TOWARDS EXPLICIT METHODS FOR DAES *

C. W. GEAR[1]

[1] *17 Honey Brook Drive, Princeton, NJ 08540, USA. email: wgear@princeton.edu*

Abstract.

Explicit methods have previously been proposed for parabolic PDEs and for stiff ODEs with widely separated time constants. We discuss ways in which Differential Algebraic Equations (DAEs) might be regularized so that they can be efficiently integrated by explicit methods. The effectiveness of this approach is illustrated for some simple index three problems.

AMS subject classification (2000): 65-L80, 34-04.

Key words: Regularization, Projective Methods.

1 Introduction.

Recently there has been interest in explicit method for stiff ODEs. Methods, such as Runge-Kutta Chebyshev [1, 15, 16] typically have extended stability regions while projective [8, 6] and telescopic projective methods [9] can be tailored to have multiple stability regions which would be useful if we knew where the "stiff" eigenvalues were. DAEs are not usually stiff, but the solution of higher-index DAEs is often accomplished by using index reduction and regularizing techniques that change them to lower index DAEs or ODES that are sometimes stiff. In this short paper we explore ways in which regularizing techniques can be applied to give known stiff eigenvalues and additional characteristics that make explicit projective integrators particularly efficient.

Efficient integration of general stiff problems (that is, ones with a large spread of eigenvalues and little structure) requires an implicit method. As computers increase in size and capacity, the size of problems being modelled increases, so the matrix arithmetic involved in an implicit method becomes a larger fraction of the total work (unless the Jacobian has a structure that allows for very good preconditioning). Thus there is a growing potential for the application of explicit methods that involve little or no matrix manipulation, even if they are slightly less efficient as integrators in terms of function evaluations per unit step.

The paper will concentrate on index-three problems arising from an Euler-Lagrange formulation with constraints. We will show that it is possible to integrate some of these problems with simple explicit projective methods. We will apply regularization techniques that convert the DAE into an ODE which has a slow manifold which is an approximation to the manifold of the DAE. Because

*Received 31 Dec, 2005

we can now start away from the manifold and move quickly to it, these methods also provide a mechanism for determining the unknown initial values that would be needed by a direct method for DAEs. It must be emphasized that we are describing preliminary work that does not yet address issues that arise in some DAE problems.

2 Background.

Two major sources of DAEs are electrical networks and constrained mechanical systems. It has been observed by many people that few, if any, real problems give rise to DAEs, but rather that they occur because of simplifications of the real problem. For example, in electrical networks, Kirchhoff current laws give rise to algebraic relationships, as do many idealizations of devices such as current and voltage sources. If these were modelled as they really existed we would have a differential equation (in fact, we would have a partial differential equation if we included too many details). In mechanical systems with constraints, the constraints are a idealization of reality. The simple pendulum modelled in cartesian coordinates, for example, has a constraint on the pendulum length, whereas any real material will stretch very slightly - and it is this stretching that provides the force that is provided by the Lagrange multiplier in the idealized DAE.

Interestingly, electrical networks were originally modelled with ODEs and a lot of sophisticated techniques were developed to reduce a network to ODEs. (See, for example, [4, 13] However, the ODEs were generally stiff and it was found that implicit methods had to be used for the stiff equations. It was also found that other types of networks could not be reduced to ODEs so we arrived at the idea of solving DAEs directly [7] and the tableau approach to network problems [10]. Initially these were index one problems, so did not present the DAE difficulties of higher index problems, but newer modelling appraoches have lead to index two problems (e.g., [14]) and the problems have become so large and non-linear that some aspects of them, such as finding consistent initial conditions, are extremely challenging.

Mechanical systems with constraints usually lead to index three problems that cannot be solved directly. The constraints in a DAE restrict the solution to a manifold and usually we cannot easily find the ODE on that manifold. Instead, two other approaches have been used. Either the index has been reduced by differentiation (which gives an ODE or lower index DAE on a larger region of space containing the manifold) followed by some method applied to prevent drift of the computed solution off the manifold (such as projection back onto it [3, 11]) or a regularization technique has been applied. Regularization techniques extend the equations to the whole space such that the constraint manifold (or an approximation to it) is a slow manifold of the reduced system. Many regularization techniques have been proposed. Baumgarte's regularization [5] has the appealing feature that the slow manifold is precisely the constraint manifold so, in principle, the solution is not changed (although numerical errors will move the solution off the manifold slightly). However, as we have noted, the real problem does not satisfy the constraints exactly either. Ideally, in a regularized problem,

the fast solution would decay to the slow manifold very rapidly. In fact, if the slow manifold is not the constraint manifold, its distance from the constraint manifold is usually order of the time constant of the fast manifold (because we have effectively created a singularly perturbed system) so that the faster the decay to the slow manifold, the better the approximation provided by the slow manifold. This is a reason why, except in Baumgarte's regularization, we prefer to have a fast return to the slow manifold. However, as [2] point out, if the rate of return to the manifold is fast compared to the step size, all of the ill-posedness problems of the DAE can come back to haunt us.

