Computational Speedup for General Nonlinear Higher Index DAE Integrators

Stephen L. Campbell*

and

Yangchun Zhong*

Department of Mathematics
North Carolina State University
Raleigh, NC 27695-8205 USA
email: slc@math.ncsu.edu
FAX: 1-919-515-3798

IMACS'94

March 1, 1994

Abstract

There has been considerable research on numerical methods for differential algebraic equations (DAEs) $f(x', x, t) = 0$ where $f_{x'}$ is identically singular. The index provides one measure of the singularity of a DAE. Most of the numerical analysis literature on DAEs to date has dealt with DAEs with indices no larger than three. Even in this case, the systems were often assumed to have a special structure. Recently a numerical method has been proposed that can, in principle, be used to integrate general unstructured higher index solvable DAEs. Modifications of this approach can be used to design constraint preserving integrators for general nonlinear higher index DAEs. Previous work on these more general approaches has focused on their feasibility and theoretical basis. This paper examines some of the issues involved with their computationally efficient implementation.

*Research supported in part by the National Science Foundation under DMS-9122745.

1 Introduction

Many physical problems are most easily initially modeled as a nonlinear implicit system of differential and algebraic equations (DAEs),

$$f(x', x, t) = 0\texttt{[dae]} \tag{1}$$

with $f_{x'} = \partial f/\partial x'$ identically singular [7]. The index ν, which will be defined shortly, is one measure of how singular a DAE is. An ordinary differential equation is index zero, and increasing index implies more complex behavior. Initially most of the numerical work on DAEs assumed that the DAE was index one. However, many of the problems in constrained mechanics are initially formulated as index two and three DAEs. DAEs of index up to six naturally occur in mechanics if actuator dynamics, joint flexibility, and other effects are included [13]. Higher index DAEs ($\nu \geq 2$) also occur in several other areas [7,13].

Most of the numerical methods for DAEs require that the systems have special structure, such as being a mechanical system with holonomic constraints [26], or have indices of only one or two. A more general approach is introduced in [9]. That approach is explicit and does not preserve constraints. The first approach for the constraint preserving numerical integration of general unstructured higher index DAEs is introduced in [18]. An alternative approach could probably be developed based on the ideas in [30]. The approach of [9,18] is geared toward the integration of moderate sized nonlinear DAEs with indices of approximately six or less such as typically arise in control and mechanics problems. This approach should be especially useful in the early stages of design and simulation when various computer generated models are being used to investigate system behavior. It will also be useful as a truth model for investigating other integration methods and the validity of various simplified models. A theoretical analysis of these methods and the establishing of their practicality is done in [9,12,17,18,19,20]. However little work has been done on how to efficiently implement these approaches.

This paper will examine several of the issues involved in efficient implementation of these methods. Section 2 will present a few basic ideas about DAEs. The general numerical approaches are briefly outlined in Section 3. Section 4 discusses reuse of Jacobians. Section 5 examines a specialized RQ algorithm.

2 Higher Index DAEs

Suppose that the DAE (1) is a system of n equations in the $(2n + 1)$-dimensional variable (x', x, t) and that $f_{x'}$ is always singular. We also assume that f is sufficiently differentiable in the variables (x', x, t) so that all needed differentiations can be carried out.

Intuitively the DAE (1) is *solvable* in an open set $\Omega \subseteq \mathbf{R}^{2n+1}$ if the graphs $(x'(t), x(t), t)$ of the solutions x form a smooth $2m+1$ dimensional manifold in Ω called the solution manifold and solutions are uniquely determined by their value x_0 at any t_0 such that $(v_0, x_0, t_0) \in \Omega$. Alternatively, there is a function $\Theta(t, \lambda)$ such that $(\Theta_t(t, \lambda), \Theta(t, \lambda), t,)$ is a diffeomorphism from an $(m+1)$-dimensional connected open set into Ω and $\Theta(t, \lambda)$ is a solution for each value of λ. More precise definitions appear in [12,14,15,30]. Solvable DAEs are also sometimes called regular [31].

In general, the solution x of (1) is known to depend on derivatives of f. If (1) is differentiated k

times with respect to t, we get the $(k+1)n$ *derivative array equations* [12]

$$F_k(x', w, x, t) = \left[\begin{array}{c} f(x', x, t) \\ f_t(x', x, t) + f_x(x', x, t)x' + f_{x'}(x', x, t)x'' \\ \vdots \\ \dfrac{d^k}{dt^k}[f(x', x, t)] \end{array} \right] = 0\texttt{[gderiv]} \qquad (2)$$

where

$$w = [x^{(2)}, \ldots, x^{(k+1)}]\texttt{[wdef]} \qquad (3)$$

The *index* ν of the DAE (1) is often taken to be the least integer k for which (2) uniquely determines x' for consistent (x, t). If such a k exists, then x' is a function of just (x, t) (for consistent (x, t)) so that

$$v = g(x, t)\texttt{[ode]} \qquad (4)$$

For general unstructured DAEs, the index is actually a somewhat more subtle concept than this definition suggests [14]. The index given here is often called the *differentiation index* and denoted ν_d when more than one index is being considered.

3 The General Approach

The widespread occurrence of DAEs has lead to an intensive examination of the numerical solution of DAE initial value problems. Several numerical methods have been developed. Backward differentiation (BDF) methods have been carefully examined. They are most useful for index one and some index two problems. Implicit Runge Kutta (IRK) methods have been analyzed for systems in Hessenberg form. Finally, specialized methods have been developed for certain DAEs with a special structure such as in constrained mechanics. These methods are discussed in [7,8,23] and some of the cited literature. Here we briefly summarize a more general approach. We shall assume that the DAE is a solvable DAE in a moderate number of variables and that formulas are known for the equations making up the DAE. This requirement can be relaxed to the existence of computer subroutines that return the values of f given values of x', x, t.

The matrix C of the equation $Cx = b$ is said to be *1-full* with respect to x_1 if there is a nonsingular matrix Θ such that

$$\Theta C = \left[\begin{array}{cc} I & 0 \\ 0 & M \end{array} \right], \quad x = \left[\begin{array}{c} x_1 \\ x_2 \end{array} \right]$$

We assume throughout this paper that there is an integer k such that the following assumptions hold.

(A1) Sufficient smoothness of F_k.

(A2) Consistency of $G = F_k = 0$ as an algebraic equation.

(A3) $\overline{J}_k = [\ G_{x'} \quad G_w\ ]$ is 1-full with respect to x' and has constant rank independent of (x', w, x, t).

(A4) $J_k = [\ G_{x'} \quad G_w \quad G_x\ ]$ has full row rank independent of (x', w, x, t).

These assumptions permit a robust numerical least squares solution of the derivative array equations. Let ν be the least value of k for which (A1)–(A4) hold. That is, ν is the *uniform differentiation index* ν_{UD} [14]. For Hessenberg systems, $\nu_d = \nu_{UD}$, but in general ν_{UD} is the maximum of ν_d over a set of perturbations.

Note that (A1)–(A4) are directly in terms of the original equations and their derivatives and do not require any sort of coordinate changes. Also (A3) and (A4) hold in a full neighborhood since $\{x', x, w\}$ are considered to be independent variables. Conditions (A1)–(A4) frequently hold in practice and are numerically verifiable using a combination of symbolic and numeric software [15]. They are almost equivalent to a type of uniform solvability [14].

3.1 Explicit Integration

Suppose that we have a value (x, t) which is *close* to a consistent value. Holding (x, t) fixed, the derivative array $F_k(v, w, x, t) = 0$ becomes a possibly inconsistent system. However, (A3) says that its Jacobian has constant rank.

