Collocation Software for Boundary Value Differential-Algebraic Equations

Uri M. Ascher*
Department of Computer Science
University of British Columbia
Vancouver, British Columbia
Canada V6T 1Z2
ascher@cs.ubc.ca

Raymond J. Spiteri†
Department of Mathematics
University of British Columbia
Vancouver, BC V6T 1Z2
spiteri@math.ubc.ca

January 19, 1995

Abstract

We describe the methods and implementation of a general-purpose code, COLDAE. This code can solve boundary value problems for nonlinear systems of semi-explicit differential-algebraic equations (DAEs) of index at most 2. Fully implicit index-1 boundary value DAE problems can be handled as well.

The code COLDAE is an extension of the package COLNEW (COLSYS) for solving boundary value ODEs. The implemented method is piecewise polynomial collocation at Gaussian points, extended as needed by the projection method of Ascher-Petzold. For general semi-explicit index-2 problems, as well as for fully implicit index-1 problems, we define a *selective projected collocation* method, and demonstrate its use. The mesh selection procedure of COLSYS is modified for the case of index-2 constraints. We also discuss shooting for initial guesses.

The power and generality of the code are demonstrated by examples.

Keywords. Differential-algebraic equations, boundary value problems, collocation, projection, software

AMS(MOS) subject classifications. 65L10, 65L20

*The work of this author was partially supported under NSERC Canada Grant OGP0004306.

†The work of this author was partially supported by an NSERC Centennial Postgraduate Fellowship.

1 Introduction

Many mathematical models arising in various applications can be written as systems of differential-algebraic equations (DAEs); see, e.g. [16, 22, 23, 24]. The numerical solution of initial value DAEs has received a lot of attention recently, and robust general-purpose codes have been written and extensively used [16, 24]. Similar developments for boundary value DAEs have been considerably lagging behind, although such problems arise in a variety of applications; see, e.g. [2, 13, 14, 15, 18, 25, 26, 28].

In this paper we describe the methods and implementation of a general-purpose code, COLDAE, which can handle boundary value problems for nonlinear systems of semi-explicit DAEs of index at most 2 and of fully implicit index-1 DAEs. The code COLDAE extends the package COLNEW [10], using the projected collocation method of [7] and extensions thereof. To our knowledge, there is currently no other general-purpose code which handles such a class of problems.

The package COLNEW [10] is a modification of the package COLSYS [4]. COLNEW is a robust code which has been used by many scientists and engineers to solve practical problems that can be formulated as systems of ordinary differential equations (ODEs) with boundary or interface conditions. The ODE system which COLNEW handles has a generally nonlinear, mixed order form:

$$\mathcal{D}^{m_i} u_i = f_i(t, \mathbf{z}(\mathbf{u})), \qquad i = 1, \dots, d \tag{1.1}$$

where t is the independent variable, $a \le t \le b$, $\mathcal{D}$ denotes differentiation with respect to t, so $\mathcal{D}^j \equiv \frac{d^j}{dt^j}$, $\mathbf{u} = (u_1, \dots, u_d)^T$ are the dependent variables, and for each t

$$\mathbf{z}(\mathbf{u}) = (u_1, \mathcal{D}u_1, \dots, \mathcal{D}^{m_1-1}u_1, u_2, \dots, u_d, \mathcal{D}u_d, \dots, \mathcal{D}^{m_d-1}u_d)^T$$

In $\mathbf{z}(\mathbf{u})(t)$ there are $m^* = \sum_{i=1}^d m_i$ *differential solution components.* The system (1.1) is subject to m^* side (boundary) conditions, which are each given as a nonlinear relationship in $\mathbf{z}(\mathbf{u})$ at one point,

$$b_j(\beta_j, \mathbf{z}(\mathbf{u})(\beta_j)) = 0, \qquad j = 1, \dots, m^* \tag{1.2}$$

where $a \le \beta_1 \le \beta_2 \le \dots \le \beta_{m^*} \le b$. Thus, (1.2) includes the usual initial value and two-point boundary value problems with separate boundary conditions as special cases.

The underlying numerical method used in COLNEW (COLSYS) is piecewise polynomial collocation at Gaussian points. Thus, with k collocation points at each subinterval of a given mesh,

$$\pi : a = t_0 < t_1 < \dots < t_N = b \tag{1.3}$$

the approximate solution for $u_i(t)$ is sought as a piecewise polynomial of order $k + m_i$ (degree $< k + m_i$) on each mesh subinterval $[t_{n-1}, t_n]$, which is globally continuous together with its first $m_i - 1$ derivatives, viz., $u_i^\pi \in \mathcal{P}_{k+m_i,\pi} \cap C^{(m_i-1)}[a, b]$. Then

$\mathbf{z}^\pi = \mathbf{z}(\mathbf{u}^\pi)$ [1] is determined by requiring that the ODE (1.1) be satisfied at Nk collocation points and that the m^* boundary conditions (1.2) be satisfied as well. We refer, e.g., to [12, 6, 1, 10] for the theory justifying this method. Loosely speaking, the highlights of this theory are that, assuming sufficient smoothness and convergence of a Newton method to an isolated solution of the problem (1.1), (1.2), the following statements hold:

1. The discrete system for the approximate solution has a stability constant which is close to that of the differential problem for h sufficiently small. Here, $h = \max h_n$, where $h_n = t_n - t_{n-1}$, $n = 1, \ldots, N$.

2. At mesh points, there is superconvergence,

$$|z_i(t_n) - z_i^\pi(t_n)| = O(h^{2k}),\ i = 1, \ldots, m^*,\ n = 0, \ldots, N \tag{1.4}$$

3. At points other than mesh points, the convergence order is lower, but local:

$$\begin{aligned} |\mathcal{D}^j u_i(t) - \mathcal{D}^j u_i^\pi(t)| &= [c_i|\mathcal{D}^{k+m_i} u_i(t_n)| + O(h_n)] h_n^{k+m_i-j} + O(h^{2k}), \quad (1.5) \\ & \quad i = 1, \ldots, d,\ j = 0, \ldots, m_i \end{aligned}$$

   where c_i are known constants and $t_{n-1} \leq t \leq t_n$.

These results hold for problems which are not very stiff. They form the theoretical foundation for the mesh selection and error estimation procedures in COLSYS/COLNEW. In particular, the local nature of the leading error term in (1.5) when $k > \max_i m_i$ is exploited.

