Spline Approximations for Systems of Ordinary Differential Equations

Ph.D. Thesis

Presented by: **Dr. Michael M. Tung**
Advisor: **Dr. Emilio Defez Candel**

Universidad Politécnica de Valencia, May 2013

Departamento de Matemática Aplicada

Dr. Emilio Defez Candel, associate professor at the Valencia Polytechnic University,

CERTIFIES that the present thesis by Dr. Michael M. Tung, with the title *Spline Approximations for Systems of Ordinary Differential Equations*, has been directed under my supervision in the Department of Applied Mathematics of the Valencia Polytechnic University. It constitutes his thesis dissertation to obtain the Ph.D. degree in Mathematics.

In compliance with the current legislation, I authorize the presentation of the above Ph.D. thesis, signing this certificate.

Valencia, May 2013

Dr. Emilio Defez Candel

"Es ist nicht das Wissen, sondern das Lernen, nicht das Besitzen, sondern das Erwerben, nicht das Dasein, sondern das Hinkommen, was den größten Genuß gewährt."

Carl Friedrich Gauß

To my wife
Rosa

and

To my parents and brother
Heidi, Ga-Meng, Sascha

Thanks

In first place, I am greatly indebted to my advisor, Emilio Defez, for his kind support and companionship, initiating my work in this area of mathematics and guiding the research for this thesis with him. Many thanks also go to my tutor, Antonio Hervás, for plenty of useful advise, and to Javier Ibáñez for his expertise on MATLAB in a fruitful collaboration with him.

Further thanks to all my colleagues of the *Departamento de Matemática Aplicada* and the *Instituto de Matemática Multidisciplinar* of the *Universidad Politécnica de Valencia* for providing a stimulating environment and encouragement to pursue and complete this work.

Last but not least, my biggest appreciation belongs to my family for their continuous and full backup in whatever matters.

Contents

List of Figures

List of Tables

Summary

A great variety of phenomena in science and engineering are modelled by using ordinary matrix differential equations of first order. The fundamental objective of this Ph.D. dissertation is the development of novel methods for the approximate solution of these equations, including equations of the linear type, differential matrix Sylvester equations, and differential matrix Riccati equations–all of them with variable coefficients. Matrix differential equations pose a higher level of difficulty for their resolution, compared to the conventional scalar equations due, to the obvious intricacy of the matrix structure: the dimensional increase of the problem, which comes hand in hand with an elevated number of necessary operations to obtain the solution, the absence of commutativity and other salient properties of the usual scalar case.

One of the proposed methods employs approximations based on cubic matrix splines. These methods have been implemented in the form of packages in the scientific programming language MATLAB. A selection of this code is made available in the appendix of this dissertation. The aim in mind was to present algorithms readily exportable to other programming environments with great ease of use.

The study continues with an extension of the method for ordinary matrix differential equations of second order, but by avoiding the traditional approach, which consists of the transformation of the problem to a system of first order, since this would increase the dimension of the new problem and come with additional computational overhead.

Regarding the usage of splines of higher than third order, we have obtained good numerical approximations, for both, first- and second-order problems.

The results comprised in the present dissertation have been published in several scientific journals of high impact. Furthermore, they have been communicated at various ECMI conferences (European Conference on Mathematics for Industry) to accomplish the highest possible degree of dissemination.

The topical classification of this thesis, following the standard *Mathematics Subject Classification* (MCS 2010) of the *American Mathematical Society* (AMS) is given by: **41A15**, **65D07**, **65F30**, **65L05**.

Resumen

Una gran variedad de fenómenos en la ciencia y en la ingeniería son modelizados utilizando ecuaciones diferenciales matriciales ordinarias de primer orden. El objetivo fundamental de este proyecto de tesis doctoral es el desarrollo de nuevos métodos de resolución aproximada de dichas ecuaciones, incluyendo las ecuaciones de tipo lineal, de tipo Sylvester y de tipo Riccati, todas ellas con coeficientes variables. Las ecuaciones diferenciales matriciales añaden a las dificultades de resolución de las ecuaciones habituales escalares la complejidad propia del caso matricial: aumento de la dimensión del problema, y por tanto del número de operaciones necesarias para hallar la solución, así como la ausencia de conmutatividad y de otras propiedades escalares habituales.

Uno de los métodos que se proponen utiliza aproximaciones construidas mediante splines cúbicos matriciales. Estos métodos se encuentran implementados en los paquetes de software científico MATLAB. Una selección de dichos códigos se adjuntan en un anexo final de la memoria. El objetivo era obtener algoritmos exportables a otros entornos, y fácilmente utilizables.

Se estudia también la extensión del método para ecuaciones diferenciales matriciales ordinarias de segundo orden, pero evitando el enfoque tradicional, que consiste en transformar el problema en un sistema de primer orden, lo que aumenta la dimensión del nuevo problema, incrementando los costes computacionales.

En lo que se refiere a la utilización de splines de orden superior al tercero, se han conseguido buenas aproximaciones, tanto para problemas de primer orden como de segundo.

Los resultados contenidos en la presente memoria han sido publicados en varias revistas de alto nivel y se han presentado como ponencias en diversas ediciones del congreso internacional ECMI (European Conference on Mathematics for Industry) para que de este modo obtuvieran la mayor difusión posible.

La clasificación temática de esta memoria, atendiendo por áreas a la *Mathematics Subject Classification* (MCS 2010) según la *American Mathematical Society* (AMS) es: **41A15**, **65D07**, **65F30**, **65L05**.

Resum

Una gran varietat de fenòmens en la ciència i en l'enginyeria són modelitzats en utilitzar les equacions diferencials matricials ordinàries de primer ordre. L'objectiu fonamental d'aquest projecte de tesi doctoral és el desenvolupament de nous mètodes de resolució aproximada d'aquestes equacions, on s'hi inclouen les equacions de tipus lineal, de tipus Sylvester i de tipus Riccati, totes amb coeficients variables. Les equacions diferencials matricials afegeixen a les dificultats de resolució de les equacions habituals escalars la complexitat pròpia del cas matricial: l'augment de la dimensió del problema, i per tant del nombre d'operacions necessàries per trobar la solució, així com l'absència de la propietat commutativa i d'altres propietats escalars habituals.

Un dels mètodes que es proposen utilitza aproximacions construïdes mitjançant splines cúbics matricials. Aquests mètodes es troben implementats en els paquets de programari científic MATLAB. Una selecció d'aquests codis s'hi adjunten en un annex final de la memòria. L'objectiu era obtenir algoritmes exportables a altres entorns, i fàcilment utilitzables.

