Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 1 (2012) 2569–2577

Procedia Computer Science

www.elsevier.com/locate/procedia

International Conference on Computational Science, ICCS 2010

A Family of BDF Algorithms for Solving Differential Matrix Riccati Equations Using Adaptive Techniques

Jesús Peinado[1a], Javier Ibañez[a], Vicente Hernández[a], Enrique Arias[b]

[a]*ITACA, Instituto de Telecomunicaciones y Aplicaciones Multimedia, Universidad Politécnica de Valencia, Camino de Vera s/n, 46022-Valencia (Spain)*

[b]*Departamento Sistemas Informáticos, Universidad de Castilla-La Mancha, Avenida España, s/n, 02071-Albacete (Spain)*

Abstract

Differential Matrix Riccati Equations play a fundamental role in control theory, for example, in optimal control, filtering and estimation, decoupling and order reduction, etc. One of the most popular codes to solve stiff Differential Matrix Riccati Equations (DMREs) is based on Backward Differentiation Formula (BDF). In previous papers the authors of this paper showed two algorithms for solving DMREs based on an iterative Generalized Minimum RESidual (GMRES) approach and on a Fixed-Point approach.

In this paper we present two contributions to improve the above algorithms. Firstly six variants of previous algorithms are carried out by using one of above algorithms in the first step and another algorithm to carry out the other steps until reaching convergence. Numerous tests on four case studies have been done comparing both precision and computational costs of MATLAB implementations of the above algorithms. Experimental results show that in some cases these algorithms improve on the speed and convergence of the original algorithms. Secondly, using the previous experimental results and since all algorithms have a similar structure and there is no best algorithm to solve all problems, two general-purpose adaptive algorithms have been designed for selecting the most appropriate algorithm, which can be chosen using a parameter that indicates the stiffness of the DMRE to be solved.

Keywords: Differential Matrix Riccati Equation (DMRE), BDF methods, GMRES methods, Algebraic Matrix Riccati Equation (AMRE), Algebraic Matrix Sylvester Equation (AMSE), Fixed-Point method.

1. Introduction

In this paper we consider DMREs of the form

$$\begin{aligned} \dot{X}(t) &= A_{21}(t) + A_{22}(t)X(t) - X(t)A_{11}(t) - X(t)A_{12}(t)X(t), \\ t_0 &\le t \le t_f, \\ X(t_0) &= X_0 \in \mathbb{R}^{m\times n}, \end{aligned} \tag{1}$$

Email addresses: jpeinado@dsic.upv.es (Jesús Peinado), jjibanez@dsic.upv.es (Javier Ibañez), vhernand@dsic.upv.es (Vicente Hernández), earias@dsi.uclm.es (Enrique Arias)

[1]Corresponding author

1877-0509
doi:10.1016/j.procs.2010.04.290

Table 1: Parameters of BDF method (order r=1, 2, 3, 4 and 5)

r	β	α_1	α_2	α_3	α_4	α_5
1	1	1				
2	2/3	4/3	-1/3			
3	6/11	18/11	-9/11	2/11		
4	12/25	48/25	-36/25	16/25	-3/25	
5	60/137	300/137	-300/137	200/137	-75/137	12/137

where $A_{11}(t) \in \mathbb{R}^{n\times n}$, $A_{12}(t) \in \mathbb{R}^{n\times m}$, $A_{21}(t) \in \mathbb{R}^{m\times n}$, $A_{22}(t) \in \mathbb{R}^{m\times m}$.

DMREs arises in several applications, in particular in Control Theory, for example the Time-Invariant Linear Quadratic Optimal Control Problem. Another application of (1) consists of solving a two point boundary value problem by decoupling this problem in two initial value problems [1]. Since the mid seventies, many different methods have been proposed, being the BFD methods the most popular codes to solve stiff Differential Matrix Riccati equations.