In COLDAE, we extend the COLSYS class of problems by considering the case where the ODEs in (1.1) involve also m additional dependent variables $\mathbf{y}(t)$ (referred to as *algebraic solution components*) and are supplemented by m additional algebraic relations (constraints). Thus, we have, in place of (1.1),

$$\begin{aligned} \mathcal{D}^{m_i} u_i &= f_i(t, \mathbf{z}(\mathbf{u}), \mathbf{y}), \qquad i = 1, \ldots, d && \text{(1.6a)} \\ 0 &= f_i(t, \mathbf{z}(\mathbf{u}), \mathbf{y}), \qquad i = d+1, \ldots, d+m && \text{(1.6b)} \end{aligned}$$

This is a semi-explicit DAE. The reasons that we have opted not to implement directly the fully implicit case are that while many applications, especially of higher index, are in the semi-explicit form, the theory, both for the analytic problem and for numerical approximations, is much less solid in the more general case (see [2, 22, 20]). Even in the index-1 fully implicit case, none of the above three stability and convergence results hold for a straightforward collocation approximation at Gaussian

[1] We employ the following notational convention: a superscript denotes power, unless it is a Greek letter (in which case it denotes the approximate solution).

points (although there is often basic convergence [2]). We handle fully implicit index-1 problems by imbedding them in semi-explicit index-2 ones – this is discussed in §4.

For the semi-explicit DAE (1.6), we can do much better by collocating the differential solution components as before while collocating the algebraic solution components $\mathbf{y}(t)$ using a generally discontinuous piecewise polynomial of order k: $\mathbf{y}^\pi \in \mathcal{P}_{k,\pi}$. This corresponds to treating the constraints (1.6b) like ODEs of order 0, an obvious idea which was conceived by many and implemented a while ago as well [25]. However, the code of [25] does not extend to the more challenging classes of DAEs considered in the present paper.

If the DAE has index 1, i.e. if the matrix E is nonsingular, where $E = (e_{ij})$,

$$e_{ij} = \frac{\partial f_{d+i}}{\partial y_j}, \qquad i, j = 1, \ldots, m \tag{1.7}$$

then $\mathbf{y}$ can be eliminated in principle, using (1.6b), and substituted into (1.6a) to form an ODE system. Furthermore, a collocation approximation of (1.6) (i.e. collocating both (1.6a) and (1.6b) at the same Gaussian points) can then be viewed as the usual collocation approximation for this obtained ODE with $\mathbf{y}$ eliminated, and the above cited convergence results are recovered (cf. [22, 2]).

But if E is singular (i.e. the DAE has a higher index), then merely approximating $\mathbf{y}$ in the "natural space" $\mathcal{P}_{k,\pi}$ does not produce a collocation method for which the ODE results hold. The simplest way to see this is to note that by writing $\mathcal{D}\mathbf{w} \equiv \mathbf{y}$ we obtain for $(\mathbf{z}(\mathbf{u}), \mathbf{w})$ at best a fully implicit index-1 DAE [19].

A projection method, which achieves the desired stability and convergence behaviour, was therefore proposed in [7, 9]. This method applies to a pure index-2 DAE[2], i.e. for the case where $E \equiv 0$ and CB is nonsingular wherever it is evaluated, where $C = (c_{ij})$, $B = (b_{ij})$,

$$c_{ij} = \frac{\partial f_{d+i}}{\partial z_{p_j}}, \qquad i = 1, \ldots, m, \; p_j = \sum_{l=1}^{j} m_l, \; j = 1, \ldots, d \tag{1.8}$$

$$b_{ij} = \frac{\partial f_i}{\partial y_j}, \qquad i = 1, \ldots, d, \; j = 1, \ldots, m \tag{1.9}$$

With this method, at the right end of each mesh subinterval, following a collocation step as described above (applied within a quasilinearization step), we modify the $(m_i - 1)$st derivatives of $\mathbf{u}$ by a vector from the range space of B so as to satisfy the constraints (1.6b) at the right end mesh point.

We restrict our implementation to DAEs of index at most 2 (this includes ODEs, which are DAEs with index 0). For reasons of inherent problem stability and (related)

[2]A pure index-2 DAE is often referred to as being in Hessenberg form. We will see below that the term "pure" is rather natural.

lack of reliable direct discretization methods, we require that higher index problems be (stably!) converted to lower index ones by analytic differentiation (cf. [8]) prior to applying the code.

Many practical higher index problems are formulated in a pure index-2 form. But still many others have a mix of index-1 and index-2 constraints (i.e. E is singular but not 0). The rank of E may well depend on the iterate where this Jacobian matrix is evaluated and it may vary depending on t on the interval $[a, b]$ (e.g. the index may change on a singular arc in optimal control problems). COLDAE has the option of handling the more general case at the price of a singular value decomposition (SVD) of E at each mesh point. This feature also allows handling fully implicit index-1 DAEs. That and other implementation details are described in §§2, 3 and 4. Specifically, in §2 we briefly recall the piecewise polynomial collocation method [12, 2], and in §3 we recall projected collocation [7] and describe our mesh selection modification. In §3 we also explain the requirements that the user-specified multipoint side conditions (1.2) must satisfy. The problem of finding consistent boundary conditions can be nontrivial even in the initial value case. With the generality of COLDAE we must require that the user supply m^* boundary conditions, even for an index-2 DAE. In the index-2 case only $m^* - m$ independent conditions are needed in theory, but we require that the given boundary conditions include specifying the m constraints (1.6b) at the left end point a – see the examples in §5. In §4 we then treat general semi-explicit index-2 DAEs, and describe a *selective projected collocation* method. We also describe a simple trick for handling fully-implicit index-1 problems. A number of illustrative examples are then presented in §5. We use these examples to also discuss further modifications to COLDAE which also improve the usage of COLNEW, for both DAEs and ODEs.

We end this section with a warning: like any boundary value ODE code, COLDAE will not be successful for all applications. Chances of success can improve significantly, if the user takes care to provide information such as an appropriate initial iterate for difficult nonlinear problems (possibly using continuation), an appropriate initial mesh in some extreme cases, and appropriate boundary conditions (the latter being somewhat trickier here than in the ODE case).

2 DAE Collocation

For a nonlinear boundary value DAE (1.6),(1.2), we implement a quasilinearization method with a damped Newton scheme, as described in [4, 6, 10]. Thus we may assume below, for purposes of the presentation, that the DAE problem is linear(ized).

Let $\rho_1 < \rho_2 < \ldots < \rho_k$ be the k canonical collocation points on $[0, 1]$. In COLDAE we use the zeros of the Legendre polynomial. (This yields a symmetric, algebraically stable difference scheme with $\rho_1 > 0$, $\rho_k < 1$.) The collocation points on a mesh subinterval $[t_{n-1}, t_n]$ are then

$$t_j = t_{n-1} + h_n \rho_j, \qquad j = 1, \ldots, k$$

An (unprojected) collocation step requires that the DAE (1.6) be satisfied by the approximate solution at the collocation points, viz.