S'estudia també l'extensió del mètode per a equacions diferencials matricials ordinàries de segon ordre, però hem evitat l'enfocament tradicional, que consisteix a transformar el problema en un sistema de primer ordre, cosa que augmenta la dimensió del nou problema, i incrementa el costos computacionals.

Pel que fa a la utilització de splines d'ordre superior al tercer, s'han aconseguit bones aproximacions, tant per als problemes de primer ordre com per als de segon.

Els resultats continguts en la present memòria han estat publicats en diverses revistes d'alt nivell i s'han presentat com a ponències en diverses edicions del congrés internacional ECMI (European Conference on Mathematics for Industry) perquè d'aquesta manera obtingueren la major difusió possible.

La classificació temàtica d'aquesta memòria, per àrees de la *Mathematics Subject Classification* (MCS 2010) segons l'*American Mathematical Society* (AMS) és: **41A15**, **65D07**, **65F30**, **65L05**.

CHAPTER 0

Background and motivation

Many phenomena in science and engineering are modeled by using matrix differential equations of first order in the form of

$$\left.\begin{array}{l} Y'(x) = f(x,Y) \\ Y(a) = Y_a \end{array}\right\} \ , \ a \leq x \leq b \tag{1}$$

where $Y_a, Y(x) \in \mathbb{C}^{r\times q}$, $f : [a,b] \times \mathbb{C}^{r\times q} \to \mathbb{C}^{r\times q}$. For example, the matrix analogue of the ordinary differential equations of Riccati type:

$$\left.\begin{array}{l} Y'(x) = C(x) - D(x)Y(x) - Y(x)A(x) - Y(x)B(x)Y(x) \\ Y(0) = Y_0 \end{array}\right\} \ , \ 0 \leq x \leq c$$

where $Y(x) \in \mathbb{C}^{p\times q}, A(x) \in \mathbb{C}^{q\times q}, B(x) \in \mathbb{C}^{q\times p}$ y $D(x) \in \mathbb{C}^{p\times p}$. This type of problem is known for its central relevance in optimal control problems [1], optimization of systems with distributed parameters [2], optimal filters [3], boundary-value problems and game theory, among many other fields in applied mathematics and engineering, see [4] and the references therein.

Another special case of (1) is the linear problem $Y'(x) = AY(x)$, where the matrix $A \in \mathbb{C}^{r\times r}$ is a matrix with constant coefficients. Over the past years, this problem has attracted considerable attention in the scientific literature due to its many-faceted applications, the difficulty to find its analytic solutions, which in principal is provided by the matrix exponential $Y(x) = e^{Ax}$. In fact, more or less accurate solutions for $Y'(x) = AY(x)$ may be found by employing different approximation techniques for the matrix exponential (Padé, Taylor, etc) [5, 6].

It is obvious that the resolution of matrix differential equations, (1), is much more involved than the resolution of the simple scalar case of conventional differential equations. The additional difficulty of the matrix case derives from the increased dimensionality of the problem and thus the higher number of operations necessary to yield an approximate solution. Moreover, salient scalar properties such as commutativity among others are entirely absent. In fact, for the majority of non-linear problems, (1), the analytic solution is unobtainable or unknown, and one has to resort to numerical methods to obtain an approximate solution. One important class of these techniques is based on the transformation of the continuous problem into another equivalent, discrete problem, such as the well-known vectorial Runge-Kutta method or the BDF (Backward Differentiation Formula) method; see [7, 8] for the best established ap-

proaches. Another technique, frequently encountered in the literature, is the matrix multi-step method [9, 10]. During the last decade, various linearization methods have been proposed; see, *e. g.*, Refs. [11]– [15] and Technical Report [16].

Regarding matrix differential equations of second order and type

$$\left.\begin{aligned} Y''(x) &= f(x, Y(x), Y'(x)) \\ Y(a) &= Y_1 \\ Y'(a) &= Y_2 \end{aligned}\right\}, \; a \le x \le b \tag{2}$$

they may be recast into an extended system of first order [17]. However, this standard approach raises the dimensionality of the problem and therefore goes in hand with a considerable increase of computational cost.

Numerical methods for calculating the approximate solutions of the special matrix problem $Y''(x) = f(x, Y(x), Y'(x)$ have been studied in the context of linear multi-pass matrix methods with constant step size [18]. Although in these cases one is able to specify an upper *a priori* error bound for the discretization error in terms of the data, these error bounds inconveniently depend on the exponential of integration step-size, which then, in practise, has to be chosen rather small. Moreover, these methods require some sort of interpolation technique to even out the discrete results in order to obtain a continuous solution [19]. An example of an incomplete problem is the following matrix differential problems of second order:

$$\left.\begin{aligned} \frac{d^2}{dt^2} Y(t) + \sqrt{A}\, Y(t) &= 0 \\ Y(0) &= Y_0 \\ Y'(0) &= Y_1 \end{aligned}\right\}, \; 0 \le x \le a \tag{3}$$

where Y_0, Y_1 are vectors. Eq. 3 emerges when one applies a semi-discretization technique to the wave equation

$$v^2 \frac{\partial^2 \psi}{\partial x^2} = \frac{\partial^2 \psi}{\partial t^2},$$

which is quite common in telecommunications engineering. Furthermore, they frequently appear in the study of diverse mechanical systems [20].

For the scalar case, cubic splines have been used in Ref. [21] to solve differential equations of first order via approximations which are, among other advantages, differentiable in a given interval. These cubic splines are calculated in a straightforward manner and yield an approximation error of order $O(h^2)$, where h is the step size. Recently, these approach has also been employed for the resolution of other scalar problems, see [22]– [23], vector problems [24], and linear matrix problems [25]. Moreover, cubic matrix splines have also successfully been used in image processing [26].

This thesis comprises various new results obtained by employing cubic matrix splines to solve Eqs. (1) and (2) and to some degree matrix splines of higher order as well. Its contents is divided in four chapters and an appendix. Each chapter corresponds work already published as scientific articles in international journals–for this reason in each chapter the original notation and nomenclature has been preserved and is accompanied by its own bibliography. The appendix enlists all relevant MATLAB and Octave code with the essential algorithms developed for various numerical tasks.

In the following, we conclude with a brief description of each chapter:

- Chapter 1 centers on the application of cubic matrix splines for the resolution of Eq. (1). It corresponds to Ref. [27].

