A Numerical Integrator for Simulation of Unstructured Implicit Models

Stephen L. Campbell Monica Selva Carmen Arévalo

Abstract— **Object oriented modeling naturally leads to implicitly defined dynamical systems or, as they are also called, differential algebraic equations (DAEs). Most existing DAE integrators require the equations defining the dynamical system to have a special structure. Progress on the development of a new integrator for general unstructured DAEs is presented. The new integrator is called UCP. Computational tests are given to show that the new method can successfully integrate problems that other methods have trouble with.**

I. INTRODUCTION

As computer modeling considers increasingly complex systems there is an increased need to have object oriented modeling packages where the system is defined by relations between the variables and submodels rather than a specified information flow such as in a function. In such an environment the modeler takes various submodels and interconnects them into a larger model. The submodels can come from different disciplines such as chemical, electrical, and mechanical so that the user may not have expertise in their individual usage. The resulting models are often Differential Algebraic Equations or DAEs. That is, they are a mix of differential and algebraic equations.

A robust software package must be able to do several things [3]. It must be able to analyze the model and either perform the simulation or, as often as possible, come back and give the user information on what the problem is if integration is not possible or internal diagnostics expect the answer might be inaccurate. While the latter outcome is helpful, the wider the class of problems that can be simulated, the more productive the user will be. This paper will describe recent significant progress in developing an integrator for general DAEs which preserves both explicit and implicit constraints. It is being designed in order to provide a reliable solution when other methods fail to do so as well as providing a truth model to check the answer of more conventional methods on those special classes of DAEs where conventional methods can be used.

II. DAE BACKGROUND

A DAE is an implicit differential equation of the form

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

Steve Campbell is with North Carolina State University, Department of Mathematics, Raleigh, NC. 27695-8205 USA. e-mail: slc@math.ncsu.edu

Monica Selva is with Humboldt University of Berlin, Institut of Mathematics, Berlin, Germany.

Carmen Arévalo is with Lund University, Department of Numerical Analysis, Lund, Sweden.

where $x(t)$ is a vector valued function, the Jacobian $F_{x'}$ is a singular matrix, and $u(t)$ is a vector valued function of source or control terms. The nature of u is important in applications and in simulation studies, but in terms of the integrator, it is just part of the dependence on t of the differential equation. Accordingly we will omit it in the rest of this paper.

An integer quantity called the index measures how much a DAE differs from an explicit ordinary differential equation (ODE). The frequent occurrence of DAEs in applications has given rise to a variety of numerical techniques that simulate and analyze DAEs. However most of these methods only apply to specific classes of problems with special structure and low index.

Depending on the application area, DAEs are also called singular, descriptor systems, constrained, semi-state, differential equation on a manifold, and implicit. DAEs occur not only in composite models. There are several reasons to also sometimes make the underlying submodels DAEs. These reasons include: models formulated implicitly, reduction to an ODE may not be practical, reduction reduces sparsity in equations and Jacobians, in simulation studies can have one implicit model rather than several explicit ones of varying dimension, changing numbers of constraints in contact problems, and extra design flexibility.

The solutions of a higher index DAE can depend on derivatives of coefficients and forcing functions. Thus the question is often not whether to differentiate but rather where to do so. If we differentiate the DAE (1) k times with respect to t, we obtain the *derivative array* equations [4]

$$\begin{aligned} F(x', x, t) &= 0 \qquad \text{(2a)} \\ &\vdots \\ \frac{d^k}{dt^k} F = F_k(x^{(k+1)}, x^{(k)}, \ldots, x, t) &= 0 \qquad \text{(2b)} \end{aligned}$$

The smallest k such that (2) uniquely determines x' in terms of consistent (x, t), $x' = Q(x, t)$ is the *index of the DAE* (1). For fully implicit problems, x' is unique in a relatively open set containing x'_0. An implicit ODE has index zero. Intuitively, the index is a measure of how many differentiations would be required to convert the DAE into an ODE. The actual description of what takes place is technically much more difficult since constraints may not be explicit and one need not have constant ranks [6], [8]. Also, in numerical algorithms one wants to avoid differentiating computed quantities. DAEs which are not index one or

not semi-explicit always have additional hidden constraints. Other types of index can be defined such as the perturbation index [14]. Originally it was thought that the different indices were closely related. [8] shows they can differ greatly for unstructured nonlinear DAEs and develops a unifying theory. See also [2].

