INTEGRATION ALGORITHMS FOR THE DYNAMIC SIMULATION OF PRODUCTION PROCESSES.

M. Berzins , P. M. Dew , A. J. Preston.

ABSTRACT

The essential features needed to construct an efficient integrator for dynamic simulation problems arising in the gas/ oil industry are examined. A series of numerical experiments are performed using two widely used codes: DASSL [8] and SPRINT [1], that are designed to solve differential-algebraic equations. From these experiments, a number of improvements to the codes have been identified and incorporated in a new integrator.

1. Introduction.

The development of general purpose codes, such as SPRINT [1] and DASSL [8], for the numerical solution of differential-algebraic equations (d.a.e.s), has made it possible to efficiently solve many of the routine equations that arise in modelling dynamic gas and oil networks [3]. Nevertheless, there still exists a number of important networks that cannot be adequately handled by existing integrators. These problems can be classified as Index two (or higher) d.a.e.s (see below). A number of numerical experiments have been performed using both DASSL [8] and SPRINT [1], and from these the following areas where problems may arise, have been identified:

(i) the solution of the non-linear equations that arise when using an implicit time-integration scheme;

(ii) the accurate estimation of the local error for the fixed or variable stepsize form of the backward differentiation formulae (b.d.f.) of Gear [4];

(iii) the stepsize and order selection mechanism used by the integrators, which are based on b.d.f. formulae.

This paper provides a summary of the work that has been undertaken in the above and related areas. For further details, the interested reader is referred to the companion report, [2].

The general class of differential-algebraic equations is defined by

(1) $\underline{f}(t\ ,\ \underline{y}(t)\ ,\ \dot{\underline{y}}(t)) = \underline{0}$ given constants $\underline{y}(0)\ ,\ \dot{\underline{y}}(0)$ and where $\underline{f} : [t_0\ ,\ T] \times \mathbf{R}^n \times \mathbf{R}^n \rightarrow \mathbf{R}^n$

In the dynamic simulation problems considered in this paper, the function $\underline{f}$ can be written in the form :

J. Manley et al. (eds.), Proceedings of the Third European Conference on Mathematics in Industry, 265–271.

(2) $\underline{f} = A(t, \underline{y})\,\dot{\underline{y}} - \underline{g}(t, \underline{y})$, where A is a square matrix and can be singular.

A classification of d.a.e.s, the index of the system of equations, has been introduced by Gear and Petzold [5]. For details on the definition of the index and its implications see [2] and [6]. This paper is concerned with the *standard index two* problem [6] pp. 24-27, defined by

$$y_1 = \phi(t) \qquad t \in [t_0, T], \quad \phi \in C^2[t_0, T] \rightarrow \mathbf{R}$$

(3) $$y_2 = \dot{y}_1 \qquad \underline{y} = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} : [t_0, T] \rightarrow \mathbf{R}^2, \quad \underline{y}(0) = \begin{bmatrix} \phi(0) \\ \dot{\phi}(0) \end{bmatrix}$$

where we have chosen $\phi(t) = \sin(2\pi t)$, $t \in [0, 3]$.

Practical experiments conducted by the authors have shown that DASSL [8] is one of the most robust codes for solving index two problems, although it is far from being efficient. The purpose of this paper is to carry out a series of numerical experiments which compare the performance of DASSL [7] with that of SPRINT [1] when solving the *standard index two* problem. A number of recommendations are given to improve the performance of SPRINT [1] for this class of problems.

2. An Investigation of the Factors Affecting the Performance of DASSL.

In order that the particular features which make DASSL [8] successful in solving d.a.e.s. are understood, the following aspects are investigated:

(i) the form of the non-linear equations to be solved, and the scaling of these equations;

(ii) the error estimates based on [6] and the stepsize and order selection algorithm.