We propose to use projective integration methods ([8])to solve regularized DAEs explicitly. In projective integration a few explicit steps are taken with a small step size (the *inner steps*) to damp fast components and get close to the slow manifold and then a large explicit step (the *outer step*) commensurate with the slow manifold behavior is taken. Once we are near the slow manifold, the large step projective integrator step should not cause a large deviation from the slow manifold, so the next small step need only focus on damping the deviation, not accurately tracking its return. For this reason, a forward Euler method with step size equal to the fast time constant is ideal since $1 + h\lambda$ is zero and the component is damped immediately (at least, for a linear problem). Because the inner step is of the order of the fast time constants, we avoid the problems mentioned by [2]. Thus the idea for the use of projective integrators with regularization is to regularize with known very fast time constants and then use an inner step to damp these very efficiently.

In electrical networks there are normally other "stiff" components that one needs to retain for accurate modelling and which have to be tracked accurately during phases of the computation when the relevant components are active. At other times these lead to stiffness, but often with a known eigenvalue. In that case, telescopic methods ([9]) can be used to damp multiple clusters. There are also issues of how to regularize the DAE in a physically realistic way by including additional capacitances and it will probably be necessary to revert to some of the former network analysis techniques to derive ODEs. We will not pursue these problems in this paper. Rather we will look at index-three mechanical systems and examine regularizations that are compatible with projective integration.

3 Regularization and Damping.

Ideally we would like to regularize the system by modeling the small perturbations that are present in a real system. This would guarantee us that the model is physically realistic. Note that this regularization, unlike that of Baumgarte, changes the solution slightly. We will illustrate it with the simple pendulum:

$$x' = v \tag{3.1}$$

$$v' = f - G^T\lambda \tag{3.2}$$

$$0 = \sqrt{x_1^2 + x_2^2} - 1 = g(x) \tag{3.3}$$

where x, v and f are 2-dimensional vectors and $G = g_x \approx x^T$. The Baumgarte regularization replaces the eq. (3.3) with

$$g'' + \alpha_1 g' + \alpha_2 g = 0 \tag{3.4}$$

where α_1 and α_2 are chosen to make $g = 0$ a stable solution. If we choose $\alpha_1 = 2\beta$ and $\alpha_2 = \beta^2$ then we have a double eigenvalue at $-\beta$ which is often recommended. However, this regularization, which can also be viewed as a reduction to index one (via the two differentiations of g followed by stabilization) does not put it in an ODE form suitable for projective integration. A physically realistic regularization is to view the "connecting rod" of the pendulum as a very "stiff" spring (this is a different use of the word "stiff" since it refers to a large spring constant, not its damping properties). In this case, we need to define the length of the rod, $L = \sqrt{x_1^2 + x_2^2}$ and to replace eq. (3.3) by

$$\lambda = E(L - 1)$$

where E is a large spring constant. Since the true solution of the problem will now permit fast oscillations in the length of the rod, we might also want to damp them by using instead

$$\lambda = E(L - 1) + FL'$$

where F is the "friction" coefficient of the rod which absorbs some of the energy from any changing of length.

The advantage of the above approach is that, since it specifies a value of λ, it yields an ODE system to which projective integration can be applied directly. Indeed, if we choose $E = \beta^2$ and $F = 2\beta$ we will find that we have introduced a double eigenvalue at $-\beta$ which would seem to be ideal for projective integration. However, a direct application of the method with an inner step size of $-1/\beta$ will fail miserably. By looking at the reason for the failure, we can see other approaches that will work (and perhaps ways to make this approach work).