Let $H(v, w) = F_k(v, w, x, t)$ for a given (x, t). Given an initial guess (v_0, w_0), we can solve $H = 0$ for (v, w) using a Gauss-Newton iteration

$$[v_{n+1}, w_{n+1}] = [v_n, w_n] - \rho_n [H_v(v_n, w_n), H_w(v_n, w_n)]^\dagger H(v_n, w_n) \, \texttt{[eq22]} \tag{5}$$

where $A^\dagger b$ is the minimum norm least squares solution of $Az = b$ [16].

It is shown in [9,12] that, under our assumptions, (5) converges to a limit (v^*, w^*). This limit satisfies the *least squares equation (LSE)*

$$[H_v(v^*, w^*), H_w(v^*, w^*)]^T H(v^*, w^*) = 0 \, \texttt{[eq55]} \tag{6}$$

Note that the LSE is not equivalent to $F_k = 0$ but has additional nonunique solutions. However, v^* defined by (6) depends only on (x, t). Thus $v^* = \overline{h}(x, t)$ defines a smooth *completion* [12].

$$x' = \overline{h}(x, t) \, \texttt{[eq56]} \tag{7}$$

That is, (6) defines an ODE whose solutions include those of the DAE. This ODE can be integrated by standard ODE integrators provided $\overline{h}$ is smooth enough. We shall refer to this procedure as the *explicit approach*. It has the additional nice features that the completion is unique as long as sufficient differentiation is done and the completion is defined on all of $\mathcal{I}$. Neither of these are true for completions defined by other methods, such as using subsets of the derivative array equations [11].

A detailed discussion of some of the technical issues with this approach can be found in [9,12,17,20]. The computational effort in carrying out this approach centers on evaluating G, evaluating the Jacobians, and carrying out the linear algebra in the least squares solves.

Because the Jacobians have both column and row nullity we have chosen not to use differencing. We have investigated two approaches for computing G and the Jacobians.

One is to form the Jacobians symbolically in a language such as MAPLE V and then have MAPLE convert this symbolic expression to optimized FORTRAN code. Programs can be written which require only the original DAE with all subsequent work being done automatically [17]. While the MAPLE portion of this calculation can be time consuming, it need only be done once for a given set of equations since MAPLE will pass any parameters to the FORTRAN subroutine. All subsequent simulations can be done using only the FORTRAN code.

We have also investigated the computation of the Jacobians and G using the automatic differentiation code ADOL-C [20]. This approach is preferable for large problems. Automatic differentiation also tends to use substantially less memory on larger problems. Even for moderate sized nonlinear DAEs, the use of automatic differentiation reduces the cost of Jacobian evaluation so much that the least squares solution is now the dominate cost.

3.2 Implicit Coordinate Partitioning

It is often desirable to preserve constraints. As shown in [17] one cannot just combine the explicit integrator and the obvious projection to stay on the solution manifold if the problem is unstructured. An alternative is developed in [18]. There subsets of the state variables are used to set up a local set of coordinates for the solution manifold. Within this local set of coordinates, an explicit integrator is used. This is similar in principle to the generalized coordinate partitioning used in mechanics [33]. However, a fundamental difference is that in our setting we do not assume that any of the constraints are known explicitly. Thus the coordinates must be picked in an implicit manner. Also we do not assume that the equations have any specific structure such as being Euler-Lagrange equations. A brief outline of this approach is the following.

Let $G = 0$ be the derivative array equations (2). As described in [18] we take a partition of the state variables

$$x = (x_1, x_2)\,\texttt{[eq400]} \tag{8}$$

so that

$$\widetilde{J} = [G_v \; G_w \; G_{x_1}]\,\texttt{[imacs2]} \tag{9}$$

has full row rank. The variables x_2 of (8) are used to parameterize the solution manifold locally.

Define the variables $\widetilde{z}, \omega$ by

$$\widetilde{z} = ([x', w], x_1) = (\omega, x_1)\,\texttt{[eq401]} \tag{10}$$

Thus $\widetilde{z}$ is everything but x_2 and ω is everything but x. Then $G_{\widetilde{z}}$ is not only 1-full with respect to x' but it is also full row rank by the choice of x_1. Thus

$$G_{\widetilde{z}}(\widetilde{z}, x_2, t)^T G(\widetilde{z}, x_2, t) = 0\,\texttt{[eq402]} \tag{11}$$

is equivalent to $G(\widetilde{z}, x_2, t) = 0$. Unlike with the explicit approach, we may now use any method of finding $\widetilde{z}$ given (x_2, t) which minimizes $C(\widetilde{z}) = \frac{1}{2}G(\widetilde{z}, x_2, t)^T G(\widetilde{z}, x_2, t)$. We have chosen to use a Gauss-Newton iteration

$$\widetilde{z}^{[m+1]} = \widetilde{z}^{[m]} - \rho_m G_{\widetilde{z}}(\widetilde{z}^{[m]}, x_2, t)^{\dagger} G(\widetilde{z}^{[m]}, x_2, t)\,\texttt{[eq404]} \tag{12}$$

Suppose then that we have a point $(\widetilde{z}_0, x_{20}, t_0)$ where $G(\widetilde{z}_0, x_{20}, t_0) = 0$ and our assumptions hold. Then by [18] there is a partitioning such that $\widetilde{z} = (x', [\xi, \eta], [x_1, x_2])$ and open neighborhoods

$\widetilde{\mathcal{N}}, \overline{\mathcal{N}}$ such that within these neighborhoods we get that the limit $({x'_1}^*, {x'_2}^*, w^*, x_1^*)$ of (12) satisfies the fundamental equations [eq406]

$$\begin{aligned} {x'_1}^* &= f_1(x_2, t)\,\texttt{[eq406a]} && (13a) \\ {x'_2}^* &= f_2(x_2, t)\,\texttt{[eq406b]} && (13b) \\ x_1^* &= h(x_2, t)\,\texttt{[eq406c]} && (13c) \end{aligned}$$

where f_1, f_2, h are defined by the limit of the iteration.

One step of the integration of the DAE now goes as follows. Given $x_{n-1} = (x_{1,n-1}, x_{2,n-1})$ we apply a multistep integrator to (13a)–(13b) to get $\widehat{x}_n = (\widehat{x}_{1,n}, \widehat{x}_{2,n})$. A final function evaluation gives $x'_n = (x'_{1,n}, x'_{2,n})$ and $\overline{x}_{1,n} = h(\widehat{x}_{2,n}, t_n)$. Then the value for x_n is taken to be $(\overline{x}_{1,n}, \widehat{x}_{2,n})$. Thus x_n lies on the solution manifold and satisfies all constraints. The feasibility of this approach has been developed in [17,27]. This approach will be referred to here as the *coordinate partitioning approach.*

4 Reuse of Jacobians

As shown in [17,20] the evaluation of J and G is cheap in comparison with the cost of the linear algebra. One natural way to speed up the computation is to reuse the Jacobians.