A variety of numerical methods have been developed for DAEs [2], [14], [20], [21], [22]. None of the variants of standard methods such as BDF, the various Implicit Runge Kutta, extrapolation, or multistep methods, work on even general index two problems. All of these methods require special structure, such as being Hessenberg, and also usually require the index to be no more than two or three even though many applications are initially formulated as higher index DAEs [2], [7]. All more general methods are based on some variant of the derivative array (2).

A. General DAE Integrators

As noted earlier, standard DAE integrators only work on special classes of systems of low index. The development of general DAE integrators requires developing mathematical results of several types. Nonlinear iterative solvers are used. These result in equations which are not equivalent to the original ones. [4], [6], [8] develop fundamental mathematical results that underlie all the integration techniques being developed and show that the assumptions made are very general. [6] develops computational tests for key quantities such as the index and dimension of the solution manifold.

A BDF based general method, also based in part on the theory developed in [4], [6], [8], is being developed [16], [17], [18].

Two different types of numerical methods have previously been examined by the authors and colleagues. The explicit integration (EI) approach [10] generates a completion of the DAE's vector field. However, during integration it can experience drift off the constraints. The implicit coordinate partitioning (ICP) approach [9] requires some additional computational effort but has the important property of preserving all constraints, implicit and explicit. ICP is a generalization of the coordinate partitioning used in some mechanical simulation codes using subsets of coordinates to locally parameterize the solution manifold. Numerical issues such as Jacobian reuse or error control are different for EI and ICP and often require different theoretical justifications. In some cases, such as [12], where some terms of the limiting solution are shown to have dynamics dependent on the choice of predictor, the theory is fundamentally different from the usual ODE theory. Significant progress on implementation issues has been made [10]. These approaches require the generation of Jacobians which involve derivatives with respect to derivatives. These Jacobians can be cheaply computed, relative to the rest of the computation, by MAPLE generated FORTRAN code for small problems and by automatic differentiation codes such as ADOL-C [13] for larger problems [5], [11].

A better alternative to EI and ICP appears to be to combine the idea of ICP with some of the ideas on tangent space parameterization due to Potra and Rheinboldt [20], [21]. This new approach, called UCP for Unitary Coordinate Partitioning, appears promising but a number of theoretical and algorithmic issues remain to be determined. The basic idea for UCP is described in [1]. This paper provides the first significant computational experience with this approach, establishes new advantages, and discusses actual implementation. UCP is currently in prototype form. When complete the code will be available at the first author's web site: `www.math.ncsu.edu/~slc`.

III. How UCP works

Assume the DAE (2a) is sufficiently differentiable and has a well defined solution manifold. Differentiate the equations in the DAE k times with respect to t to get the derivative array. As noted, this can be done with symbolic or automatic differentiation software. This yields routines for evaluating the derivative array equations

$$G(x', w, x, t) = 0$$
$$w = [x^{(2)}, \ldots, x^{(k+1)}]$$

and the Jacobians

$$\overline{J}_k = [\ G_{x'} \quad G_w\ ], \quad J_k = [\ G_{x'} \quad G_w \quad G_x\ ],$$

If some structure is present, it can be exploited by not differentiating all equations the same number of times.

It is assumed that for large enough k the following assumptions hold:

- (A1) Sufficient smoothness of G.
- (A2) Consistency of $G = 0$
- (A3) $\overline{J} = [\ G_{x'} \quad G_w\ ]$ 1-full and has constant rank independent of (t, x, x', w).
- (A4) $J = [\ G_{x'} \quad G_w \quad G_x\ ]$ has full row rank independent of (t, x, x', w)

Assumptions (A1)–(A4) are directly in terms of the original equations and their derivatives. They are numerically verifiable and are almost equivalent to the existence of a well defined solution manifold [6] so they are not overly restrictive.

ICP often works well but sometimes has difficulties. In particular, it has the following problems which it inherits from the coordinate partitioning strategy:

- Error control is difficult because of error introduced by changes in the coordinate partition.
- There is no a priori guarantee that one can do better when considering a coordinate change and one has to search among a number of possible subsets of coordinates.
- The partition chosen may not be the best choice of coordinates even if a robust partition strategy is used.

