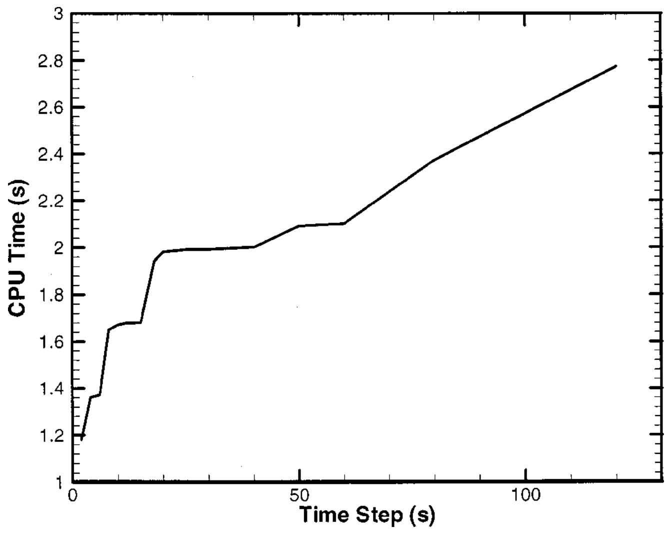

Figure 3. CPU time *vs.* time step for the Li-ion code described here for a 3 C discharge with 45 × 32 grid.

our preconditioned GMRES method is very efficient. Notice that we use the GMRES method to get the Newton direction in the Newton method; therefore, there is no need to use the GMRES method to drive the residual to a very low degree. Experience suggests that decreasing the residual from 1 to 10^{-3} to 10^{-6} is sufficient.

In summary, the method presented in this paper involves discretizing the PDEs using the finite volume method in order to express them as a set of nonlinear algebraic equations, which are then solved simultaneously. This is achieved by first linearizing the equations using Newton's method, with a nonlinear Gauss-Seidel method to obtain the initial guess, after which the linear system is solved using the GMRES method, with a block Gauss-Seidel and multigrid preconditioning coupled with TDMA as a smoother in each level. The procedure results in very efficient solution of the coupled thermal-Li-ion code which involves gradients in two dimensions, as seen from Fig. 2, where the reaction current distribution across the cell is shown after 842 s during a 3 C discharge. The significant change in the reaction current both across the cell and with cell height suggests the need for efficient algorithms. The efficiency of the present algorithm is seen from the convergence central processing unit (CPU) times shown in Table III, where a 3 C discharge is simulated using a time step of 2 s. Table III also shows that as the number of vertices increases, the CPU time increases linearly, *i.e.*, $O(n)$, as is theoretically expected.[10,16-18] This points to the optimal nature of the method, which along with its scalable nature makes it particularly amenable for parallel computing.

In addition to providing efficient convergence in the order of minutes for reasonable grid sizes, the method is also seen to converge at large time steps, where the convergence speeds are larger. This is seen clearly in Fig. 3, where the CPU time, during one Newton iteration, is shown with the time step enlarged. The simulations were performed for different time steps for a 3 C discharge using a 45 × 32 grid. One can see that the CPU time increases much more slowly than the time step, suggesting that the time step can be increased even further, thereby improving convergence speed, as long as accuracy is maintained.

In order to study the effect of increasing the time step on the accuracy, simulations were conducted by increasing the time step from 2 s (where the solution time is approximately 10 min) to 120 s (where the solution time is approximately 1 min), and the cell voltage and temperature are plotted in Fig. 4 and 5, respectively. The curves were generated during 3 C discharge using a grid size of 45 × 32. The negligible difference between the curves generated at the two time steps assert to the usefulness of utilizing this methodology for battery simulation. The 10-fold increase in the convergence speed makes this an attractive feature of the presently developed methodology.

Partial Newton's Method

The numerical procedures illustrated entail generating the Jacobians of the matrix during the Newton iteration, which was performed analytically in this study. This task can, however, be extremely time consuming under some conditions. For example, the

Table III. CPU time with grid size for the Li-ion code generated during a 3 C discharge with a time step of 2 s.

