SAND86-8893
Unlimited Release
Printed September 1986

A DIFFERENTIAL/ALGEBRAIC EQUATION FORMULATION OF THE METHOD-OF-LINES SOLUTION TO SYSTEMS OF PARTIAL DIFFERENTIAL EQUATIONS *

Robert J. Kee
Computational Mechanics Division
Sandia National Laboratories
Livermore, CA 94550

and

Linda R. Petzold
Computing and Mathematics Research Division
Lawrence Livermore National Laboratory
Livermore, CA 94550

Abstract

The method-of-lines is a computational approach for solving partial differential equations (PDE) where some of the derivatives are discretized, and the resulting lower dimensional system is solved, usually with existing software. The approach is most often used to solve parabolic PDEs where the spatial derivatives are discretized (for example, by finite difference approximations) and the resulting initial value ordinary differential equation (ODE) system is solved using software for initial value problems. This paper shows that considering the discretized problem as a system of differential/algebraic equations (DAEs) rather than ODEs accommodates more complex systems of equations and often provides a more convenient user interface than the traditional method-of-lines approach.

* This work supported by the United States Department of Energy, Scientific Computing Staff.

A DIFFERENTIAL/ALGEBRAIC EQUATION FORMULATION OF THE METHOD-OF-LINES SOLUTION TO SYSTEMS OF PARTIAL DIFFERENTIAL EQUATIONS

INTRODUCTION

Numerical methods for solving partial differential equations (PDE) usually involve the replacement of all derivatives by discrete difference approximations. The method-of-lines (MOL) does this also, but in a special way that takes advantage of existing software. For parabolic PDEs the typical case is to discretize the spatial derivatives (for example, by finite differences), and thus convert the PDE system into an ordinary differential equation (ODE) initial value problem. The ODE system can then be solved using standard software such as LSODE.[1] The success and popularity of this method stem from the availability of quality software for initial value problems.

There are two important advantages to the MOL approach. First, it is computationally efficient. The ODE software takes on the burden of the time discretization and of choosing the time steps in a way that maintains accuracy and stability in the evolving solution. Moreover, most production ODE software is written to be both robust and computationally efficient. The software is extensively tested, and thus reliable. The second advantage is one of human efficiency. The person using the MOL only has to be concerned with discretizing spatial derivatives, thus reducing significantly the work required to write a computer program to solve a given problem.

Until recently, all production ODE software has been written to solve initial value problems in a "standard form," $y' = f(t, y)$. Here, y denotes the vector of dependent variables, t denotes time, and f is any function of t and y. The "prime" denotes a time derivative. This form is perfectly adequate for a large class of problems, including many MOL problems. However, it is restrictive, and many well-posed problems of practical interest are not easily written in the standard form.

The differential/algebraic equation (DAE) form $F(t, y, y') = 0$ is much more general than the standard form, and thus contains a larger class of problems. We find significant advantages for some problems to posing the MOL as a system of DAEs rather than as a standard ODE initial value problem. The recent development of software, such as DASSL,[2] to solve DAEs has made this alternative formulation feasible and attractive.

In the following section we describe DAEs in general terms and discuss briefly the numerical methods used to solve them. Then we show by two examples how the MOL can take advantage of the DAE form.

DIFFERENTIAL/ALGEBRAIC EQUATIONS

Implicit systems of DAEs are written in the form

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

where F, y, and y' are N-dimensional vectors and t is time (the independent variable). Solution of the system requires initial conditions

$$y(t_0) = y_0, \quad y'(t_0) = y'_0 \tag{2}$$

which satisfy Eq. (1) at t_0. Many systems of physical interest are most naturally posed in this form.

There are several advantages to solving a system directly in the DAE form rather than trying to rewrite it as a standard-form ODE. First, it is simpler to leave the problem in its original form rather than to try to rewrite it. This is an important consideration especially for large systems or for systems of equations which are generated automatically by simulation packages. Second, even where it is theoretically possible to solve for y' and convert to a standard-form ODE, it may not be efficient to do so. For example, converting $Ay' = By$ to standard form requires multiplying by A^{-1}. For large systems arising in method-of-lines applications, A is usually a sparse matrix, whereas A^{-1} is usually dense, so that multiplying by A^{-1} destroys the natural sparsity of the system. If F depends nonlinearly on y', then converting Eq. (1) to a standard-form ODE requires solving a nonlinear equation on each time step for y', which is again inefficient. Third, for many systems of physical interest, $\partial F/\partial y'$ is singular, and it may be quite complicated or impractical to solve for y' in these cases.