UCP uses orthogonal transformations other than permutations to pick the local coordinates to parametrize the solution manifold. In principle, it should not suffer from the three failings of ICP listed above. This is illustrated in Figure 1 where an elliptical solution manifold is drawn. Suppose that the numerical solution is at the point on the curve. On the left, ICP has to choose either the vertical or horizontal axis. This may not be the best choice and error is introduced when the choice is changed. On the right we see that UCP can pick an optimal coordinate system (the slanted line) and vary it as the point moves.

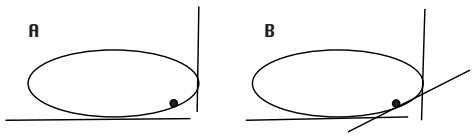

Fig. 1. ICP vs UCP: Available Coordinates

UCP picks a new coordinate system in the following way at a given point on the solution curve.

- Given the Jacobian $[G_{x'}, G_w | G_x]$, left orthogonal transformations are performed to give

$$\left[\begin{array}{c|c} S & X \\ 0 & D \end{array}\right]$$

  where S, D are full row rank.
- An orthogonal transformation U is computed so that $DU = \left[\begin{array}{cc} D_5 & 0 \end{array}\right]$ where D_5 is nonsingular.
- Let $U = \left[\begin{array}{cc} U_1 & U_2 \end{array}\right]$ so that our new coordinates are

$$\underline{x} = U^T x = \left[\begin{array}{c} U_1^T x \\ U_2^T x \end{array}\right] = \left[\begin{array}{c} \underline{x}_1 \\ \underline{x}_2 \end{array}\right]$$

- $\underline{x}_2$ are new local coordinates which parameterize the solution manifold.
- Then the nonlinear system

$$\underline{G}(x', w, \underline{x}_1, \underline{x}_2) = G(x', w, U\underline{x}) = 0$$

  is solved for $x', w, \underline{x}_1$ in terms of $\underline{x}_2$ using a Gauss-Newton iteration and the Jacobians

$$J_U \stackrel{\text{def}}{=} [\underline{G}_{x'}, \underline{G}_w, \underline{G}_{\underline{x}_1}, \underline{G}_{\underline{x}_2}] = [G_{x'}, G_w, G_x U]$$

$$\widetilde{J}_U \stackrel{\text{def}}{=} [\underline{G}_{x'}, \underline{G}_w, \underline{G}_{\underline{x}_1}] = [G_{x'}, G_w, G_x U_1]$$

  This yields the numerical relationships

$$x' = g(\underline{x}_2, t) \quad (3a)$$
$$\underline{x}_1 = h(\underline{x}_2, t) \quad (3b)$$

- (3a) is integrated using a variable step Adams Bashforth Moulton in E(PC)E mode.
- $x(t_{n+1})$ is updated by $x = U \left[\begin{array}{c} \underline{x}_1 \\ U_2^T \widehat{x} \end{array}\right] = U_1 \underline{x}_1 + U_2 U_2^T \widehat{x}$

This last step means that while we have made an integration step and preserved the constraints, at the end of each integration step, we have returned to the original coordinate system.

IV. Implementation Issues

Convergence proofs for the iterative solver require that $\widetilde{z}^{[0]}$ be in an appropriate neighborhood of a point $\widetilde{z}_0$ such that $G(\widetilde{z}_0, x_2, t) = 0$. The simplest way to insure convergence is to have at least first order estimates of all derivatives that appear in the derivative array. If a multistep method has order equal to the highest derivative that appears in the derivative array, then first order estimates will be readily available, say from the Nordsieck vector.

In the UCP code being developed we use an Adams-Bashforth-Moulton method. There are several orders available within the code, but the one we use most often is the fourth order one. This provides estimates for up to fifth derivatives and thus is suitable for general DAEs of index up to 4. Note that this assumption on order is only to guarantee the estimate for the w variables. In fact, we have solved a number of problems of even higher index using this fourth order integrator with no difficulty.

V. Test Problems

We now turn to presenting the results of some new computational studies. They establish several new facts.

A. Problem 1: Tight orbits

The first problem is

$$\begin{aligned} x_1' &= -bx_2 - x_1 x_3 \\ x_2' &= x_1 - bx_2 x_3 \\ 0 &= x_1^2 + bx_2^2 - L \end{aligned}$$

Here $b > 0,\ L > 0$. We take $L = 1$. This is an index two DAE. The solution lies on an ellipse

$$\begin{aligned} x_1 &= -L\sin(\sqrt{b}t) \\ x_2 &= \frac{L}{\sqrt{b}}\cos(\sqrt{b}t) \\ x_3 &= 0 \end{aligned}$$

and is illustrated in Figure 2 for $x_1(0) = 1, x_2(0) = 0$.