For the coordinate partitioning method, the reuse of the Jacobians in (12) would lead to

$$\widetilde{z}^{[m+1]} = \widetilde{z}^{[m]} - \rho_m G_{\widetilde{z}}(\widetilde{z}^0, \widehat{x}_2, \widehat{t})^\dagger G(\widetilde{z}^{[m]}, x_2, t)\,\texttt{[eq4041]} \tag{14}$$

where $\widehat{x}_2 \neq x_2, \widehat{t} \neq t$ if the Jacobian is reused for several time steps. Convergence of (14) can be proved in a straightforward manner. The limit $\widetilde{z}^*$ of (14) satisfies

$$G_{\widetilde{z}}(\widetilde{z}^0, x_2, t)^T G(\widetilde{z}^*, x_2, t) = 0\,\texttt{[eq4042]} \tag{15}$$

Since the matrix in (15) has full column rank we get that

$$G(\widetilde{z}^*, x_2, t) = 0\,\texttt{[eq4043]} \tag{16}$$

and the parts of the solution that are used are the same as if the Jacobian had been updated each iteration. Thus Jacobians can be reused with the coordinate partitioning approach as long as there is satisfactory convergence of the iterations (14).

However, the situation is more complex with the explicit approach. Suppose (v_0, w_0) is close to (v^*, w^*). The simplest modification of (5) would be

$$[v_{i+1}, w_{i+1}] = [v_i, w_i] - [H_v(v_0, w_0), H_w(v_0, w_0)]^\dagger H(v_i, w_i)\,\texttt{[it-1]} \tag{17}$$

There are two issues concerning (17) that must be examined. One is convergence of the iteration. The other is the dependence of v^* on (v_0, w_0) and possibly (t_0, x_0) if the Jacobian is reused for several time steps. A key result is the following theorem from [4]. We let $\mathcal{R}, \mathcal{N}$ denote the range and nullspace of a matrix.

Theorem 1 [ben-thm] *Let F be a function, $F: E^n \to E^m$, y_0 a vector in E^n, M a real positive number, such that:*

$$\texttt{[cond1]} F \in C'(S(y_0, \delta)), \; S(y_0, \delta) = \{y : \; \|y - y_0\| < \delta\} \tag{18}$$

$$\texttt{[cond2]} M \; \|J^\dagger(y_0)\| < 1 \tag{19}$$

$$\texttt{[cond3]} \|J^\dagger(y_0)\| \; \|F(y_0)\| < (1 - M\|J^\dagger(y_0)\|) \; \delta \tag{20}$$

Then the sequence

$$\texttt{[seq-1]} y_{i+1} = y_i - J^\dagger(y_0)F(y_i) \quad i = 0, 1, 2, \cdots \tag{21}$$

converges to the unique solution of $J^T(y_0)F(y) = 0$ which lies in $S(y_0, \delta) \cap \{y_0 + \mathcal{R}(J^T(y_0)\}$, where $\mathcal{R}(A)$ is the range space of A. Moreover,

$$\texttt{[y-n]} \|y_{i+1} - y_i\| \leq k_0^i(1 - k_0)\delta, \; \textit{where } k_0 = M\|J^\dagger(y_0)\| \tag{22}$$

We now prove the following local result:

Proposition 1 [p-1] *Suppose that for a given (t, x) we have (v_0, w_0) is sufficiently near the solution manifold of* (2), *and* (A1)-(A4) *hold. Then*

(i) *The sequence (v_i, w_i) defined by* (17) *converges to the solution (v^*, w^*) of*

$$J^T(v_0, w_0)H(v, w) = 0 \texttt{[neweq1]} \tag{23}$$

(ii) *(v^*, w^*) is continuous in x and t.*

Proof. To simplify notation, let

$$y = [v, w], \quad R(y) = H(v, w), \quad J(y) = R_y(y).$$

We shall verify that (18), (19), (20) are satisfied here. Let $S(y_0, \delta) = \{y : \; \|y - y_0\| < \delta\}$, $B(x_k, \delta_x) = \{x : \; \|x - x_k\| < \delta_x\}$, and $I(t_k, \delta_t) = \{t : \; |t - t_k| < \delta_t\}$. Suppose that $y \in S(y_0, \delta)$, $x \in B(x_k, \delta_x)$, and $t \in I(t_k, \delta_t)$. By assumption (A3), we have that in $S(y_0, \delta)$, $B(x_k, \delta_x)$, and $I(t_k, \delta_t)$, $J^\dagger$ is continuously differentiable. By continuity of J there is a c_1 so that

$$\texttt{[J-lips]} \|J(y) - J(y_0)\| \leq c_1\|y - y_0\| \tag{24}$$

Thus by the notation used in [4], we have $R(y) \in C'(S(y_0, \delta))$. Also note that on the sets $S(y_0, \delta)$, $B(x_k, \delta_x)$, and $I(t_k, \delta_t)$, we have

$$\|J^\dagger(y)\| \leq c_2, \quad \|R(y)\| \leq \epsilon \tag{25}$$

Let M be a real positive number, such that

$$\texttt{[Mc-2]} Mc_2 < 1 \tag{26}$$

By taking y_0 close to the solution manifold of (2) which is $R(y) = 0$, we can make ϵ small enough to insure that

$$\texttt{[J0-r0]}\,\|J^\dagger(y_0)\|\ \|R(y_0)\| < (1 - M\|J^\dagger(y_0)\|)\ \delta \tag{27}$$

Then by Theorem 1, y_i converges to $y^* \in S(y_0, \delta) \cap \{y_0 + \mathcal{R}(J^T(y_0)\}$, and y^* satisfies

$$\texttt{[lse-1]}\,J^T(y_0)R(y^*) = 0 \tag{28}$$

To see (ii), we first define a function by treating x and t as parameters

$$\texttt{[g-xt]}\,g_{x,t}(y) = y - J^\dagger(y_0)R(y) \tag{29}$$

From [4], for $y \in S(y_0, \delta),\ x \in B(x_k, \delta_x),\ t \in I(t_k, \delta_t)$, we have

$$\texttt{[contract]}\,\|g_{x,t}(y_1) - g_{x,t}(y_2)\| \le M\|J^\dagger(y_0)\|\ \|y_1 - y_2\| \le Mc_2\|y_1 - y_2\| \tag{30}$$

Since Mc_2 is independent of x and t, it follows that $g_{x,t}(y)$ is an uniform contraction mapping. Rewriting (17) as

$$\texttt{[seq-gyt]}\,y_1 = g_{x,t}(y_0),\ y_{i+1} = g_{x,t}(y_i) \tag{31}$$

we have (17) converges to y^* and y^* is continuous in x and t by the Contraction Mapping Theorem. $\square$

With these preliminary results we now turn to examining the reuse of Jacobians.

Lemma 1 `[l-1]` *Let $\Phi(y) = J^T(y_0)R(y)$, $J(y)$ be 1-full with constant rank for all y. Then there exists $\delta_1 > 0$, such that in $S(y_0, \delta_1)$, we have $\Phi_y(y)$ is 1-full with constant rank.*

Proof. When $y = y_0$, we have $\Phi_y(y_0) = J^T(y_0)J(y_0)$. Then $\text{rank}(\Phi_y(y_0)) = \text{rank}(J^T(y_0)) = \text{rank}(J^\dagger(y_0)) = p$ where p is a integer constant, and $\mathcal{N}(\Phi_y(y_0)) = \mathcal{N}(J(y_0))$. Thus $\Phi_y(y_0)$ is 1-full. Since $J^T(y_0)J(y_0)$ has rank p, there exists a nonsingular $p \times p$ submatrix $M(y_0)$ of $J^T(y_0)J(y_0)$. Let $M(y)$ be the $p \times p$ submatrix of $\Phi_y(y) = J^T(y_0)J(y)$ obtained by deleting the same rows and columns of $J^T(y_0)J(y)$ as done for $M(y_0)$. By the continuity of invertability of a matrix, there exists $\delta_1 > 0$, such that the matrix $M(y)$ is invertible in $S(y_0, \delta_1)$. Thus $\text{rank}(J^T(y_0)J(y)) \ge p$. On the other hand, we have $\text{rank}(J^T(y_0)J(y)) \le \text{rank}(J^T(y_0)) = p$. Thus $\text{rank}(\Phi_y(y)) = \text{rank}(J^T(y_0)J(y)) = p$ in $S(y_0, \delta_1)$. The 1-fullness of $\Phi_y(y)$ follows immediately from the 1-fullness of $J(y)$. $\square$