Implicit ODEs: There are three important special cases of the general DAE form. Implicit ODEs are DAEs where the matrix $\partial F/\partial y'$ is everywhere nonsingular near the solution. These systems are really ODEs "in disguise", but for reasons of convenience or efficiency as described above might be best left in the DAE form. There is usually little or no difficulty in solving an implicit ODE with a code written for DAEs. This is because the problem is really an ODE, and numerical methods for ODEs are, in comparison to DAEs, very well understood and very well developed.

Semi-Explicit DAEs: The second important subset of DAEs is the class of semi-explicit DAEs. These are posed as ODE systems coupled to nonlinear algebraic constraints.

$$\begin{aligned} y' &= f(t, y, z) \\ 0 &= g(t, y, z), \end{aligned} \tag{3}$$

where y and z are vectors of dependent variables. These problems are not implicit ODEs because the algebraic constraints cause the matrix $\partial F/\partial y'$ in Eq. (1) to be singular. One way to think about these problems is that they are in a sense "more singular" than implicit ODEs. In fact, we can go a step further and distinguish several different degrees of singularity for semi-explicit DAE systems. This is a useful thing to do because it helps us to understand the underlying structure of the systems and also to identify classes of highly singular systems with which DAE codes might have trouble. It is useful to define a term called the *index*,[3] which can be thought of as a measure of the degree of singularity of the DAE. An ODE (or an implicit ODE) has index 0, or the least amount of singularity. For the simplest case of semi-explicit DAEs, we have that $\partial g/\partial z$ is nonsingular. Thus for these problems it is at least theoretically possible to solve the algebraic equation for z as a function of y and t and substitute that expression in place of z in the ODE, to obtain a standard-form ODE. The index of this type of problem is one. If $\partial g/\partial z$ is singular, then the index is larger than one. Another way to find the index, which is sometimes useful for converting a higher-index problem into a lower-index one, is to differentiate the algebraic constraints with respect to time, and substitute for any derivatives from the ODE. Each differentiation of the algebraic constraints reduces the index by one, so that the number of times required to differentiate the algebraic constraint to obtain an ODE is the index of the system. Numerical methods tend to have difficulties with systems where the index is greater than one, although at least for semi-explicit DAEs, they are not always insurmountable. Several references[3–6] provide further discussion of higher-index systems. For the purposes of this paper, we restrict our attention to index-one systems as they arise in a method-of-lines context.

Fully Implicit DAEs: The final class of problems is that of fully implicit DAEs formulated as in Eq. (1). For these fully implicit systems, it is possible in theory to decide whether the index is one or higher by linearizing the problem, transforming the linearized problem to semi-explicit form and then examining $\partial g/\partial z$. However this is a difficult and time-consuming operation. Because we have found that most method-of-lines applications give rise to index-one systems, we recommend instead to try one of the codes described below for DAEs, and only if it does not work to start thinking about whether the problem has a high index.

Differential/algebraic equations are closely related to stiff systems of standard-form ordinary differential equations. For example the semi-explicit index-one DAE system is closely related to the stiff system $y' = f(t, y, z)$ coupled with $\epsilon z' = g(t, y, z)$, where $\partial g/\partial z$ is nonsingular and has eigenvalues with negative real parts, and ϵ is a small number. As $\epsilon \rightarrow 0$, the solution of this stiff system tends to the solution of the semi-explicit DAE. Thus in some sense we can think of DAEs as "infinitely stiff" ODEs.

Numerical Methods: The original observation that DAEs could be solved by numerical methods for stiff ODEs was as far as we know due to Gear,[7] and Gear and Brown[8] wrote an early code for this purpose. The main production codes for DAEs today are LSODI[9] due to Hindmarsh and Painter, which solves linearly implicit systems of the form $A(t, y)y' = f(t, y)$, and DASSL[2] written by Petzold, which solves fully implicit DAE systems $F(t, y, y') = 0$. Both of these codes are based on backward differentiation formulas (BDF), and are designed mainly for index-one DAEs. There is theory [4] to support the use of backward differentiation formulas on fully implicit index-one DAEs. Deuflhard et al.[10,11] have developed a code called LIMEX that is based on extrapolation of the semi-implicit backward Euler method. There is a good theoretical foundation for the methods used in this code for semi-explicit DAEs, but as yet there is little supporting theory for the fully implicit case.