This paper is organized as follows. First, Section 2 describes a family of BDF methods for solving DMREs based on three approaches: matrix Sylvester equations [1, 2], GMRES method [3] and Fixed-Point iteration [4]. Section 3 explains all developed algorithms. In Section 4, four case studies and the results obtained are presented. From experimental results, an adaptive algorithm is explained in Section 4. Finally, the conclusions and future work are outlined in Section 5.

2. A family of BDF methods

In this section we will describe a family of BDF methods for solving DMREs. In a BDF scheme, the integration interval $[t_0, t_f]$ is divided so that the approximate solution at t_k, X_k, is obtained by solving an AMRE. Several methods have been implemented for solving AMREs, however, in the context of stiff DMREs, one of the better choices for solving the associated AMRE is to apply implicit schemes based on Newton's or quasi-Newton methods. Let $F(t, X)$ be the right hand side of (1),

$$F(t, X) = A_{21}(t) + A_{22}(t)X(t) - X(t)A_{11}(t) - X(t)A_{12}(t)X(t).$$

If we consider a mesh $t_0 \le t_1 \le t_2 \ldots \le t_f$ of interval $[t_0, t_f]$ and we apply a BDF scheme, then the approximate solution X_k at t_k is obtained by means of solving the following matrix equation

$$-X_k + \sum_{j=1}^{r} \alpha_j X_{k-j} + \Delta t_{k-1}\beta F(t_k, X_k) = 0, \tag{2}$$

where $\Delta t_{k-1} = t_k - t_{k-1}$, and α_j $(j = 1, 2, \ldots, r)$ and β are values that appear in Table 1, being r the order of BDF method. Equation (2) can be expressed as the AMRE

$$\bar{A}_{21} + \bar{A}_{22}X_k + X_k\bar{A}_{11} + X_k\bar{A}_{12}X_k = 0, \tag{3}$$

where

$$\begin{aligned}
\bar{A}_{21} &= -\beta\Delta t_{k-1}A_{21}(t_k) - \sum_{j=1}^{r} \alpha_j X_{k-j},\\
\bar{A}_{22} &= -\beta\Delta t_{k-1}A_{22}(t_k) + I_m,\\
\bar{A}_{11} &= \beta\Delta t_{k-1}A_{11}(t_k),\\
\bar{A}_{12} &= \beta\Delta t_{k-1}A_{12}(t_k).
\end{aligned}$$

2.1. A Sylvester method

This approach was used by Dieci in [1] and by Choi and Laub in [2]. Equation (3) can be solved by the Newton's iteration

$$\begin{aligned} \hat{G}'_{(l)}(X_k^l - X_k^{l-1}) &= -G(X_k^{l-1}), l \geq 1, \\ X_k^0 &= X_{k-1}, \end{aligned} \tag{4}$$

where $\hat{G}'_{(l)}$ is the Fréchet derivative [5, p. 310] of

$$G(X) = \bar{A}_{21} + \bar{A}_{22}X + X\bar{A}_{11} + X\bar{A}_{12}X.$$

For a fixed l the following AMSE is obtained:

$$C_{22}^{l-1}\Delta X_k^{l-1} + \Delta X_k^{l-1}C_{11}^{l-1} = C_{21}^{l-1}, \tag{5}$$

where

$$\begin{aligned} C_{22}^{l-1} &= \bar{A}_{22} + X_k^{l-1}\bar{A}_{12}, \\ C_{11}^{l-1} &= \bar{A}_{11} + \bar{A}_{12}X_k^{l-1}, \\ C_{21}^{l-1} &= -\bar{A}_{21} - \bar{A}_{22}X_k^{l-1} - X_k^{l-1}C_{11}^{l-1} \end{aligned}$$

and $\Delta X_k^{l-1} = X_k^l - X_k^{l-1}$.