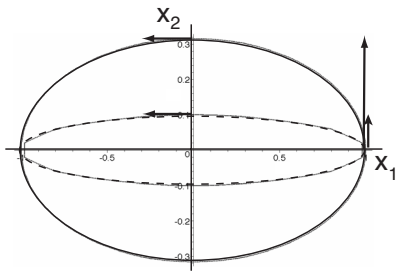

Fig. 2. x_1, x_2 for $b = 10$ (solid line) and $b = 100$ (dashed line).

Note that the solutions have the same speed when they cross the x_2 axis. However, there are sharper and faster corners at the x_1 axis as b increases. These trajectories with rapid tight corners are similar to gravity assisted space trajectories where long coast trajectories are mixed with rapid flybys of planets.

With all the methods tested, x_3 is found to very high accuracy. Accordingly we focus on x_1 and x_2. UCP and ICP reuse partitions/coordinates from one step to another with a strategy for deciding when to update. In this paper we let UCPP and ICPP be UCP and ICP set to compute new coordinates at every time step.

Consider first $b = 10$. All of the graphs for this example are of the error. It is computed by comparing the numerical solution to the exact solution.

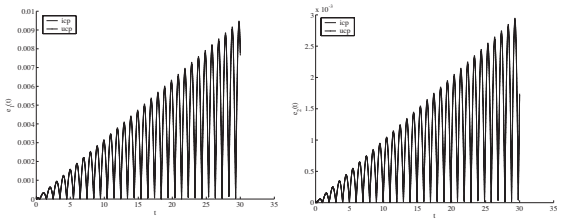

Fig. 3. Error in x_1 (left) and x_2 (right) using UCP, ICP with fixed $h = 0.01$, $b = 10$.

Figure 3 gives the results using a fixed stepsize. We see that on this easy problem UCP and ICP had the same error. Since all of the error graphs for the $b = 10$ case are qualitatively similar, we will omit them and merely give the quantitative results.

If the partition is recomputed on every time step, again with the fixed step of $h = 0.01$, we see that the error on $[0,\ 30]$ is reduced from 0.01 and 0.003 to 0.008 and 0.0025 respectively. This shows that there is some accuracy loss due to partition reuse even on this easy problem.

Suppose that we use our current variable step implementation which will be discussed more carefully in a later section. Suppose that the requested error tolerance is TOL $= 10^{-6}$. The error is the same for both UCP and ICP. If we request the same error tolerance, but use the UCPP and ICPP versions, the errors are reduced from 0.00424 to 0.0036 and from 0.013 to 0.0128 showing that even with variable step methods and well conditioned problems there is some loss due to partition reuse.

Now suppose that we increase b to 100. The graphs of the error for x_1 and x_2 are the same except for magnitude so that we shall give only x_1. If the step size is constant we have that ICP and UCP perform the same as shown in Figure 4. The results for ICPP and UCPP are shown in Figure 5. We immediately notice two things. First, with the same stepsize as in Figure 4, we see that the error in UCPP is less than in UCP and much less than ICPP. The old form of coordinate partitioning is no longer able to deliver the same accuracy as the new approach. These two figures show that not only is UCP superior on this problem but that great care needs to be taken in the reuse of coordinates or one can undo the advantage of UCPP.

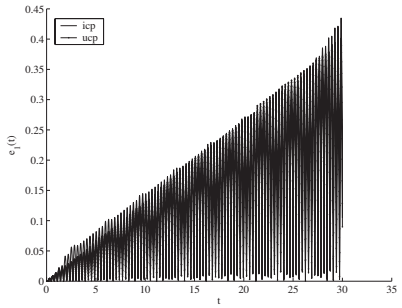

Fig. 4. Error in variable one using ICP, UCP with constant $h = 0.01$, $b = 100$.

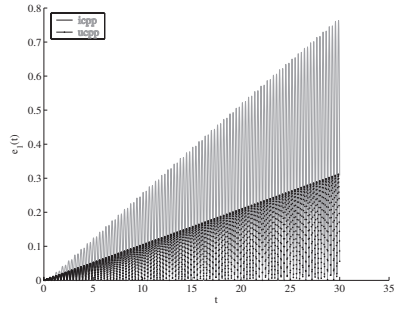

Fig. 5. Error in variable one using ICPP, UCPP with constant $h = 0.01$, $b = 100$.