In our work we use the DASSL code. The DASSL user interface is similar to most ODE software in that it requires the user to provide a subroutine that defines the system of equations to be solved, that is, Eq. (1). In DASSL this subroutine takes as its input the time T and the vectors Y and YPRIME, and produces as output the vector DELTA, where DELTA=F(T,Y,YPRIME) are the residuals of the DAE (the amount by which the function F fails to be zero for the input values T, Y, and YPRIME). DASSL's job is to estimate iteratively the values of y and y', and the user-supplied subroutine tells it how well the estimates are satisfying the equations.

The underlying ideas about how the DAE methods work are relatively easy to visualize for the backward Euler method, which is the simplest case of the BDF methods and also the foundation of the extrapolation method for DAEs. Several references[1,2,10,11] provide more details about how the DAE methods and codes work; here we only illustrate the general ideas using the backward Euler method. After time discretization, Eq. (1) becomes:

$$F\left(t, y^{n+1}, \frac{y^{n+1} - y^n}{\Delta t}\right) = 0. \tag{4}$$

To advance from time level n to time level $n+1$, Eq. (4) represents a system of nonlinear algebraic equations for y^{n+1}. The Solution is accomplished via a modified Newton iteration. That is, first the Jacobian matrix of the system is evaluated,

$$\mathbf{J} = \frac{\partial F}{\partial y^{n+1}}, \tag{5}$$

and then used to iteratively solve for $y^{n+1}_{(k+1)}$

$$J(y^{n+1}_{(k+1)} - y^{n+1}_{(k)}) = -F\left(t, y^{n+1}_{(k)}, \frac{y^{n+1}_{(k)} - y^n}{\Delta t}\right), \tag{6}$$

where (k) is the iteration index. The initial iterate $(k = 0)$ comes from a predictor step, which is usually computed as a polynomial extrapolant based on the solution at previous time steps. The iteration continues until it converges, i.e., until the correction computed for the next iterate is sufficiently small. For the sake of computational efficiency, the Jacobian is only evaluated periodically rather than at each step and at each iteration, as would be done in a full Newton's method. The rate of convergence is monitored, and when it is sufficiently slow, or failing to converge, a new Jacobian is computed. If the iteration is still too slow, then the time step is reduced. Once the iteration has converged for a given step, the local truncation error is estimated. Based on the error estimate, the code determines whether or not the time step Δt was too large. If it was, the step is rejected, the time step reduced, and the whole procedure repeated. If the errors are much smaller than required, then the time step for the next step is increased.

A problem that is encountered in practical application, especially in MOL applications, is that of obtaining consistent initial conditions. When a system has algebraic constraints, i.e., equations without time derivatives, it is essential that the initial conditions satisfy the constraints. This is unlike the situation for standard-form ODEs where any set of initial conditions is acceptable. It makes sense that the constraints should be satisfied at the initial conditions, but we find that a common error committed by those unaccustomed to using the MOL in a DAE form is failing to ensure that the initial conditions are consistent. The result is that the code is unable to satisfy its error test on the very first step, and thus it can never proceed.

EXAMPLES USING THE METHOD-OF-LINES IN DAE FORM

There are many applications where the DAE formulation of the method-of-lines leads to effective numerical solution of physical problems. However, because DAE software has only recently become available, there are still relatively few practitioners of the method. Some examples of the approach can be found in applications relating to boundary layer flow in chemical vapor deposition reactors,[12–15] combustion ignition,[16,17] convective flow in the earth's mantle,[18] gas transfer in piping networks,[19], and adaptive meshing methods.[20] Here, we use two examples to explain how the discretization of PDEs often leads to systems of equations in the DAE format and why the DAE approach has important advantages for effective numerical solution.