$$\begin{aligned} \mathcal{D}^{m_i} u_i^\pi(t_j) &= f_i(t_j, \mathbf{z}^\pi(t_j), \mathbf{y}^\pi(t_j)), \qquad i = 1, \ldots, d && (2.1a) \\ 0 &= f_i(t_j, \mathbf{z}^\pi(t_j), \mathbf{y}^\pi(t_j)), \qquad i = d+1, \ldots, d+m && (2.1b) \end{aligned}$$

for $j = 1, \ldots, k$. We use a monomial representation for the piecewise polynomial approximate solution $(i = 1, \ldots, d)$,

$$u_i^\pi(t) = \sum_{l=0}^{m_i - 1} \frac{(t - t_{n-1})^l}{l!} z_{n-1,p+l} + (h_n)^{m_i} \sum_{l=1}^{k} \psi_l\left(\frac{t - t_{n-1}}{h_n}\right) w_{il} \tag{2.2}$$

where $z_{n-1,p+l} = z_{p+l}^\pi(t_{n-1})$, $p = \sum_{j=1}^{i-1} m_j + 1$, and ψ_l are k polynomials of order $k + m_i$ on $[0,1]$ satisfying

$$\mathcal{D}^j \psi_l(0) = 0,\ j = 0, \ldots, m_i - 1, \qquad \mathcal{D}^{m_i} \psi_l(\rho_j) = \delta_{jl},\ j, l = 1, \ldots, k \tag{2.3}$$

It follows that $w_{il} = \mathcal{D}^{m_i} u_i^\pi(t_l)$. [3] A similar representation is used for the algebraic components $\mathbf{y}$ with $m_i = 0$ in the above formulae (hence $\mathbf{w}_l = \mathbf{y}^\pi(t_l) \equiv \mathbf{y}_l$).

Substituting this representation in the collocation equations (2.1) we obtain $(d + m)k$ equations which we can use in order to locally eliminate the m_ith derivatives $\mathbf{w}_l$ and the algebraic components $\mathbf{y}_l$ in terms of the mesh values of $\mathbf{z}$,

$$\mathbf{z}_{n-1} = (z_1^\pi(t_{n-1}), \ldots, z_{m^*}^\pi(t_{n-1}))^T \tag{2.4}$$

It is not difficult to see that for $h_n > 0$ small enough, the local linear system of order $(d + m)k$ which is solved in this process is nonsingular. The matrix involved has a bounded inverse in the case of a DAE of index 1. In the index-2 case, the inverse bound is $O(h_n^{-1})$, but at least for a pure index-2 form this merely corresponds to scaling of the collocation rows corresponding to the constraints. Thus, the approximate solution at any point t in $[t_{n-1}, t_n)$ can be obtained in terms of $\mathbf{z}_{n-1}$ using (2.2). The continuity requirements on $\mathbf{z}^\pi$ are then obtained by evaluating the approximate solution at t_n and equating it to $\mathbf{z}_n$:

$$\mathbf{z}_n = \Gamma_n \mathbf{z}_{n-1} + \gamma_n, \qquad n = 1, \ldots, N \tag{2.5}$$

A complete linear system of $(N+1)m^*$ equations for $(N+1)m^*$ unknowns is now obtained upon requiring m^* side conditions to be satisfied by $\mathbf{z}^\pi$. The structure of

[3] The functions ψ_l depend of course on m_i as well, but note that they need be constructed only once, not $d + m$ times: if $m_1 = \max_i m_i$, say, then ψ_l for all solution components are appropriate derivatives of those used for u_1^π.

this system, and hence the method for its solution, are the same as encountered with ODEs in COLNEW, and need not be elaborated further here (cf. [10]). We note, however, that our approach to DAEs in all cases is to eliminate the algebraic solution components, obtaining an ODE discretization. This allows us to obtain stability, and affects our decisions about the class of problems solved by COLDAE and their error control and mesh selection: in the latter, only the differential solution components $\mathbf{z}$ are considered.

3 Projected collocation for pure index-2 DAEs

While the straightforward collocation method described above is feasible, and converges as $h \to 0$ for all well-conditioned linear problems of index at most 2 (except for a few pathological cases, cf. [2] and references therein), the properties of nonstiff ODE collocation discussed in §1 are retained only for index-1 problems. For a pure index-2 DAE (i.e. $E \equiv 0$, cf. (1.7)), we therefore apply projected collocation [7, 9]. For the mixed-order DAE (1.6), let

$$\mathbf{x} = (\mathcal{D}^{m_1-1}u_1, \mathcal{D}^{m_2-1}u_2, \ldots, \mathcal{D}^{m_d-1}u_d)^T \tag{3.1}$$

and denote similarly by $\mathbf{x}^\pi$ and $\mathbf{x}_n$ the corresponding subvectors of $\mathbf{z}^\pi$ and $\mathbf{z}_n$. Following a collocation step as described in the previous section, we project

$$\mathbf{x}_n \leftarrow \mathbf{x}^\pi(t_n) + B(t_n)\lambda_n \tag{3.2}$$

with λ_n chosen so that the linearized (1.6b) be satisfied at t_n, $1 \le n \le N$.

The projection requirements allow one to eliminate λ_n locally, obtaining

$$\mathbf{x}_n \leftarrow (I - B(CB)^{-1}C)\mathbf{x}_n - B(CB)^{-1}\mathbf{r} \tag{3.3}$$

(see (1.8), (1.9)), where $\mathbf{r}$ is the corresponding inhomogeneity of the linearized constraints and all quantities are evaluated at t_n. This, in turn, is incorporated into (2.5), where rows $m_1, m_2, \ldots, m_d$ of Γ_n and the corresponding elements of γ_n get multiplied by $(I - B(CB)^{-1}C)(t_n)$ and the term $(B(CB)^{-1}\mathbf{r})(t_n)$ is subtracted from the projected γ_n as well. Once this is done, the rest of the solution process is again the same as for boundary value ODEs.

The constraints (1.6b) are thus satisfied at all mesh points except t_0. We use in COLDAE the convention that satisfying the constraints (1.6b) at the left end point $t_0 = a$ is a requirement on the boundary conditions (1.2). Indeed, recall that for an index-2 DAE only $m^* - m$ side conditions independent of the constraints are required to hold. In order to be able not to distinguish a priori between index-1 and index-2 problems, COLDAE expects m^* side conditions on $\mathbf{z}(\mathbf{u})$ in any case: for index-2 problems these include the specification of the constraints at the left end point.

With the above described projection, the superconvergence results (1.4) hold [7, 9]. [4] Let us quickly recall that the proof is obtained by considering a $(d-m)\times d$ matrix function $R(t)$ with normalized, linearly independent rows, which satisfies for each t

$$RB = 0 \tag{3.4}$$

Thus, multiplying (1.6a) by R eliminates $\mathbf{y}$, and an *essential underlying ODE* (EU-ODE) is obtained for

$$\mathbf{v}(t) = R\mathbf{u} \tag{3.5}$$

The results (1.4), (1.5) are retrieved for $\mathbf{v}$ (this is simple to see particularly when R is constant), and the projection on the constraint manifold at mesh points then helps to retrieve the superconvergence (1.4) for $\mathbf{u}$ as well.