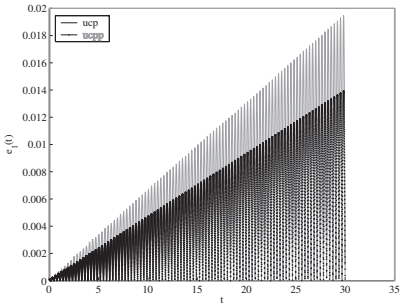

Fig. 6. Error in variable one using UCP, UCPP with variable h, TOL$= 10^{-6}$, $b = 100$.

What about using variable step with $b = 100$? The results are plotted in Figure 6. Neither ICP nor ICPP could meet requested tolerances and integration failed. Implicit coordinate partitioning is not able to deliver the requested accuracy. Both UCPP and UCP were successful and were able to integrate the problem. Note, however, that even though the error estimate and step size variation was the same for both UCP and UCPP that, in fact, the UCPP answer was about 25% more accurate.

B. Problem 2: Shuttle Reentry

The second problem was chosen as representative of a typical highly nonlinear, higher index, moderate sized DAE with several parameters arising in an engineering situation. It is the space shuttle reentry problem from [2] except that we use slightly different initial conditions. This trajectory prescribed path control problem for the shuttle in relative coordinates [2] is given by the dynamics equations:

$$H' = V_R \sin(\gamma) \tag{4a}$$

$$\xi' = \frac{V_R \cos(\gamma)\sin(A)}{r\cos(\lambda)} \tag{4b}$$

$$\lambda' = \frac{V_R}{r}\cos(\gamma)\cos(A) \tag{4c}$$

$$V_R' = \frac{-D}{m} - g\sin(\gamma) - \Omega_E^2 r\cos(\lambda)\times \Big(\sin(\lambda)\cos(A)\cos(\gamma) - \cos(\lambda)\sin(\gamma)\Big) \tag{4d}$$

$$\gamma' = \frac{L\cos(\beta)}{mV_R} + \frac{\cos(\gamma)}{V_R}\left(\frac{V_R^2}{r} - g\right) + 2\Omega_E\cos(\lambda)\sin(A) + \frac{\Omega_E^2 r\cos(\lambda)}{V_R}\Big(\sin(\lambda)\cos(A)\sin(\gamma) + \cos(\lambda)\cos(\gamma)\Big) \tag{4e}$$

$$A' = \frac{L\sin(\beta)}{mV_R\cos(\gamma)} + \frac{V_R}{r}\cos(\gamma)\sin(A)\tan(\lambda) - 2\Omega_E\Big(\cos(\lambda)\cos(A)\tan(\gamma) - \sin(\lambda)\Big) + \frac{\Omega_E^2 r\cos(\lambda)\sin(\lambda)\sin(A)}{V_R\cos(\gamma)} \tag{4f}$$

along with a path constraint

$$\frac{D}{m} - \big[C_0 + C_1(V_R - V_0) + C_2(V_R - V_0)^2 + C_3(V_R - V_0)^3\big] = 0 \tag{4g}$$

Some of the variables are given in terms of the state variables as: $\rho(H) = .002378\exp(-H/23800)$, $r = H + a_e$, $C_L(\alpha) = .84 - .48(38 - \alpha C_{rd})/26$, $g = \mu/r^2$, $D = .5\rho C_D S V_R^2$, $L = .5\rho C_L S V_R^2$, and $C_D(\alpha) = .78 - .58(38 - \alpha C_{rd})/26$. There are also parameters and constants, $\mu = 0.1407653916 \times 10^{17}$ ft^3/s^2, $a_e = 20902900$ ft, $\Omega_E = .72921159 \times 10^{-4}$ rads/sec, $m = 5964.4965$ slugs, $C_0 = 3.974960446019$, $S = 2690$ ft^2, $C_1 = -0.01448947694635$, $C_{rd} = 360/2\pi$, $C_2 = -0.2156171551995 \times 10^{-4}$, $\alpha = 40^o$, and $C_3 = -0.1089609507291 \times 10^{-7}$.