The results were obtained using finite differencing to compute the partial derivatives of the Jacobian matrix. Details of the statistics used to measure code performance are given in the Appendix. The number of evaluations of the residual vector $\underline{r}$, not counting those used in decomposing the Jacobian, is defined (in the case of *the standard index two d.a.e*) at the nth accepted step by

$$\underline{r}_n = \begin{bmatrix} y_{1,n} - \phi(t_n) \\ y_{2,n} - \dot{y}_{1,n} \end{bmatrix}$$

where $y_{1,n}$ denotes the computational approximation to the exact value $y_1(t_n)$ and similarly for $y_{2,n}$ and $\dot{y}_{1,n}$. The weighted vector norm of estimated local errors in $\underline{y}$, $\underline{le}$, is controlled at each step by means of a standard mixed error test and using the *averaged l_2 norm*, [2].

2.1. The Form of the Non-Linear Equations.

DASSL [8] uses the kth order b.d.f. of Gear ($k = 1,2,3,4,5$) to approximate $\dot{\underline{y}}_n$ by $\dot{\underline{y}}_n = \dfrac{\underline{y}_n}{h_o \gamma_o} + \sum_{i=1}^{k} \alpha_i \underline{y}_{n-i}$, where h_o is the stepsize used on the previous step and $\gamma_0, \alpha_1, \alpha_2, \ldots, \alpha_k$ are the method dependent parameters.

The rate of convergence of the non-linear equations can be accelerated by using relaxation techniques. In DASSL [8] the relaxation factor used is ρ (see below), so that the j+1th iterative value of Newton's method, $\underline{y}_n^{(j+1)}$, is found by solving for the correction $\Delta \underline{y}_n^{(j)}$:

$$\left[\frac{\partial f}{\partial \underline{y}} + \frac{1}{h_o \gamma_o}\frac{\partial f}{\partial \dot{\underline{y}}}\right]\left[\Delta \underline{y}_n^{(j)}\right] = -\underline{f}\left[t_n\ ,\ \underline{y}_n^{(j)}\ ,\ \frac{y_n^{(j)}}{h\gamma} + \sum_{i=1}^{k}\alpha_i \underline{y}_{n-i}\right] \times \rho$$

$$\underline{y}_n^{(j+1)} = \underline{y}_n^{(j)} + \Delta \underline{y}_n^{(j)}\ ,\ j = 1,2,3,...$$

where $\rho = \left[\dfrac{2\, h_c\, \gamma_c}{h_o\, \gamma_o\ +\ h_c\, \gamma_c}\right]$ and the subscript c denotes current values, while the subscript o denotes old values. A further modification to the way in which the non-linear equations are solved is given by Petzold and Lösted [9]. In this case, if the jth equation contains derivatives, then the factor $2\, h_c\, \gamma_c$ in ρ is replaced by $h_o\gamma_o$ and a scaling of $\rho = 1$ is applied to the algebraic constraints.

An improvement to the above is to adopt the approach used in SPRINT[1], but select the relaxation factors by the following procedure. The row(s) of the iteration matrix representing the differential equation(s) are multiplied by $h_o\gamma_o$ and the corresponding residual entry is multiplied by $h_c\gamma_c$. This is expected to lead to a faster convergence of the iterates. The system of equations to be solved is then given by

$$\begin{bmatrix} -1 & 0 \\ 1 & -h_o\gamma_o \end{bmatrix}\left[\Delta^* \underline{y}_n^{(j)}\right] = \begin{bmatrix} y_{1,n}^{(j)} - \phi\ (t_n) \\ \dfrac{2h_o\gamma_o\ h_c\gamma_c\ (y_{2,n}^{(j)} - \dot{y}_{1,n}^{(j)})}{h_o\gamma_o + h_c\gamma_c} \end{bmatrix}$$

$$\underline{y}_n^{(j+1)} = \underline{y}_n^{(j)} + \Delta^*\, \underline{y}_n^{(j)}\ \ j = 1,2,3,...$$

$$\underline{y}_n^{(1)} = \underline{y}_n^{(p)} + \Delta^*\, \underline{y}_n^{(0)}$$

By comparing the results in Tables 1 and 2a, a large improvement to the errors in y_1 for the *standard index two* problem can be observed. Indeed, the algebraic equation is now satisfied to within unit round-off error. It should be noted that although there is little improvement to the errors in $\dot{y}_1$, there is some improvement for the errors in $\dot{y}_2$ and also a considerable reduction in the amount of work performed, compared with DASSL [8]. However, the results show that the maximum global error in $\dot{y}_2$ is still well beyond accepted bounds.