The Heat Equation

We first use a very simple problem, the heat equation, to introduce the MOL ideas and some basic nomenclature.

$$\frac{\partial y}{\partial t} = \frac{\partial^2 y}{\partial x^2} \,. \tag{7}$$

Recall that the general form for DAEs is $F(t, y, y') = 0$. In the case of PDEs, we should perhaps explicitly include the spatial derivatives in the DAE form and write $F(t, x, y, y_x, y_{xx}, y') = 0$. For the heat equation this yields

$$F(t, x, y, y_x, y_{xx}, y') = -\frac{\partial^2 y}{\partial x^2} + y' = 0. \tag{8}$$

Discretizing the spatial derivatives by finite differences on a uniform mesh yields

$$F_j = -\left(\frac{y_{j-1} - 2y_j + y_{j+1}}{\Delta x^2}\right) + y_j' = 0, \tag{9}$$

where j is the mesh index and Δx is the mesh spacing. The boundary conditions are handled in a similar way. For example, at $x = 0$,

$$F_1 = y_1 - y(t,0) = 0 \tag{10}$$

specifies the boundary value to be y_1. The software to solve DAEs is designed in such a way that the user writes a subroutine which, given the y and y' vectors returns the residual vector F.

The DASSL code solves the DAEs with variable order, variable step backward differentiation formula (BDF) methods. However, to give an idea of the equations the code solves at each timestep consider how the equations would look if a backward Euler method were used. Equation (9) would become

$$F_j^{n+1} = -\left(\frac{y_{j-1}^{n+1} - 2y_j^{n+1} + y_{j+1}^{n+1}}{\Delta x^2}\right) + \frac{y_j^{n+1} - y_j^n}{\Delta t} = 0, \tag{11}$$

where n is the time index. This particular system of equations is linear, but in general the equations are nonlinear and are solved by some variant of Newton's method.

Of course, the spatially discretized heat equation can be posed as a standard system of ODEs by incorporating the spatial boundary conditions into the differential equations. This traditional approach removes the algebraic equation $F_1 = 0$ and redefines F_2 as

$$F_2 = \frac{dy_2}{dt} - \frac{y(t,0) - 2y_2 + y_3}{\Delta x^2} \,. \tag{12}$$

However, in many situations, including the following example, the algebraic equations cannot be discarded so easily; the DAE formalism and DASSL are important to solving such problems efficiently.

Combustion Ignition

This example comes from our work on the ignition behavior of combustible gases.[16,17] The specific problem is one in which a quiescent uniform gas mixture is contained in a closed cylindrical vessel. At the start of the problem an external source (such as a laser) deposits energy near the vessel's center for a prescribed period of time, after which the source is no longer applied. Depending on the composition of the mixture and the strength of the ignition source, a flame may propagate from the source and burn all the gas. The objective of the simulation is to help understand the ignition process itself and to determine minimum ignition energies for certain gas mixtures. The purpose of this example, however, is to demonstrate how a DAE formulation facilitates the modeling process.

The one-dimensional conservation equations in cylindrical coordinates are the starting point.
Continuity:

$$\frac{\partial}{\partial t}(\rho r) + \frac{\partial}{\partial r}(\rho u r) = 0 \tag{13}$$

Momentum:

$$\rho\frac{\partial u}{\partial t} + \rho u\frac{\partial u}{\partial r} + \frac{\partial p}{\partial r} - \frac{\partial}{\partial r}\left(2\mu\frac{\partial u}{\partial r}\right) - \frac{2\mu}{r}\left(\frac{\partial u}{\partial r} - \frac{u}{r}\right) = 0 \tag{14}$$

Thermal Energy:

$$\rho c_p\frac{\partial T}{\partial t} + \rho u c_p\frac{\partial T}{\partial r} - \frac{1}{r}\frac{\partial}{\partial r}\left(r\lambda\frac{\partial T}{\partial r}\right) - \frac{\partial p}{\partial t} - u\frac{\partial p}{\partial r} + \sum_{k=1}^{K}\rho Y_k V_k c_{pk}\frac{\partial T}{\partial r} + \sum_{k=1}^{K}\dot{\omega}_k h_h W_k = 0 \tag{15}$$

Species Transport:

$$\rho\frac{\partial Y_k}{\partial t} + \rho u\frac{\partial Y_k}{\partial r} + \frac{1}{r}\frac{\partial}{\partial r}(\rho r Y_k V_k) - \dot{\omega}_k W_k = 0 \tag{16}$$

Equation of State:

$$p - \frac{\rho R T}{\overline{W}} = 0 \tag{17}$$

In these equations t is time and r is the spatial coordinate; T is the temperature; Y_k is the mass fraction of the kth species; p is the pressure; u is the velocity of the fluid mixture; ρ is the mass density; W_k is the molecular weight of the kth species; $\overline{W}$ is the mean molecular weight of the mixture; R is the universal gas constant; λ is the thermal conductivity of the mixture; μ is the viscosity of the mixture; c_p is the constant pressure heat capacity of the mixture; c_{pk} is the constant pressure heat capacity of the kth species;

$\dot{\omega}_k$ is the molar production rate by chemical reaction of the kth species per unit volume; h_k is the specific enthalpy of the kth species; and V_k is the diffusion velocity of the kth species, which depends on the species concentration gradients.

