Physica D 60 (1992) 269–279
North-Holland

PHYSICA D

Numerical solution of differential-algebraic equations in mechanical systems simulation

Linda R. Petzold [1]
Department of Computer Science, University of Minnesota, Minneapolis, MN 55455, USA

The numerical solution of the differential-algebraic equations of motion of mechanical systems offers many computational challenges. In this paper we describe progress which has been made in understanding the formulation of the equations of motion from the viewpoint of numerical stability, and outline some of the difficulties which must be resolved for efficient and reliable numerical methods in real-time simulation of mechanical systems.

1. Introduction

In recent years much activity has been devoted to the development of numerical methods and underlying theory for the solution of differential-algebraic equation (DAE) systems. These types of systems occur frequently as initial value problems in the computer-aided design and modeling of mechanical systems subject to constraints, electrical networks, chemically reacting systems such as distillation, flow of incompressible fluids, and in many other applications. Differential-algebraic systems, which in general can take the form $F(t,y,y') = 0$, are different from standard-form ODE systems $y' = f(t,y)$ in that, while they include ODE systems as a special case, they also include problems which are quite different from ODEs.

In a sense, the more singular a DAE system is, the more difficult it is to solve numerically. The *index* of a system is a measure of the degree of singularity of the system. Roughly speaking, ODE systems $y' = f(t,y)$ are index-zero, differential equations coupled with algebraic constraints, $y' = f(y,z)$, $0 = g(y,z)$, where $\partial g/\partial z$ is nonsingular, are index-one, and differential equations coupled with algebraic constraints where z cannot be solved for uniquely in g as a function of y are of index higher than one. The index can be defined also for systems which are not expressed in the semi-explicit form of differential equations coupled with algebraic constraints. Additional difficulties can occur for these systems because the singularity may be moving from one part of the system to another.

Much progress has been made on understanding the underlying structure and numerical solution of DAE systems. Fundamental concepts such as index and solvability have been extended to classes of DAEs describing a broad range of scientific and engineering problems. Convergence results have been given for numerical methods such as multistep and Runge–Kutta applied to several important classes of DAEs. Production-level computer codes such as DASSL [7] have been employed extensively for the solution of (index-one) engineering problems. Much of this work is described in the recent monographs [7,21,22].

There is much still that needs to be done for the effective solution of certain classes of DAEs. In this paper we will focus on the algorithms and analysis which are needed for the effective real-

[1] This work was partially supported by the US Army Research Office at the University of Minnesota Army High Performance Computing Research Center.

Elsevier Science Publishers B.V.

time simulation of mechanical systems. Real-time simulation of mechanical systems is needed in robotics, as well as in the design and simulation of vehicles, including automobiles, high-speed trains, tanks and construction equipment.

The modeling of multibody systems gives rise to Euler–Lagrange equations. Any effective numerical method for these systems must be very fast and extremely robust because the systems must often be solved repetitively by design engineers who do not have time to develop a working knowledge of complex computer software or numerical methods. For some important applications such as vehicle simulation and design, the systems must be solved in real-time.

Euler–Lagrange equations are usually posed initially in the form of a system of differential equations (Newton's laws of motion) coupled with nonlinear constraints which are enforced via a Lagrange multiplier. Direct discretization of this index-three system yields numerical methods which are often not very robust because of well-known [7] difficulties with error estimation and stepsize control, as well as severe ill-conditioning of linear systems at each time step and other problems. A wide variety of reformulations of the problem and associated numerical methods have been suggested in an attempt to find a system of equations describing the system which can be effectively solved numerically. However, each has some apparent disadvantage in terms of speed and/or robustness. Because the constraints are sometimes highly nonlinear and have a strong physical relevance, it is generally considered important that the constraints, and sometimes the time derivative of the constraints, be satisfied very accurately. In addition, there are other potential difficulties: the constraints can become rank-deficient or nearly rank-deficient, the solution may have components which are oscillating at a high frequency, and there is the possibility of frequent discontinuities which are especially troublesome because the solution of a high-index DAE can be less continuous than its input. Real-time simulation imposes severe requirements on the solution method. The solution must be computed extremely rapidly, necessitating the use of massively parallel computers. The challenge for multibody systems is to develop a problem formulation and associated class of numerical methods which preserves the stability of the system, ensures that the constraints are satisfied, adapts to possibly rapid or discontinuous changes in the solution and to nearly rank-deficient constraints, and accomplishes this task in an absolute minimum of computer time and extremely reliably. In section 2 we will outline some recent results on the stable formulation of the equations of motion for numerical solution, and in section 3 we will outline some of the computational challenges for efficient and reliable numerical methods.