The expression obtained in place of (1.5), however, is for $\mathbf{v}$, not $\mathbf{u}$. The first term on the right hand side of (1.5) is what gets equidistributed when selecting a new mesh, and this now relates to higher derivatives of $\mathbf{v}$. In other words, the error equidistribution should be done on the constraint manifold. In practice, we approximate R for this purpose by a piecewise constant function, and thus project the approximate solution $\mathbf{x}^\pi$ on each mesh subinterval $[t_{n-1}, t_n]$, multiplying it by the already computed $(I - B(CB)^{-1}C)(t_n)$. Divided differences are then used to estimate the higher derivatives of $\mathbf{v}$, as in COLSYS (see, e.g., §9.3 of [6]). For nonlinear problems, this is done only after convergence of the nonlinear iteration has been achieved on a current mesh.

4 Selective projected collocation for the general case

In this section we consider the general case for (1.6) where the matrix $E(t)$ is possibly singular but not necessarily 0. We will in fact allow the rank of $E(t)$ to vary with t, except that we require a constant rank on each mesh subinterval in the numerical approximation. This means, in practical terms, that we allow for switching points $\hat{t}_l$ where the rank of E may change. These points must become part of any mesh in COLDAE, so their location should be known in advance; however, there are standard tricks (see, e.g., [6]) to convert a problem with unknown switching points to one with known switching points, provided we know how many such points there are.

Given a matrix $E(t)$ (using (1.7) for the linearized DAE (1.6)), consider its pointwise singular value decomposition:

$$E = U\Sigma V^T \tag{4.1}$$

[4] Note that now the global continuity has been lowered: $u_i^\pi \in \mathcal{P}_{k+m_i,\pi} \cap C^{(m_i-2)}[a,b]$, with continuity from the right in $\mathbf{x}^\pi$ at mesh points.

where U and V are orthogonal matrices and

$$\Sigma = \begin{pmatrix} S & 0 \\ 0 & 0 \end{pmatrix} \tag{4.2}$$

with S a nonsingular diagonal matrix of rank r. (In general, $r = r(t, \mathbf{z}(\mathbf{u})(t), \mathbf{y}(t))$.) Let

$$U = (U_1\, U_2), \qquad V = (V_1\, V_2) \tag{4.3}$$

where U_1 and V_1 consist of the first r columns of U and V, respectively, and U_2 and V_2 are the rest. Writing

$$\mathbf{y} = V\psi = V_1\psi_1 + V_2\psi_2 \tag{4.4}$$

we have

$$E\mathbf{y} = U_1 S \psi_1 \tag{4.5}$$

so ψ_1 can be eliminated from the constraints. This is the "index-1 part" of the algebraic unknown vector $\mathbf{y}$. Moreover, clearly $U_2^T E = 0$, so upon multiplying the linear (1.6b) by U_2^T and substituting $\psi = V^T\mathbf{y}$ in (1.6a) we obtain a DAE in pure index-2 form for $\mathbf{u}$ and ψ_2, the latter being the "pure index-2 part" of the algebraic unknown vector $\mathbf{y}$. In this transformed DAE of pure form, the condition for index-2 is clearly that $U_2^T C B V_2$ be nonsingular.

Consider next applying a collocation scheme without projection, as described in §2, to the linear (1.6). The transformations just described are all pointwise and involve no derivatives, so the collocation equations for the transformed DAE are identical to the transformed collocation equations applied to the original form.

The only difference arises for the projected collocation method, where the projection should be carried out only onto the constraint manifold corresponding to the pure index-2 variables ψ_2. But this is conceptually simple now: the procedure described in §3 applies here with no change, except that

$$B \leftarrow BV_2, \qquad C \leftarrow U_2^T C \tag{4.6}$$

The resulting method is referred to as selective projected collocation.

In COLDAE, the user has the following three options:

1. *Collocating with no projection.* This is good for ODEs and for semi-explicit index-1 boundary value problems. For index-2 problems and for fully implicit index-1 problems, one of the next two options is usually preferable.

2. *Projected collocation for pure index-2.* This is the preferred option if the user knows that the problem to be solved is indeed of this form (everywhere in t).

3. *Selective projected collocation.* This "rich man's option" corresponds to the case described in the present section. At the right end of each subinterval (within a quasilinearization iteration) the code uses LINPACK routines to decompose E

as in (4.1). The unprojected collocation procedure is as usual, except that at the end of each subinterval processing, a projection step is performed as in §3, but using (4.6). These updated matrices are also used for the mesh selection procedure. [5]

Fully implicit index-1 DAEs

The mixed index-2 option discussed in this section also allows dealing with fully implicit index-1 problems. Consider the system of m equations

$$\mathbf{g}(t, \mathbf{x}, \mathcal{D}\mathbf{x}) = \mathbf{0} \tag{4.7}$$

(we suppress higher derivatives for notational simplicity), and denote

$$\mathbf{y} = \mathcal{D}\mathbf{x}, \qquad E = \mathbf{g_y}, \qquad C = \mathbf{g_x} \tag{4.8}$$

Applying SVD to E of (4.8) as before and using the notation (4.1)–(4.5), it follows that near a given solution (4.7) has index 0 if the rank is $r \equiv m$ and index 1 if $r < m$, r constant, and $U_2^T C V_2$ is nonsingular (see, e.g., [19, 23]). But this allows us to pose the index-1 (4.7) as the semi-explicit DAE of index at most 2

$$\begin{array}{rcl} \mathcal{D}\mathbf{x} & = & \mathbf{y} \\ \mathbf{0} & = & \mathbf{g}(t, \mathbf{x}, \mathbf{y}) \end{array} \tag{4.9}$$

This DAE, subject to the same boundary conditions on $\mathbf{x}$ which come with (4.7), can be solved by COLDAE using the general index option of selective projected collocation.

5 Numerical examples and further discussion

In the tables below we employ the following notation, unless otherwise specified: N_j denotes the size of the final mesh needed to satisfy the error tolerances (as estimated by the code), when using the jth projection option among the three mentioned at the end of the previous section – if no such convergence is reached then this is indicated by $*$; $erru_j$ denotes the maximum error on all components of $\mathbf{u}$ measured at 101 equidistant points, in case that the exact solution is known, when the jth projection method is used; $erry_j$ denotes similar errors on $\mathbf{y}$ components; $ermsh_j$ denotes similarly the maximum error in $\mathbf{u}$ at mesh points on the final mesh.

5.1 Projected collocation and mesh selection

[5] The expense of SVD computations is not very siginificant for small problems. Note that the decompositions are carried out only on $m \times m$ matrices. To deal with larger problems more efficiently, we are considering smooth factorizations [27, 21]. This will be reported in the near future.