2.2. Estimating the Local Error.

In DASSL [8], after convergence of the iterates has been achieved, the error estimates at orders k , k–1 , k–2 are given by $\underline{e} = \alpha^*\ (\underline{y}_n - \underline{y}_n^{(p)})$ where the superscript p denotes the predicted value and where $\alpha^* = \alpha^*\ (h\ ,\ h_1\ ,\ h_2\ ,\ \cdots\ ,\ h_k)$ depends on the current stepsize, h, and on the previous k stepsizes at order k, In particular, at order one, the value of α^* is given by $\alpha^* = \dfrac{h}{h + h_1}$.

Clearly, if $[t_{n-1}\ ,\ t_n]$ spans a discontinuity or sudden change in gradient the error test can always be satisfied in DASSL [8] at order 1, since the code can always select a small enough step size. However α^* does not have a convenient form for any of the higher orders. In contrast, the error estimate used in the codes of Hindmarsh [7] and in the b.d.f. SPGEAR module of SPRINT [1] is given by $\underline{E} = \alpha\,(\,\underline{y}_n - \underline{y}_n^{(p)})$ where α is a constant which, at order one, has the value $\alpha = \frac{1}{2}$. In this case, if $[t_{n-1}\ ,\ t_n]$ spans a discontinuity in $\dot{\underline{y}}$, it is not always possible to satisfy the error test at order one.

The results in Table 2b show that a strong relationship exists between the number of error test failures, the work done by the code, and the order selected. When the number of error test failures is high, the code spends a long time at order one, taking advantage of the stability properties, but in order to achieve the required accuracy, the code takes many steps.

An alternative is to use only the error estimates that arise from the differential equations in the system. The local error estimate, $\underline{le}$, is now defined by $\underline{le} = M^{-1}\ A\ \underline{e}$ rather than $\underline{le} = \underline{e}$ as in SPRINT [1], where M is the iteration matrix used in solving the non-linear equations, and A is the matrix defined by equation (2). These equations are then solved by performing an LU decomposition on M, and using back-substitution.

Tables 2b and 3b show that with this modification to the local error estimate , the new code takes advantage of the efficient high order methods. The number of error test failures has been reduced considerably and there is a drastic reduction in the global error.

2.3. Stepsize and Order Selection.

Suppose that at order k, the estimates of the error at the orders one above and below are denoted by $\underline{e}^{k+1}$ and $\underline{e}^{k-1}$. Then in the order selection algorithm of DASSL [8], $\underline{e}^{k-1}$, $\underline{e}^{k}$, and $\underline{e}^{k+1}$, are replaced by $A\ \underline{e}^{k-1}$, $A\ \underline{e}^{k}$, and $A\ \underline{e}^{k+1}$ respectively. This is expensive as it requires three matrix-vector products, each of which costs "half" an o.d.e. residual evaluation in the SPRINT [1] software. However, this appears to be one of the few ways of ensuring that the code selects the correct order. DASSL [8] also tends to make sure that $||\underline{e}^{k-2}|| > ||\underline{e}^{k}||$, to ensure that the order selection algorithm does not force the code to stay at an unnecessarily high order. In the case when this condition has been violated, the order is reduced whenever $||A\ \underline{e}^{k-2}|| < ||A\ \underline{e}^{k}||$.