2D grid size	CPU time (min)
45 × 45	10
90 × 62	45
178 × 122	190
354 × 242	800

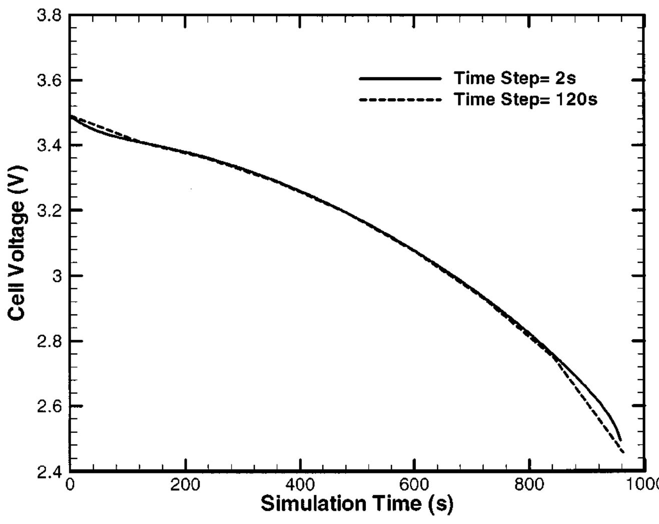

Figure 4. Cell potential during 3 C discharge for simulation conducted using a time step of 2 and 120 s with a grid size of 45 × 32.

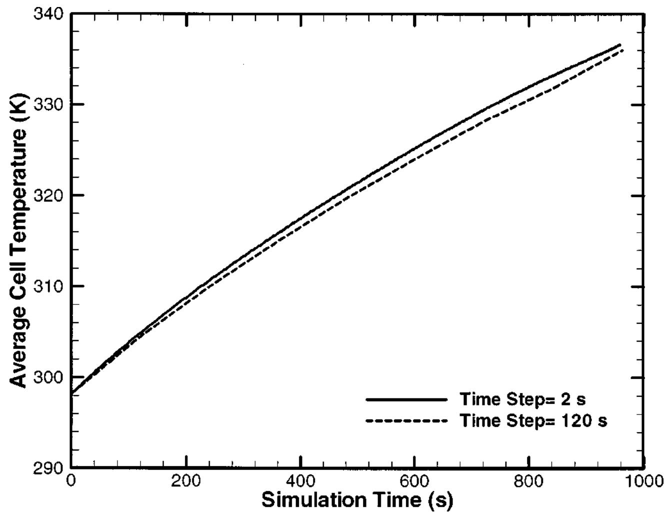

Figure 5. Cell temperature evolution during 3 C discharge of the cell using time steps of 2 and 120 s with a grid size of 45 × 32.

first two terms in Eq. 10 are difficult to differentiate as the conductivity of the solution changes with concentration. A similar problem is encountered when differentiating the diffusion term in Eq. 3, as the diffusion coefficient is dependent on the concentration, and hence changes across the cell. It would, therefore, be advantageous to simplify the process by neglecting these differentials and utilizing the resultant inexact Jacobian in the Newton iteration. While this has the disadvantage of changing the iteration path with a potential for slowing it down, it simplifies the solution of the Jacobians considerably because of the larger number of zeros in the matrix. Two cases are thus devised and investigated, namely

Case A: The differentiation of the term $\nabla \cdot (\kappa_D^{\text{eff}} \nabla \ln\ c_e^{\text{Li}})$ in Eq. 8 is not included in the Jacobian.

Case B: In addition, the gradient of κ^{eff} in the term $\nabla \cdot (\kappa^{\text{eff}} \nabla \phi_e)$ in Eq. 8 and the gradient of D^{eff} in the term $\nabla \cdot (D^{\text{eff}} \nabla c_e^{\text{Li}})$ in Eq. 3 are not included in the Jacobian.

Figure 6 plots the CPU time *vs.* the simulation time during discharge, obtained from the full Newton method and the two cases illustrated, respectively. The similarity for the three cases is clearly seen. Detailed analysis shows that both the decrease in performance due to the inexact Jacobian and the increase due to the increased number of zeros have a negligible effect, leading to the overall negligible effect shown in Fig. 6. Therefore, these terms can be neglected when generating the Jacobian matrix for battery simulation, thereby making this time-consuming process considerably easier.