Example 1

Consider for $0 \le t \le 1$

$$
\begin{aligned}
x_1' &= (\nu - \frac{1}{2-t})x_1 + (2-t)\nu y + q_1(t) \\
x_2' &= \frac{\nu - 1}{2-t}x_1 - x_2 + (\nu - 1 - \frac{\nu p(t)}{2+t})y + q_2(t) \\
0 &= (t + 2 - p(t))x_1 + (t^2 - 4)x_2 + r(t)
\end{aligned}
$$

with $x_1(0) = 1$ and $p(t)$ a known function to be specified below. This example has been analyzed, in slight variations and with $p \equiv 0$, in [7, 8]. Here $\nu \ge 1$ is a parameter. The inhomogeneities $\mathbf{q}$ and r are chosen to be

$$
\begin{aligned}
\mathbf{q} &= \begin{pmatrix} \frac{3-t}{2-t}e^t \\ (2 + \frac{(\nu+2)p(t)+p'(t)}{t^2-4} - \frac{2tp(t)}{(t^2-4)^2})e^t \end{pmatrix} \\
r &= -(t^2 + t - 2)e^t
\end{aligned}
$$

so that the exact solution is $x_1 = e^t$, $x_2 = (1 + \frac{p(t)}{t^2-4})e^t$, $y = -\frac{e^t}{2-t}$.

First we let $p(t) \equiv 0$. While this is a linear, pure index-2 initial value problem, it can be particularly nasty, depending on the value of ν:

- The stability constant for unprojected collocation is exponentially large in ν, while that for projected collocation (and for the problem itself) grows only linearly in ν.
- Even the projected collocation scheme exhibits a behaviour common to nonstiff integration methods, requiring a small stepsize h when $\nu h \gg 1$ (see [8]).

In Table 5.1 we record results of running COLDAE with and without projection. The initial mesh in all cases was uniform with $N = 5$, the number of collocation points per mesh subinterval was $k = 4$, a maximum mesh size of 100 subintervals was imposed and the tolerance on both components of $\mathbf{u} = \mathbf{x}$ was $1.e-5$. (Note also the additional boundary condition used, $x_1(0) - 2x_2(0) = -1$, as dictated by the constraint.)

ν	N_1	$erru_1$	$erry_1$	$ermsh_1$	N_2	$erru_2$	$erry_2$	$ermsh_2$
1	10	.86e-8	.10e-4	.86e-8	10	.12e-8	.87e-5	.89e-15
10	10	.13e-4	.23e-3	.13e-4	10	.15e-7	.87e-5	.80e-11
50	*				10	.44e-6	.86e-5	.80e-7
100	*				10	.37e-6	.87e-5	.11e-6

Table 5.1: Errors for Example 1

The advantage of using the projected collocation method is clearly demonstrated for this example. Note the recovery of the superconvergence results for moderate values of ν.

Next, we consider this DAE with

$$p(t) = -(1 + erf(\frac{t - 1/3}{\sqrt{2\varepsilon}}))$$

with the parameter $0 < \varepsilon \ll 1$. This function varies rapidly (like $\varepsilon^{-1/2}$) around $t = 1/3$, and this is reflected in the form of the solution $x_2(t)$. Note that the EUODE remains stable here, because $p \leq 0$, $p' \leq 0$.

We have used this example to test our mesh selection procedure (the fact that this is an initial value problem is immaterial here: the approach remains as if this were a boundary value problem). In Figs. 5.1 and 5.2 we display the mesh consisting of every second point of the final COLDAE mesh (cf. [4]) and the solution x_2 based on that mesh, when running for $\varepsilon = 1.e-5$, $\nu = 20$ (the latter excludes the non-projection option $proj_1$), using a tolerance $1.e-5$ on both components of $\mathbf{x}$, a uniform initial mesh of 5 subintervals and $k = 4$. For Fig. 5.1 we used essentially the COLSYS mesh selection procedure for $\mathbf{x}$. While the layer region has been detected, the mesh to the left of the layer is clearly much finer than it could have been. In general, it has been our experience that this procedure needs a smaller h in the current mesh, in order to recognize the need to redistribute it, than the same procedure with the projected solution requires. For Fig. 5.2 we projected $\mathbf{x}$ on the algebraic solution manifold, as described at the end of §3, before applying the mesh selection procedure. The resulting mesh has the layer better located, and therefore a smaller mesh (80, instead of 139 subintervals) is needed for achieving roughly the same accuracy.
□

5.2 Selective projected collocation

Example 2

The following example has been developed from one proposed by S. Reich (private communication). It is an instance of a nonlinear, semi-explicit DAE of index at most 2:

$$\begin{aligned}
x_1' &= (\varepsilon + x_2 - p_2(t))y + p_1'(t) \\
x_2' &= p_2'(t) \\
x_3' &= y \\
0 &= (x_1 - p_1(t))(y - e^t)
\end{aligned}$$