- Chapter 2 moves on to solve Eq. (2) with the help of cubic matrix splines and corresponds to Ref. [28]. Special cases of Eq. (2) with incomplete differential equations were presented as communications at several international conferences [29–33], and other applications appeared as book chapters by the prestigious publisher Springer-Verlag.
- Chapter 3 focusses on the application of cubic matrix splines to tackle linear differential equations $Y'(x) = A(x)Y(x)+b(x)$ with splines of order higher than 3. This work corresponds to Ref. [34].
- Chapter 4 deals with first-order matrix differential equations, (1), and their solution in terms of splines of order higher than 3. This method corresponds to Ref. [35], and preliminary results were presented in a talk at an international conference [36].
- Chapter 5 uses matrix splines of order higher than cubic splines to solve matrix differential equations of second order, (2). A special, incomplete case was presented in a talk at an international conference and published as proceedings by Springer-Verlag [37].
- The Appendix lists a selection of MATLAB programs with the essential algorithms to produce the data for various numerical tables and graphs in the previous chapters.

References

1. R. E. Kalman, *Contributions to the theory of optimal control*, Bol. Soc. Mat. Mex. **5** (1961), 102–119.
2. E. Huntley, *A note on the application of the matrix riccati equation to the optimal control of distributed parameter systems*, IEEE Trans. on Automatic Control **24** (1979), 487–489.
3. R. E. Kalman and R. S. Bucy, *New results in linear filtering and prediction theory*, Trans. ASME, J. Basic Eng. Series D **83** (1961), 95–108.
4. W. T. Reid, *Riccati differential equation*, Academic Press, New York, 1971.
5. C. B. Moler and C. F. Van Loan, *Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later*, SIAM Review **45** (2003), no. 1, 3–49.
6. N. J. Higham, *Functions of matrices, theory and computation*, SIAM, 2008.
7. J. J. Ibáñez, *Computación de altas prestaciones para el cálculo de funciones de matrices y su aplicación a la resolución de ecuaciones diferenciales*, Ph.D. Thesis, 2006.
8. E. Arias, V. Hernández, J. J. Ibáñez, and J. Peinado, *Solving differential Riccati equations by using BDF methods*, Tech. Report Technical Report. DSIC-II/05/05, Departamento de Sistemas Informáticos y Computación de la Universidad Politécnica de Valencia.
9. L. Jódar, J. L. Morera, and G. Rubio, *Computable explicit bounds for the discretization error of variable stepsize multistep methods*, Comp. Math. Appl. **39** (2000), 63–77.
10. J. L. Morera, G. Rubio, and L. Jódar, *Higher order implicit multistep methods for matrix differential equations*, Comp. Math. Appl. **33** (1997), no. 4, 39–48.
11. C. García López, *Métodos de linealización para la resolución numérica de ecuaciones diferenciales*, Ph.D. Thesis, 1998.
12. E. Arias, V. Hernández, I. Bl. Espert, and J. J. Ibáñez, *Nonsingular Jacobian free piecewise linearization of ordinary differential equations (EDO)*, pp. 1–5, Institut National de Recherche en Informatique et en Automatique, 1999.
13. E. Arias, V. Hernández, and J. J. Ibáñez, *High performance algorithms for computing nonsingular Jacobian free piecewise linealization of differential algebraic equations*, pp. 7–12, Birkhäuser, Boston, USA, 2004.
14. E. Arias, V. Hernández, J. J. Ibáñez, and J. Peinado, *A fixed point-based BDF method for solving differential Riccati equations*, Appl. Math. Comput. **188** (2007), no. 2, 1319–1333.
15. V. Hernández, J. J. Ibáñez, J. Peinado, and E. Arias, *A GMRES-based BDF method for solving differential Riccati equations*, Appl. Math. Comput. **196** (2008), no. 2, 613–626.
16. E. Arias, V Hernández, J. J. Ibáñez, and P. Ruiz, *Solving initial value problems for ordinary differential equations by a piecewise-linearized method based on diagonal Padé approximations*, Tech. Report Technical Report DSIC-II/04/07, Departamento de Sistemas Informáticos y Computación de la Universidad Politécnica de Valencia.
17. E. A. Coddington and N. Levinson, *Theory of ordinary differential equations*, McGraw-Hill, New York, 1995.
18. L. Jódar, J. L. Morera, and R. J. Villanueva, *Numerical multistep matrix methods for $y'' = f(t, y)$*, Appl. Math. Comput. **59** (1993), 257–274.
19. L. Jódar and E. Ponsoda, *Continuous numerical solutions and error bounds for matrix differential equations*, Int. Proc. First Int. Colloq. Num. Anal. (Utrecht, The Netherlands) (D. Bainov and V. Covachev, eds.), VSP, 1993, pp. 73–88.
20. E. Defez, J. Sastre, J. J. Ibáñez, and P. Ruiz, *Computing matrix functions solving coupled differential models*, Math. Comput. Model. **50** (2009), 831–839.
21. F. R. Loscalzo and T. D. Talbot, *Spline function approximations for solutions of ordinary differential equations*, SIAM J. Numer. Anal. **4** (1967), no. 3, 433–445.
22. E. A. Al-Said, *The use of cubic splines in the numerical solution of a system of second-order boundary value problems*, Comput. Math. Appl. **42** (2001), 861–869.