To look at the issue, we will consider an even simpler problem - an index 3 DAE that has no degrees of freedom. In this example, all variables are scalars.

$$\begin{aligned} x' &= v \\ v' &= -G^T\lambda \\ 0 &= x - p(t) = g(x) \end{aligned} \tag{3.5}$$

In this example, $G^T = 1$. Using the approach suggested above, we replace eq. (3.5) with

$$\lambda = E(x - p(t)) + F(v - p'(t)) \tag{3.6}$$

Since p does not really play a role here except to give a non-zero answer, we will ignore it in the discussion below. If the look at the modified ODE we get

$$\begin{bmatrix} x' \\ v' \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -E & -F \end{bmatrix} \begin{bmatrix} x \\ v \end{bmatrix} \tag{3.7}$$

and, if we choose $E = \beta^2$ and $F = 2\beta$, we have a double eigenvalue of the Jacobian, J, at $-\beta$. $J + \beta I$ is nilpotent with index 2, which means that two steps of forward Euler with step size $1/\beta$ will annihilate the errors. However, the first such forward Euler step introduces an amplification of any error in x by $E/\beta = \beta$ which is large. If the problem is nonlinear the Jacobian at the second step has changed by a large amount and we will not get any damping. It is possible that one might get around this by "freezing" the Jacobian for the two steps, but that implies an ability to modify the equations in some way that may prove to be computationally expensive. Instead, we consider other regularizations.

The difficulty that a simple explicit ODE method like forward Euler is going to have with equations in the form eq. (3.3) is that once x has wandered off the constraint, at least two steps are needed to move it back since the constraining force only enters into the velocity derivative. Hence, if a one-stage explicit method is going to move x back to the constraint in one step there will have to be a regularization term in the expression for x'. Hence we consider a regularization of the form

$$x' = v - G^T\lambda_1, \quad v' = f - G^T\lambda_2 \tag{3.8}$$

where λ_i will be chosen to return any drift back to the manifold. The obvious choices are to use deviations of g and g' from zero as returning "forces," so we consider

$$\lambda_i = E_{i1}g + E_{i2}g'$$

With this definition of λ_i, the Jacobian of eq. (3.8) is

$$\begin{bmatrix} -E_{11} & 1 - E_{12} \\ -E_{21} & -E_{22} \end{bmatrix}$$

Since we would like to place both eigenvalues at $-\beta$ and not have elements larger than the order of β we naturally consider $E_{11} = E_{22} = \beta$ and $E_{12} = E_{21} = 0$.

Applying this approach to the general form of eqs. (3.1), (3.2) and (3.3), namely

$$\begin{aligned} x' &= v \\ v' &= f - G^T\lambda \\ 0 &= g(x) \end{aligned} \tag{3.9}$$

where now x and v are n-dimensional and g is m-dimensional ($m \leq n$), we will replace these equations with

$$\begin{aligned} x' &= v - \beta G^T(GG^T)^{-1}g(x) \\ v' &= f - \beta G^T(GG^T)^{-1}g'(x) \end{aligned} \tag{3.10}$$

If β is large, this forces the solution onto a manifold where g and g' are correspondingly small, which is hence close to the manifold of the DAE. (The $(GG^T)^{-1}$ scaling is chosen because

$$G^T(GG^T)^{-1}\frac{\partial g(x)}{\partial x}$$

is idempotent which makes the additional eigenvalues close to $-\beta$.)

The ODE in eq. (3.10) is now of order $2n$ which is $2m$ larger that the order of the original DAE in eq. (3.9). Its $2m$ additional eigenvalues are asymptotically close to $-\beta$ as β gets large. While this would not be particularly nice system for a conventional stiff solver, it is very well suited to projective integration as will be discussed in the Section 4.

An alternative formulation is to set $E_{11} = E_{22} = 0$ and $-E_{12} = E_{21} = \beta$ giving

$$\begin{aligned} x' &= v + \beta G^T (GG^T)^{-1} g'(x) \\ v' &= f - \beta G^T (GG^T)^{-1} g(x) \end{aligned} \tag{3.11}$$

This adds m pairs of eigenvalues almost at $\pm\beta$ on the imaginary axis. This may seem a curious thing to do, but, as we will see in the Section 4, we can arrange for the inner steps of projective integration to damp these components.

4 Implementation and Numerical Tests.

The formulations given in eq. (3.10) and eq. (3.11) require the computation of the term $G^T(GG^T)^{-1}z$ for various vectors z. This can be accomplished by orthonormalizing the rows of G. Suppose that S is an m by m matrix such that $N = SG$ has orthonormal rows. Then

$$\begin{aligned} G^T(GG^T)^{-1}z &= G^T S^T S^{-T} (GG^T)^{-1} S^{-1} Sz \\ &= G^T S^T (SGG^T S^T)^{-1} Sz \\ &= N^T Sz \end{aligned} \tag{4.1}$$

In other words, we simply orthonormalize G to get N and at the same time, modify z. Then we multiply the modified z by N^T. If the constraints are redundant it will show up in the orthonormalization when we are left with nearly zero rows of the transformed matrix. As long as the modified z is consistent, the process simply drops the remaining rows.

