Project spec for CPodes: coordinate projection in CVODES, phase 1

Radu Serban[*], Michael Sherman[†]

DRAFT Version 0.0, May 1, 2006.

Abstract

Multibody dynamics is a key part of the core software offerings from the NIH-funded National Center for Biomedical Computing at Stanford (Simbios), embodied in the Simbody[1] code. It is critical to the success of the project that the best available numerical methods be used to solve the resulting Index-3 DAE systems, which range widely in size, stiffness, and numerical conditioning, and are used both for simulations and sensitivity studies. Simbios has identified the DOE's CVODES[2] code as the most promising method available in the public domain; however, CVODES does not currently accept problems in the Simbody form. This is a specification for a set of minimal additions to CVODES, suitable for incorporation in the CVODES source distribution, which would allow it to solve these Index-3 DAEs using the Coordinate Projection scheme.[3] It is expected that this will be widely useful for other mechanical systems, perhaps with some additional enhancements. However, we focus here first on a specific deliverable which can efficiently solve problems as formulated by Simbody, and discuss possible future extensions. We propose to do this work as a collaboration between Simbios and DOE's Center for Applied Scientific Computing, with funding from Simbios.

1 Background 1
2 The form of a Simbody mechanical system 2
2.1 Implicit integration 4
2.2 Sensitivity 5
2.3 Some practical considerations 5
3 Specific changes to CVODES 5
4 Future directions 6
5 Logistics 6
Acknowledgments 7
References 7

1 Background

Large scale mechanical systems arise frequently in models of biological structures, from biomechanical studies of gait to molecular dynamics simulation of biopolymers. Many of these systems are most naturally formulated as differential equations in internal (relative) mechanical coordinates, restricted by a smaller number of algebraic constraints. To address the need to model these systems, the NIH National Center for Physics-based Simulation of Biological Structures at Stanford (Simbios) is developing the Simbody™ code. Simbody is part of the SimTK Core suite of software being produced by Simbios.

Each evaluation of the system equations representing these models is extremely expensive, especially for molecular systems, due to the complex force models that must be used to model biological systems. Sophisticated numerical integration methods must be employed with these systems in order to minimize

[*] Lawrence Livermore National Laboratory, Center for Applied Scientific Computing.

[†] Stanford University, Simbios Center for Physics-based Simulation of Biological Structures.

the number of function evaluations required to simulate a given length of time. Multistep methods can be extremely effective since they achieve high order with minimal function evaluations, especially when the systems are non-stiff, which is often but not always the case. The CVODES integrator provided as part of the DOE Sundials suite would be ideal for this purpose for many reasons, including high order stiff and nonstiff multistep methods, sensitivity analysis, suitability for large-scale problems, high-quality modern software engineering, and public domain availability (BSD license).

However, the set of equations resulting from a Simbody model forms an Index 3 DAE, which would seem to preclude use of an ODE integrator like CVODES. In fact, there are few options for solving Index 3 DAEs directly—for example, the popular DASSL code and the Sundials DAE solver IDA are limited to Index 2 problems. Fortunately, the special structure of internal coordinate multibody systems permits them to be efficiently converted to an ODE which automatically satisfies the Index 1 (acceleration level) constraints, leaving only a problem of gradual drift of Index 2 and 3 constraints away from their manifolds. There are two common methods for dealing with this, Baumgarte stabilization[4] and Coordinate Projection. The former has theoretical and practical difficulties, but is very easy to implement and results in just an ODE. Instead, we intend to employ Coordinate Projection which is a theoretically rigorous and in practice extremely robust method for dealing with this problem. Coordinate Projection requires only an ODE integrator combined with a projection scheme for eliminating drift, and some minor changes to error estimation and step size selection. The projection scheme not only eliminates the drift, it also provably reduces the integration error at each step so permits larger step sizes than would otherwise be possible.

We propose here to develop the necessary minor modifications to CVODES that would permit it to be used as a high-quality Coordinate Projection scheme for solving Index 3 DAEs of the particular form generated by Simbody (and many other mechanical modeling codes) and described below.

2 The form of a Simbody mechanical system

The numerical integration literature has many examples of mechanical systems; see for example section VII.7 in Hairer and Wanner.[5] Of necessity, these have been simplified examples to make the issues clear. Simbody on the other hand must deal with the fully general systems that arise in biosimulation, so many of the simplifying assumptions made in the literature are not valid. This has a significant practical impact on the interface between the model and the numerical integrator.