23. E. A. Al-Said and M. A. Noor, *Cubic splines method for a system of third-order boundary value problems*, Appl. Math. Comput. **142** (2003), 195–204.
24. G. Micula and A. Revnic, *An implicit numerical spline method for systems for ODE's*, Appl. Math. Comput. **111** (2000), 121–132.
25. E. Defez, L. Soler, A. Hervás, and C. Santamaría, *Numerical solutions of matrix differential models using cubic matrix splines*, Comput. Math. Appl. **50** (2005), 693–699.
26. E. Defez, A. Hervás, A. Law, J. Villanueva-Oller, and R. Villanueva, *Matrix-cubic splines for progressive transmission of images*, J. Math. Imaging Vision **17** (2002), no. 1, 41–53.
27. E. Defez, L. Soler, A. Hervás, and M. M. Tung, *Numerical solutions of matrix differential models using cubic matrix splines II*, Math. Comput. Model. **46** (2007), 657–669.
28. M. M. Tung, E. Defez, L. Soler, and A. Hervás, *Numerical solutions of second-order matrix models using cubic-matrix splines*, Comput. Math. Appl. **56** (2008), 2561–2571.
29. M. M. Tung, E. Defez, A. Hervás, and L. Soler, *Cubic-matrix splines and second-order matrix models*, The 14th European Conference on Mathematics for Industry (ECMI 2006) (Universidad Carlos III de Madrid, Leganés, Spain), 2006.
30. E. Defez, M. M. Tung, J. J. Ibáñez, and A. Hervás, *Approximate numerical solutions of autonomous second-order matrix models using cubic-matrix splines*, The 15th European Conference on Mathematics for Industry (ECMI 2008) (University College, London, UK), 2008.
31. M. M. Tung, E. Defez, A. Hervás, and L. Soler, *Cubic-matrix splines and second-order matrix models*, Mathematics in Industry, no. 12, pp. 949–956, Springer Verlag, Berlin, 2008.
32. E. Defez, M. M. Tung, and J. J. Ibáñez, *A numerical approximation for incomplete second-order matrix models in engineering*, pp. 53–63, Instituto de Matemática Multidisciplinar de la Universidad Politécnica de Valencia, Valencia, 2008.
33. E. Defez, M. M. Tung, J. J. Ibáñez, and A. Hervás, *Approximate numerical solutions of autonomous second-order matrix models using cubic matrix splines*, Mathematics in Industry, no. 15, pp. 785–790, Springer Verlag, Berlin, 2010.
34. E. Defez, A. Hervás, J. J. Ibáñez, and M. M. Tung, *Numerical solutions of matrix differential models using higher-order matrix splines*, Mediterr. J. Math. **9** (2012), 865–882.
35. E. Defez, M. M. Tung, J. J. Ibáñez, and J. Sastre, *Approximating and computing nonlinear matrix differential models*, Math. Comput. Model. **55** (2012), 2012–2022.
36. ———, *Numerical solutions of matrix differential models in engineering using higher-order matrix splines*, pp. 82–99, Instituto de Matemática Multidisciplinar de la Universidad Politécnica de Valencia, Valencia, 2009.
37. E. Defez, M. M. Tung, J. J. Ibáñez, and L. Soler, *Higher-order matrix splines and second order matrix models*, The 16th European Conference on Mathematics for Industry (ECMI 2010) (Bergische Universität Wuppertal, Wuppertal, Germany), 2010.

CHAPTER 1

Numerical solutions of first-order matrix differential equations using cubic splines

1 Introduction

A great variety of phenomena in physics and engineering can be modelled in the form of matrix-differential equations. Although linear matrix-differential equations, whose numerical solutions using cubic matrix splines were presented in [1], are valid for a wide range of applications, non-linear equations are also of great interest. This work generalizes the approach of [1], providing a novel scheme to numerically solve non-linear differential matrix equations of the first-order. Concretely, in this work we will develop a method for the numerical integration of the first order matrix differential equation given by

$$\left.\begin{aligned} Y'(x) &= f(x, Y(x)) \\ Y(a) &= Y_a \end{aligned}\right\} \; a \leq x \leq b\,, \tag{1.1.1}$$

where $Y_a, Y(t) \in \mathbb{C}^{r\times q}$, $f : [a,b] \times \mathbb{C}^{r\times q} \mapsto \mathbb{C}^{r\times q}$.

Different examples of problem (1.1.1) can be found in [2]. Numerical schemes to obtain approximate solutions for (1.1.1) by means of linear multistep methods with constant steps have been devised in [3]. Although there exist *a priori* error bounds for these methods expressed in function of the data problem, these error bounds are given in terms of an exponential which depends on the integration step h. Therefore, in practice, h will take too small values. Furthermore, these methods require some interpolation techniques in order to get a continuous solution [3].

Generalizing the method proposed for the linear case in [1], here we elaborate an extension using cubic-matrix splines in the numerical approximation for the solutions of (1.1.1). In the scalar case, cubic splines were used in [4] for the resolution of ordinary differential equations obtaining approximations that, among other advantages, were of class $\mathscr{C}^1$ in the interval $[a,b]$. These splines are easy to compute and produce an approximation error of only $O(h^4)$. Recently, this method has been used in the resolution of other scalar problems as discussed in [5], and even linear matrix problems (see [1]). The present work extends this powerful scheme to the

resolution of matrix problems of the non-linear type (1.1.1).

This chapter is organized as follows. In Section 2 we develop the proposed method, whose algorithm is then given in Section 3. Finally, in Sections 4, 5 and 6 practical examples are presented.

Throughout this work, we will adopt the notation for norms and matrix cubic splines as in the previous work [1] and common in matrix calculus. Following this nomenclature, we define the Kronecker product of $A = (a_{ij}) \in \mathbb{C}^{m\times n}$ and $B \in \mathbb{C}^{r\times s}$, denoted by $A \otimes B$, as the block matrix

$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{bmatrix} .$$

The column-vector operator on a matrix $A \in \mathbb{C}^{m\times n}$ is given by

$$vec(A) = \begin{bmatrix} A_{\bullet 1} \\ \vdots \\ A_{\bullet n} \end{bmatrix} , \text{ where } A_{\bullet k} = \begin{bmatrix} a_{1k} \\ \vdots \\ a_{mk} \end{bmatrix} .$$

If $Y = (y_{ij}) \in \mathbb{C}^{p\times q}$ and $X = (x_{ij}) \in \mathbb{C}^{m\times n}$, then the derivative of a matrix with respect to a matrix is defined by [6, p. 62 and 81]:

$$\frac{\partial Y}{\partial X} = \begin{bmatrix} \frac{\partial Y}{\partial x_{11}} & \cdots & \frac{\partial Y}{\partial x_{1n}} \\ \vdots & & \vdots \\ \frac{\partial Y}{\partial x_{m1}} & \cdots & \frac{\partial Y}{\partial x_{mn}} \end{bmatrix} , \text{ where } \quad \frac{\partial Y}{\partial x_{rs}} = \begin{bmatrix} \frac{\partial y_{11}}{\partial x_{rs}} & \cdots & \frac{\partial y_{1q}}{\partial x_{rs}} \\ \vdots & & \vdots \\ \frac{\partial y_{p1}}{\partial x_{rs}} & \cdots & \frac{\partial y_{pq}}{\partial x_{rs}} \end{bmatrix} .$$

If $X \in \mathbb{C}^{m\times n}, Y \in \mathbb{C}^{n\times v}, Z \in \mathbb{C}^{p\times q}$, then the following rule for the derivative of a matrix product with respect to another matrix applies [6, p. 84]:

$$\frac{\partial XY}{\partial Z} = \frac{\partial X}{\partial Z}\left[I_q \otimes Y\right] + \left[I_p \otimes X\right]\frac{\partial Y}{\partial Z} , \tag{1.1.2}$$

where I_q and I_p denote the identity matrices of dimensions q and p, respectively. If $X \in \mathbb{C}^{m\times n}, Y \in \mathbb{C}^{u\times v}, Z \in \mathbb{C}^{p\times q}$, the following chain rule [6, p. 88] is valid :

$$\frac{\partial Z}{\partial X} = \left[\frac{\partial \left[vec(Y)\right]^t}{\partial X} \otimes I_p\right]\left[I_n \otimes \frac{\partial Z}{\partial \left[vec(Y)\right]}\right] . \tag{1.1.3}$$