which we want to solve for the boundary conditions

$$x_1(0) = p_1(0), \qquad x_2(1) = p_2(1), \qquad x_3(0) = 1$$

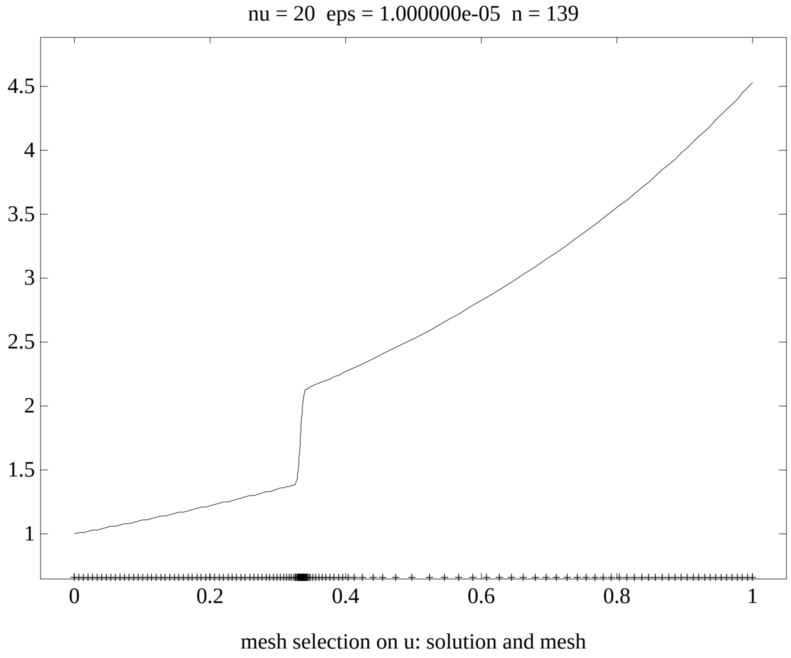

Figure 5.1: Example 1 - unprojected mesh selection

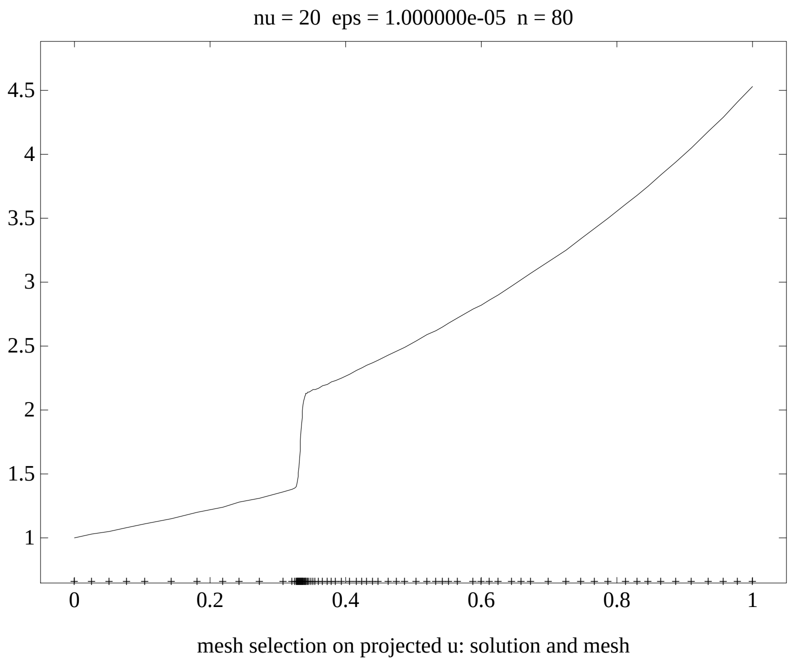

Figure 5.2: Example 1 - projected mesh selection

Here $\varepsilon > 0$ is a parameter and p_1, p_2 are given functions. For the results reported in Table 5.2 we chose

$$p_1 = p_2 = \sin t$$

This problem has two isolated solutions which can be easily computed.

- One solution is obtained by setting $y = e^t$. This yields $x_3 = e^t$, $x_2 = p_2(t)$ (in any case), and $x_1 = p_1(t) + \varepsilon(e^t - 1)$. The linearized problem around the exact solution has index 1 and nothing exciting happens in addition, except that the conditioning deteriorates as $\varepsilon \to 0$.

- The other solution is obtained by setting $x_1 = p_1(t)$, which is the other possibility for satisfying the constraint. This yields $x_2 = p_2(t)$, $y = 0$ and $x_3 = 1$. The variational problem at this solution has index 2, with conditioning growing with ε^{-1}. For $\varepsilon = 0$, y and x_3 are not defined in a locally unique way. Yet, letting $e_1 = (x_1^\pi)' - p_1'$ and $e_2 = x_2^\pi - p_2$, we see from the equation for x_1' that for all $\varepsilon \geq 0$ small enough compared to h, the error in y depends only on e_1 and e_2, which in turn depend only on h and not on ε. Thus, the same numerical solution is obtained when $\varepsilon h^{-1} \to 0$. The numerical solution is locally unique and the stability constant deteriorates only as a negative power of h.

In Table 5.2 we record results of running COLDAE with and without projection. The solution to which the code converges depends, of course, on the initial guess. Since the index may be 1 or 2, depending on the current iterate, the option of always projecting is not used, and the options compared are never projecting (option 1) vs. selective projected collocation (option 3). The initial parameter setup for COLDAE was as in the previous example. In addition to the information described for Table 5.1 we also write under 'sln' whether the first or the second exact solution is approximated.

sln	ε	N_1	$erru_1$	$erry_1$	$ermsh_1$	N_3	$erru_3$	$erry_3$	$ermsh_3$
1	1	10	.75e-9	.16e-6	.11e-13	10	.75e-9	.16e-6	.11e-13
1	1.e-4	10	.65e-9	.16e-6	.41e-10	10	.65e-9	.16e-6	.41e-10
1	1.e-8	10	.58e-7	.46e-5	.54e-7	10	.58e-7	.46e-5	.54e-7
2	1	10	.10e-7	.20e-5	.10e-7	10	.12e-8	.19e-6	.14e-14
2	1.e-4	10	.10e-3	.20e-1	.10e-3	10	.12e-4	.19e-2	.48e-9
2	1.e-8	*	*	*	*	40	.11e-3	.73e-1	.16e-6

Table 5.2: Errors for Example 2

We note that the values listed in Table 5.2 depend on the initial guess (e.g. for the last entry, significantly better errors were obtained when starting from a different initial guess). For the index-1 solution, both options give precisely the same results, of course. For the index-2 solution, the projection option is significantly better.

When we set $\varepsilon = 1.e - 8$, we could not obtain convergence for any method, until we increased the initial mesh size to $N = 20$. Then convergence was obtained for the selective projection method, but not for the unprojected one.
□

5.3 Fully implicit, index-1 problems

A class of problems where (selective) projected collocation proves useful, is fully implicit index-1 DAEs (4.7) converted to semi-explicit index-2 DAEs (4.9). We have used COLDAE to solve the example in [3] in this way (the unprojected collocation method does very poorly here and the projected method does very well, similarly to Example 1). We have also solved a version of the detonation problem considered in [15]. This is an instance where the availability of a general-purpose tool like COLDAE allows one to spend less time on developing special-purpose methods and codes.

The price paid in solving (4.9) instead of (4.7) is that the size of the system appears to have doubled. However, note that in applications (e.g. the detonation problem) often only part of the given system is not in semi-explicit form to begin with. Thus, the size of the resulting higher index DAE can be much less than doubled. Also, the resulting index-2 DAE is often already in pure form, so no SVD is needed for its solution in such cases. Finally, we remark that if the problem contains inhomogeneities with jump discontinuities then such functions should be defined like the approximate solution, i.e. with a mesh point placed at the discontinuity location and the inhomogeneity defined to be continuous from the right. Appropriately discontinuous approximate solutions are then possible.

5.4 Shooting for initial guesses

Let us turn now to the question of solving initial value DAEs using COLDAE. This is certainly possible in principle, viewing initial value problems as a special case of boundary value problems. But the code does not take any special advantage of the relative simplicity and locality of initial value problems, and therefore it is not competitive with initial value codes in general. One major difference is in the amount of storage which COLDAE requires, which increases linearly with the number of steps N, whereas the amount of storage required by initial value codes is usually independent of N. The latter does not hold, however, if the approximate solution is to be known for all values t in a given interval $[a, b]$ simultaneously. That is the case when the intended use of the initial value solution is as an initial approximation ("guess") for a boundary value problem solution.

This is sometimes a useful idea (not only for DAEs, but also for boundary value ODEs). It may happen that a major difficulty in practice when solving nonlinear boundary value problems is in obtaining a sufficiently good initial guess to start a convergent damped Newton iteration from. Some users have in fact claimed that this difficulty gives shooting methods an advantage over COLSYS. But if a shooting method can indeed be applied (i.e., if the conditioning of the initial value problem is not much worse than the conditioning of the boundary value problem, and initial value solutions starting from a do reach b, cf. e.g. [6]), then we can also shoot *once* in order to obtain an initial approximation in the context of COLDAE.