Therefore X_k^l can be obtained by solving (5) for ΔX_k^{l-1} and computing

$$X_k^l = \Delta X_k^{l-1} + X_k^{l-1}.$$

The standard solution process for (5) is the Bartels-Stewart method [6].

2.2. A Fixed-Point method

This method has been developed in [4]. From (3) we obtain

$$\bar{A}_{21} + X_k\bar{A}_{11} + (\bar{A}_{22} + X_k\bar{A}_{12})X_k = 0,$$

$$(\bar{A}_{22} + X_k\bar{A}_{12})X_k = -(\bar{A}_{21} + X_k\bar{A}_{11}),$$

which allows to define the following Fixed-Point iteration

$$\begin{aligned} (\bar{A}_{22} + X_k^{l-1}\bar{A}_{12})X_k^l &= -(\bar{A}_{21} + X_k^{l-1}\bar{A}_{11}), l \geqslant 1, \\ X_k^0 &= X_{k-1}. \end{aligned} \tag{6}$$

Similarly, from (3) we obtain the Fixed-Point iteration

$$\begin{aligned} X_k^l(\bar{A}_{11} + \bar{A}_{12}X_k^{l-1}) &= -(\bar{A}_{21} + \bar{A}_{22}X_k^{l-1}), l \geqslant 1, \\ X_k^0 &= X_{k-1}. \end{aligned} \tag{7}$$

2.3. A GMRES method

This method was developed in [3] by the authors of this article. If we apply the vec operator [7, pp. 20] to (5), then

$$\left[I_n \otimes C_{22}^{l-1} + (C_{11}^{l-1})^T \otimes I_m\right]\Delta x_k^{l-1} = \mathrm{vec}(C_{21}^{l-1}).$$

This linear system can be solved efficiently without explicitly building the matrix $I_n \otimes C_{22}^{l-1} + (C_{11}^{l-1})^T \otimes I_m$ by the GMRES method, and then

$$X_k^l = X_k^{l-1} + \mathrm{mat}(\Delta x_k^{l-1}, m, n).$$

3. A family of BDF algorithms for solving DMREs

The authors of this paper developed three basic algorithms for solving AMREs: algorithm `dgearesyl` based on a Sylvester method (Algorithm 1 of [4]), algorithm `dgearefpo` based on a Fixed-Point method (Algorithm 2 of [4]) and algorithm `dgearegmr` based on a GMRES method (Algorithm 3 of [3]). In this paper we have modified these algorithms to improve speed and convergence. The idea consists of combining two BDF algorithms applying a first step using a method, and then, applying several steps using a second method making the necessary iterations (steps) to reach convergence. The six algorithm developed are (see Figure 1):

1. Apply a Sylvester iteration and the Fixed-Point method (`dgearesfp`).
2. Apply a GMRES iteration and the Fixed-Point method (`dgearegfp`).
3. Apply a Fixed-Point iteration and the Sylvester method (`dgearefsy`).
4. Apply a Fixed-Point iteration and the GMRES method (`dgearefgm`).
5. Apply a GMRES iteration and the Sylvester method (`dgearegsy`).
6. Apply a Sylvester iteration and the GMRES method (`dgearesgm`).

Another contribution of this paper is the development of a general algorithm which allows to choose "the best algorithm" for each DMRE problem depending on the stiffness of problem. With this scheme up to nine different BDF algorithms can be applied.

Therefore there is a driver algorithm for the time-invariant case `dgeidrbdf` (not shown) and another for the time-varying case (Algorithm 1) `dgevdrbdf`. This algorithm solves a time-varying DMRE by calling other algorithms to solve AMREs. The computational algorithms solve AMREs by the nine algorithms developed.