Proposition 2 `[p-2]` *If the conditions in Proposition 1 hold for a fixed (v_0, w_0), then v^* in (23) is uniquely determined by (x, t).*

Proof. Rewrite equation (28) as

$$\texttt{[lse-2]}\,\Phi(y) = J^T(y_0)R(y) = 0 \tag{32}$$

Then the Jacobian of $\Phi(y)$ is

$$\texttt{[lse-Ja]}\,\Phi_y(y) = J^T(y_0)J(y) \tag{33}$$

By Lemma 1, there exists a set $S(y_0, \delta_1)$, where δ_1 is defined as in Proposition 1 such that $\Phi_y(y)$ is 1-full with constant rank in S_1.

Now let $\delta_{\min} = \min(\delta, \delta_1)$, where δ is defined as in Proposition 1. Then we see that the proof in Proposition 1 is still valid if we use the set $S(y_0, \delta_{\min})$ instead of $S(y_0, \delta)$. By the 1-fullness of $\Phi_y(y)$ in $S(y_0, \delta_{\min})$,

$$\mathcal{N}(J^T(y_0)J(y^*)) = \mathcal{N}(J(y)) \subset \left\{ \begin{bmatrix} v \\ 0 \end{bmatrix} \right\}^{\perp}$$

Recall that $y = [v, w]$, the 1-fullness and constant rank of $\Phi_y(y)$ means that $[R_v \ R_{w_1}]$ has full column rank by permuting w, where $w = [w_1, w_2]$. Thus we can solve for v and w_1

$$\begin{aligned} v &= \widetilde{h_1}(w_2, x, t) \\ w_1 &= \widetilde{h_2}(w_2, x, t) \end{aligned}$$

Then we have:

$$J^T(v_0, w_0)R(\widetilde{h_1}(w_2, x, t), \widetilde{h_2}(w_2, x, t), w_2, x, t) = 0.\texttt{[lse-in-yt]} \tag{34}$$

Differentiating (34) with respect to w_2 yields

$$J^T(v_0, w_{10}, w_{20})(R_v \widetilde{h_1}_{w_2} + R_{w_1}\widetilde{h_2}_{w_2} + R_{w_2}) = 0.$$

This is

$$J^T(v_0, w_{10}, w_{20})[R_v \ R_{w1} \ R_{w_2}] \begin{bmatrix} \widetilde{h_1}_{w_2} \\ \widetilde{h_2}_{w_2} \\ I \end{bmatrix} = 0.\texttt{[lse-diff]} \tag{35}$$

Note that

$$J^T(v_0, w_{10}, w_{20})[R_v \ R_{w1} \ R_{w_2}] = J^T(v_0, w_{10}, w_{20})J(v, w_1, w_2)$$

is 1-full, (35) implies $\widetilde{h_1}_{w_2} = 0$ Thus v^* is uniquely determined. □

Note that Proposition 2 is a local result.

4.1 Reuse of Jacobian During One Time Step

The simplest situation is when v^* does not depend on (v_0, w_0) since then we get a smooth completion. We introduce one more assumption

(A5) $\mathcal{R}([G_{x'} \ G_w])$ depends only on t and x.

Assume for this subsection that (A1)-(A5) hold and that we update the Jacobian at every time step. Then there exists an orthogonal matrix $U(x, t)$ such that

$$U(x, t)[G_{x'} \ G_w] = \begin{bmatrix} K_1 & K_2 \\ 0 & 0 \end{bmatrix}$$

where $[K_1 \ K_2]$ has full row rank. As showed in [10], we may assume without loss of generality that

$$[G_{x'} \ G_w] = \begin{bmatrix} K_1 & K_2 \\ 0 & 0 \end{bmatrix} \texttt{[U-jac]} \tag{36}$$

As before, let $v = x'$. Then we can rewrite the derivative array equations (2) as `[dae-range]`

$$G = \begin{array}{ll} G_1(v, w_1, w_2, x, t) & (37a) \\ G_2(v, w_1, w_2, x, t) & (37b) \\ G_3(x, t) & (37c) \end{array}$$

where G_1 has $\dim(v)$ equations. We can permute w so that

$$\begin{bmatrix} G_{1v} & G_{1w_1} \\ G_{2v} & G_{2w_1} \end{bmatrix} \texttt{[nons-jac]} \tag{38}$$

is nonsingular. Then the least squares equation $[G_v \ G_w]^T(v_0, w_0)G(v, w) = 0$ is

$$\begin{bmatrix} G_{1v} & G_{1w_1} & G_{1w_2} \\ G_{2v} & G_{2w_1} & G_{2w_2} \\ 0 & 0 & 0 \end{bmatrix}^T_{(v_0, w_0)} \begin{bmatrix} G_1 \\ G_2 \\ G_3 \end{bmatrix} = 0. \texttt{[lse-range]} \tag{39}$$

By (37) and (38), we get that (39) is equivalent to

$$\begin{bmatrix} G_1 \\ G_2 \end{bmatrix} = 0 \texttt{[lse-G1G2]} \tag{40}$$

The implicit function theorem and the nonsingularity of (38) imply that (39), or (40), has a solution `[compl-range]`

$$\begin{array}{rcll} v & = & \phi_1(w_2, x, t) & (41a) \\ w_1 & = & \phi_2(w_2, x, t) & (41b) \end{array}$$

with w_2, x, t arbitrary in a neighborhood. Then we have `[lse-sol]`

$$\begin{array}{rcll} G_1(\phi_1, \phi_2, w_2, x, t) & = & 0 & (42a) \\ G_2(\phi_1, \phi_2, w_2, x, t) & = & 0 & (42b) \end{array}$$

Differentiating (42) with respect to w_2 yields

$$\texttt{[diff-lse-sol]} \begin{bmatrix} G_{1v} & G_{1w_1} \\ G_{2v} & G_{2w_1} \end{bmatrix} \begin{bmatrix} \phi_{1w_2} \\ \phi_{2w_2} \end{bmatrix} + \begin{bmatrix} G_{1w_2} \\ G_{2w_2} \end{bmatrix} = 0 \tag{43}$$

The 1-fullness implies that

$$\begin{bmatrix} G_{1v} & G_{1w_1} \\ G_{2v} & G_{2w_1} \end{bmatrix}^{-1} \begin{bmatrix} G_{1w_2} \\ G_{2w_2} \end{bmatrix} = \begin{bmatrix} 0 \\ * \end{bmatrix} \tag{44}$$

Then we have $\phi_{1w_2} = 0$. Thus $v = \phi_1(x, t)$ and ϕ_1 is as smooth in (x, t) as desired.

Theorem 2 [compl-thm] *Suppose that (A1)-(A5) hold on a finite t interval. Then the least squares iteration (17) on the derivative array (2), with updating at every time step of the integrator, produces a completion*

$$x' = \phi(x,t)\texttt{[compl-reuse-j]} \tag{45}$$

which is as smooth in (x,t) as G is in (x', w, x, t). The completion is defined on a neighborhood of the solution manifold. It is the same as the completion produced by (5).