2. Stable formulations of the equations of motion

In this section we will be concerned with formulations and numerical methods for the Euler–Lagrange equations of constrained mechanical motion. These are systems of the form

$$M(t,p)\,\ddot{p} = f(t,p,v) - G(t,p)^{\mathrm{T}}\lambda, \tag{1a}$$

$$0 = g(t,p), \tag{1b}$$

where the positions and velocities satisfy $p, v \in \mathbb{R}^{n_p}$, and $M(p)$ is a $n_p \times n_p$ regular (symmetric positive definite) mass matrix, f is a vector of applied forces, and λ represents the n_λ Lagrange multipliers or constraint forces coupled to the system by the $n_\lambda \times n_\lambda$ constraint matrix $G := \partial g/\partial p$. These types of systems arise frequently in the modeling of multibody systems [23], for example in vehicle simulation, computer-aided design of mechanical systems, and modeling of robotic manipulators.

The Euler–Lagrange system (1) poses difficulties for numerical methods in part because it is index-three. In particular, direct discretization

of (1) yields numerical methods which are often not very robust because of the well-known [7] difficulties for higher-index systems with error estimation and stepsize control, as well as severe ill-conditioning of the linear systems at each time step, and a variety of other problems. In addition, for some problems the constraints can be poorly conditioned; in these cases methods applied to (1) and to some of its reformulations can behave numerically as if they were solving a problem for which the index is even higher.

To overcome the problems inherent in the direct numerical solution of the index-three form of the Euler–Lagrange equations, quite a number of reformulations of (1) have been suggested; some are in use in multibody codes [37]. Many of these formulations of the equations are based on differentiation of the constraints. The constraints

$$0 = g(p), \tag{2a}$$

$$0 = G(p)\,v, \tag{2b}$$

$$\begin{aligned} 0 &= G(p)\,\dot{v} + v^{\mathrm{T}} G_p(p)\,v \\ &=: G(p)\,\dot{v} + z(p,v) \end{aligned} \tag{2c}$$

are called the *position, velocity, and acceleration-level constraints*, respectively. An index-two problem can be formed, for example, by replacing the position constraint in (1) with the velocity constraint. The resulting problem has, with appropriate initial conditions, the same solutions as (1) and is somewhat easier to solve numerically. However, the solutions can *drift* away from satisfying the position constraints because of numerical errors at each time step. This drift is often considered unacceptable in engineering problems because of the strong physical relevance of the position constraints (these are often holding components together), and because of their sometimes severe nonlinearity. An index-one problem can be formed by replacing the position constraint in (1) with the acceleration constraint. The resulting system is generally much easier to solve numerically, but now the solution can drift away from both the position and velocity constraints; the drift away from the position constraint can be quite significant.

Various formulations and solution procedures have been proposed to deal with or eliminate the problem of drift. Gear and others [18] have suggested that instead of replacing the position constraint with the velocity constraint, both constraints could be explicitly enforced by means of an additional Lagrange multiplier. This leads to a system of the form

$$\dot{p} = v - G^{\mathrm{T}}(t,p)\,\mu, \tag{3a}$$

$$M(t,p)\,\dot{v} = f(t,p,v) - G^{\mathrm{T}}(t,p)\,\lambda, \tag{3b}$$

$$0 = g(p), \tag{3c}$$

$$0 = G(p)\,v. \tag{3d}$$

The resulting problem is index-two. There is a similar formulation which enforces additionally the acceleration constraint by means of yet another Lagrange multiplier. These types of systems are generally called *stabilized formulations* of the Euler–Lagrange equations (as opposed to the unstabilized forms discussed earlier for which there may be drift). Although the stabilized formulations quite cleverly eliminate the drift problems, we have unfortunately found in recent numerical experiments that most ODE methods (including BDF and most implicit Runge–Kutta) applied to these equations may become very inefficient in certain situations, for example if the system is heterogeneous (includes components with widely disparate masses) or the constraints are poorly conditioned.

Eq. (1) has $m = n_p - n_\lambda$ degrees of freedom. Using the constraints, we can reduce (1) locally to a system of m ODEs called a *state-space form*. The choice of coordinates is not unique; Haug and Wehage [43] use Cartesian coordinates. The resulting method is called *generalized coordinate partitioning*, and is the basis of the code DADS [37]. Potra and Rheinboldt [33,34]

suggest a different local parameterization. For the purposes of analysis, we will propose to define an *essential underlying ODE*, which is a certain class of state-space forms. By its construction, the original constraints are satisfied by a state-space form. The same set of coordinates may not work over an entire problem; thus the coordinates must be chosen adaptively.