To thoroughly evaluate the proposed method we will need to implement an automatic code (as well as provide further theory). For the purposes of demonstrating that the approach has sufficient potential to be worth additional exploration we made a minimal modification to an off-the shelf ODE package, specifically `ode113` in Matlab. This is a variable order Adams code for non stiff equations. When requested to integrate over an interval, `ode113` calls a function to evaluate the derivatives at specified "y" values (in this case the values of the vectors x and v). We wrote a derivative evaluation function that first performed the appropriate inner steps and then calculated the derivative. During these inner steps the inner integrator changes the state variables (x and v) so they are in the vicinity of the slow manifold. Hence it is necessary to change the corresponding "y" values inside `ode113`. This was done by adding a return argument to the derivative evaluation function that returned the new "y" values, and modifying `ode113` so that it changed its internal "y" values - but only after a step was accepted. Because `ode113` is an Adams-based code, all of the error estimation and

step and order changing computations are done on the derivatives, so changing the "y" values does not impact the logic of the code.

For the regularization given in eq. (3.10) we used two inner forward Euler steps of length $1/\beta$. These advance "time" a short amount unbeknownst to `ode113`. Since we are using β values of around 10^{-7} or smaller on a problem where the typical outer time step is about 10^{-2} to 10^{-1}, it is not a significant amount and we did not correct for it (although a good code should take care of that).

The regularization given in (3.11) requires a different inner integrator. Clearly it would not be computationally efficient to use imaginary steps sizes (neither would it work) but we need an inner integrator that will damp complex eigenvalue pairs. This can be done with a two-stage Runge-Kutta-like process. The following integrator:

$$\begin{aligned} k_1 &= f(y) \\ k_2 &= f(y + hk_1) \\ y_{new} &= y + h(k_2 - k_1) \end{aligned} \tag{4.2}$$

has a stability polynomial for the test equation $y' = \lambda y$ of $1 + (h\lambda)^2$. This vanishes for $h\lambda = \pm\mathbf{i}$ as desired. (Also, it does not advance time!) The tests mentioned below used two iterations of this process in the inner integration and then evaluated a derivative for return to `ode113`. Two problems were run for the tests reported below.

Problem 1 This was a simple pendulum of length 1 started from a position at 30 degrees from the vertical with zero velocity. For comparison the "true" solution was computed for the reduced equation for the angular displacement, also using `ode113`.

Problem 2 This consisted of a set of M pendula hanging from equally spaced points on a horizontal circle of radius 1. Each pendulum weight was connected to its neighbor by a rod such that, when at rest, the pendula weights were equally spaced around a circle of radius 0.5. The lower circle was 2 units below the upper circle.

In problem 2 we did not specify a consistent initial state (it is not easy to find other than the rest state!) Instead, $M-1$ pendula were placed in their rest position with identical initial velocities (these violate $g' = 0$), while the last was placed in the center of the lower circle (which violates $g = 0$ for all rods to which it is attached). The first inner integration steps moved the pendula to consistent positions and velocities.

The first problem was integrated by five methods. Method S1 used the regularization of eq. (3.5) using eq. (3.6) and was integrated using a stiff solver from Matlab (`ode15s`) with $E = \beta^2$, $F = 2\beta$ and $\beta = 100$. (Larger β leads to failure of the integrator.) Method S2 used the regularization in eq. (3.10). It was also integrated with `ode15s` with $\beta = 10^4$. (Again, larger values lead to integrator failure.) Projective method P1 used the regularization (3.10) and was integrated with the modified `ode113` using projective integration and various values of β from 10^7 to 10^{10} while projective method P2 used the regularization

in eq. (3.11) with projective integration and the two-stage RK inner integrator described above with β ranging from 10^5 to 10^7. The different values of β in the projective integration tests had insignificant impact on the running time or the max error so we only report the results for the larger β values. The fifth method, labelled B, was Baumgarte's stabilization with $\alpha_1 = 200$ and $\alpha_2 = 10,000$ in eq. (3.4).