Proof. Observe that using the same process of producing (40) from (39), we can show that $[G_v\ G_w]^T G(v,w) = 0$, is still equivalent to (40). Note that ϕ is uniquely defined by (40) independent of the actual partitions of w used in (42). Then the result follows. $\Box$

Assumption (A5) holds for index 1 systems, Hessenberg systems of index 2, and some Hessenberg index 3 systems. Examples show that (A5) is not necessary. Example 1 shows that if (A5) does not hold, then v^* can depend on (v_0, w_0) even if the Jacobian is updated at every time step.

4.2 Reuse of Jacobian for More Than One Time Step

In the previous section, we obtained convergence results for reusing the Jacobian within one time step. From a computational point of view, we wish to update the Jacobian as rarely as possible. Proposition 1 actually tells us that it is possible to keep the Jacobian for several time steps and still have the iteration converge if the time steps of the ODE integrator are small enough. Let the sets $S(x_0, \delta)$, $B(y_k, \delta_y)$, and $I(t_k, \delta_t)$ be the same as before.

Proposition 3 [p-4] *Assume that the conditions in Proposition 1 hold at (t_k, x_k). Suppose that the step size is chosen such that $t_i \in I(t_k, \delta_t)$, $x_i \in B(x_k, \delta_x)$, the starting value y_0^i is close to the solution manifold of $R(y) = 0$ at t_i, and there exists δ_i such that $S(y_0^i, \delta_i) \subset S(y_0, \delta)$ for $i = k+1, \cdots, k+l$. Then the same Jacobian can be used for l time steps.*

Proof. Let $\delta_{\min} = \min(\delta_i, \delta_1)$. The proof for Proposition 1 is valid as long as $t_i \in I(t_k, \delta_t)$, $x_i \in B(x_k, \delta_x)$, and $S(y_0^i, \delta_i) \subset S(y_0, \delta)$ at time t_i. $\Box$

While the iterates may converge with reuse of the Jacobian for several time steps, we will usually have to deal with dependence on the starting values and possibly also on x_0 and t_0, the time at which the Jacobian is evaluated. This is illustrated by the Examples in Section 4.4. The various possibilities are summarized in the next table.

	Dependency of v^* on Starting Values	
Conditions	Reuse During One Time Step	Reuse for Several Time Steps
(A1)-(A4)	v_0, w_0	v_0, w_0, x_0, t_0
(A1)-(A5)	none	x_0, t_0

The completions for reusing Jacobians

4.3 Local error control

In general, if we want to use the same Jacobian in more than one time step, as we have seen in the previous sections, the completion may depend on the starting values $x_0^k,\ t_0^k,\ v_0^k,\ w_0^k$ for some k. That is,

$$x' = h(x, t, x_0^k, t_0^k, v_0^k, w_0^k).\texttt{[compl-reuse]} \tag{46}$$

If we need to use a new Jacobian for the iteration, then we will have different completions during the integration. In this section, we will discuss the local error control strategies for the transition between local completions.

Note that in (46), we use particular values of x, t, v, w as parameters. Rewrite (46) as

$$x' = h(x, t, \lambda), \texttt{[compl-para]} \tag{47}$$

where $\lambda = \{y_0^k,\ t_0^k,\ v_0^k,\ w_0^k\}$. Now suppose that we are going to integrate the DAE (1) on the interval $I = [t_0\ \ t_{end}]$. By compactness, the interval I can be covered by a finite subcover $\bigcup\limits_{\alpha} I_\alpha$. Recall that in the proof of Propositions 1 and 2, given any time t_k and x_k, on the set

$$S(x_0, \delta_{\min}) = \{x :\ \|x - x_0\| < \delta_{\min}\}$$

$$B(y_k, \delta_{y_k}) = \{y :\ \|y - y_k\| < \delta_{y_k}\}$$
$$I(t_k, \delta_{t_k}) = \{t :\ |t - t_k| < \delta_{t_k}\}$$

the iteration (17) converges. The compactness means that one can pick a finite number of t_k such that $\bigcup\limits_{t_k} I(t_k, \delta_{t_k})$ covers I. Let $\delta_I = \min_{t_k}\ (\delta_{t_k})$, then we have a finite cover $\bigcup\limits_{\alpha} I_\alpha(\delta_I)$ for I. On each $I_\alpha(\delta_I)$, along with the corresponding sets S and B, the iteration will converge. We now see this cover is independent of time t_k. In summary, we have the following:

Proposition 4 `[p-44]` *The Jacobian in iteration (17) can be used for at least δ_I time.*

There remains the problem of how to deal with the dependence of v^* on values of x_0, t_0, v_0, w_0. A careful analysis will be given elsewhere. However, we wish to point out intuitively why the dependence can be dealt with. For convenience, suppose that we are working on an interval of length δ_I and from (46) we have that $v^* = h(x, t, \lambda)$. Let $\overline{x}$ be the solution to the DAE of interest. Note that $\overline{x}' = h(\overline{x}, \lambda, t)$ since $\overline{x}$ and its derivatives satisfy the derivative array equations. If $\overline{h}$ is the least squares completion we have that $h(\overline{x}, \lambda, t) = \overline{h}(\overline{x}, t)$. Then, assuming smoothness of h in λ, x we get $h(x, t, \lambda) = \overline{h}(\overline{x}, t) + O(\|\overline{x} - x\|)$. Now suppose that we are performing a numerical integration on this interval and at time t_k we have an estimate x_k for $\overline{x}(t_k)$ that is $O(h^r)$ accurate and we are using an rth order integrator. Then the above shows that we will get an rth order estimate for $\overline{x}'(t_k)$ and hence an rth order estimate for $\overline{x}(t_{k+1})$ with a local error of $O(h^{r+1})$. Thus, in principle, we can still perform an r order integration.

4.4 Examples

To illustrate the results of the previous sections, we discuss two elementary examples.

Example 1 [example1] The DAE

$$\begin{bmatrix} 0 & yy' \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}' + \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} f(t) \\ g(t) \end{bmatrix} \texttt{[ex1]} \tag{48}$$

is solvable and has the unique solution given by

$$\begin{aligned} x &= f(t) - g(t)(g'(t))^2 \\ y &= g(t) \end{aligned}$$

Differentiating (48) twice, we get the derivative array equations

$$\begin{aligned} y(y')^2 + x - f &= 0 \\ y - g &= 0 \\ (y')^3 + 2yy'y'' + x' - f' &= 0 \\ y' - g' &= 0 \\ 5(y')^2y'' + 2y(y'')^2 + 2yy'y''' + x'' - f'' &= 0 \\ y'' - g'' &= 0 \end{aligned}$$

The Jacobian

$$[G_v \quad G_w] = \begin{bmatrix} 0 & 2yy' & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 3(y')^2 + 2y'y'' & 0 & 2yy' & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 10y'y'' + yy''' & 1 & 5(y')^2 + 4yy'' & 0 & 2yy' \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Note that the range of the Jacobian depends on y and y' so that condition (A5) does not hold. For the iteration (5), we have the least squares equation $[G_v \quad G_w]^T \, G(v, w) = 0$. After some simplifications, the least squares equations are

$$5(y')^2y'' + 2y(y'')^2 + 2yy'y''' + x'' - f'' = 0 \texttt{[ex1-lse1]} \tag{49a}$$
$$(y')^3 + 2yy'y'' + x' - f' = 0 \texttt{[ex1-lse2]} \tag{49b}$$
$$y'' - g'' = 0 \texttt{[ex1-lse3]} \tag{49c}$$
$$2y^2(y')^3 + (2xy - 2yf + 1)y' - g' = 0 \texttt{[ex1-lse4]} \tag{49d}$$