Algorithm 1 Solves DMREs by means of a BDF algorithm

Function $\{X_k\}_{k=1}^{p}$ = `dgevdrbdf`$(alg, data, t_0, X_0, t_f, \Delta t, tol, maxiter)$
Inputs: *alg* is the algorithm used to solve the AMRE associated to DMRE (Figure 1); *data(t)* is the function that computes the coefficient matrices of (1) at instant t; initial time $t_0 \in \mathbb{R}$; starting guess matrix $X_0 \in \mathbb{R}^{m\times n}$; final time $t_f \in \mathbb{R}$; step size $\Delta t \in \mathbb{R}$; order of BDF method $r \in \{1, 2, 3, 4, 5\}$; $tol \in \mathbb{R}^+$ is the tolerance used in BDF method; $maxiter \in \mathbb{N}$ is the maximum number of Newton iterations in BDF method
Outputs: Matrices $X_k \in \mathbb{R}^{m\times n}$, $k = 1, 2, \ldots$

Initialize α and β with the values given in Table 1
$t = t_0$
while $t < t_f$ **do**
  if $\|X_k - X_{k-1}\|_\infty < \varepsilon_1 \|X_k\|_\infty$ **then**
    $\Delta t = \delta \Delta t$
  else if $\|X_k - X_{k-1}\|_\infty > \varepsilon_2 \|X_k\|_\infty$ **then**
    $\Delta t = \max(\Delta t/\delta, \varepsilon_3)$
  end if
  $s = \min(r, k)$
  $[A_{11}, A_{12}, A_{21}, A_{22}] = data(t)$
  $\bar{A}_{21} = -\beta_s \Delta t A_{21} - \sum_{j=1}^{s} \alpha_{sj} X_{k-j}$
  $\bar{A}_{22} = -\beta_s \Delta t A_{22} + I_m$
  $\bar{A}_{11} = \beta_s \Delta t A_{11}$
  $\bar{A}_{12} = \beta_s \Delta t A_{12}$
  $[X_k, e] = alg(\bar{A}_{11}, \bar{A}_{12}, \bar{A}_{21}, \bar{A}_{22}, X_{k-1}, tol, maxiter)$
  $t = t + \Delta t$
end while
if $e == -1$ **then**
  error ('there is no convergence')
end if

It is possible to design an efficient and adaptive algorithm that solves DMREs depending on the stiffness of problem (Algorithm 1). The selection of step size is based on variable-coefficient strategies [8] by using an interpolating