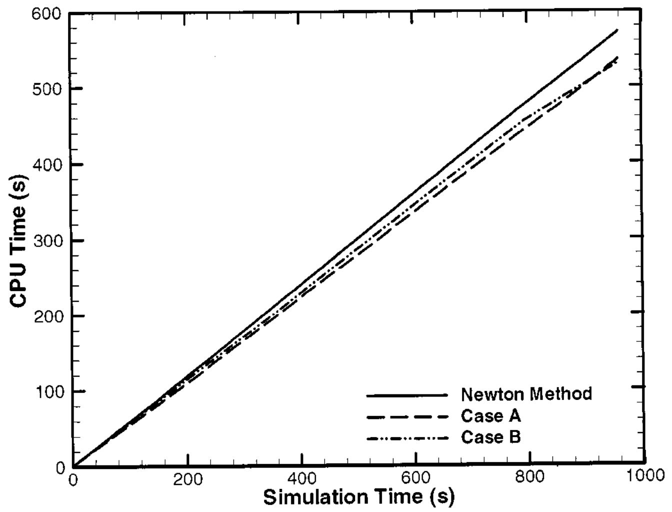

Figure 6. CPU time *vs.* simulation time for the Li-ion code using the complete and the partial Jacobians. The plot was generated during a 3 C discharge using a grid size of 45 × 32.

Coupling of Governing Equations

While use of advanced solution algorithms are essential to efficiently solve a system of equations, a compromise between complexity and efficiency can be achieved only by understanding the physics of the system. In other words, knowledge of the nature of the equations would provide us with guidelines for the choice of the solver. One important criterion is the coupling between the various equations solved, namely, the concentration (Eq. 3), potential (Eq. 9 and 10), and temperature (Eq. 12) equations. If the equations were considerably decoupled from each other, the iterative procedure (like Picard's iteration) would suffice, while completely coupled equations would need to be solved simultaneously (using the Newton method). While the former offers simplicity (*e.g.*, no need for Jacobians), its efficiency decreases as the coupling between the equations becomes more significant. The latter, while efficient, would impose undue complexity when the system is simple. In order to gauge the level of the coupling between the various equations in battery modeling, four test simulations were conducted, namely

Case 1: All the equations are coupled (full Newton method).

Case 2: The temperature equation is separated, while the others are solved simultaneously.

Case 3: The concentration equation is also separated, along with the temperature equation.

Case 4: All four are separated (equivalent of Picard's method).

Figure 7 shows the CPU time during the simulation for three of the cases. It can be seen that while case 2 and 3 result in a slight increase in the CPU time, the change is negligible, suggesting that the temperature and concentration equations are reasonably decoupled from the potential equations. When the two potential equations are also decoupled (case 4) the simulation time changes dramatically with as much as 75 times decrease in performance. Clearly the two potential equations are the major cause for the inefficiency of sequential iteration techniques and require simultaneous solution. These two equations are dependent on each other through the reaction current, where the overpotential appears in the exponential term. Hence, small changes in the potential can lead to large changes in the current, especially when the exchange current density is large (*e.g.*, the Zn electrode in alkaline cells). Hence, the possibility of the solution changing significantly during each iteration step is considerable, hence making it more difficult. This coupling of the two potential equations with the relatively decoupled nature of the temperature and concentration equations has been seen in other battery systems also. It appears that an effective way of taking advantage of the ease of iteration techniques while combining the robustness of simultaneous solvers is to solve the potential equations simultaneously while solving the rest sequentially. This technique offers the most potential for the solution of systems that have similar physicochemical behavior as that shown in this paper.

Conclusion

In this paper, we illustrate a set of numerical techniques that have been found to be most efficient to solve the equations that describe battery systems. We illustrate these techniques using a 2D coupled thermal electrochemical first-principles model of a Li-ion cell. The methods developed are based on Newton linearization of nonlinear algebraic equations along with an iterative solver, GMRES, instead of a direct solver (Gauss elimination) used in Band(J) algorithm. This is expected to be critical in such large sparse systems where direct solvers are very inefficient. The Newton iteration method is also made more robust by using a nonlinear Gauss-Seidel method to provide a good initial guess. In addition, we explore the robustness of the GMRES subroutine by using the block Gauss-Seidel and multigrid preconditions, where significant improvements in performance