Note that (49d) is a third order equation in y', so there is always at least one real solution. Suppose we have $y' = \phi(x, y, f, g')$. Then from the other equations, we get the completion

$$y' = \phi(x, y, f, g') \texttt{[ex1-comp1]} \tag{50a}$$
$$x' = f' - \phi^3 - 2y\phi g'' \texttt{[ex1-comp2]} \tag{50b}$$

Now for the iteration (17), we have the least square equation

$$\texttt{[lse-ex1]}[G_v \quad G_w]^T_{(v_0,w_0)}\, G(v,w) = 0 \tag{51}$$

Or equivalently,

$$\begin{aligned}
5(y')^2y'' + 2y(y'')^2 + 2yy'y''' + x'' - f'' &= 0\texttt{[ex1-lse11]} && (52a)\\
(y')^3 + 2yy'y'' + x' - f' &= 0\texttt{[ex1-lse12]} && (52b)\\
y'' - g'' &= 0\texttt{[ex1-lse13]} && (52c)\\
2y_0'y^2(y')^2 + y' + (2y_0'yx - 2y_0'yf - g') &= 0\texttt{[ex1-lse14]} && (52d)
\end{aligned}$$

In (52b) and (52d), the Jacobian with respect to x' and y' is

$$\left[\begin{array}{cc} 1 & 3(y')^2 + 2yy'' \\ 0 & 4y^2y_0'y' + 1 \end{array}\right],$$

which is nonsingular if the starting values are close to the manifold $G = 0$. Then we have

$$\begin{aligned}
y' &= \psi(x, y, f, g', y_0')\texttt{[ex1-comp11]} && (53a)\\
x' &= f' - \psi^3 - 2y\psi g''\texttt{[ex1-comp12]} && (53b)
\end{aligned}$$

The example shows that if we reuse the same Jacobian during one time step, the completion can be dependent on starting values (v_0, w_0) if (A5) does not hold.

Example 2 `[example2]` The DAE

$$\left[\begin{array}{cc} 0 & y \\ 0 & 0 \end{array}\right]\left[\begin{array}{c} x \\ y \end{array}\right]' + \left[\begin{array}{c} x \\ y \end{array}\right] = \left[\begin{array}{c} f(t) \\ g(t) \end{array}\right]\texttt{[ex2]} \tag{54}$$

is solvable and has the unique solution given by

$$\begin{aligned}
x &= f(t) - g(t)g'(t)\\
y &= g(t)
\end{aligned}$$

Differentiating (48) twice, we get the derivative array equations

$$\begin{aligned}
yy' + x - f &= 0\\
y - g &= 0\\
(y')^2 + yy'' + x' - f' &= 0\\
y' - g' &= 0\\
3y'y'' + yy''' + x'' - f'' &= 0\\
y'' - g'' &= 0
\end{aligned}$$

The Jacobian is

$$[G_v \quad G_w] = \begin{bmatrix} 0 & y & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2y' & 0 & y & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 3y'' & 1 & 3y' & 0 & y \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Note that the Range space of the Jacobian depends only on y so that (A5) holds. For the iteration (17), if the Jacobian is reused for several time steps, the least square equation is

$$\texttt{[lse-ex4]}[G_v \quad G_w]^T_{(v_0,w_0,x_0,y_0,t_0)}\ G(v,w) = 0 \tag{55}$$

Or equivalently,

$$\begin{aligned} 3y'y'' + yy''' + x'' - f'' &= 0\texttt{[ex4-lse41]} && (56a) \\ (y')^2 + y'y'' + x' - f' &= 0\texttt{[ex4-lse42]} && (56b) \\ y'' - g'' &= 0\texttt{[ex4-lse43]} && (56c) \\ y_0yy' + y_0x - y_0f + y' - g' &= 0\texttt{[ex4-lse44]} && (56d) \end{aligned}$$

The completion is

$$\begin{aligned} y' &= \frac{g' - xy_0 + y_0f}{1 + y_0y}\texttt{[ex4-comp41]} && (57a) \\ x' &= f' - (y')^2 - yg''\texttt{[ex4-comp42]} && (57b) \end{aligned}$$

The example shows that if we reuse the same Jacobian during more than one time step, then the completion can be dependent on starting values (x_0, y_0, t_0) even if (A5) holds.

Example 3 [example3] Take the same DAE as in Example (1).

If we use the same Jacobian for more than one time step, the least square equation becomes

$$\texttt{[lse-ex3]}[G_v \quad G_w]^T_{(v_0,w_0,x_0,y_0,t_0)}\ G(v,w) = 0 \tag{58}$$

Or equivalently,

$$\begin{aligned} 5(y')^2y'' + 2y(y'')^2 + 2yy'y''' + x'' - f'' &= 0\texttt{[ex3-lse31]} && (59a) \\ (y')^3 + 2yy'y'' + x' - f' &= 0\texttt{[ex3-lse32]} && (59b) \\ y'' - g'' &= 0\texttt{[ex3-lse33]} && (59c) \\ 2y_0y_0'y(y')^2 + y' + (2y_0y_0'x - 2y_0y_0'f - g') &= 0\texttt{[ex3-lse34]} && (59d) \end{aligned}$$

In (59b) and (59d), the Jacobian with respect to x' and y' is

$$\begin{bmatrix} 1 & 3(y')^2 + 2yy'' \\ 0 & 4y_0yy_0'y' + 1 \end{bmatrix},$$

which is nonsingular if the starting values are close to the manifold $G = 0$. Then we have

$$\begin{aligned} y' &= \psi(x, y, f, g', y_0', y_0)\,\texttt{[ex3-comp31]} \qquad (60a) \\ x' &= f' - \psi^3 - 2y\psi g''\,\texttt{[ex3-comp32]} \qquad (60b) \end{aligned}$$

Thus if we use the same Jacobian for more than one time step, the completion can be dependent on starting values $(v_0, w_0, x_0, y_0, t_0)$.

5 The Linear Algebra

Not only is it desirable to perform the least squares solves less frequently but it is highly desirable to reduce the cost of performing them. In this section we focus on speeding up the linear algebra in the coordinate partitioning approach.

Recall that the Jacobian for the coordinate partitioning approach is

$$\widetilde{J} = \left[\begin{array}{ccc} G_{x'} & G_w & G_{x_1} \end{array} \right] = \left[\begin{array}{ccccccc} J_{00} & 0 & 0 & 0 & \cdots & 0 & X_0 \\ J_{10} & J_{11} & 0 & 0 & \cdots & 0 & X_1 \\ J_{20} & J_{21} & J_{22} & 0 & \cdots & 0 & X_2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ J_{r0} & J_{r1} & J_{r2} & J_{r3} & \cdots & J_{rr} & X_r \end{array} \right] \texttt{[j-matrix]} \qquad (61)$$

where the $J_{ij}, 0$ are $n \times n$ matrices and the X_i are $n \times m$ matrices for $i, j = 0, 1, 2, ..., r$, and $1 \leq m \leq n$. Also $G = F_r, r \geq \nu$.

The least squares solves take the form

$$\widetilde{J}z = F_r\,\texttt{[im5]} \qquad (62)$$

Normally one would consider performing a QR on (62). However, we have that $\widetilde{J}$ is full row rank and it is easier to exploit the structure in (61) by performing orthogonal column operations on $\widetilde{J}$ and compute a RQ decomposition. That is, we compute a factorization $\widetilde{J} = PRQ$ where Q is an orthogonal matrix, R is lowertriangular, and P is a permutation matrix. Then the solution of (62) is

$$z = (PRQ)^\dagger F_r = Q^T R^\dagger P^T F_r\,\texttt{[im6]} \qquad (63)$$

We shall see that R has a special structure which simplifies the computation of $R^\dagger$ over the QR case. In general we restrict the use of row permutations so as to not destroy the special block structure of $\widetilde{J}$ in (61). We will ignore P in what follows to simplify notation.