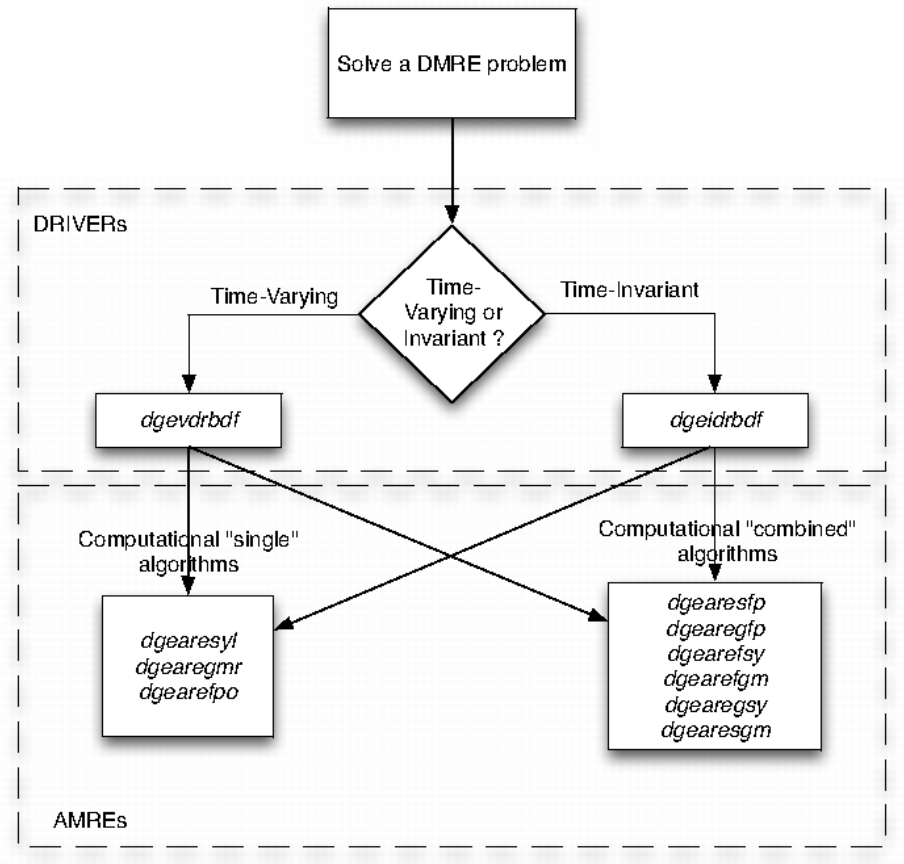

Figure 1: Implemented algorithms and their inter-dependencies

polynomial P_k which interpolates the points $(t_k, X_{k-r}), (t_{k+1}, X_{k-r+1}), \ldots, (t_k, X_k)$ and then the implicit equation

$$\dot{P}_k(t_k) = A_{21}(t) + A_{22}(t_k)X_k - X_k A_{11}(t_k) - X_k A_{12}(t)X_k$$

is solved for X_k by using one of the algorithms of Figure 1. For selecting the step size in the k step we consider the parameters ε_1, ε_2 and s such as $0 < \varepsilon_1 \lesssim E_k \lesssim \varepsilon_2$, where

$$E_k = \frac{\|X_k - X_{k-1}\|_\infty}{\|X_k\|_\infty}.$$

The step size is selected in steps 4-8 of Algorithm 1, where ε_3 ($0 < \varepsilon_3 < \varepsilon_1$) is a necessary parameter to avoid that the step size becomes very small.

4. Experimental Results

The main objective of this section is to compare the developed algorithms. The implementations have been tested on an Apple Macintosh iMac 2.16 Ghz Core 2 Duo processor with 2 Gb of RAM, MacOsX(Unix) OS and MATLAB version 7.7.

As test battery, four stiff problems were used: two time-invariant DMREs and two time-varying DMREs. An exhaustive study of all tests of this paper can be found in [9].

Case study 1. The first case study [1, 10] consists of the following time-invariant DMRE

$$\dot{X}(t) = A_{21} + A_{22}X(t) - X(t)A_{11} - X(t)A_{12}X(t), 0 \le t \le t_f,$$

where $A_{11} = 0_n$, $A_{12} = A_{21} = \alpha I_n$, ($\alpha > 0$ controls the stiffness of the problem), $A_{22} = 0_n$, and $X(0) = X_0 \in \mathbb{R}^{n\times n}$.

The exact solution is given by

$$X(t) = (\alpha(X_0 + I_n)e^{\alpha t} - \alpha(X_0 - I_n)e^{-\alpha t})^{-1}(\alpha(X_0 + I_n)e^{\alpha t} + \alpha(X_0 - I_n)e^{-\alpha t}),$$

which allows the approaches presented in this document to be compared in terms of accuracy.

Case study 2. This case study [2] consists of the following time-invariant DMRE

$$\dot{X}(t) = X(t)T_{2^k} + T_{2^k}X(t) - X(t)T_{2^k}X(t) + \alpha^2 T_{2^k}, t \ge 0,$$
$$X(0) = I_{2^k},$$

where $k \in \mathbb{N}$ and $X(t), T_{2^k} \in \mathbb{R}^{2^k\times 2^k}$. Matrices T_{2^k} are generated as follows:

$$T_2 = \begin{bmatrix} -1 & 1 \\ \alpha^2 & 1 \end{bmatrix},$$
$$T_{2^k} = \begin{bmatrix} -T_{2^{k-1}} & T_{2^{k-1}} \\ \alpha^2 T_{2^{k-1}} & T_{2^{k-1}} \end{bmatrix}, k \ge 2,$$

where α controls the stiffness of the problem. The solution of this DMRE is given by

$$X(t) = I_{2^k} + \frac{\alpha^2 + 1}{\omega}\tanh \omega t T_{2^k},$$

where $\omega = (\alpha^2 + 1)^{\frac{k+1}{2}}$.