Still another possible method for solving the Euler–Lagrange equations consists of appending the velocity and acceleration constraints to (1). The resulting system is called an *overdetermined DAE (ODAE)*, and has been investigated by Führer [13] and Leimkuhler [14], and others [31]. The ODAE is discretized by a numerical method such as BDF, and the resulting nonlinear system is solved by a Gauss–Newton iteration. In [14] it is shown that for a model problem where the constraints are linear with constant coefficients and under certain other conditions, the solution to the ODAE which is determined using a certain *ssf-iteration* to solve the nonlinear system is the same as that obtained by solving one of the stabilized forms, and that these solutions are equivalent to those obtained by numerically integrating the state-space form using the same discretization method. Unfortunately, these results do not appear to carry over to the more general case.

Various other methods have been proposed for solving the Euler–Lagrange equations, including the regularizations of Baumgarte [6], Lötstedt [25], Kalachev and O'Malley [24], Zeid and Overholt [45] and others. Sometimes these regularizations can be quite effective; variations of the method of Baumgarte are in use in many engineering codes. Unfortunately, it is not always easy to pick the regularization parameters which work.

As we have seen, a wide variety of formulations and associated numerical methods have been suggested for the solution of the Euler–Lagrange equations of constrained mechanical motion. In recent work [2], we have systematically evaluated the formulations and associated numerical methods from the standpoint of stability, to determine whether some formulations and methods are inherently better at preserving the conditioning of the original problem than others. The basic idea is to define a class of *essential underlying ODEs* (EUODE). The EUODE is defined for higher-index linear Hessenberg DAEs of the form

$$x^{(m)} = \sum_{j=1}^{m} A_j z_j + By + q, \tag{4a}$$

$$0 = Cx + r, \tag{4b}$$

where $z(x) = (x, x', \ldots, x^{(m_1)})^T$, A_j, B and C are smooth functions of t, $0 \le t \le 1$, $A_j(t) \in \mathbb{R}^{n_x \times n_x}$, $j = 1, 2, \ldots, m$, $B(t) \in \mathbb{R}^{n_x \times n_x}$, $n_y \le n_x$ and CB nonsingular for each t (this assures that the DAE has index $m + 1$). All matrices involved are assumed to be uniformly bounded in norm by a constant of moderate size. The inhomogeneities are $q(t) \in \mathbb{R}^{n_x}$ and $r(t) \in \mathbb{R}^{n_y}$.

The EUODE is derived as follows. As in [1], there exists a smooth, bounded matrix function $R(t) \in \mathbb{R}^{(n_x - n_y) \times n_x}$ whose linearly independent rows form a basis for the nullspace of B^T (R can be taken to be orthonormal). Thus, for each t, $0 \le t \le 1$,

$$RB = 0. \tag{5}$$

We assume that there exists a constant K_1 of moderate size such that

$$\|(CB)^{-1}\| \le K_1 \tag{6}$$

uniformly in t, and obtain (lemma 2.1 in [1]) that there is a constant K_2 of moderate size such that

$$\left\| \begin{pmatrix} R \\ C \end{pmatrix} \right\| \le K_2. \tag{7}$$

The constant K_2 depends, in addition to K_1, also on $\|B\|$, $\|C\|$ and $\|R\|$. Let K_3 be a moderate bound on R and its derivatives:

$$\|R^{(j)}\| \leq K_3, \quad j = 0, 1, \ldots, m. \tag{8}$$

Define new variables

$$u = Rx, \quad 0 \leq t \leq 1. \tag{9}$$

Then, using (4b), the inverse transformation is given by

$$x = \begin{pmatrix} R \\ C \end{pmatrix}^{-1} \begin{pmatrix} u \\ -r \end{pmatrix} = Su - Fr, \tag{10}$$

where $S(t) \in \mathbb{R}^{n_x \times (n_x - n_y)}$ satisfies

$$RS = I, \quad CS = 0 \tag{11}$$

and

$$F := B(CB)^{-1}. \tag{12}$$

By our assumptions and (7) this mapping is well-conditioned. Both S and F are smooth and bounded. The first m derivatives of S and F are bounded by a constant involving K_2 and K_3. Taking m derivatives of (9) yields

$$\begin{aligned} u^{(m)} &= (Rx)^{(m)} \\ &= \sum_{j=1}^{m} \left[RA_j + \binom{m}{j-1} R^{(m-j+1)} \right] z_j \\ &\quad + Rq. \end{aligned} \tag{13}$$