5.1 The RQ decomposition

We perform a special RQ decomposition as follows.

We begin by performing a RQ on the full row rank matrix C which is the first block row of (61). The Q is applied to all of $\widetilde{J}$. This is done exactly like performing the usual QR algorithm on the C^T. Note that this operation only affects the first and last block column of $\widetilde{J}$ in (61). We now have

$$\begin{bmatrix} \widetilde{J}_{00} & 0 & 0 & 0 & \cdots & 0 & 0 \\ \widetilde{J}_{10} & J_{11} & 0 & 0 & \cdots & 0 & \widetilde{X}_1 \\ \widetilde{J}_{20} & J_{21} & J_{22} & 0 & \cdots & 0 & \widetilde{X}_2 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \widetilde{J}_{r0} & J_{r1} & J_{r2} & J_{r3} & \cdots & J_{rr} & \widetilde{X}_r \end{bmatrix}, \texttt{[j-matrix2]} \tag{64}$$

where a tilde denotes a new entry and $\widetilde{J}_{00}$ is lower triangular. Continuing in this manner we get that

$$\widetilde{J}Q = [U\ 0]\,\texttt{[uq]}, \tag{65}$$

where U is lower triangular. Thus in (63) we have

$$R^{\dagger} = \begin{bmatrix} U^{-1} \\ 0 \end{bmatrix} \texttt{[im77]} \tag{66}$$

We examine the amount of work saved by this algorithm over performing a QR with no regard to the structure present. The algorithm given above works equally well when the structure of the DAE permits reduced differentiation of some of the equations. See [17] for example. However, we will only consider the fully implicit case here where all n equations are differentiated the same number of times r and the X_{ij} are $n \times m$. Using a standard QR type implementation of each block row step of the RQ algorithm we get that the total flop cost for computing the RQ decomposition of $\widetilde{J}$ is:

$$\texttt{[rqcost]}\frac{1}{6}(r+1)(3r+4)n^3 + (r+1)\left((r+1)m + \frac{1}{2}r + \frac{3}{2}\right)n^2 + (r+1)\left(2m + \frac{13}{6}\right)n \tag{67}$$

To get at better idea of these estimates we assumed that $m = n$ and computed (67) and the cost of a QR algorithm for different state dimensions n, and index ν. Table 1 shows how the values changed for different indices with $n = 10$.

For a given index the ratio QR/RQ was essentially independent of size as shown by Table 2. Of course, in practice only $n \geq \nu$ occurs but Table 2 is useful in showing the invariance of the ratio with respect to n. The tables show that the modified RQ leads to a substantial savings in computation.

5.2 Using RQ to check partitions

One of the key aspects of the coordinate partitioning approach is the monitoring of the partition so that it can be changed when necessary to avoid ill-conditioning caused by curvature of the solution manifold. If a standard QR as described in [18,27] is being done on (61), then the partition check is done with little extra computational effort. Let

$$Q\widetilde{J} = \begin{bmatrix} R_1 & X_1 \\ 0 & X_2 \end{bmatrix} \texttt{[im8]} \tag{68}$$

where R_1 has full row rank, the X_i are $n_i \times m$, and X_2 is supposed to be full row rank. The partition is changed when X_2 becomes less well conditioned. That is, when the singular values of X_2 become

Index	QR (10^6)	RQ (10^5)	$\frac{QR}{RQ}$
1	0.010150	0.07177	1.41
2	0.028675	0.15415	1.86
3	0.061500	0.26753	2.30
4	0.112625	0.41192	2.73
5	0.186050	0.58730	3.17
6	0.285775	0.79369	3.60
7	0.415800	1.03107	4.03
8	0.580125	1.29945	4.46
9	0.782750	1.59883	4.90
10	1.027675	1.92922	5.33

Table 1: The number of flops for $n = 10$.

[tab1]

Sizes(n)	QR (10^5)	RQ (10^5)	$\frac{QR}{RQ}$
3	0.05415	0.01848	2.93
5	0.24025	0.07840	3.06
7	0.64715	0.20720	3.12
9	1.36125	0.43128	3.16
11	2.46895	0.77704	3.18
13	4.05665	1.27088	3.19
15	6.21075	1.93920	3.20

Table 2: The number of flops for index 5

[tab2]

small. Now it is important to monitor the partition regularly. However, frequent use of the QR would undo the savings of the RQ algorithm. We would like to be able to use the RQ to monitor the partition and only compute the QR when it becomes necessary to compute a new partition.

If one assumes, as we do here, that R_1 remains well conditioned and constant rank, then X_2 can have a small singular value only if $\widetilde{J}$ has a small singular value. But this means that U in (65) must have a small singular value. This suggests two ways to monitor the partition based on the RQ factorization. Here U is a $(r+1)n \times (r+1)n$ lower triangular matrix.

Method 1: Let $U = [u_{ij}]$, $U^{-1} = [u_{ij}^{-}]$, $i, j = 1, 2, ..., (r+1)n$, and $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{(r+1)n}$ be the singular values of U. A lower bound for the condition number, in the 2-norm is [25]:

$$\texttt{[cond-low]} K = \frac{\sigma_1}{\sigma_{(r+1)n}} \geq \frac{\max_{ij} |u_{ij}|}{\min_i |u_{ii}|} \tag{69}$$

(69) tells us that if any of the diagonal elements u_{ii} are too small, then we need to choose a new partition. This check can be done with little extra cost since when we compute the RQ decomposition, we compute the diagonals first. Since $\|U\|_2 \leq \sqrt{(r+1)n}\ \|U\|_\infty$, we can also get an upper bound for K

$$\begin{aligned} K &= \|U\|_2\|U^{-1}\|_2 \\ &\leq (r+1)n\|U\|_\infty\|U^{-1}\|_\infty \\ &= (r+1)n\left(\max_{1\leq i\leq(r+1)n}\sum_{j=1}^{(r+1)n}|u_{ij}|\right)\left(\max_{1\leq i\leq(r+1)n}\sum_{j=1}^{(r+1)n}|u_{ij}^{-}|\right) \texttt{[cond-upper]} \qquad (70) \end{aligned}$$

Method 2: Apply the condition number estimator [21] to compute the condition number of U. This estimator requires $\mathrm{O}((r+1)^2n^2)$ operations.

Since Method 1 is considerably cheaper it will be desirable to use it as much as possible. However, it is known that Method one can often be a poor estimator of $\sigma\min$ [6]. To examine how accurate an estimator this is we have taken an index three fully implicit nonlinear DAE with $n = 5$ from [18]. This problem has a solution which has some sharp turns in some choices of coordinates at $t = m\pi/2.m = 1,2,3$. Experiments with the Jacobian updating strategies discussed here have been made but we wish to focus on the problem of partition monitoring. Three times were chosen where conditioning was a problem. Row pivoting was used during the RQ but only within block rows. Table 3 gives top and bottom entries in (69), the actual largest and smallest singular values of $\widetilde{J}$, the condition estimate from (69), the condition estimate `dtrco` from LINPACK and the actual condition number of $\widetilde{J}$.

time	$\min u_{ii}$	$\max u_{ij}$	$\sigma_{\min}$	$\sigma_{\max}$	Meth. 1.	Meth. 2.	Cond.
1.57	0.32	1784	0.00101	3643	5565	6.1×10^6	3.6×10^6
3.14	3.65	1223	0.00711	5515	335	9.3×10^5	7.7×10^5
4.71	0.96	1200	0.00304	2602	1250	11.6×10^6	8.6×10^5

Table 3: Condition Estimators

`[tab9]`

On this example $\max_{ij}|u_{ij}|$ does a reasonably good job of estimating $\sigma_{\max}$. This is to be expected since $\sigma_{\max}(U) \leq (((r+1)n)^{3/2}\max_{ij}|u_{ij}|$. But $\min_{ii}|u_{ii}|$ is a much less desirable estimate of $\sigma_{\min}$.