Case study 3. This stiff time-varying DMRE is a widely used test problem, known as the "knee problem" [11, 1] defined as

$$\varepsilon\dot{x}(t) = \varepsilon - tx(t) + x^2(t), t \ge -1, x(-1) = -1, 0 < \varepsilon << 1,$$

associated to coefficient matrix

$$A(t) = \begin{bmatrix} a_{11}(t) & a_{12}(t) \\ a_{21}(t) & a_{22}(t) \end{bmatrix} = \begin{bmatrix} t/\varepsilon & -1/\varepsilon \\ 1/2 & 0 \end{bmatrix}, n = m = 1.$$

The reduced solution $x = t$ is stable before 0 and $x \cong 0$ is stable past it. Parameter ϵ controls the stiffness of the problem.

Case study 4. This stiff time-varying DMRE ([1, 12]) comes from a stiff two-point boundary value problem. This DMRE is defined as

$$A_{11}(t) = \begin{bmatrix} -t/2\varepsilon & 0 \\ 0 & 0 \end{bmatrix}, A_{12}(t) = \begin{bmatrix} 1/\varepsilon & 0 \\ 0 & 1/\varepsilon \end{bmatrix},$$

$$A_{21}(t) = \begin{bmatrix} 1/2 & 1 \\ 0 & 1 \end{bmatrix}, A_{22}(t) = \begin{bmatrix} 0 & t/2\varepsilon \\ 0 & 0 \end{bmatrix},$$

where $t \geq -1$, $0 < \varepsilon << 1$. The initial condition is

$$X(-1) = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

The solution has an initial layer and then it approaches

$$X(t) = \begin{bmatrix} -\varepsilon/t & 2(\sqrt{\varepsilon} + t)/2(1 - t\sqrt{\varepsilon}) \\ 0 & \sqrt{\varepsilon} \end{bmatrix}.$$

For t away from 0, there is a smooth transition around the origin and then

$$X(t) \cong \begin{bmatrix} t/2 & \sqrt{\varepsilon} \\ 0 & \sqrt{\varepsilon} \end{bmatrix}.$$

All algorithms explained before have been tested, but only the ones with the best results are shown:

1. Sylvester algorithm (`dgearesyl`).
2. GMRES algorithm (`dgearegmr`).
3. Fixed-Point algorithm (`dgearefpo`).
4. Sylvester algorithm with an initial Fixed-Point iteration (`dgearesfp`).
5. GMRES algorithm with an initial Fixed-Point iteration(`dgearegfp`).

The driver algorithms developed are:

- `dgeidrbdf` (not shown) solves time-invariant DMREs by means of a BDF method.
- `dgevdrbdf` (Algorithm 1) solves time-varying DMREs by means of a BDF method.

For both algorithms the following parameters have been used:

These parameters have been adjusted in each case study to get the best execution time (Te) and the least relative error (using the analytic solution and the obtained solution). Then, for each test we show results for execution time and relative error. The relative error is computed as

$$E_r = \frac{\|X - X^*\|_\infty}{\|X\|_\infty},$$

where X^* is the computed solution and X is the analytic solution.

We used the following values for our tests: $\varepsilon_1 = \Delta t_0 \cdot 10^{-2}$, $\varepsilon_2 = \Delta t_0 \cdot 10^{-1}$, $\varepsilon_3 = \Delta t_0 \cdot 10^{-3}$, $\delta = 1 + \varepsilon_3$, where Δt_0 is a initial step size provided by the user.