In shuttle reentry simulations NASA typically holds the angle of attack α constant at about forty degrees. This leaves the bank angle β as a control variable. The resulting system is a semi-explicit nonlinear index three DAE in the seven state variables $\{(H, \xi, \lambda, V_R), (\gamma, A), \beta\}$. It is important to note that given the state vector, there are multiple solutions for the bank angle control. Most other DAE integrators would have to perform some sort of analytic reduction procedure. One exception is RADAU5, which should, in principle, be able to solve an index three Hessenberg system. But we have been unable to get RADAUV to integrate this problem. Both UCP and ICP were able to solve this problem without any difficulty in its full index three formulation. Solutions for the control β and the velocity V_R are graphed in Figure 7. Notice the dramatic difference in scale. In this problem variables range over six orders of magnitude. This, combined with the high nonlinearity and existence of nearby incorrect implicitly defined control histories, makes this a challenging problem.

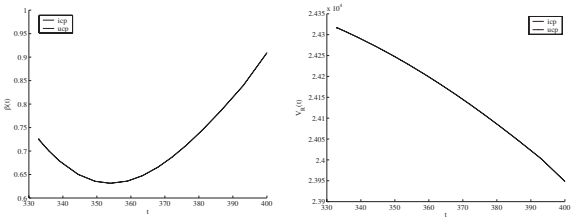

Fig. 7. Control (left) and velocity V_R (right) solutions found using variable step UCP and ICP.

VI. The Stepsize Controller

For this problem where the difficulties were due to high index and nonlinearities and conditioning, but not due to wildly varying coordinates, the stepsize selections were comparable, but not identical, for ICP and UCP. Both successfully integrated this problem.

We use a smooth error control strategy developed in [23], which entails discarding heuristic schemes for the adaptive step size selection. Instead, linear digital control theory is applied rigorously to construct smoother and stabler step size selection algorithms.

The elementary algorithm widely used for step size selection is a first order adaptive deadbeat controller,

$$h_{n+1} = \left(\frac{\theta \cdot \text{TOL}}{r_n}\right)^{1/(p+1)} h_n \tag{5}$$

where TOL is the local error tolerance, $\theta < 1$ is a safety factor and r_n is the local error estimate. This estimate is calculated in the usual way as $r_n = C_n\left(h_n x_n' - h_n x_n'^{[0]}\right)$, where $x_n'^{[0]}$ and x_n' are the Adams-Bashforth and Adams-Moulton approximations to $x'(t_n)$ obtained by solving (3a) with the predictor-corrector method. The coefficient C_n depends on the step size sequence and on the order of the Adams-Bashforth-Moulton method used to solve (3a). For stability purposes, the step size ratios are bounded by a step-function type limiter. This controller frequently produces non-smooth and sometimes oscillatory step size changes.

To overcome these difficulties we chose Söderlind's H211b (b=4) controller,

$$h_{n+1} = \left(\frac{\theta \cdot \text{ TOL}}{r_n}\right)^{\frac{1}{4(p+1)}} \left(\frac{\theta \cdot \text{TOL}}{r_{n-1}}\right)^{\frac{1}{4(p+1)}} \left(\frac{h_{n-1}}{h_n}\right)^{\frac{1}{4}} h_n.$$

Being a controller with a first-order step size low-pass filter, it produces smoother step size sequences. To reduce control errors, and thus prevent unwarranted step rejections, the safety factor θ must be chosen sufficiently small. It is important to note that the use of this controller does not incur higher computational costs.

Figure 8 shows the step size history of solving the shuttle reentry problem using UCP with two different step size strategies. The figure on the left is the standard method (5) while the one on the right uses the H211b strategy with the step limiter.

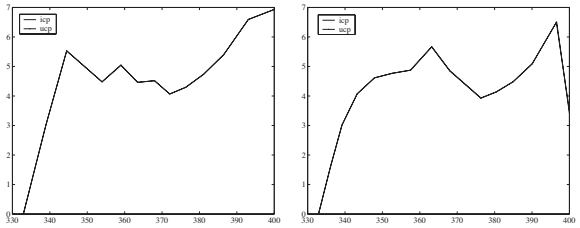

Fig. 8. Step size history for UCP using (5) and H211b strategies.

Note that the step in the right graph varies smoothly until it hits the step limiter. The use of heuristic non-differentiable limiters causes additional step size jumps and loss of predictability in the relation TOL/local error. This may be smoothed out by using a limiter based on the $\arctan$ function. This is currently being implemented.