The equations could be solved in the form stated above and they often are.[17] However, some modifications lead to a system that is easier to solve and that illustrates some important points about DAEs. First, we make a Lagrangian coordinate transformation, which satisfies the continuity equation exactly and which effectively eliminates the convective terms from the transport equations. The Lagrangian coordinate ψ is defined by

$$\left(\frac{\partial \psi}{\partial r}\right)_t = \rho r \tag{18}$$

$$\left(\frac{\partial \psi}{\partial t}\right)_r = -\rho u r \tag{19}$$

The transformation from (t, r) to (t, ψ) is therefore given by

$$\left(\frac{\partial}{\partial r}\right)_t = \rho r \frac{\partial}{\partial \psi} \tag{20}$$

$$\left(\frac{\partial}{\partial t}\right)_r = -\rho u r \frac{\partial}{\partial \psi} + \frac{\partial}{\partial t} \tag{21}$$

substituting Eqs. (18) and (19) into Eq. (13) easily verifies that this transformation satisfies the continuity equation identically.

For problems in which the ignition source is weak, it is reasonable to assume that the sound speed is high and the pressure disturbances are of such low amplitude that the pressure can be taken as spatially uniform, but time varying. This assumption leads to replacing the momentum equation by the uniform pressure condition, i.e.,

$$\frac{\partial p}{\partial r} = 0 \tag{22}$$

Finally, for the purposes of the discussion here, we will drop the species conservation equation. The species equations add nothing further to the discussion of the DAE formulation of the MOL, and dropping them now avoids complicating the discussion by introducing the chemical kinetics nomenclature. Thus the following discussion is for a single-component nonreacting gas.

The reduced system of equations in Lagrangian coordinates is summarized as:
Physical coordinate:

$$0 = \frac{\partial r}{\partial \psi} - \frac{1}{\rho r} \tag{23}$$

Momentum:

$$0 = \frac{\partial p}{\partial \psi} \tag{24}$$

Energy:

$$\frac{\partial T}{\partial t} - \frac{1}{\rho c_p}\frac{\partial p}{\partial t} = \frac{1}{c_p}\frac{\partial}{\partial \psi}\left(\rho r^2 \lambda \frac{\partial T}{\partial \psi}\right) \tag{25}$$

State:

$$0 = p - \frac{\rho R T}{\overline{W}} \tag{26}$$

The new independent variable is ψ and it ranges from zero at the vessel center to ψ_R, the total mass in the vessel, at the outer wall, $r = R$. That is,

$$\psi_R = \int_0^R \rho r dr, \tag{27}$$

where ρ is the initial density. Since no mass can enter or leave the vessel, ψ_R must remain unchanged.

These equations have three dependent variables, r, p, and T. We choose not to consider ρ as a dependent variable because $\rho = p\overline{W}/RT$ can be easily substituted for density everywhere it appears. There are thus three equations and three dependent variables. The order of the system is four; the energy equation having second-derivative terms, and pressure and physical-coordinate equations each involving only first-derivative terms. Therefore, four boundary conditions are required. Independent initial conditions are specified on pressure and temperature, and consistent initial conditions are computed for the radius.

The Equations as DAEs: We now call attention to some facts that make the method-of-lines solution of these equations by DAE software attractive. By contrast, these same facts make solution by standard ODE software troublesome. These points regard both the form of the equations and the specification of the initial and boundary conditions. Looking at the equations themselves, one sees that the energy equation (25) has two time derivatives while Eqs. (23) and (24) have no time derivatives at all. The traditional MOL requires that each of the spatially discretized equations have one time-derivative of one dependent variable. Thus, solution via standard ODE software for initial value problems is impractical, while this distribution of time derivatives is routine in a DAE setting.