This can be conveniently done in one calling (driver) program, which first calls COLDAE to solve a (user-defined) initial value problem, and then calls COLDAE again to solve the desired boundary value DAE problem by continuation from the solution of the initial value one. Thus, the user has only to guess the initial values — the initial guess for other values of t is taken e.g. to be the same, i.e. constant. The quasilinearization iteration of COLDAE then amounts to a sort of waveform method. In order to facilitate this possibility further, we have implemented in COLDAE an option which performs uncontrolled (standard) Newton iterations without damping, until convergence of the nonlinear iteration is hopefully obtained. This has been used in order to obtain good initial guesses for some examples in the class of problems described next.

5.5 Optimal control and parameter estimation

There are many applications in which a DAE for state variables involves control functions. The control is to be determined so as to minimize some objective functional. For instance, in robotics one considers problems of trajectory optimization (see [28, 18])

$$\min J = \psi(\mathbf{p}(T), \mathbf{p}'(T)) + \int_0^T L(\mathbf{p}(t), \mathbf{p}'(t), \mathbf{u}(t), t)dt \tag{5.1}$$

subject to the constrained multibody equations

$$\begin{aligned} M(\mathbf{p})\mathbf{p}'' &= \mathbf{f}(\mathbf{p}, \mathbf{p}', \mathbf{u}) - G^T(\mathbf{p})\lambda & (5.2a) \\ \mathbf{0} &= \mathbf{g}(\mathbf{p}) & (5.2b) \end{aligned}$$

and some side conditions

$$\mathbf{b}(\mathbf{p}(0), \mathbf{p}'(0), \mathbf{p}(T), \mathbf{p}'(T)) = \mathbf{0} \tag{5.3}$$

where $\mathbf{p}$ are generalized body positions, M is a positive definite mass matrix and $G = \frac{\partial \mathbf{g}}{\partial \mathbf{p}}$ is a constraint matrix with a full row rank. The control $\mathbf{u}$ appears in the applied forces $\mathbf{f}$, and the objective can be, e.g., to find the trajectory which takes the system from one specified position to another in a minimum amount of time T.

Replacing (5.2b) by its derivative

$$\mathbf{0} = \mathbf{g}'(\mathbf{p}) = G(\mathbf{p})\mathbf{p}'$$

or by a combination

$$\mathbf{0} = \gamma\mathbf{g}(\mathbf{p}) + G(\mathbf{p})\mathbf{p}' \tag{5.4}$$

($\gamma > 0$, cf. [11]), the state equations form an index-2 DAE. The necessary conditions for this problem yield a boundary value DAE for $\mathbf{p}$, $\mathbf{p}'$, λ and their adjoint variables (i.e. the obtained size is double that of (5.2); see, e.g., [17]). Assume that L is such that the resulting DAE has index 2. This is a boundary value problem even if the conditions (5.3) are given only at $t = 0$. When specifying the boundary conditions for COLDAE, note that the constraint (5.2b) and its adjoint constraint equation must be specified by the boundary conditions at $t = 0$, and that these constraints *are* satisfied automatically at $t = T$, so the conditions at T must be *complementary* (see Example 3 below).

The use of COLDAE to solve a system like (5.1)–(5.3) has the advantage of availablity of a general-purpose software. However, we do not recommend it as a general "cure". Firstly, there may be many equations in (5.2); secondly, the controls are often restricted by inequalities, and this cannot be handled directly by COLDAE. Another note is that by concentrating on the necessary conditions for an extremum of (5.1)–(5.3), the sense of minimization is lost, so some information is not being used. Nonetheless, COLDAE is an available tool which can minimize the human effort in writing special-purpose programs for certain applications.

A somewhat related problem is that of parameter estimation. A given DAE of a type which COLDAE covers depends on unknown parameters ω. The problem is to recover the parameters so that the DAE solution best fits given observations on the solution. Such problems are ill-posed, and can be difficult to solve.

Example 3

Consider

$$\begin{aligned} x_1' &= x_2 + x_1 y \\ x_2' &= -\omega^2 x_1 + x_2 y \\ 0 &= (\frac{\pi}{3})^2 x_1^2 + x_2^2 - 1 \\ & \quad x_1(0) = 0, \;\; x_2(0) = 1 \end{aligned}$$

where ω is a constant parameter. This is a pure index-2 DAE and the exact solution for $\omega = \bar{\omega} := \frac{\pi}{3}$ is

$$x_1 = \omega^{-1} \sin \omega t, \;\; x_2 = \cos \omega t, \;\; y = 0$$

Now, suppose that we are to find ω which best fits an observed function $r(t)$, where the observations are on $x_1(t) + x_2(t)$, $0 \le t \le 2$. The necessary conditions for minimizing $\frac{1}{2}\int_0^2 (x_1 + x_2 - r)^2 dt$ yield the boundary value DAE problem with 6 ODEs and 2 constraints, for $x_1, x_2, \omega, \lambda_1, \lambda_2, \nu, y$ and μ,

$$\begin{aligned} x_1' &= x_2 + x_1 y \\ x_2' &= -\omega^2 x_1 + x_2 y \end{aligned}$$

$$
\begin{aligned}
\omega' &= 0 \\
\lambda_1' &= -y\lambda_1 + \omega^2\lambda_2 - 2(\frac{\pi}{3})^2 x_1\mu - (x_1 + x_2 - r) \\
\lambda_2' &= -\lambda_1 - y\lambda_2 - 2x_2\mu - (x_1 + x_2 - r) \\
\nu' &= 2\omega x_1\lambda_2 \\
0 &= (\frac{\pi}{3})^2 x_1^2 + x_2^2 - 1 \\
0 &= x_1\lambda_1 + x_2\lambda_2
\end{aligned}
$$

$$
\begin{aligned}
x_1(0) &= 0,\ x_2(0) = 1,\ \nu(0) = 0,\ \lambda_2(0) = 0 \\
\nu(2) &= 0,\ x_2(2)\lambda_1(2) - (\frac{\pi}{3})^2 x_1(2)\lambda_2(2) = 0
\end{aligned}
$$

We now solve this problem twice, using COLDAE:

1. Setting $r(t) = \bar{\omega}^{-1}\sin\bar{\omega}t + \cos\bar{\omega}t$, we recover $\omega = \bar{\omega}$ to machine precision (with 20 uniform mesh elements and 4 collocation points per element).

2. Setting $r(t)$ to be the piecewise linear interpolant of the values $\bar{\omega}^{-1}\sin\bar{\omega}t + \cos\bar{\omega}t$ at 21 equidistant points, we recover $\omega = 1.3792$, which, as an approximation to $\bar{\omega}$, is in a relative error of 31.7%. Smoothing $r(t)$ by passing a cubic spline through these 21 points and re-running has not helped much. Obviously, this problem is not well-conditioned.