Using $m - 1$ derivatives of (10) we obtain the EUODE

$$\begin{aligned} u^{(m)} = &\sum_{j=1}^{m} \left[RA_j + \binom{m}{j-1} R^{(m-j+1)} \right] \\ &\times [(Su)^{(j-1)} - (Fr)^{(j-1)}] + Rq. \end{aligned} \tag{14}$$

The EUODEs of a system are certain state-space forms which are uniquely defined up to a bounded, nonsingular change of variables. It is shown [2] that if the EUODE is stable, i.e., if its Green's function is bounded by a constant of moderate size, then a similar conclusion holds for the original DAE. Since the boundedness of the Green's function is invariant under bounded, nonsingular changes of variables, the question of stability for the EUODEs is well-defined. In [2], we used the EUODE to investigate the stability of some of the many equation formulations for Euler–Lagrange systems. We found that all of the formulations preserved the stability except unstabilized index reduction.

While several different equation formulations might equally preserve the conditioning of the Euler–Lagrange equations, the properties of numerical methods applied to these systems are often quite different. For example, it is well-known [7] that higher-index systems are in a sense ill-posed, and can lead to difficulties for numerical methods with error control, ill-conditioning of linear systems at each time step, etc. For higher-index Hessenberg DAEs such as the Euler–Lagrange equations, there is a problem with *numerical* instability for many methods. Consider, for example, a linear homogeneous Hessenberg index-two system:

$$x' = A(t)x + B(t)y, \tag{15a}$$

$$0 = C(t)x. \tag{15b}$$

This system has the EUODE

$$u' = RASu + R'Su. \tag{16}$$

Now discretize with implicit Euler:

$$x_{n+1} = x_n + hA_{n+1}x_{n+1} + hB_{n+1}y_{n+1}, \tag{17a}$$

$$0 = C_{n+1}x_{n+1}. \tag{17b}$$

Transforming back to the variables of the EUODE yields the discretization

$$u_{n+1} = u_n + hRASu_{n+1} + hR'Su_n. \tag{18}$$

Comparing (18) with the EUODE (16) shows that the implicit Euler method corresponds to a discretization of the EUODE which handles the term $R'Su$ *explicitly*! Thus, although convergence results [7] predict that this method

will converge globally to $O(h)$, there is a problem with respect to numerical stability which restricts the stepsize when $R'S$ is large. This problem is verified by experiment; there is very definitely a nonstiff behavior of methods ranging from BDF to most implicit Runge–Kutta for certain stable linear Hessenberg index-two systems. On the other hand, it is possible to argue, with a finer analysis, that under 'reasonable' conditions, this type of numerical instability should not occur for certain projections (for example, in (17) if $B = C^T$). The best cure for this numerical instability seems to be to reformulate the system in a form for which the instability cannot occur. This is done by reformulating the system in a form where the projection can be controlled, rather than dictated by the M matrix. In particular, we would like to formulate the system so that $B = C^T$. We call these formulations the methods of 'projected invariants'. The methods are constructed as follows:

(i) Starting with the original Euler–Lagrange equation, use the acceleration constraint to eliminate λ and obtain an ODE in p, v which has as invariants the position and velocity constraints:

$$\dot{p} = v, \tag{19a}$$

$$\dot{v} = (I - H)M^{-1}f - Fz(p,v), \tag{19b}$$

where $F = M^{-1}G^T(GM^{-1}G^T)^{-1}$, and $H = FG$.

(ii) Project the solution onto the desired invariants using G^T or other stable projection. For example, project onto the position constraints:

$$\dot{p} = v - G^T\mu, \tag{20a}$$

$$\dot{v} = (I - H)M^{-1}f - Fz(p,v), \tag{20b}$$

$$0 = g(p). \tag{20c}$$

(iii) Note that the above system has the same numerical solution as the following implicit formulation which can be implemented more efficiently:

$$\dot{p} = v - G^T\mu, \tag{21a}$$

$$M\dot{v} = f(p,v) - G^T\lambda, \tag{21b}$$

$$0 = G\dot{v} + z(p,v), \tag{21c}$$

$$0 = g(p). \tag{21d}$$

Depending on whether we do the projection onto the position constraints alone, or onto the position and velocity constraints, this leads us to two forms of projected invariants methods for constrained mechanical systems:

(i) Project onto position constraint:

$$\dot{p} = v - G^T\mu, \tag{22a}$$

$$M\dot{v} = f(p,v) - G^T\lambda, \tag{22b}$$

$$0 = G\dot{v} + z(p,v) \text{ (acceleration)}, \tag{22c}$$

$$0 = g(p) \qquad \text{(position)}. \tag{22d}$$

(ii) Project onto position and velocity constraints:

$$\dot{p} = v - G^T\mu - L^T\tau \tag{23a}$$

$$M\dot{v} = f(p,v) - G^T\lambda - MG^T\tau, \tag{23b}$$

$$0 = G\dot{v} + z(p,v) + GG^T\tau \text{ (acceleration)}, \tag{23c}$$

$$0 = Gv \qquad \text{(velocity)}, \tag{23d}$$

$$0 = g(p) \qquad \text{(position)}, \tag{23e}$$

where $L = G_p v$. These equations are studied in more detail in [31] and [3]. There is some controversy over whether it is really necessary to include the term $L^T\tau$, however, numerical experiments in [3] seem to indicate that including this term is advantageous for numerical stability, in certain cases where the solution is oscillating at a high frequency.

There is also a nice geometrical interpretation for the method of projected invariants – it corresponds to the orthogonal projection onto the invariant constraints of the ODE.

3. Computational challenges

3.1. *Efficient solution techniques*

Virtually all the proposed formulations for Euler–Lagrange equations have a similar structure with regard to the linear systems which must be solved at each time step. Even the solution of a state-space form, which at first glance might seem to have quite a different structure, can be expressed using Lagrange multipliers in a form with this structure [33,34]. Thus it is important to be able to solve efficiently systems with this structure. There are several important cases:

3.1.1. *Nonstiff*

In the nonstiff case, half-explicit methods [22] and/or iterations [14,18,32] can be devised which require much less work than in the stiff case. There is still a linear system, which arises because of the constraints, to be solved at each time step. However, the matrix, which has the form $\left(\begin{smallmatrix} M & G^{\mathrm{T}} \\ G & 0 \end{smallmatrix}\right)$, has some nice properties: it is symmetric positive definite, and it does not depend on the stepsize or order of the discretization. Further, linear systems of this form have been studied extensively, e.g., in constrained optimization, and some of these algorithms may be appropriate. For example, it may be feasible to update the matrix or its decomposition over a sequence of steps/iterations by quasi-Newton updates. This method seems to be promising for solution on massively parallel architectures. The mechanical systems have a special structure which can be further exploited; for example, the $O(n)$ methods [35] can also be used to solve the linear systems. However, this method leads to a recurrence which seems to be difficult to parallelize. A variety of formulations of the equations and methods are possible.

For many systems, for example if the stiffness arises because of a controller, only a small, readily identifiable part of the system may be stiff [42]. Here we expect that the GMRES iterative method [36], with the appropriate half-explicit methods as a preconditioner, should be effective. For real-time simulation, it is essential to be able to make use of explicit methods wherever possible; if the solution is not computed in the allotted time, it will be useless.

3.1.2. *Stiff*

Stiff problems can arise for example in the modeling of flexible bodies subject to constraints. In the fully-stiff case, the linear systems to be solved at each time step still exhibit a special structure, but they are no longer symmetric and now depend on the stepsize. For example, the stabilized index-two form of the equations of motion (3) leads to the linear system

$$\begin{pmatrix} I/h\beta_0 & -I & G^{\mathrm{T}} & 0 \\ K & M/h\beta_0 + D & 0 & G^{\mathrm{T}} \\ G & 0 & 0 & 0 \\ \Gamma & G & 0 & 0 \end{pmatrix} \begin{pmatrix} p \\ v \\ \mu \\ \lambda \end{pmatrix} = \begin{pmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \end{pmatrix}.$$

The matrix above is rather large, of dimension $2n_p + 2n_\lambda$, and its LU decomposition is generally dense. If the number of constraints is of the same order of magnitude as the number of positions, methods which are analogous to the null-space method of numerical optimization [20] can be considered. At present, we do not have sufficient experience to determine whether this is preferable to other alternatives. In addition, for flexible structures, the considerable structure inherent in the linear system arising from the discretization should be exploited.