If $A = (a_{ij}) \in \mathbb{C}^{m\times n}$, the Frobenius norm of A is [7] given by:

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n}\left|a_{ij}\right|^2} . \tag{1.1.4}$$

The following relationship between the 2-norm and Frobenius norm holds [7]:

$$\|A\|_2 \le \|A\|_F \le \sqrt{n}\,\|A\|_2 . \tag{1.1.5}$$

2 Proposed general method

Let us consider the problem

$$\left.\begin{array}{l} Y'(x) = f(x, Y(x)) \\ Y(a) = Y_a \end{array}\right\} \; a \le x \le b \; , \tag{1.2.6}$$

where $Y_a, Y(t) \in \mathbb{C}^{r\times q}$, $f : [a,b] \times \mathbb{C}^{r\times q} \mapsto \mathbb{C}^{r\times q}$, $f \in \mathscr{C}^1(T)$, with

$$T \;=\; \{(x,Y)\,;\; a \le x \le b\;,\; Y \in \mathbb{C}^{r\times q}\} \;, \tag{1.2.7}$$

and f fulfills the global Lipschitz's condition

$$\left\| f(x,Y_1) \;-\; f(x,Y_2) \right\| \;\le L \left\| Y_1 - Y_2 \right\| \;,\; a \le x \le b \;, Y_1, Y_2 \in \mathbb{C}^{r\times q} \;, \tag{1.2.8}$$

which guarantees the existence and uniqueness of the continuously differentiable solution $Y(x)$ of problem (1.2.6), see [8, p. 99].

Let us consider $h = (b-a)/n$, n being a positive integer, so that the partition of the interval $[a,b]$ is given by

$$\Delta_{[a,b]} = \{a = x_0 \;<\; x_1 \;<\; \ldots \;<\; x_n = b\} \;,\; x_k = a + kh \;,\; k = 0, 1, \ldots, n \;. \tag{1.2.9}$$

We will construct in each subinterval $[a+kh, a+(k+1)h]$ a matrix-cubic spline approximating the solution of problem (1.2.6). For the first interval $[a, a+h]$, we consider that the matrix-cubic spline is defined by

$$S_{\big|_{[a,a+h]}}(x) \;=\; Y(a) \;+\; Y'(a)(x-a) \;+\; \frac{1}{2!}Y''(a)(x-a)^2 \;+\; \frac{1}{3!}A_0(x-a)^3 \;, \tag{1.2.10}$$

where the matrix $A_0 \in \mathbb{C}^{r\times q}$ is a parameter to be determined. It is straightforward to check:

$$S_{\big|_{[a,a+h]}}(a) = Y(a) \;, \quad S'_{\big|_{[a,a+h]}}(a) = Y'(a) = f(a, Y(a)) \;.$$

To fully determine the matrix-cubic spline we still must obtain $Y''(a)$ and A_0. We consider the functions h_1 and h_2 defined by

$$\begin{array}{ll} h_1 : [a,b] \mapsto [a,b] & h_2 : [a,b] \mapsto \mathbb{C}^{r\times q} \\ h_1(x) \;=\; x & h_2(x) \;=\; Y(x) \end{array} \;,$$

where $Y(x)$ is the theoretical solution of (1.2.6). We describe now $f(x, Y(x))$ as a composition of functions f and (h_1, h_2), that is, let $\phi : [a,b] \;\mapsto\; \mathbb{C}^{r\times q}$ be defined by

$$\phi(x) \;=\; \left[f \circ (h_1, h_2)\right](x) \;=\; f\left(h_1(x), h_2(x)\right) \;=\; f(x, Y(x)) \;.$$

Thus, ϕ is a real variable function of x, and applying theorem 8.9.2 of [6, p. 170] its derivative takes the form:

$$D\phi \;=\; D\left(f\circ(h_1,h_2)\right) \;=\; \left(\left(D_1 f\right)(h_1,h_2)\right)\cdot Dh_1 \;+\; \left(\left(D_2 f\right)(h_1,h_2)\right)\cdot Dh_2\ ,$$

where the partial derivatives of f, $D_1(f)$, $D_2(f)$ exist and are continuous since it is assumed that $f\in\mathscr{C}^1(T)$. By (1.2.6) it is clear that

$$\frac{d\,(vec\,Y(x))^T}{dx} \;=\; \left[vec\,f(x,Y(x))\right]^T\ .$$

Next, applying the chain rule for matrix functions (1.1.2) and then taking the derivative of a matrix with respect to a matrix, (1.1.3), one obtains

$$Y''(x) \;=\; \frac{\partial f(x,Y(x))}{\partial x} \;+\; \left[\left[vec\,f(x,Y(x))\right]^T\otimes I_r\right]\frac{\partial f(x,Y(x))}{\partial\, vec\,Y(x)}\ . \tag{1.2.11}$$

We are now in the position to evaluate $Y''(a)$ using (1.2.11).

By imposing that (1.2.10) is a solution of problem (1.2.6) in $x=a+h$, we have:

$$S'_{\big|_{[a,a+h]}}(a+h) \;=\; f\left(a+h,S_{\big|_{[a,a+h]}}(a+h)\right)\ , \tag{1.2.12}$$

and obtain from (1.2.12) the matrix equation with only one unknown matrix A_0:

$$A_0 \;=\; \frac{2}{h^2}\left[f\left(a+h,Y(a)+Y'(a)h+\frac{1}{2}Y''(a)h^2+\frac{1}{6}A_0h^3\right)-Y'(a)-Y''(a)h\right]\ . \tag{1.2.13}$$

Assuming that the matrix equation (1.2.13) has only one solution A_0, the matrix-cubic spline is totally determined in the interval $[a,a+h]$.