Initial Conditions: At the initial condition, the pressure and temperature must be specified and, if desired, the temperature can have a spatial profile. Since it is a dependent variable, an initial condition is also required for the radius r. However, unlike in ODEs, the initial condition for r can not be specified arbitrarily. Once p and T are specified, r must be uniquely determined to satisfy Eq. (23), i.e.,

$$r = \int_0^{\psi_R} \frac{1}{\rho r} d\psi. \tag{28}$$

Moreover, evaluation of the initial r profile from Eq. (28) must be done in a manner that is consistent with the eventual differencing of Eq. (23). The numerical methods would likely fail if the initial r is evaluated from Eq. (28) with a high-order quadrature when Eq. (23) is differenced with a two-point forward difference formula. In other words, it may be more important to make the initial conditions consistent than to make them accurate. In any case there is no need to evaluate r more accurately at the initial condition than it is evaluated in the subsequent solution.

Boundary Conditions: The boundary conditions in physical terms are that no mass leaves the vessel, and the vessel walls are held at a fixed temperature T_w. At the vessel center, $\psi = 0$, the symmetry condition leads to the following boundary conditions,

$$r = 0 \quad \text{and} \quad \frac{\partial T}{\partial \psi} = 0. \tag{29}$$

At the vessel boundary, $\psi = \psi_R$ the boundary conditions are

$$r = R \quad \text{and} \quad T = T_w. \tag{30}$$

These boundary conditions have an unusual aspect that makes the DAE formulation especially appropriate, and would cause difficulty for a standard-form MOL approach. Notice that there is no explicit boundary condition on pressure, and there are two conditions on the radius. The problem is perfectly well-posed in that the two conditions on r are sufficient to satisfy the requirements of the two first-order equations. As expected, the two conditions on temperature are sufficient to satisfy the energy equation's requirements.

There are three unknowns at each boundary (r, p, and T) yet we have only provided two boundary conditions so far. At $\psi = \psi_R$ the physical coordinate equation itself becomes the remaining "boundary condition." At $\psi = 0$ the pressure equation is solved. This procedure requires that one of the first-order equations, say the physical coordinate equation, be backward differenced in ψ and that the other, the continuity equation, be forward differenced. This point is discussed further in the following section on spatial differencing.

The DAE formulation is well-suited to an implicit boundary condition formulation. As in this example, there is no need to associate boundary conditions with equations. The only requirement is that the correct number of residual functions involving the correct number of dependent variables can be written at each boundary. This flexibility can be especially powerful in problems where the boundary conditions themselves are complicated nonlinear functions involving all the dependent variables. For instance, in problems where complex surface chemistry can occur at the walls the implicit boundary condition capabilities are quite important.[12,13]

Spatial Differencing: The spatial differencing can be accomplished in many different ways, for example by finite differences or finite elements. Here, the spatial differencing is illustrated by finite difference approximations on a fixed grid in the independent variable ψ. The second derivatives in the energy equation are approximated with second-order central differences,

$$\frac{\partial}{\partial\psi}(\lambda\frac{\partial T}{\partial\psi}) \approx \left(\frac{2}{\psi_{j+1}-\psi_{j-1}}\right)\left(\lambda_{j+1/2}\left(\frac{T_{j+1}-T_j}{\psi_{j+1}-\psi_j}\right)-\lambda_{j-1/2}\left(\frac{T_j-T_{j-1}}{\psi_j-\psi_{j-1}}\right)\right), \tag{31}$$

where the subscript j denotes the jth grid point. In the case of the first order equations, Eqs. (23) and (24), one must be forward differenced and the other backward differenced. That is,

$$\frac{r_j-r_{j-1}}{\psi_j-\psi_{j-1}}-\frac{2}{(\rho_j r_j+\rho_{j-1}r_{j-1})}=0, \tag{32}$$

and

$$\frac{p_{j+1}-p_j}{\psi_{j+1}-\psi_j}=0. \tag{33}$$

It does not matter which one is the forward differenced equation, but because the boundary conditions on r are applied at both ends of the domain, it is essential that the two first order equations are differenced in the opposite directions. It may help to think of the first order equations as initial value problems in space ψ, with one propagating into the vessel from the inner boundary and the other propagating from the outer boundary. If the differencing is not done in this way, the boundary conditions can not be specified correctly.