3.1.3. Automatic stiffness detection

It is our impression that, with the important exception of flexible multibody systems, most problems in the simulation of multibody systems are nonstiff (or involve only a very few stiff components, as described above). However, stiff problems certainly do occur. A robust system for computer aided design should be able to treat both types of systems, hopefully with no intervention from the user. For example, it should be possible to construct a method similar to those which have been proposed and implemented for ODEs [30,38,39], which would monitor the convergence of the iteration and automatically switch to the appropriate method. The stability investigations yield some clues for the problem of what constitutes a stiff differential-algebraic system.

3.2. Rank-deficient systems

It can sometimes happen that the constraint matrix G^T becomes rank-deficient or nearly rank-deficient. In such a case, the problem becomes poorly conditioned, and numerical methods can experience serious difficulties. There are a number of possibilities for dealing this, including regularizing the system via a penalty term (see for example [28] or eliminating the redundant constraints, however, there are difficulties with either alternative.

3.3. Discontinuities

Frequent discontinuities are possible in a multibody system. Some of these discontinuities will be located very efficiently by a root-finder such as in DASSLRT [7]. However, others may arise from user-defined functions or other unanticipated situations, and need to be located automatically and handled efficiently. In the case of a collision, conservation properties of the solution should be preserved across the interface. The situation for DAEs presents difficulties in addition to the ODE case because the solution of a high-index system can be less continuous than the input, and singularities in the system can lead to numerical behavior which is quite similar to that caused by discontinuities. Impulsive solutions are possible.

3.4. Highly oscillatory systems

Often in multibody systems the solution may have components which are oscillating at a high frequency. This may arise, for example, from components which are rotating, or from the natural frequencies of the system. In a numerical method such as multistep or Runge–Kutta, which are based on approximating the solution locally, the stepsize must be chosen very small to resolve the oscillation in the solution. In some cases this could ruin the possibility for obtaining a solution in real-time. Usually, the details of the oscillating solution are not so important as the long-term solution behavior. Methods need to be developed to handle this situation.

3.5. Parallel methods

Real-time simulation is required when it is necessary to simulate part of the system and use people or hardware in other parts of the system. An example is vehicle simulation. This introduces a number of additional requirements for numerical integration. For an introduction to some of the problems and challenges of real-time simulation, see [23]. In particular, it is essential that the solution be computed in an absolute minimum of computer time; if the computation is not complete in the allotted time, it will be useless.

Since an explicit method is generally much simpler than an implicit method to parallelize, it seems important to be able to identify the nonstiff parts of the system and treat them with explicit methods or functional iteration. For example, consider the Hessenberg index-two system

$$x' = f(x,y), \quad 0 = g(x), \tag{24a,b}$$

where the matrix $g_x f_y$ is nonsingular. In the systems of interest, $f_y = -M^{-1}G^T y$, where $G = g_x$ and M is a symmetric positive definite mass matrix.

Following discretization with an explicit method, one obtains a nonlinear system which must be solved at each time step:

$$x = h\beta f(x,y) + b, \quad 0 = g(x). \tag{25a,b}$$

For example, for implicit Euler discretization we would have $\beta = 1, b = x_n, x = x_{n+1}, y = y_{n+1}$. We will consider solving this with a half-explicit Newton method:

$$x^{k+1} = b + h\beta f(x^k, y^{k+1}), \tag{26a}$$

$$0 = g(x^{k+1}). \tag{26b}$$

Alternatively, methods have been devised [22] which discretize explicitly in x and implicitly in y, but the nonlinear system for y looks pretty much the same. This leads to a nonlinear system in y to be solved at each time step, coupled to a functional iteration in x. Using Newton for y leads to

$$y^{k+1} = y^k - (h\beta)^{-1}(g_x f_y)^{-1} \times g(b + h\beta f(x^k, y^k)), \tag{27a}$$

$$x^{k+1} = b + h\beta f(x^k, y^{k+1}). \tag{27b}$$

Now, noting that $g_x f_y$ is symmetric positive definite, we will update the inverse of this matrix using the parallel quasi-Newton methods (Broyden and BFGS) devised by Still [40,41] for updating the matrix decomposition. Quasi-Newton methods have been tried previously in ODE solvers by Hindmarsh and others, but were not very successful because the matrix in ODEs, $(I - hJ)$, changes by as much as rank n whenever the stepsize h changes. However, in this DAE application the matrix does not depend on the stepsize and hence changes relatively slowly from step to step. Convergence of this iteration needs to be investigated, and its performance evaluated on the appropriate parallel (MIMD) computers. Several approaches to parallelization for multibody systems have been considered recently in the literature. Bae and Haug [4,5] express the elimination as a recursion which is then distributed among processors. This method of using the recursion is the most efficient for serial computation but is difficult to parallelize; Bae and Haug give some results on an Alliant. Zeid and Overholt [45] regularize the system, turning it into a moderately stiff system which is then solved by explicit methods in parallel. It is difficult to evaluate this method in comparison with the others because the regularization introduces a high frequency which influences the step size, hence they cannot be compared on the basis of work per step alone.