3. Conclusions and Recommendations.

In this paper and also in [2], a number of important issues concerning the design of an efficient integrator for index two d.a.e.s. have been discussed. In particular, the strategies used in both DASSL [8] and SPRINT [1] concerning the method of solution and scaling of the non-linear equations have been mentioned. The experiments have shown that the form of the non-linear equations used in SPRINT [1] works well, and that scaling of the equations is an important issue. For this reason, the form of the non-linear equations as solved in SPRINT [1] is adopted with the DASSL [8] convergence strategy. The numerical results have also shown that the formulation of

the local error estimate can have a major impact on the efficiency of the d.a.e. integration. Therefore the local error estimate based only on the differential variables is adopted. To ensure that the code selects the correct order, the strategy adopted is identical to that used in DASSL [8], except for an additional criterion based on the comparison of two matrix-vector products.

Applying these modifications has enabled an index two module to be constructed, SPDASL, which can be used within the SPRINT [1] software. The results from this new index two solver are given in Tables 4a and 4b. In fact, a comparison of the results in Table 4a with those in Table 1 shows an extremely large reduction in the work performed by the new code. Table 4b shows that the order selection strategy has worked extremely well, allowing a large proportion of time to be spent at a high order. The errors in $\dot{y}_2$ have been reduced considerably, and the algebraic equations are better satisfied since the errors in $\dot{y}_1$ have been reduced. This indicates a very large improvement over the original code.

We have also tested our module on a dynamic simulation problem supplied by Shell Research Ltd, and the preliminary results are encouraging [2].

Acknowledgements.

We wish to thank Shell Research Limited for permission to publish this paper and for funding the SERC CASE studentship for Andrew Preston. We are also grateful for the help that has been given by S Frost and L Scales of Shell Research Ltd.

REFERENCES.

[1] Berzins M., Dew P.M., and Furzeland R.M. (1986), Developing P.D.E. Software Using The Method of Lines and Differential Algebraic Integrators. High-lighted Talk presented at 1986 O.D.E. Conference, Albuquerque, New Mexico, July 1986 (to appear in Appl. Numer. Math.).

[2] Berzins M., Dew P.M., Preston A.J., (1988), Integration Algorithms for the Dynamic Simulation of Production Processes, Report 88.20, School of Computer Studies, Leeds University, Leeds LS2 9JT.

[3] Capstick M.A., (1987), On Improving the Performance of Dynamic Process Simulators, Ph. D. Thesis, Departments of Chem. Eng. and Comp. Studies, Leeds University, Leeds LS2 9JT.

[4] Gear C.W., (1971), Numerical Initial Value Problems in Ordinary Differential Equations, Prentice Hall, Englewood Cliffs, NJ, U.S.A.

[5] Gear C.W., Petzold L.R., (1984), ODE Methods for the solution of Differential/Algebraic Systems, SIAM Journal on Numerical Analysis, 21, pp. 716-728.

[6] Gupta G.K., Gear C.W., Leimkuhler B., (1985), Implementing Linear Multistep Formulas for solving D.A.E.s, Report UIUCDCS-R-85-1205, Department of Computer Science, University of Illinois, Urbana IL61801.

[7] Hindmarsh A.C., (1981), O.D.E Solvers for use with the method of lines, Advances in Computer Methods for Partial Differential Equations IV, R. Vichnevetsky and R.S. Stepleman, eds., IMACS, New Brunswick, NJ, pp. 312-316.

[8] Petzold L.R., (1982), A Description of DASSL : A Differential-Algebraic System Solver, SAND82-8637, Applied maths division 8331, Sandia National Laboratories, Livermore, California C.A 94550.

[9] Petzold L.R., Lötstedt P., (1983), Numerical solution of Nonlinear Differential Equations with Algebraic Constraints, Report SAND 83-8877, SSISC, Sandia National Laboratories, Livermore, California C.A 94550.

Authors' Address.

School of Computer Studies, Leeds University, LEEDS LS2 9JT.

Appendix.