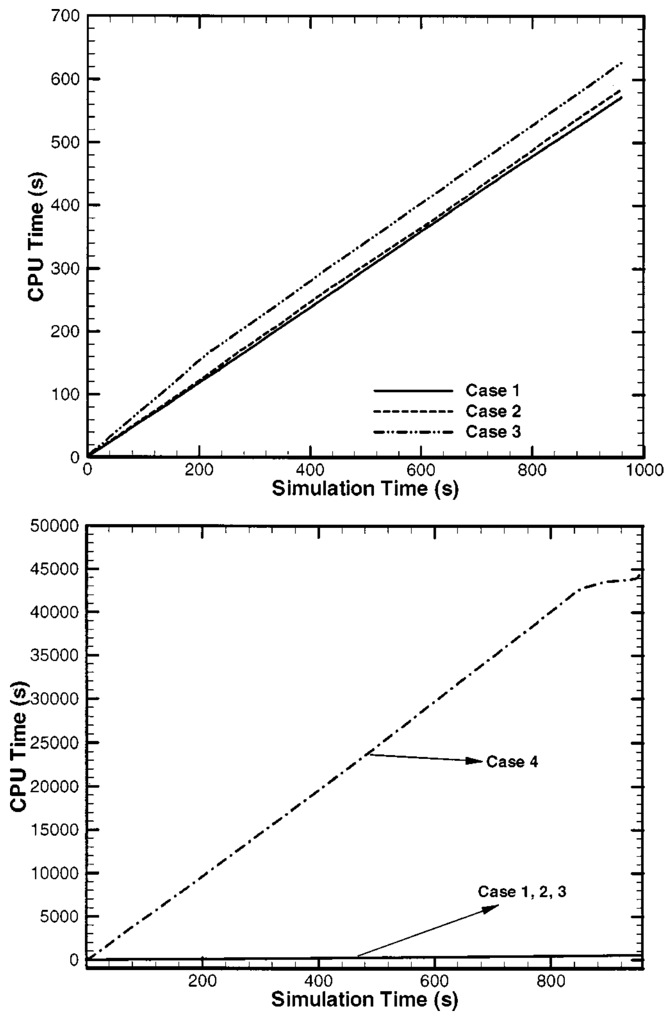

Figure 7. CPU time *vs.* simulation time for the Li-ion code with the various equations coupled with each other. While the top frame shows cases 1-3, the bottom shows all four where the considerably larger time when all the equations are decoupled is clearly seen.

were seen. In addition to identifying the set of algorithms that are most efficient for the solution of such systems (batteries and fuel cells), we also provide insight into the physics of the equations by studying their coupling. It is seen that the two potential equations are strongly coupled with each other and have to be solved simultaneously. On the other hand, the concentration and temperature equations are relatively decoupled and can be solved separately. It is proposed that an optimum algorithm for such systems, which combines robustness with ease of usage, would be to solve the potential equations simultaneously and couple this with the rest in a sequential iteration procedure.

Acknowledgments

The authors are grateful to Dr. Wen-Bin Gu (GM Global Propulsion) for developing the equations that are used in this manuscript. The work was supported, in part, by DARPA Tactical Technology Office, Electric and Hybrid Vehicle Technology Program, under cooperative agreement no. MDA 972-95-3-0019, and by DOT Advanced Vehicle Program under cooperative agreement no. DTRS56-99-T-0016.

The Pennsylvania State University assisted in meeting the publication costs of this article.

List of Symbols

a	interfacial area per unit volume, cm^2/cm^3
c_s^{Li}	concentration of lithium in the solid phase, mol/cm^3
$c_{s\,max}^{Li}$	maximum concentration in solid phase, M
$\bar{c}_{se}^{Li}$	concentration of lithium at the electrode/electrolyte interface, M
c_e^{Li}	concentration of lithium in the electrolyte, mol/cm^3
C_p	specific heat capacity, J/kg K
D^{eff}	effective diffusion coefficient in the liquid phase, cm^2/s
D_s	diffusion coefficient in the solid phase, cm^2/s
$f_\pm$	mean molar activity coefficient of the electrolyte, mol/cm^3
F	Faraday's constant, C/equiv
h	equivalent convective heat-transfer coefficient, W/cm^2 K
i_{0j}	exchange current density of an electrode reaction j, A/cm^2
I	input current density, A/cm^2
j^{Li}	pore wall flux of lithium ion, mol/cm^2 s
l_{se}	diffusion length (R_s/5 for spherical particles), cm
q	volumetric heat generation rate, J/cm^3 s
R	universal gas constant, J/mol K
R_s	radius of the spherical particle, cm
t_+^0	transference number of the Li$^+$ with respect to the solvent velocity
t	time, s
T	absolute temperature of the cell system, K
U_j	open-circuit potential data for the insertion materials, V
x	coordinate along the cell width, cm
y	coordinate along the cell height, cm