The user indicates the stiffness by a parameter s that indicates if the problem is non stiff or low stiff ($s = 0$) or stiff ($s \neq 0$):

- s=0 (non stiff or low stiff): The DMRE is solved by the Fixed-Point algorithm.
- s=1 (stiff): The DMRE is solved by the combined BDF-Sylvester/Fixed-Point algorithm.
- s=2 (stiff): The DMRE is solved by the combined BDF-GMRES/Fixed-Point algorithm.

Table 2: Relative errors for adaptive algorithm r=2, tol= 1e-5, $maxiter$=100

Er	Case Study 1	Case Study 2	Case Study 3	Case Study 4
$s = 1$	0	3.29e-20	4.999e-7	5.92e-15
$s = 2$	0	3.81e-16	4.999e-7	6.00e-15
$s = 3$	0	1.64e-20	4.999e-7	1.56e-15
$s = 4$	0	3.81e-16	4.999e-7	1.56e-15

Table 3: Execution times for adaptive algorithm r=2, $tol = 10^{-5}$, $maxiter$=100

Te	Case Study 1	Case Study 2	Case Study 3	Case Study 4
$s = 1$	1.30	1.06	2.76	4.71
$s = 2$	0.11	0.13	2.67	3.41
$s = 3$	1.33	1.11	2.61	6.4
$s = 4$	0.11	0.13	2.46	5.0

- s=3 (stiff): The DMRE is solved by the Sylvester algorithm.
- s=4 (stiff): The DMRE is solved by the GMRES algorithm.

Tables 2 and 3 contain relative errors and execution times of Case studies when $s \in \{1, 2, 3, 4\}$. The parameters of problems and algorithms were:

- Case study 1: $n = 16$, $\alpha = 1000$, $t_f = 3$, $r = 2$, $tol = 10^{-5}$, $\Delta t_0 = 0.1$, $maxiter = 100$.
- Case study 2: $n = 16$, $\alpha = 100$, $t_f = 1$, $r = 2$, $tol = 10^{-5}$, $\Delta t_0 = 0.1$, $maxiter = 100$.
- Case study 3: $n = 1$, $\epsilon = 10^{-4}$, $t_f = 100$, $r = 2$, $tol = 10^{-5}$, $\Delta t_0 = 0.01$, $maxiter = 100$.
- Case study 4: $n = 2$, $\epsilon = 10^{-4}$, $t_f = 50$, $r = 2$, $tol = 10^{-5}$, $\Delta t_0 = 0.01$, $maxiter = 100$.

Table 2 shows that the combined algorithms ($s = 1, 2$) have the same relative errors as the single algorithms ($s = 3, 4$) in two of the four case studies and lower relative errors than the single algorithms in the other case studies. Table 3 shows that combined algorithms have lower or equal execution times than single algorithms in three of the four case studies.

5. Conclusions and Future Work

We developed a family of algorithms for solving DMREs using adaptive techniques. Because there is no best algorithm to solve all problems, two general purpose adaptive algorithms which select the most adequate adaptive algorithm depending on the DMRE has been designed. The algorithms can be chosen by using a parameter that indicates the stiffness of the problem. MATLAB implementations of these algorithms have been also done. The source codes can be downloaded from the following http address:

http://www.grycap.upv.es/dmretoolbox/DMRE_BDF.htm

There are three items for future work:

- Use other BDF methods for solving AMREs as Adams-Moulton methods [13, pp. 127].

- Parallel implementation of the algorithms presented in this work will be carried out in a distributed memory platform, using the message passing paradigm, MPI [14] and BLACS [15] for communications, and PBLAS [16] and ScaLAPACK [17] for computations.

- Use GPU computing. It is possible to use some kind of GPUS as a way to speed up several computations in a computer. Using the CUDA (Compute Unified Device Architecture) [18] environment from the NVIDIA maker. This environment has tools as CUBLAS [19] (a CUDA BLAS), and very recently appeared CUDA/CUBLAS based implementations of some LAPACK routines: MAGMA [20], CULAPACK [21] and CULATOOLS [22].