For fully-stiff systems, the iteration matrix is no longer symmetric, and it depends on the stepsize. Hence the quasi-Newton updating approach does not seem to be advantageous. The iteration matrix bears a strong resemblance to matrices encountered in constrained optimization, where null-space and range-space methods have proven to be useful in serial computation [20], depending on the relative dimensions of the number of positions n_p versus the number of constraints n_λ. For simulation of vehicles or rigid body mechanisms, where we usually expect n_λ to be of the same order of magnitude as n_p, a variant of the null-space method seems to be appropriate. Parallelization of this algorithm has been considered in [44] for medium-scale parallelism. Extension to massively parallel computers will require fast algorithms for Cholesky decomposition, QR factorization and backsolve [15].

References

[1] U. Ascher and L. Petzold, Projected implicit Runge–Kutta methods for differential-algebraic equations,

SIAM J. Numer. Anal. 28 (1991) 1097–1120.
[2] U. Ascher and L. Petzold, Stability of computational methods for constrained dynamics systems, SIAM J. Sci. Stat. Comput., to appear.
[3] U. Ascher and L. Petzold, Projected collocation for higher-order higher-index differential-algebraic equations, Comput. Appl. Math.,to appear.
[4] D.S. Bae and E.J. Haug, A recursive formulation for constrained mechanical system dynamics. I and II, Mech. Struct. Mach. 15 (1987) 359–382 , 481–506.
[5] D.S. Bae and E.J. Haug, A recursive formulation for constrained mechanical system dynamics: Part III, Parallel processor implementation, University of Iowa Report 1987.
[6] J. Baumgarte, Stabilization of constraints and integrals of motion in dynamical systems, Comput. Math. Appl. Mech. Eng. 1 (1976) 1–16.
[7] K. Brenan, S. Campbell and L. Petzold, Numerical Solution of Initial-Value Problems in Differential-Algebraic Equations (Elsevier Science Publishers, Amsterdam, 1989).
[8] S. Campbell and B. Leimkuhler, Differentiation of constraints in differential algebraic equations, J. Mech. Struct. Mach., to appear.
[9] J.E. Dennis and R.B. Schnabel, Numerical Methods for Unconstrained Optimization and Nonlinear Equations (Prentice-Hall,Englewood Cliffs, NJ, 1983).
[10] E. Eich, K. Führer, B. Leimkuhler and S. Reich, Stabilization and projection methods for multibody dynamics, Research report (Institute of Mathematics, Helsinki University of Technology, 1990).
[11] W.H. Enright, K.R Jackson, S.P. Norsett and P.G. Thomsen, Effective solution of discontinuous IVPs using a Runge–Kutta formula pair with interpolants, University of Toronto Numerical Analysis Report (1986).
[12] W.H. Enright and M.S. Kamel, Automatic partitioning of stiff systems and exploiting the resulting structure, ACM Trans. Math. Software 5 (1979) 374–385.
[13] C.Führer, Differential-algebraische Gleichungssysteme in Mechanischen Mehrkörpersystemen Theorie, Numerische Ansätze und Anwendungen, Ph. D. Thesis (Technische Universität München, 1988).
[14] K. Führer and B. Leimkuhler, Formulation and numerical solution of the equations of constrained mechanical motion, Numer. Math. 59 (1991) 55–69.
[15] K.A. Gallivan, R.J. Plemmons and A.H. Sameh, Parallel algorithms for dense linear algebra computations, SIAM Rev. 32 (1990) 54–135.
[16] C.W. Gear, Differential-algebraic equation index transformations, SIAM J. Sci. Stat. Comput. 9 (1988) 39–47.
[17] C.W. Gear, Maintaining solution invariants in the numerical solution of ODEs, SIAM J. Sci. Stat. Comput. 7 (1986) 734–743.
[18] C.W. Gear, G.K. Gupta and B. Leimkuhler, Automatic integration of Euler–Lagrange equations with constraints, J. Comput. Appl. Math. 12 / 13 (1985) 77–90.
[19] C.W. Gear and O. Osterby, Solving ordinary differential equations with discontinuities, ACM Trans. Math. Software 10 (1984) 23–44.
[20] P.E. Gill, W. Murray and M.H. Wright, Practical Optimization (Academic Press, New York, 1981).
[21] E. Griepentrog and R.März, Differential-Algebraic Equations and Their Numerical Treatment, Teubner-Texte zur Mathematik, Band 88 (1986).
[22] E. Hairer, C. Lubich and M. Roche, The Numerical Solution of Differential-Algebraic Systems by Runge–Kutta Methods, Springer Lecture Notes in Mathematics, No. 1409 (Springer, Berlin, 1989).
[23] E.J. Haug and R.C. Deyo, eds., Real-Time Integration Methods for Mechanical System Simulation (Springer, Berlin, 1989).
[24] L. Kalachev and R. O'Malley, Regularization of linear differential algebraic equations, preprint (1991).
[25] P. Lötstedt, On a penalty function method for the simulation of mechanical systems subject to constraints, Royal Institute of Technology TRITA-NA-7919 (Stockholm, Sweden).
[26] Ch. Lubich, h^2-extrapolation methods for differential-algebraic systems of index 2, IMPACT 1 (1989) 260–268.
[27] Ch. Lubich, Extrapolation integrators for constrained multibody systems, Technical Report(Universität Innsbruck, 1990).
[28] K.C. Park and J. Chiou, Stabilization of computational procedures for constrained dynamical systems, J. Guid. 11 (1988) 365–370.
[29] L.R. Petzold, An efficient numerical method for highly oscillatory ordinary differential equations, SIAM J. Numer. Anal. 18 (1981) 455–479.
[30] L.R. Petzold, Automatic selection of methods for solving stiff and nonstiff systems of ordinary differential equations, SIAM J. Sci. Stat. Comput. 4 (1983) 136–148.
[31] L.R. Petzold and F.A. Potra, ODAE methods for the numerical solution of Euler–Lagrange equations, Lawrence Livermore National Laboratory Report UCRL-JC-107157(1991).
[32] F.A. Potra, Multistep method for solving constrained equations of motion (University of Iowa, Department of Mathematics, 1991).
[33] F.A. Potra and W.C. Rheinboldt, Differential-geometric techniques for solving differential algebraic equations, in: NATO Advanced Research Workshop in Real-time Integration Methods for Mechanical System Simulation, eds. R. Deyo and E. Haug (Springer, Berlin, 1990).
[34] F.A. Potra and W.C. Rheinboldt, On the numerical solution of the Euler–Lagrange equations, University of Iowa, Center for Simulation and Design Optimization of Mechanical Systems, Technical Report R-81 (1990).
[35] D.E. Rosenthal, An order n formulation for robotic systems, J. Astronaut. Sci. 38 (1990) 511–529.
[36] Y. Saad and M.H. Schultz, GMRES: A generalized minimum residual algorithm for solving nonsymmet-