Here are some of the real-world issues we have to address:

- large size: ~1000 coupled coordinates, ~100 constraints. (For multiscale systems we also expect to couple to time-dependent PDEs which are much larger but structured differently.)
- deviation from pure 2nd order structure (more position coordinates than velocity coordinates, and the presence of auxiliary 1st order ODEs)
- poorly conditioned and rank-deficient (redundant) constraints; state-dependent rank changes
- differing numbers of constraints at the position, velocity, and acceleration levels
- the necessity and ability to exploit special problem structure to replace matrix operations with linear-time operators, and to directly satisfy the acceleration constraints.

Conceptually, Simbody formulates the following overdetermined Index 3 system:

(1a)	$\dot{q} = \mathbf{Q}u$	*kinematics*
(1b)	$\mathbf{M}\dot{u} = \mathbf{f} - \mathbf{A}^{\mathrm{T}}\lambda$	*dynamics*
(1c)	$\mathbf{A}\dot{u} = \mathbf{b}$	*acceleration constraints*
(1d)	$\mathbf{v}(t,q,u) = 0$	*velocity constraints*
(1e)	$\mathbf{p}(t,q) = 0$	*position constraints*
(1f)	$\dot{z} = \dot{z}(t,q,u,z)$	*auxiliary odes*

However, multibody systems have significant exploitable structure to be described below.

Here the q's are the n_q generalized coordinates (position variables), $\underline{u}$'s are the $n_u \leq n_q$ generalized speeds (velocity variables), t is time and overdot represents differentiation with respect to time. z is a set of n_z auxiliary differential equations which can affect forces but not positions or velocities. λ are m_a unknowns (Lagrange multipliers) associated with the acceleration constraints such that $\mathbf{A}^{\mathrm{T}}\lambda$ are the internal forces needed to enforce the constraints.

$\mathbf{Q}$=$\mathbf{Q}(q)$ is an n_qxn_u matrix mapping u's to q time derivatives. In practice it is an identity matrix except for quaternions, where there are four q's but three u's and a 4x3 block on the diagonal. $\mathbf{M}$=$\mathbf{M}(q)$ is a square, dense, symmetric, positive definite, well conditioned mass matrix. $\mathbf{f}$=$\mathbf{f}(t,q,u,z)$ is the system of applied and internally-generated forces, mapped to act along the generalized speeds. In most problems, almost all the CPU time goes into computing $\mathbf{f}$, so it is very important to reduce the number of required evaluations of eq. 1b.

$\mathbf{A}$=$\mathbf{A}(q)$ is an m_axn_u, poorly conditioned or rank deficient matrix, sometimes with state-dependent rank changes, and $\mathbf{b}$=$\mathbf{b}(t,q,u)$. Many, but not all, of the acceleration constraints come from differentiating velocity constraints. Similarly, many of the m_v velocity constraints are just time derivatives of the m_p position constraints, but new ones are also introduced at that level ("non-holonomic" constraints). Also, the use of quaternions at the position level introduces normalization constraints which exist *only* at that level; there are no corresponding velocity or acceleration constraints for them because the change of variables to u's replaces four dependent states with three independent ones. Each of the constraint equations involves only a subset of the state variables (often disjoint), and knowledge of that structure allows efficient solution of constraint equations as a series of small decoupled least squares problems.

Simbody is able to evaluate equations 1b and 1c simultaneously in O(n) time using the recursive structure of the multibody system. This leads to the following system, suitable to be solved with a Coordinate Projection scheme:

(2a)	$\dot{q} = \mathbf{Q}u$	*kinematics*
(2b)	$\dot{u} = \dot{u}(t,q,u,z)$	*dynamics*
(2c)	$\dot{z} = \dot{z}(t,q,u,z)$	*auxiliary odes*
(2d)	$\mathbf{v}(t,q,u) = 0$	*velocity constraints*
(2e)	$\mathbf{p}(t,q) = 0$	*position constraints*

Note that the acceleration constraints have been eliminated; they are satisfied automatically by equation 2b. In practice, then, a Simbody system provides operators for instantaneous solution of the above equations, rather than access to the individual matrices, which in most cases are never formed at all. In general, it is not necessary to distinguish among the three kinds of state variable or between the two kinds of constraints to be maintained. So the above equations can be reduced for external purposes to just these two operations:

$$\text{(3a)} \quad \dot{y} = f(t, y) \qquad \textit{derivatives}$$
$$\text{(3b)} \quad y = g(t, \hat{y}) \qquad \textit{projection}$$

representing the solution through coordinate projection of the following ODE with invariants:

$$\begin{aligned} \dot{y} &= f(t, y) \\ 0 &= g(t, y) \end{aligned}$$

Equation 3a provides the state derivatives $\dot{y} = \{\dot{q}, \dot{u}, \dot{z}\}$ as a function of time and state. Then equation 3b takes a state $\hat{y} = \{\hat{q}, \hat{u}, z\}$ which violates equation 2d or 2e and performs a normal projection back to the constraint manifold.