6. References

[1] L. Dieci, Numerical integration of the differential Riccati equation and some related issues, SIAM Journal on Numerical Analysis. 29 (3) (1992) 781–815.
[2] C. H. Choi, A. J. Laub, Efficient matrix-valued algorithms for solving stiff Riccati differential equations, IEEE Trans. on Automatic Control 35 (7) (1990) 770–776.
[3] V. Hernández, J. Ibáñez, E. Arias, J. Peinado, A GMRES-based BDF method for solving differential Riccati equations, Applied Mathematics and Computation 196 (2) (2008) 613–626.
[4] E. Arias, V. Hernández, J. Ibáñez, J. Peinado, A fixed point-based BDF method for solving Riccati equations, Applied Mathematics and Computation 188 (2) (2007) 1319–1333.
[5] D. H. Griffel, Applied Functional Analysis, Mathematics and its Applications, Ellis Hordwood, New York, 1981.
[6] R. H. Bartels, G. W. Stewart, Solution of the matrix equation AX + XB = C: Algorithm 432, Communications of the ACM 15 (9) (1972) 820–826.
[7] R. A. Horn, C. R. Johnson, Topics in Matrix Analysis, Cambridge University Press, London, 1991.
[8] U. M. Ascher, L. R. Petzold, Computer Methods for Ordinary Differential Equations and Differential-Algebraic Equations, SIAM, 1998.
[9] J. Ibáñez, J. Peinado, V. Hernández, E. Arias, A family of BDF methods for solving differential matrix Riccati equations using adaptative techniques, Tech. Rep. DSIC-II/08/09, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia (Spain) (2009).
[10] G. H. Meyer, Initial Value Methods for Boundary Value Problems, Academic Press, New York, 1973.
[11] G. Dahlquist, L. Edsberg, G. Skllermo, G. Sderlind, Are the Numerical Methods and Software Satisfactory for Chemical Kinetics?, in Numerical Integration of DE and Large Linear Systems, Vol. 968/1992 of Lecture Notes in Computer Mathematics, Springer Berlin / Heidelberg, 1982, pp. 149–164.
[12] D. L. Brown, J. Lorenz, A high-order method for stiff boundary value problems with turning points, SIAM Journal on Scientific and Statistical Computing 8 (1987) 790–805.
[13] U. Ascher, L. Petzold, Computer Methods for Ordinary Differential Equations, SIAM, Philadelphia, 1998.
[14] W. Gropp, E. Lusk, A. Skjellum, Using MPI: Portable Parallel Programming with the Message-Passing Interface, MIT Press, 1994.
[15] J. Dongarra, R. C. Whaley, A user's guide to the BLACS v1.1, Tech. Rep. UT-CS-95-281, Department of Computer Science, University of Tennessee (1995).
[16] J. Choi, J. Dongarra, S. Ostrouchov, A. Petitet, D. Walker, A proposal for a set of parallel basic linear algebra subprogram, Tech. Rep. UT-CS-95-292, Department of Computer Science, University of Tennessee (1995).
[17] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, ScaLAPACK Users' Guide, SIAM, 1997.
[18] NVIDA, Nvidia CUDA compute unified device architecture (2009).
[19] NVIDIA, CUDA. CUBLAS library (2009).
[20] Enhancing the performance of dense linear algebra solvers on GPUs (in MAGMA project), Supercomputing '08 Poster.
[21] S. Barrachina, M. Castillo, F. Igual, R. Mayo, E. Quintana, G. Quintana, Exploiting the capabilities of modern gpus for dense matrix, Concurrency and Computation: Practice & Experience 21 (2009) 2457–2477.
[22] Culatools, http://www.culatools.com.