Greek

α_a, α_c	transfer coefficients
ε_e	porosity
η_j	local surface overpotential of reaction j, V
κ^{eff}	effective conductivity in the liquid phase, Ω^{-1} cm^{-1}
κ_D^{eff}	diffusion conductivity, Ω^{-1} cm^{-1}
λ	thermal conductivity, W/cm K
ρ	density, g/cm^3
σ^{eff}	effective conductivity in the matrix phase, Ω^{-1} cm^{-1}
ϕ_s	matrix phase potential, V
ϕ_e	solution phase potential, V

Subscript

0	initial condition

References

1. C. O'Gorman, D. Ingersoll, R. Jungst, and T. Paez, in *Selected Battery Topics*, G. Halpert, M. L. Gopikanth, K. M. Abraham, W. R. Cieslak, and W. A. Adams, Editors, PV 98-15, p. 248, The Electrochemical Society Proceedings Series, Pennington, NJ (1998).
2. J. Newman, *Electrochemical Systems*, 2nd ed., Prentice-Hall, Inc., Englewood Cliffs, NJ (1973).
3. W. B. Gu and C. Y. Wang, in *Proceedings of the Symposium on Lithium Batteries*, S. Surampudi, R. A. Marsh, Z. Ogumi, and J. Prakash, Editors, PV 99-25, p. 748, The Electrochemical Society Proceedings Series, Pennington, NJ (2000).
4. M. Doyle, J. Newman, A. S. Gozdz, C. N. Schmutz, and J.-M. Tarascon, *J. Electrochem. Soc.*, **143**, 1890 (1996).
5. T. F. Fuller, M. Doyle, and J. Newman, *J. Electrochem. Soc.*, **141**, 1 (1994).
6. A. Greenbaum, *Iterative Methods for Solving Linear Systems, Frontiers in Applied Mathematics*, SIAM, Philadelphia, PA (1997).
7. J. Newman, in *The Fundamental Principles of Current Distribution and Mass Transport in Electrochemical Cells, Electroanalytical Chemistry*, A. J. Bard, Editor, Marcel Dekker, Inc., New York (1973).
8. L. R. Petzold, in *Scientific Computing*, R. S. Stepelman and M. Carver, Editors, North Holland Publishing Company, Amsterdam (1983).
9. P. N. Brown, A. C. Hindmarsh, and L. R. Petzold, *SIAM J. Sci. Comput. (USA)*, **15**, 1467 (1994).
10. W. Hackbusch, *Multigrid Methods and Applications*, Springer-Verlag, Berlin (1985).
11. S. V. Patankar, *Numerical Heat Transfer and Fluid Flow*, Hemisphere, Washington, DC (1980).
12. S. C. Chapra and R. P. Canale, *Numerical Methods for Engineers*, 3rd ed., McGraw-Hill, New York (1998).
13. X.-C. Tai and J. Xu, *Math. Comput.*, Submitted.
14. Y. Saad and M. H. Schlutz, *SIAM (Soc. Ind. Appl. Math.) J. Sci. Stat. Comput.*, **7**, 865 (1986).
15. L. H. Thomas, *Elliptical Problems in Linear Difference Equations over a Network*, Watson Science Computer Laboratory Report, Columbia University, New York (1949).
16. V. U. Trottenberg, *Multigrid Methods: Fundamental Algorithms, Model Problem Analysis and Applications, Multigrid Methods*, Springer, Cologne (1981).
17. J. Xu, *SIAM Rev.*, **34**, 581 (1992).
18. J. Xu, in *Wavelets, Multilevel Methods and Elliptic PDEs*, Oxford Science Publications, New York (1997).