Because projection defined this way eliminates errors normal to the constraint manifold, the integrator's error vector should be projected before it is used to determine whether the required accuracy has been obtained. I think that requires a routine something like this:

$$\text{(3c)} \quad \varepsilon = g_{\varepsilon}(t, y, \hat{\varepsilon}) \qquad \textit{error projection}$$

There may be some additional interface requirements due to scaling of both the state variables and constraint errors—see "practical considerations" below.

2.1 Implicit integration

In many cases Simbody can efficiently calculate the analytical system Jacobian $\mathbf{J} = \partial\dot{y}/\partial y$ so we expect to use the explicit Jacobian interface to CVODES. Problem sizes are such that dense linear algebra is likely to be most effective, with cluster parallelism generally being used to evaluate the forces rather than the linear system. Some mechanical systems are so stiff that a finite or central difference Jacobian is inadequate and only an analytic or automatic differentiation Jacobian is good enough.

At some point it may be useful to exploit the internal structure of this matrix for efficiency. The system Jacobian has block structure:

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{qq} & \mathbf{J}_{qu} & \mathbf{J}_{qz} \\ \mathbf{J}_{uq} & \mathbf{J}_{uu} & \mathbf{J}_{uz} \\ \mathbf{J}_{zq} & \mathbf{J}_{zu} & \mathbf{J}_{zz} \end{bmatrix} = \begin{bmatrix} \partial\dot{q}/\partial q & \partial\dot{q}/\partial u & \partial\dot{q}/\partial z \\ \partial\dot{u}/\partial q & \partial\dot{u}/\partial u & \partial\dot{u}/\partial z \\ \partial\dot{z}/\partial q & \partial\dot{z}/\partial u & \partial\dot{z}/\partial z \end{bmatrix} \approx \begin{bmatrix} \mathbf{0} & \mathbf{I} & {\scriptstyle \mathbf{0}} \\ \partial\dot{u}/\partial q & \partial\dot{u}/\partial u & {\scriptstyle \partial\dot{u}/\partial z} \\ {\scriptstyle \partial\dot{z}/\partial q} & {\scriptstyle \partial\dot{z}/\partial u} & {\scriptstyle \partial\dot{z}/\partial z} \end{bmatrix}$$

Except for some minor annoyances due to quaternions, the 2nd-order structure of the system makes the top half of the matrix trivial, and typically $n_z \ll n_q \approx n_u$, so it can be very effective to exploit this structure when factoring the iteration matrix $\mathbf{I}$–$h\mathbf{J}$. Some integrators provide direct support for 2nd-order systems in this form (e.g. Radau5), but because of the not-quite-2nd-order problems we produce a general method won't work while a slightly customized one can.

Attainable speedup with this trick is about 2x, which in many cases will not be significant, so exploiting the internal structure of **J** does not need to be supported in phase 1 of this project. We mention it just as a possible future improvement should the dense linear algebra become a bottleneck, and so it can be kept in mind when designing the interface.

2.2 Sensitivity

A Simbody model contains many parameters *p* in addition to the state variables *q*, *u*, and *z*. The sensitivity with respect to those parameters of a small number of functions calculated over an integrated trajectory is often of great interest. We would thus like to be able to use the sensitivity features of CVODES along with the coordinate projection method. Again it is not necessary to provide any specific support for this in phase 1, but we note it for future consideration.

2.3 Some practical considerations

With generalized coordinates, scaling is very different for different variables and can be state dependent. Also, many variables are cyclic and thus need to be integrated using only absolute tolerances. So Simbody provides a scaling for all its variables. Scaling is an efficient calculation so can be reasonably done every time step.

Both the derivative calculation and the projection can fail in recoverable ways which should cause a reduction in step size. Thus the integrator must be prepared to get all the way through a step and then fail at the final projection, in the same way it would for a recoverable failure in the derivative routine.

Q: Radu, how does scaling interact with the error projection? I think the error estimates in absolute terms should be projected, with the result being scaled only to determine whether desired per-variable accuracy has been achieved. Is that right?

Constraint tolerances: although acceleration constraints are always satisfied, we do not necessarily have to project the position and velocity constraints at every step, and when we do we need only project to a certain tolerance. Like the states, these have scaling differences that must be dealt with, and only the Simbody system can know the significance of various constraint violations.

Q: Radu, how should we deal with constraint tolerances? We could ask Simbody to pass back to CVODES the actual vectors of constraint violations. These would then have to be weighted using additional Simbody-supplied information. An alternative would be to have the CVODES user specify a constraint tolerance, e.g. 0.001, to be interpreted as the acceptable fraction of “unit error” that the user is willing to tolerate. Then Simbody could return a scalar which would indicate the amount by which it considers constraints to be violated by the current state (in terms of Simbody’s internally-weigthed definition of “unit error”).