By our assumptions on f, we have that $\sigma_{\max}$ is bounded of moderate size. It may still be desirable to monitor $\max_{ij}|u_{ij}|$ in order to guarantee sufficient accuracy. If $\sigma_{\max}$ gets too small then there are several consequences. The iteration may require better starting values to converge, convergence of the iteration may be slowed down, and the solution may be found to lower accuracy.

This suggests the following strategy when doing implicit coordinate partitioning. The Jacobian is recomputed whenever the iteration slows down or a fixed number of time steps have been taken.

If a new Jacobian has been computed and convergence is slow or $\min_{ii} |u_{ii}|$ becomes too small we compute the estimate for the condition number by Method 2. If the estimate shows that the problem is becoming ill conditioned, then we recompute a new local set of coordinates. If the Jacobian is not poorly conditioned then the most likely difficulty is a poor initial guess and we reduce the stepsize of the numerical integrator to get a better prediction for the starting value.

6 Conclusion

Several computational aspects of a class of general integration methods for DAEs have been investigated. It has been shown that even in the explicit case that Jacobians can be reused for several time steps although one must then worry about the error control during the transition between local completions. A specialized *RQ* algorithm has been developed to speed up the linear algebra in the least squares solves. Its use in monitoring the partitions with the coordinate projection method has been discussed.

References

[1] U. Ascher and L. R. Petzold, *Projected implicit Runge-Kutta methods for the solution of index-two differential-algebraic systems*, SIAM J. Numer. Anal., 28 (1991), 1097-1120.

[2] A. Barrlund, *Constrained least squares methods, of implicit type, for the numerical solution to higher index linear variable coefficient differential algebraic systems*, Report UMINF-166.89, University of Umea, Institute of Information Processing, Umeå, Sweden, 1989.

[3] A. Barrlund, *Constrained least squares methods for non-linear differential algebraic systems*, Report UMINF-91.15, University of Umea, Institute of Information Processing, Umeå, Sweden, 1991.

[4] A. Ben-Israel, *A modified Newton-Raphson method for the solution of systems of equations*, Israel J. Math., 3 (1965), 94-98.

[5] A. Ben-Israel, *A Newton-Raphson method for the solution of systems of equations*, J. Math. Anal. Appl., 15 (1966), 243-252.

[6] Å. Björck, *Algorithms for linear least squares problems*, Computer Algorithms for Solving Linear Algebraic Equations, Edited by E. Spedicato, NATO ASI Series, Vol. F77, Springer-Verlag, Berlin, 1991, 57-92.

[7] K. E. Brenan, S. L. Campbell, and L. R. Petzold, *Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations*, Elsevier, 1989.

[8] G. D. Byrne and W. E. Schiesser (editors), *Recent Developments in Numerical Methods and Software for ODEs/DAEs/PDEs*, World Scientific, 1991.

[9] S. L. Campbell, *A computational method for general higher index singular systems of differential equations*, 1989 IMACS Transactions Scientific Computing, Vol. 1.2 (1989), 555-560.

[10] S. L. Campbell, *Least squares completions for nonlinear higher index DAE Control Problems*, CRSC Technical Report 101590-01, NCSU, 1990.

[11] S. L. Campbell, *Uniqueness of completions for linear time varying differential algebraic equations*, Linear Algebra & Its Appl., 161 (1992), 55-68.

[12] S. L. Campbell, *Least squares completions for nonlinear differential algebraic equations*, Numer. Math., 65 (1993), 77-94.

[13] S. L. Campbell, *High index differential algebraic equations*, preprint, 1993.

[14] S. L. Campbell and C. W. Gear, *The index of general nonlinear DAEs*, preprint, 1993.

[15] S. L. Campbell and E. Griepentrog, *Solvability of general differential algebraic equations*, SIAM J. Sci. Stat. Comp., to appear.

[16] S. L. Campbell and C. D. Meyer, Jr., *Generalized Inverses of Linear Transformations*, Dover Press, New York, 1991.

[17] S. L. Campbell and E. Moore, *Progress on a general numerical method for nonlinear higher index DAEs II*, Circuits Systems & Signal Processing, 13 (1994), to appear.

[18] S. L. Campbell and E. Moore, *Constraint preserving integrators for general nonlinear higher index DAEs*, preprint, 1993.

[19] S. L. Campbell and E. Moore, *A coordinate free approach for constraint preserving higher index DAE integrators*, Proc. 1994 IMACS, to appear.

[20] S. L. Campbell, E. Moore, and Y. Zhong, *Utilization of automatic differentiation in control algorithms*, IEEE Trans. Aut. Control, to appear.

[21] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Second edition, Johns Hopkins Univ. Press, 1989.

[22] E. Griepentrog and R. März, *Differential-Algebraic Equations and Their Numerical Treatment*, Teubner-Texte zur Mathematik, Band 88, Leipzig, 1986.

[23] E. Hairer, C. Lubich, and M. Roche, *The Numerical Solution of Differential-Algebraic Systems by Runge-Kutta Methods*, Springer-Verlag, New York, 1989.

[24] E. J. Haug and R. C. Deyo, Editors. *Real-Time Integration Methods for Mechanical System Simulation*, Springer-Verlag Computer and Systems Sciences, Vol. 69, 1991.

[25] C. L. Lawson and R. J. Hanson, *Solving Least Squares Problems*, Prentice-Hall, 1974.

[26] Ch. Lubich, U. Nowak, U. Pöhle, Ch. Engstler, *MEXX-Numerical software for the integration of constrained mechanical multibody systems*, Konrad-Zuse Zentrum für Information Stechnik, Berlin, 1993.

[27] E. Moore, *Constraint Preserving Multistep Integraters for Differential Algebraic Equations*, Ph.D. Thesis, N. C. State University, Raleigh, NC, 1994.

[28] F. A. Potra, *Implementation of linear multistep methods for solving constrained equations of motion*, SIAM J. Numer. Anal., 30 (1993), 774 - 789.

[29] F. A. Potra and J. Yen, *Implicit numerical integration for Euler-Lagrange equations via tangent space parameterization*, Mechanics of Structures & Machines, 19 (1991), 77-98.

[30] P. J. Rabier and W. C. Rheinboldt, *A general existence and uniqueness theorem for implicit differential algebraic equations*, Diff. Int. Eqns., 4 (1991), 563-582.

[31] S. Reich, *On an existence and uniqueness theory for nonlinear differential-algebraic equations*, Circuits, Systems Signal Processing, 10 (1991), 343-359.

[32] D. S. Watkins, *Fundamentals of Matrix Computations*, John Wiley & Sons, 1991.

[33] R.A. Wehage and E. J. Haug, *Generalized coordinate partitioning for dimension reduction in analysis of constrained dynamic systems*, J. Mech. Design, 134 (1982), 247-255.