Now, in the interval $[a+h,a+2h]$, the matrix-cubic spline takes the form

$$\begin{aligned}S_{\big|_{[a+h,a+2h]}}(x) = {}& S_{\big|_{[a,a+h]}}(a+h)+S'_{\big|_{[a,a+h]}}(a+h)(x-(a+h))\\ &+\frac{1}{2!}S''_{\big|_{[a,a+h]}}(a+h)(x-(a+h))^2+\frac{1}{3!}A_1(x-(a+h))^3\ ,\end{aligned} \tag{1.2.14}$$

so that $S(x)$ is of class $\mathscr{C}^2([a,b])$ on $[a,a+h]\cup[a+h,a+2h]$, and all coefficients of the matrix-cubic spline $S_{\big|_{[a+h,a+2h]}}(x)$ are determined with the exception of $A_1\in\mathbb{C}^{r\times q}$. By construction, matrix-cubic spline (1.2.14) satisfies the differential equation (1.2.6) in $x=a+h$. We can obtain A_1 by requiring that the differential equation (1.2.6) holds at point $x=a+2h$:

$$S'_{\big|_{[a+h,a+2h]}}(a+2h) \;=\; f\left(a+2h,S_{\big|_{[a+h,a+2h]}}(a+2h)\right)\ .$$

Expanding, we obtain the matrix equation with only one unknown matrix A_1:

$$\begin{aligned}A_1 = \frac{2}{h^2}\Bigg[& f\left(a+2h,S_{\big|_{[a,a+h]}}(a+h)+S'_{\big|_{[a,a+h]}}(a+h)h+\frac{1}{2}S''_{\big|_{[a,a+h]}}(a+h)h^2+\frac{1}{6}A_1h^3\right)\\ &-S'_{\big|_{[a,a+h]}}(a+h)-S''_{\big|_{[a,a+h]}}(a+h)h\Bigg]\ .\end{aligned} \tag{1.2.15}$$

Let us assume that the matrix equation (1.2.15) has only one solution A_1. This way the spline is totally determined in the interval $[a+h, a+2h]$.

Iterating this process, let us construct the matrix-cubic spline taking $[a+(k-1)h, a+kh]$ as the last subinterval. For the next subinterval $[a+kh, a+(k+1)h]$, we define the corresponding matrix-cubic spline as

$$S_{|_{[a+kh,a+(k+1)h]}}(x) = \beta_k(x) + \frac{1}{3!}A_k(x-(a+kh))^3 , \tag{1.2.16}$$

where

$$\beta_k(x) = \sum_{k=0}^{2} \frac{1}{k!} S^{(k)}_{|_{[a+(k-1)h,a+kh]}}(a+kh)(x-(a+kh))^k . \tag{1.2.17}$$

With this definition, the matrix-cubic spline is $S(x) \in \mathscr{C}^2\left(\bigcup_{j=0}^{k}[a+jh, a+(j+1)h]\right)$ and fulfills the differential equation (1.2.6) at point $x = a+kh$. As an additional requirement, we assume that $S(x)$ satisfies the differential equation (1.2.6) at the point $x = a+(k+1)h$:

$$S'_{|_{[a+kh,a+(k+1)h]}}(a+(k+1)h) = f\left(a+(k+1)h, S_{|_{[a+kh,a+(k+1)h]}}(a+(k+1)h)\right) ,$$

and expanding this equation with the unknown matrix A_k yields

$$\begin{aligned} A_k = \frac{2}{h^2}\Bigg[& f\left(a+(k+1)h, S_{|_{[a+(k-1)h,a+kh]}}(a+kh) + S'_{|_{[a+(k-1)h,a+kh]}}(a+kh)h \right. \\ & \left. + \frac{1}{2}S''_{|_{[a+(k-1)h,a+kh]}}(a+kh)h^2 + \frac{1}{6}A_k h^3\right) \\ & - S'_{|_{[a+(k-1)h,a+kh]}}(a+kh) - S''_{|_{[a+(k-1)h,a+kh]}}(a+kh)h\Bigg] . \end{aligned} \tag{1.2.18}$$

Note that this matrix equation (1.2.18) is analogous to equations (1.2.13) and (1.2.15), when $k=0$ and $k=1$, respectively. We will show that these equations have an unique solution using a fixed-point argument.

For a fixed h, we will consider the matrix function of matrix variable $g : \mathbb{C}^{r\times q} \mapsto \mathbb{C}^{r\times q}$ defined by

$$\begin{aligned} g(T) = \frac{2}{h^2}\Bigg[& f\left(a+(k+1)h, S_{|_{[a+(k-1)h,a+kh]}}(a+kh) + S'_{|_{[a+(k-1)h,a+kh]}}(a+kh)h \right. \\ & \left. + \frac{1}{2}S''_{|_{[a+(k-1)h,a+kh]}}(a+kh)h^2 + \frac{1}{6}Th^3\right) \\ & - S'_{|_{[a+(k-1)h,a+kh]}}(a+kh) - S''_{|_{[a+(k-1)h,a+kh]}}(a+kh)h\Bigg] . \end{aligned} \tag{1.2.19}$$

Relation (1.2.18) holds if and only if $A_k = g(A_k)$, that is, if A_k is a fixed point for function $g(T)$.

Observe that by using (1.2.17) and applying the global Lipschitz's condition (1.2.8) it follows that

$$\|g(T_1) - g(T_2)\| \leq \frac{Lh}{3}\|T_1 - T_2\| .$$

Taking $h < 3/L$, $g(T)$ yields a contractive matrix function, which guarantees that equation (1.2.18) has unique solutions A_k for $k = 0, 1, \ldots, n-1$. Hence, the matrix-cubic spline is completely determined. Taking into account [4, Theorem 5], the following result can be established.

Theorem 1.1. *Let be L the Lipschitz constant defined by (1.2.8). If $h \leq 3/L$, then the matrix-cubic spline $S(x)$ exists in each subinterval $[a+kh, a+(k+1)h]$, $k = 0, 1, \ldots, n-1$, as defined in the previous construction. Furthermore, if $f \in \mathcal{C}^3(T)$, then $\|Y(x) - S(x)\| = O(h^4)$ $\forall x \in [a,b]$, where $Y(x)$ is the theoretical solution of (1.2.6).*

3 Algorithm

The following algorithm is designed to compute the approximate solution of (1.2.6) by means of matrix-cubic splines in the interval $[a,b]$ with an error of the order $O(h^4)$ under conditions of theorem 1.1.

- Determine the constant $Y''(a)$ given by (1.2.11). Take $n > L(b-a)/3$, $h = (b-a)/n$ and the partition $\Delta_{[a,\,b]}$ defined by Eq. (1.2.9).
- Solve the matrix equation (1.2.13) for $k = 0$ and determine $S_{|_{[a,a+h]}}(x)$ of Eq. (1.2.10).
- Solve the matrix equation (1.2.18) iteratively for $k = 1, \ldots, n-1$, and then compute the splines $S_{|_{[a+kh,a+(k+1)h]}}(x)$ according to Eq. (1.2.16).

Depending on the function $f(t,Y)$, matrix equations (1.2.13) and (1.2.18) can be solved explicitly (see [9]) or using the iterative method (see for example [10]):

$$T_{l+1}^s = g(T_l^s)\ , \text{ where } T_0^s \text{ is an arbitrary matrix in } \mathbb{C}^{r\times q}\ ,\ s = 0, 1, \ldots, n-1$$

and $g(T)$ is given for (1.2.19). In the following section, we will test the algorithm proposed.