Jacobian Structure: We have found some programming conveniences that are often useful when applying the method-of-lines. Note that we solve Eqs. (23) and (24) in their stated form rather than treating the pressure as a global variable and evaluating r by an integral. The motivation is to simplify the structure of the Jacobian matrix in the linear system of equations that must be solved at each step of Newton's method. We could consider r to be a coefficient in the diffusion term of the energy equation, and we could evaluate it from the integral equation, Eq. (28). However, doing so expands the band width of the Jacobian and results in an inefficient solution procedure. Therefore, we introduce r as a dependent variable at each mesh point, and solve for it with all the other variables. The system of equations is larger, but in a form more amenable to efficient solution. A similar argument is made for the evaluation of pressure p. The pressure appears in every other equation because it is needed to evaluate the density ρ from the equation of state. By solving Eq. (24) we introduce pressure as a variable at each mesh point, and solve a trivial equation to assure that all pressures are all equal. Once again this allows the Jacobian matrix to be banded, but at the expense of creating more dependent variables, i.e. the pressure at each mesh point. Treating simple variables like pressure in this way is a programming convenience. If treated as a global variable, the Jacobian would have a few dense rows and columns, but would not be essentially more difficult to solve.

However, as there is no standard software for these matrices, we prefer to avoid writing a special-purpose linear equation solver.

CONCLUSION

Formulating the method-of-lines as a system of DAEs is often superior formulating it as a system of standard-form ODEs. The DAE formulation provides capabilities to solve a larger class of problems, and it usually provides for a more convenient user interface. However, there are some potential pitfalls. The very general form of DAEs allows one inadvertently to pose a problem for which a solution does not exist or is not unique. Also, one has to be careful to ensure that initial conditions are consistent. Such problems are less likely to be encountered in the more restrictive standard-form ODE approach. Nevertheless, the flexibility of the DAE approach and the availability of production DAE software, such as DASSL, now makes the DAE approach one of practical importance.

ACKNOWLEDGEMENT

We are grateful for the collaboration of our colleagues James A. Miller, Andrew E. Lutz, William S. Winters, and Michael E. Coltrin, with whom we work closely in applying this approach to problems of practical interest.

REFERENCES

1. Hindmarsh, A. C., "ODEPACK, A Systemitized Collection of ODE Solvers. In "Scientific Computing" (R. S. Stepleman, et al., eds.), Vol. 1, pp. 55-64. *IMACS Trans. on Scientific Computation,* North-Holland, Amsterdam (1983).
2. Petzold, L. R. "A Description of DASSL: A Differential-Algebraic System Solver," *Proc. IMACS World Conference, Montreal,* and Sandia National Laboratories Report, SAND82-8637 (1982).
3. Petzold, L. R. "Differential/Algebraic Equations are Not ODEs," *SIAM J. Scientific and Stat. Comp.*, **3**, 367-384 (1982).
4. Gear, C. W. and Petzold, L. R. "ODE Methods for the Solution of Differential/Algebraic Systems," *SIAM J. Numer. Anal.* **21** No. 4, pp. 716-728 (1984).
5. Lötstedt, P. and Petzold, L. R. "Numerical Solution of Nonlinear Differential Equations with Algebraic Constraints I: Convergence Results for the Backward Differentiation Formulas," *Mathematics of Computation* **46** No. 174, pp. 491-516 (1986).
6. Petzold, L. R. and Lötstedt, P. "Numerical Solution of Nonlinear Differential Equations with Algebraic Constraints II: Practical Implications," *SIAM J. Sci. Stat. Comput.* **7** No. 3, pp. 720-733 (1986).