A: Scaling of both the states and the constraints can be taken into account through weighting in the appropriate norm evaluations. Indeed, the weighted-residual means square norm (WRMS), ubiquitously used in CVODES, will be also used in the projection step, thus including, through the absolute tolerances, variable scale information. For the constraint violation on the other hand, we will use a weighted norm with weights based on constraint scales provided by Simbody. These later scales could also be used in “preconditioning” the iteration matrix for the Newton iteration in the projection step.

3 Specific changes to CVODES

We propose to make the following specific changes to CVODE[S] to provide a coordinate projection capability for DAEs and/or ODEs with invariants:

- Modify the nonlinear solver to conditionally perform a projection step to correct for any drift from the constraint manifold after a successful step (based on the Nordsieck history array of previous projected solutions). Mathematically, the projection defines a minimization problem which can be solved efficiently by Newton iterations. In practice, an approximate solution to the minimization problem will suffice (corresponding to a slight relaxation of the orthogonality of the projection).

- Modify the CVODES error controller to use the projected error estimate of the original BDF method as the error estimate of the projected method. In addition, include an error test decision loop to allow for a step rejection based on the performance of the projection step itself. This error test will parallel the existing one (which attempts to control errors in the integration of the underlying ODE, regardless of the constraint violation) in the sense that, based on the outcome of the projection, it will either mark the step as successful or trigger one of the recovery attempts (Jacobian update, stepsize and/or method order modification, etc.)
- Extend the user interface to allow for specification of additional information, such as constraint scaling, constraint tolerances, and (possibly) user-provided functions for computing the weighted constraint violation norms and maybe even user-provided functions to replace the internal projection mechanism.

The above modifications will allow CVODE[S] to solve, through coordinate projection, any ODE system with invariants of the form (3a)+(3b). To enhance efficiency and solver performance, additional modifications can be made to take advantage of the special structure of mechanical systems in the form generated by Simbody:

- Implement automatic relative scaling of the position and velocity constraints based on the constraint Jacobian provided by Simbody.
- Taking advantage of the fact that, for BDF methods (the underlying linear multistep method in CVODE), semi-explicit index-1 DAEs behave under discretization like ODEs, it may be more effective to iteratively solve (1b)+(1c) instead of (2b). In this form, one of the blocks in the BDF Jacobian can be used as the iteration matrix for the modified Newton used in the projection step, thus avoiding the cost of evaluating A (recall that the Simbody multibody formalism evaluates with complexity O(n) the products Au and $A^T\lambda$, but not A itself).

4 Future directions

Upon successful completion of phase 1, we may wish to continue on to a phase 2. The specifics of that phase have yet to be determined, but here are some possibilities:

- While the BDF Jacobian of the underlying ODE obtained after explicitly excluding the Lagrange multipliers (Eq. 2b) is typically dense, the Jacobian of the index-1 DAE represented by Eqs. 1b+1c has the sparsity pattern of a so-called “optimization matrix”. A direct sparse linear solver could significantly reduce the cost of the linear algebra involved in the implicit time stepping.
- To enable additional types of analyses (e.g. model reduction, design optimization, optimal control), sensitivity analysis (SA) - evaluating derivatives with respect to problem parameters of the solution and/or functionals of the solution - provides a critical ingredient. The coordinate projection solver resulting from phase 1 could be extended to simultaneously integrate the continuous sensitivity equations (forward SA) and/or integrate backwards in time the adjoint of the linearized forward system (adjoint SA) to provide sensitivity capabilities.

5 Logistics

Funding for this project will be provided by the Simbios center, under supervision of Executive Director Jeanette Schmidt. We expect the total cost of phase 1 to be $40,000 with completion expected in July, 2006. Personnel involved will be Radu Serban from LLNL, and Michael Sherman and Jack Middleton from Simbios.

Acknowledgments

Simbios is funded by the National Institutes of Health through the NIH Roadmap for Medical Research, Grant U54 GM072970. Information on the National Centers for Biomedical Computing can be obtained from http://nihroadmap.nih.gov/bioinformatics.
The work of Radu Serban is supported under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

References

[1] https://simtk.org/home/simbody

[2] http://www.llnl.gov/CASC/sundials/main.html

[3] Eich, E. Convergence results for a coordinate projection method applied to mechanical systems with algebraic constraints. *SIAM J. on Numerical Analysis* 30(5):1467-1482 (1993).

[4] Baumgarte, J., Stabilization of constraints and integrals of motion in dynamical systems, *Comp. Methods Appl. Mech*., 1 (1972), pp. 1-16.

[5] Hairer, E. and Wanner, G. *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, 2nd revised ed., Springer, 1996.