ric linear systems, SIAM J. Sci. Stat. Comput. 7 (1986) 856–869.

[37] W. Schiehlen, ed., Multibody Systems Handbook (Springer, Berlin, 1990).

[38] L.F. Shampine, Type-insensitive ODE codes based on implicit $A(\alpha)$-stable formulas, Math. Comput. 39 (1982) 109–123.

[39] L.F. Shampine, Type-insensitive ODE codes based on extrapolation methods, SIAM J. Sci. Stat. Comput. 4 (1983) 635–644.

[40] C.H. Still, Parallel quasi-Newton methods for unconstrained optimization, in: Proc. Fifth Conf. on Distributed memory concurrent computing (April 1990)pp. 263–271.

[41] C.H. Still, The parallel BFGS method for unconstrained minimization, in: Proc. Sixth Conf. on Distributed memory concurrent computing (April 1991).

[42] R. Wehage, US Army Tank Automotive Command Center, personal communication (1991).

[43] R.A. Wehage and E.J. Haug, Generalized coordinate partitioning for dimension reduction in analysis of constrained dynamic systems, J. Mech. Design 104 (1982) 247–255.

[44] S. Wright, A fast algorithm for equality-constrained quadratic programming on the Alliant FX/8, Ann. Oper. Res. 14 (1988) 225–243.

[45] A. Zeid and J. Overholt, A bond graph formalism for automating modeling of multibody systems, preprint (Army High Performance Computing Research Center, 1991).