The second problem was run with seven pendula using the first three methods described above. (The fourth method does not currently look as promising.) Since we do not have the "true" solution, all that was done was to compute the energy loss over a time interval of 100 and the Matlab execution time.

In making these tests, the default values for all parameters in the Matlab codes were used (except for the computation of the "true" solution of Problem 1 for comparison purposes which used AbsTol $= 10^{-9}$ and RelTol $= 10^{-8}$). While the default values may not be a sensible choice, this work is still in its early stages, and there seems to be little reason to be fine tuning codes at this point (and answers obtained after endless tweaking of parameters are suspect as tests of an idea). The integration was run for 100 time units which covered almost 15 periods of problem 1. The max error (compared to the computed solution) for each of the cases and the computation time is shown in Table 4.1. It should be noted that the bulk of the run time for small problems like these is overhead, so that these results only indicate that the methods have potential, not that they are faster. The larger errors of the stiff methods are also more indicative that the stiff methods are not good methods for these problems rather than the merits of the proposed schemes.

Table 4.1: Run time and Errors for Problem 1.

Method	R1	R2	P1	P2	B
Max Error	0.0678	0.0203	0.00026	0.0063	0.00080
Run Time	2.156	3.453	0.516	0.532	3.797

The results are shown in Table 4.2 for problem 2.

Table 4.2: Run time and energy change for Problem 2.

Method	R1	R2	P1
Energy change	-0.0032	0.0055	-0.00022
Run Time	432.1	1079	31.5

REFERENCES

1. A. Abdulle, *Fourth order Chebyshev methods with recurrence relation*, SIAM J. Sci. Comput. 25 (2002) pp. 2041–2054.

2. U. M. Ascher, H. Chin, and S. Reich, *Stabilization of DAEs and invariant manifolds*, Numer. Math. 67 (1994), pp. 131–149
3. U. M. Ascher and L. R. Petzold, *Projected implicit Runge-Kutta methods for differential-algebraic equations*, SIMA J. Num. Anal. 28 (1991) pp. 1097–1120
4. T. R. Bashkow, *The A matrix, new network description*, IRE Trans. Circuit Theory, CT-4 (1957), pp. 117–120
5. J. Baumgarte, *Stabilization of constraints and integrals of motion in dynamical systems*, Comp. Math. Appl. Mech. Engrg., 1 (1976), pp. 1-16.
6. K. Eriksson, C. Johnson, A. Logg, *Explicit time-stepping for stiff ODEs*, SIAM J. Sci. Comput. 25 (2003) pp. 1142–1157.
7. C. W. Gear, *Simultaneous Numerical solution of differential-algebraic equations*, IEEE Trans. Circuit Theory, CT-18 (1971), pp. 89–95
8. C. W. Gear, I. G. Kevrekidis, *Projective methods for stiff differential equations: Problems with gaps in their eigenvalue spectrum*, SIAM J. Sci. Comput. 24 (2003) pp. 1091–1106.
9. C. W. Gear, I. G. Kevrekidis, *Telescopic projective methods for parabolic differential equations*, J. Comput. Phys. 187 (2003) pp. 95–109.
10. G. D. Hachtel, R. K. Brayton, and F. G. Gustavson, *The sparse tableau approach to network analysis and design*, IEEE Trans. Circuit Theory, CT-18 (1971) pp. 101–113
11. E. Hairer, *Symmetric projection methods for differential equations on manifolds*, BIT, 40 (2000), pp. 726–735.
12. M. Hanke, *A note on the consistent initialization for non-lineaar index-2 differential-algebraic equations in Hessenberg form*, Report 2004:02, Parallel and Scientific Computing Institute, Roy. Inst. Tech., Uppsala, Sweden, 2004
13. E. S. Kuh, R. A. Rohrer, *The state-variable approach to network Analyis*, Proc IEEE 53 (1965) pp. 672–686
14. R. Marz and C Tischendorf, *Recent results in solving index-2 differential-algebraic equations in circuit simulation*, SIAM J. Sci. Comput. 18 (1997) pp. 139-159
15. A. A. Medovikov, *High order explicit methods for parabolic equations*, BIT 38 (1998) pp. 372–390.
16. B. P. Sommeijer, L. F. Shampine, J. G. Verwer, *RKC: An explicit solver for parabolic PDEs*, J. Comput. Appl. Math. 88 (1998), pp. 315–326.
17. P. J. van der Houwen, *The development of Runge-Kutta methods for partial differential equations*, Appl. Num. Math. 20 (1996) pp. 261–272.