□

5.6 On regularization

Various authors have proposed to *regularize* a given DAE by adding to (1.6b) terms involving $\epsilon\mathbf{y}'$, turning the problem (1.6) into an ODE. The parameter ϵ, $0 < \epsilon \ll 1$, must be taken small to ensure that the solution of the obtained problem does not deviate much from the desired one. (In this description we do not consider stabilization methods, which maintain the exact solution in a reformulated problem, as regularization methods.) With COLDAE one can of course solve the obtained boundary value ODE. However, this approach is often unnecessary, and may introduce stiffness where there has not been any. It is not difficult to conjure up examples where the obtained regularized problem is more difficult to solve than the original one. An exception is a singularity point, where the local index of the DAE increases at that one point. We proceed with an example for the latter.

Example 4

The DAE

$$
x_1' = x_2 y - \alpha J
$$

$$
\begin{aligned}
x_2' &= y - 1 \\
0 &= x_1 - (\frac{J^2}{y} + y)
\end{aligned}
$$

has been investigated in [5] as a simple model for the hydrodynamic semiconductor equations. The boundary conditions are

$$y(0) = y(\beta) = \bar{n}$$

Here J, α, $\bar{n} > J$ and β are known positive constants. The resulting solution y is positive on $[0, \beta]$. The linearization of this DAE clearly has index 1, except where $y = J$. Cases where y becomes less than J (starting at $y(0) > J$) correspond to transonic flow. The solution is then discontinuous where y jumps back from ($< J$) to ($> J$) (see [5]).

In such a case, where y is discontinuous at an unknown location at which there is a singularity, one may be better off replacing the DAE by a regularizing ODE. The method used in [5] is equivalent to replacing the constraint above by

$$\epsilon y' = x_1 - (\frac{J^2}{y} + y)$$

insisting that the constraint be satisfied at $t = 0$, viz.

$$x_1(0) = \frac{J^2}{\bar{n}} + \bar{n}$$

Solutions with rather sharp layer profiles were obtained in [5] using COLDAE (i.e. COLNEW for the regularized boundary value ODE) by applying continuation in ϵ. $\square$

While this example shows that replacing the DAE by a regularizing ODE can be beneficial, we note that there are various types of possible singularities and that this is not necessarily a cure for all problems. Also, the regularized ODE and its additional boundary condition must be chosen with care.

References

[1] U. Ascher, *Collocation for two-point boundary value problems revisited*, SIAM J. Numer. Anal. 23 (1986), 596-609.

[2] U. Ascher, *On numerical differential algebraic problems with application to semiconductor device simulation*, SIAM J. Numer. Anal. 26 (1989), 517-538.

[3] U. Ascher, *On symmetric schemes and differential-algebraic equations*, SIAM J. Sci. Stat. Comput. 10 (1989), 937-949.

[4] U. Ascher, J. Christiansen and R. Russell, *Collocation software for boundary value ODEs*, ACM Trans. Math. Software 7 (1981), 209-222.

[5] U. Ascher, P. Markowich, P. Pietra and C. Schmeiser, *A phase plane analysis of transonic solutions for the hydrodynamic semiconductor model*, Mathematical Models and Methods in Applied Sciences 1 (1991), 347-376.

[6] U. Ascher, R. Mattheij and R. Russell, *Numerical Solution of Boundary Value Problems for Ordinary Differential Equation*, Prentice-Hall, 1988.

[7] U. Ascher and L. Petzold, *Projected implicit Runge-Kutta methods for differential-algebraic equations*, SIAM J. Numer. Anal. 28 (1991), 1097-1120.

[8] U. Ascher and L. Petzold, *Stability of Computational Methods for Constrained Dynamics Systems*, TR 91-3, Dept. Computer Science, Univ. of B.C., 1991. SIAM J. SISSC (1993), to appear.

[9] U. Ascher and L. Petzold, *Projected Collocation for Higher-Order Higher-Index Differential-Algebraic Equations*, TR 91-9, Dept. Computer Science, Univ. of B.C., 1991. JCAM (1992), to appear.

[10] G. Bader and U. Ascher, *A new basis implementation for a mixed order boundary value ODE solver*, SIAM J. Scient. Stat. Comput. 8 (1987), 483-500.

[11] J. Baumgarte, *Stabilization of constraints and integrals of motion in dynamical systems*, Comp. Math. Appl. Mech. Eng. 1 (1976), 1-16.

[12] C. de Boor and B. Swartz, *Collocation at Gaussian points*, SIAM J. Numer. Anal. 10 (1973), 582-606.

[13] B.P. Boudreau, *A steady-state diagenetic model for dissolved carbonate species and pH in the porewaters of oxic and suboxic sediments*, Geochimica et Cosmochimica Acta 51 (1987), 1185-1196.

[14] B.P. Boudreau and D.E. Canfield, *A provisional diagenetic model for pH in anoxic porewaters: Application to the FOAM site*, J. Marine research 46 (1988), 429-455.

[15] M. Braithwaite, T. Farran, I. Gladwell, P. Lynch, A. Minchinton, I. Parker and R. Thomas, *A detonation problem posed as a differential/algebraic boundary value problem*, Math. Eng. Ind. 3 (1990), 45-57.

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

[17] A. Bryson and Y.C. Ho, *Applied Optimal Control*, Ginn and Co., Waltham, MA, 1969.

[18] F. Delebeque and R. Nikoukah, *On simulation and control of multibody mechanical systems*, Lecture at the ARO workshop on mechanical systems and vehicle simulation, Raleigh, NC, Nov. 5-7, 1992.

[19] C.W. Gear, *Differential-algebraic equation index transformations*, SIAM J. Sci. Stat. Comput. 9 (1988), 39-47.

[20] C.W. Gear and L.R. Petzold, *ODE methods for the solution of differential-algebraic systems*, SIAM J. Numer. Anal. 21 (1984), 716-728.

[21] P. Gill, W. Murray, M. Saunders, G.W. Stewart and M. Wright, *Properties of a representation of a basis for the null space*, Math. Prog. 33 (1985), 172-186.

[22] E. Griepentrog and R. Marz, *Differential-Algebraic Equations and their Numerical Treatment*, Teubner-Texte Math. 88, Teubner, Leipzig, 1986.

[23] E. Hairer, Ch. Lubich and M. Roche, *The numerical solution of differential-algebraic systems by Runge-Kutta methods*, Lecture Notes in Math vol. 1409, Springer-Verlag, 1989.

[24] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II* , Springer-Verlag, 1991.

[25] M. Ho, *A collocation solver for systems of boundary-value differential/algebraic equations*, Computers and Chem. Eng. 7 (1983), 735-737.

[26] J. S. Logsdon and L. T. Biegler, *Accurate solution of differential-algebraic optimization problems*, Industrial and Engineering Chemistry Research 28 (1989) 1628-1639.

[27] A. Maciejewski and C. Klein, *The singular value decomposition: computation and applications to robotics*, Int. J. Robotics Res. 8 (1989), 63-79.

[28] M. Steinbach, H. G. Bock and R. W. Longman, *Time-optimal extension or retraction in polar coordinate robots: a numerical analysis of the switching structure*, Proc. AIAA Guidance, Navigation and Control, Boston, Aug. 1989.