7. Gear, C. W. "Simultaneous Numerical Solution of Differential/algebraic Equations," *IEEE Trans. on Circuit Theory*, **CT-18**, No. 1, pp.89-95, (1971).
8. Gear, C. W. and Brown, R. L. "Documentation for DFASUB – A Program for the Solution of Simultaneous Implicit Differential and Nonlinear Equations," University of Illinois, Dept. of Computer Science, UIUCDCS-R-73-575, (1973).
9. Hindmarsh, A. C. "ODE Solvers for Use with the Method of Lines," *Advances in Computer Methods for Partial Differential Equations – IV*, R. Vichnevetsky and R. S. Stepleman, eds., IMACS, New Brunswick, NJ, pp. 312-316, (1981).
10. Deuflhard, P. and Nowak, U. "Extrapolation Integrators for Quasilinear Implicit ODEs," University of Heidelberg, preprint (1985).
11. Deuflhard, P. Hairer, E. and Zugck, J. "One-Step and Extrapolation Methods for Differential/Algebraic Systems," University of Heidelberg, preprint (1985).
12. Coltrin, M. E., Kee, R. J., Miller, J. A., "A Mathematical Model of the Coupled Fluid Mechanics and Chemical Kinetics in a Chemical Vapor Deposition Reactor," *J. Electrochem. Soc.* **131** no. 2, 425-434 (1984).
13. Coltrin, M. E., Kee, R. J., Miller, J. A., "A Mathematical Model of Silicon Chemical Vapor Deposition: Further Refinements and the Effects of Thermal Diffusion," *J. Electrochem. Soc.* **133** no. 6, 1206-1213 (1986).
14. Kee, R. J., Petzold, L. R., Smooke, M. D., and Grcar, J. F. "Implicit Methods in Combustion and Chemical Kinetics Modeling," *Multiple Time Scales*, Academic Press (1985).
15. Kee, R. J. and Miller, J. A. "A Structured Approach to the Computational Modeling of Chemical Kinetics and Molecular Transport in Flowing Systems," Sandia National Laboratories Report SAND86-8841, to appear *Springer Series in Chemical Physics* (1986).
16. Raffel, B., Warnatz, J., Wolff, H., Wolfrum, J., and Kee, R. J. "Thermal Ignition and Minimum Ignition Energy on Oxygen-Ozone Mixtures," Proceedings, 10th International Colloquium on Gas Dynamics of Explosions and Reactive Systems, Berkeley, CA (1985).
17. Lutz, A. E., Kee, R. J., and Dwyer, H. A. "Ignition Modeling with Grid Adaption," Proceedings, 10th International Colloquium on Gas Dynamics of Explosions and Reactive Systems, Berkeley, CA (1985).
18. Quareni, F. and Yuen, D. A. "Time-dependent solutions of mean-field equations with applications for mantle convection," *Physics of the Earth and Planetary Interior* **36** 337-353 (1984).
19. Winters, W. S. "TOPAZ – The Transient One-Dimensional Pipe Flow Analyzer: Equations an Numerics," Sandia National Laboratories Report SAND85-8248, (1985).
20. Petzold, L. R. "Observations on an Adaptive Moving Grid Method for One-Dimensional Systems of Partial Differential Equations," Lawrence Livermore National Laboratories preprint UCRL-94629, submitted to *Applied Numerical Mathematics* (1986).

UNLIMITED RELEASE
INITIAL DISTRIBUTION

L. R. Petzold, LLNL, L-316 (20)
1126 M. E. Coltrin
1510 J. W. Nunziato
1512 J. C. Cummings
1512 J. E. Shepherd
1513 M. R. Baer
1513 D. W. Larson

8000 R. S. Claassen
Attn: 8100 E. E. Ives
8400 R. C. Wayne

8200 A. N. Blackwell
Attn: 8230 W. D. Wilson
8250 R. D. Cozine
8260 P. E. Brewer
8270 R. C. Dougherty

8233 J. S. Binkley
8233 J. F. Grcar
8235 D. L. Crawford
8235 T. H. Jefferson

8240 C. W. Robinson
Attn: 8241 G. A. Benedetti
8242 M. R. Birnbaum
8243 M. L. Callabresi
8244 C. M. Hartwig

8245 R. J. Kee (30)
8245 G. H. Evans
8245 W. G. Houf
8245 J. C. Yu
8245 S. Paolucci
8245 F. M. Rupley
8245 C. A. Lajeunesse
8245 W. S. Winters
8245 W. L. Hermina
8245 A. E. Lutz
8300 D. L. Hartley
8340 W. Bauer
8348 J. Vitko
8350 P. L. Mattern
8351 R. W. Dibble
8353 G. A. Fisk
8353 J. A. Miller
8360 W. J. McLean
8361 T. H. Fletcher
8361 M. D. Allendorf
8361 D. R. Hardesty
8361 R. E. Mitchell
8362 T. M. Dyer
8363 B. R. Sanders
8363 P. K. Barr
8363 H. A. Dwyer
8470 R. L. Rinne

8265 Publications Division/
Technical Library Processes
Division, 3141

3141 Technical Library
Processes Division (3)

P. W. Dean, 8024-2 for
Central Technical Files (3)