4 Example: A non-linear vector system

We consider the next non-linear vector differential system

$$\left.\begin{aligned} y_1'(x) &= -1 + e^x - \sin(x) + \sin(y_2(x)) \\ y_2'(x) &= \frac{1}{4 + y_1(x)^2} - \frac{1}{5 + e^{2x} + 2\,e^x\,\cos(x) - \sin^2(x)} \\ y_1(0) &= 2, \qquad y_2(0) = \tfrac{\pi}{2} \end{aligned}\right\}\ 0 \leq x \leq 1\ , \tag{1.4.20}$$

It is easy to check that this problem has the exact solution $y_1(x) = e^x + \cos(x)$, $y_2(x) = \pi/2$, so in this particular case we will be able to obtain the exact error of our numerical estimates.

We can rewrite (1.4.20) in the compact form

$$\left.\begin{aligned} Y'(x) &= F(x,Y) \\ Y(0) &= \begin{pmatrix} 2 \\ \frac{\pi}{2} \end{pmatrix} \end{aligned}\right\}\ 0 \leq x \leq 1 \tag{1.4.21}$$

with

$$Y(x)=\begin{pmatrix} y_1(x)\\ y_2(x)\end{pmatrix}\in\mathbb{R}^2,\ F(x,Y)=\begin{pmatrix} -1+e^x-\sin(x)+\sin(y_2(x))\\ \dfrac{1}{4+y_1(x)^2}-\dfrac{1}{5+e^{2x}+2e^x\cos(x)-\sin^2(x)}\end{pmatrix}\in\mathbb{R}^2.$$

Thus, $Y'(0)=F\left(0,\begin{pmatrix}2\\ \frac{\pi}{2}\end{pmatrix}\right)=\begin{pmatrix}1\\0\end{pmatrix}$. We calculate $Y''(0)$ using (1.2.11). in this case, one gets

$$vec(Y(x))=Y(x)=\begin{pmatrix} y_1(x)\\ y_2(x)\end{pmatrix},\ \frac{\partial F(x,Y(x))}{\partial x}=\begin{pmatrix} e^x-\cos(x)\\ \dfrac{2e^{2x}+2e^x\cos(x)-2e^x\sin(x)-2\cos(x)\sin(x)}{\left(5+e^{2x}+2e^x\cos(x)-\sin^2(x)\right)^2}\end{pmatrix}. \tag{1.4.22}$$

On the other hand, we have

$$[vec\ F(x,Y(x))]^T\otimes I_2$$

$$=\left(-1+e^x-\sin(x)+\sin(y_2(x))\ \frac{1}{4+y_1(x)^2}-\frac{1}{5+e^{2x}+2e^x\cos(x)-\sin^2(x)}\right)\otimes I_2$$

$$=\left(\left(-1+e^x-\sin(x)+\sin(y_2(x))\right)I_2\middle|\left(\frac{1}{4+y_1(x)^2}-\frac{1}{5+e^{2x}+2e^x\cos(x)-\sin^2(x)}\right)I_2\right)$$

$$=\left(\begin{array}{cc|cc} -1+e^x-\sin(x)+\sin(y_2(x)) & 0 & \frac{1}{4+y_1(x)^2}-\frac{1}{5+e^{2x}+2e^x\cos(x)-\sin^2(x)} & 0\\ 0 & -1+e^x-\sin(x)+\sin(y_2(x)) & 0 & \frac{1}{4+y_1(x)^2}-\frac{1}{5+e^{2x}+2e^x\cos(x)-\sin^2(x)}\end{array}\right), \tag{1.4.23}$$

and

$$\frac{\partial F(x,Y(x))}{\partial\ vec\ Y(x)}=\begin{pmatrix}\frac{\partial F(x,Y(x))}{\partial y_1}\\ \hline \frac{\partial F(x,Y(x))}{\partial y_2}\end{pmatrix}=\left(\begin{array}{c}\frac{\partial}{\partial y_1}\left(-1+e^x-\sin(x)+\sin(y_2(x))\right)\\ \frac{\partial}{\partial y_1}\left(\frac{1}{4+y_1(x)^2}-\frac{1}{5+e^{2x}+2e^x\cos(x)-\sin^2(x)}\right)\\ \hline \frac{\partial}{\partial y_2}\left(-1+e^x-\sin(x)+\sin(y_2(x))\right)\\ \frac{\partial}{\partial y_2}\left(\frac{1}{4+y_1(x)^2}-\frac{1}{5+e^{2x}+2e^x\cos(x)-\sin^2(x)}\right)\end{array}\right)=\begin{pmatrix}0\\ \frac{-2y_1(x)}{\left(4+y_1(x)^2\right)^2}\\ \hline \cos(y_2(x))\\ 0\end{pmatrix}. \tag{1.4.24}$$

Therefore, we find

$$\left[[vec\ F(x,Y(x))]^T\otimes I_2\right]\frac{\partial F(x,Y(x))}{\partial\ vec\ Y(x)}=\begin{pmatrix}\left(\frac{1}{4+y_1(x)^2}-\frac{2}{9+2e^{2x}+4e^x\cos(x)+\cos(2x)}\right)\cos(y_2(x))\\ -\frac{2y_1(x)\left(-1+e^x-\sin(x)+\sin(y_2(x))\right)}{\left(4+y_1(x)^2\right)^2}\end{pmatrix}, \tag{1.4.25}$$

and by (1.4.22)-(1.4.25) one finds the following formula

$$Y''(x) = \frac{\partial F(x,Y(x))}{\partial x} + \left[[vec\ F(x,Y(x))]^T \otimes I_2\right] \frac{\partial F(x,Y(x))}{\partial\ vec\ Y(x)}$$
$$= \begin{pmatrix} e^x - \cos(x) + \left(\frac{1}{4+y_1(x)^2} - \frac{2}{9+2e^{2x}+4e^x\cos(x)+\cos(2x)}\right)\cos(y_2(x)) \\ \frac{2e^{2x}+2e^x\cos(x)-2e^x\sin(x)-2\cos(x)\sin(x)}{\left(5+e^{2x}+2e^x\cos(x)-\sin^2(x)\right)^2} - \frac{2y_1(x)\left(-1+e^x-\sin(x)+\sin(y_2(x))\right)}{\left(4+y_1(x)^2\right)^2} \end{pmatrix}. \tag{1.4.26}$$

Taking into account that $y_1(0) = 2, y_2(0) = \frac{\pi}{2}$ and evaluating $Y''(x)$ of (1.4.26) when $x = 0$, one gets $Y''(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$.