VII. Conclusion

We have surveyed some of the theory for general DAE integrators. We then presented a promising new DAE integrator called UCP for Unitary Coordinate Partitioning. Computational experiments were run that establish that

- UCP can result in a more robust and accurate computation than the older approach of Implicit Coordinate Partitioning (ICP) at only slightly more computational cost (10-20%)
- In trying to create more numerically efficient codes, extra care must be taken when reusing Jacobians and coordinates or these advantages can be lost.
- UCP which recomputed the partition at every time step consistently did better.
- UCP can integrate problems with prescribed tolerances that ICP cannot
- UCP can integrate problems that other methods such as DASSL or RADAU cannot.
- UCP can integrate complex problems from engineering applications.

VIII. Acknowledgements

Research supported in part by NSF Grants DMS-0101802, DMS-020695, and ECS-0114095.

References

[1] C. Arevalo, S. L. Campbell, and M. Selva, *Unitary Partitioning in General Constraint Preserving DAE Integrators*, Comp. Math. Appl., to appear.

[2] K. E. Brenan, S. L. Campbell and L. R. Petzold, *The Numerical Solution of Initial Value Problems in Differential-Algebraic Equations*, SIAM, Philadelphia, 1996.

[3] S.L. Campbell, *Intelligent DAE solvers and user friendly design, simulation, and analysis packages*, Proc. IEEE International Conf. Systems, Man, and Cybernetics, San Diego, Oct., 1998, 177–3182.

[4] S. L. Campbell, *Least squares completions for nonlinear differential algebraic equations*, Numerische Mathematik, 65 (1993), 77–94.

[5] S. L. Campbell, E. Moore, and Y. Zhong, *Utilization of automatic differentiation in control algorithms*, IEEE Trans. Automatic Control, 39 (1994), 1047–1052.

[6] S. L. Campbell and E. Griepentrog, *Solvability of general differential algebraic equations*, SIAM J. Scientific Computation, 16 (1995), 257–270.

[7] S. L. Campbell, *High index differential algebraic equations*, J. Mech. Struct. & Machines, 23 (1995), 199–222.

[8] S. L. Campbell and C. W. Gear, *The index of general nonlinear DAEs*, Numerische Mathematik, 72 (1995), 173–196.

[9] S. L. Campbell and E. Moore, *Constraint preserving integrators for general nonlinear higher index daes*, Numerische Mathematik, 69 (1995), 383–399.

[10] S. L. Campbell and Y. Zhong, *Jacobian reuse in explicit integrators for higher index DAEs*, Applied Numerical Mathematics, 25 (1997), 391–412.

[11] S. L. Campbell and R. Hollenbeck, *Automatic differentiation and implicit differential equations*, in Computational Differentiation: Techniques, Applications, and Tools, Edited by M. Berz, C. Bischof, G. Corliss, and A. Griewank, SIAM, Philadelphia, 1996, 215–227.

[12] S. L. Campbell and K. Yeomans, *Behavior of the Nonunique Terms in General DAE Integrators*, Applied Numerical Mathematics, 28 (1998), 209–226.

[13] A. Griewank, D. Juedes, and J. Utke, *Algorithm 755: ADOL-C: a package for the automatic differentiation of algorithms written in C/C++*, ACM Trans. Math. Software, 22 (1996), 131–167.

[14] 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, 1989.

[15] E. Hairer and G. Wanner, *Solving Ordinary Differential Equations II: Stiff and Differential-Algebraic Problems*, Springer-Verlag, 1991.

[16] P. Kunkel and V. Mehrmann, *Canonical forms for linear differential-algebraic equations with variable coefficients*, J. Comp. Appl. Math., 56 (1994), 225–251.

[17] P. Kunkel, V. Mehrmann, and W. Rath, *GENDA: A software package for the solution of general linear differential algebraic equations*, SIAM J. Sci. Comp., 18 (1997), 115–138.

[18] P. Kunkel and V. Mehrmann, *Regular solutions of nonlinear differential-algebraic equations and their numerical determination*, Numerische Math., 79 (1998), 581–600.

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

[20] F. A. Potra and W. C. Rheinboldt, *On the numerical solution of Euler-Lagrange equations*, Mech. Struct. & Mach., 19 (1991), 1–18.

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

[22] W. C. Rheinboldt, *Solving algebraically explicit DAEs with the MANPACK-manifold-algorithms*, Comput. Math. Appl., 33 (1997), 31- 43.

[23] Gustaf Söderlind, *Digital filters in adaptive time-stepping*, ACM Transactions on Mathematical Software, 29 (2003), 1–26,