The tables below illustrate the results obtained for the numerical experiments. For the *standard index two* problem, the initial conditions for the test problem are consistent with the analytic solution. The column headings denoted by *err* y_1, *err* y_2 etc., denote the maximum global error (in y_1 , y_2 etc.) that occurred over the interval $[t_0 , T]$ for all accepted steps using a tolerance value TOL. The column *EF* denotes the number of error test failures and *STEP* the number of steps taken by the integrator. *CALL* records the number of function evaluations for the problem, not counting those used in the decomposition of the Jacobian. *JAC* is the number of decompositions of the Jacobian and *SOLVE* denotes the total number of iterations incurred by the non-linear equations solver. The number of steps taken at each of the five orders is also recorded.

Tab. 1 The standard index two problem using DASSL [8].

TOL	err y_1	err y_2	err $\dot{y}_1$	err $\dot{y}_2$	STEP	CALL	JAC	SOLVE
0.5D-02	0.21D-03	0.92D-01	0.92D-01	0.90D+03	462	1504	521	1504
0.2D-02	0.31D-04	0.17D+00	0.17D+00	0.87D+02	608	2115	644	2115
0.1D-02	0.60D-05	0.28D-01	0.28D-01	0.12D+04	908	3179	1376	3179
0.5D-03	0.26D-05	0.16D-01	0.16D-01	0.13D+03	4936	19381	7638	19381

Tabs. 2a,b The standard index two problem after modifications to the non-linear equation solver.

TOL	err y_1	err y_2	err $\dot{y}_1$	err $\dot{y}_2$	STEP	CALL	JAC	SOLVE
0.5D-02	0.00D+00	0.81D-01	0.81D-01	0.10D+04	331	1042	344	1042
0.2D-02	0.00D+00	0.33D-01	0.33D-01	0.59D+01	369	1278	509	1278
0.1D-02	0.14D-16	0.26D-01	0.26D-01	0.12D+03	1296	4841	2173	4841
0.5D-03	0.00D+00	0.95D-01	0.95D-01	0.27D+02	469	1298	399	1298

TOL	ORD 1	ORD 2	ORD 3	ORD 4	ORD 5	EF
0.5D-02	259	32	28	12	0	176
0.2D-02	290	21	18	22	18	256
0.1D-02	1167	60	36	31	2	1108
0.5D-03	311	66	45	31	16	167

Tabs. 3a,b The standard index two problem after changes made to the estimation of local error.

TOL	err y_1	err y_2	err $\dot{y}_1$	err $\dot{y}_2$	STEP	CALL	JAC	SOLVE
0.5D-02	0.00D+00	0.10D-01	0.10D-01	0.17D+01	66	141	11	344
0.2D-02	0.14D-16	0.67D-02	0.67D-02	0.62D+00	85	198	7	482
0.1D-02	0.14D-16	0.48D-02	0.47D-02	0.62D+00	95	218	7	532
0.5D-03	0.00D+00	0.22D-02	0.14D-02	0.40D+02	123	289	20	676

TOL	ORD 1	ORD 2	ORD 3	ORD 4	ORD 5	EF
0.5D-02	1	1	7	20	37	2
0.2D-02	1	1	15	7	61	10
0.1D-02	1	1	17	7	69	10
0.5D-03	6	12	11	14	80	14

Tabs. 4a,b The standard index two problem modifying the order selection strategy, convergence strategy, and error test in the SPDASL module of SPRINT [1].

TOL	err y_1	err y_2	err $\dot{y}_1$	err $\dot{y}_2$	STEP	CALL	JAC	SOLVE
0.5D-02	0.31D-12	0.51D-01	0.18D-01	0.46D+01	73	291	10	201
0.2D-02	0.12D-11	0.27D-01	0.73D-02	0.43D+01	86	319	11	217
0.1D-02	0.88D-13	0.11D-01	0.54D-02	0.70D+00	99	347	11	232
0.5D-03	0.40D-13	0.41D-02	0.24D-02	0.21D+01	118	425	9	286

RTOL=ATOL	ORD 1	ORD 2	ORD 3	ORD 4	ORD 5	EF
0.5D-02	1	11	15	13	33	3
0.2D-02	1	1	17	24	43	2
0.1D-02	1	1	22	22	53	1
0.5D-03	1	12	23	19	63	5