It is straightforward to show that F, defined by (1.4.21), fulfills the global Lipschitz's condition

$$\|f(x,Y) - f(x,Z)\| \leq \|Y - Z\|\ ,\ 0 \leq x \leq 1\ , Y, Z \in \mathbb{R}^2\ , \tag{1.4.27}$$

thus, we can take L given by (1.2.8) as $L = 1$. Therefore, we need to take $h < 3/L$ and thus $h = 0.1$ for example. The results are generated with *Mathematica* using `FindRoot` function to solve the emerging algebraic equations, and are summarized in Table 1. In each interval, we evaluated the difference between the estimates of our numerical approach and the exact solution, and then take the Frobenius norm of this difference. The maximum of these errors are indicated in the third column for each subinterval.

Interval	Approximation	Max. Error
$[0, 0.1]$	$\begin{pmatrix} 2 + x + 0.177917x^3 \\ \frac{\pi}{2} - 5.62424 \times 10^{-6}x^3 \end{pmatrix}$	2.83337×10^{-6}
$[0.1, 0.2]$	$\begin{pmatrix} 1.99995 + 1.00138x - 0.0138342x^2 + 0.224031x^3 \\ 1.5708 + 6.67857 \times 10^{-7}x - 6.67857 \times 10^{-6}x^2 + 0.0000166377x^3 \end{pmatrix}$	2.83337×10^{-6}
$[0.2, 0.3]$	$\begin{pmatrix} 1.99975 + 1.00445x - 0.0291822x^2 + 0.249611x^3 \\ 1.5708 - 4.57386 \times 10^{-6}x + 0.00001953x^2 - 0.0000270433x^3 \end{pmatrix}$	2.94712×10^{-6}
$[0.3, 0.4]$	$\begin{pmatrix} 1.99841 + 1.01783x - 0.0737602x^2 + 0.299142x^3 \\ 1.57079 + 0.0000126873x - 0.0000380073x^2 + 0.0000368871x^3 \end{pmatrix}$	2.94712×10^{-6}
$[0.4, 0.5]$	$\begin{pmatrix} 1.99655 + 1.0318x - 0.108685x^2 + 0.328246x^3 \\ 1.5708 - 0.0000271633x + 0.0000616192x^2 - 0.0000461351x^3 \end{pmatrix}$	3.0698×10^{-6}
$[0.5, 0.6]$	$\begin{pmatrix} 1.9899 + 1.07171x - 0.188509x^2 + 0.381462x^3 \\ 1.57079 + 0.0000485499x - 0.0000898071x^2 + 0.0000548159x^3 \end{pmatrix}$	3.0698×10^{-6}
$[0.6, 0.7]$	$\begin{pmatrix} 1.98277 + 1.10736x - 0.247933x^2 + 0.414475x^3 \\ 1.57081 - 0.0000785694x + 0.000122058x^2 - 0.0000628872x^3 \end{pmatrix}$	3.20977×10^{-6}
$[0.7, 0.8]$	$\begin{pmatrix} 1.96307 + 1.19177x - 0.368517x^2 + 0.471896x^3 \\ 1.57077 + 0.000117333x - 0.000157802x^2 + 0.0000703796x^3 \end{pmatrix}$	3.20977×10^{-6}
$[0.8, 0.9]$	$\begin{pmatrix} 1.94382 + 1.26395x - 0.45874x^2 + 0.509489x^3 \\ 1.57084 - 0.0001661x + 0.00019649x^2 - 0.0000772419x^3 \end{pmatrix}$	3.37764×10^{-6}
$[0.9, 1.0]$	$\begin{pmatrix} 1.89829 + 1.41574x - 0.627395x^2 + 0.571954x^3 \\ 1.57073 + 0.000224533x - 0.000237548x^2 + 0.0000835127x^3 \end{pmatrix}$	3.37764×10^{-6}

Table 1.1. Approximation for vector differential system (1.4.20) in the interval $[0, 1]$ with step size $h = 0.1$.

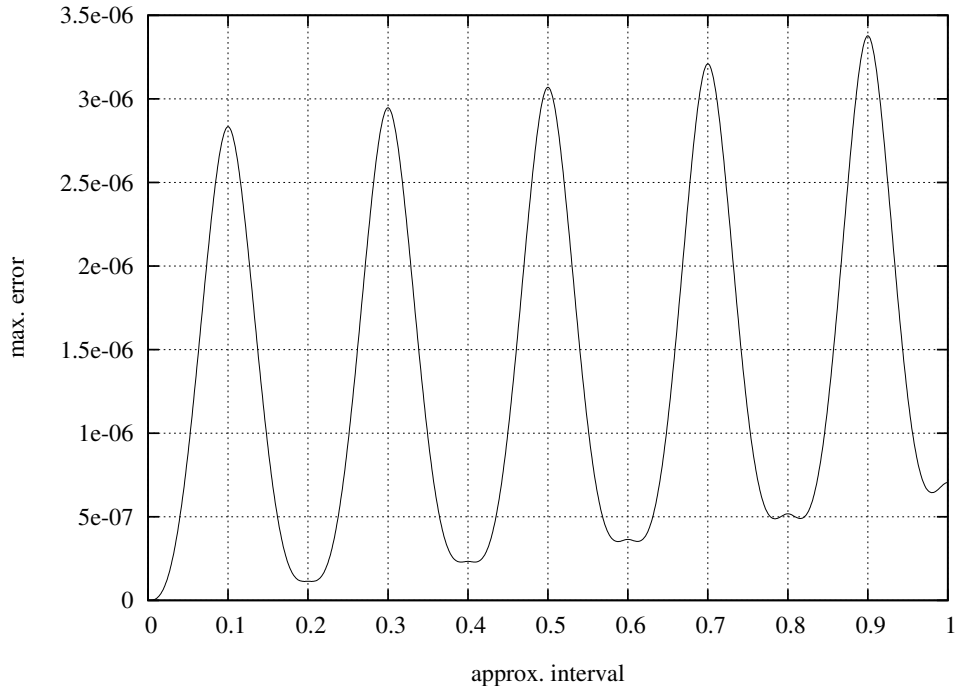

Fig. 1.1. Representing the Frobenius error margins for vector differential system (1.4.20) in the interval $[0,1]$ with step size $h = 0.1$.

5 Example: Sylvester matrix differential equation

Linear matrix differential equations of the type

$$\left.\begin{array}{l} Y'(x) = A(x)Y(x) + Y(x)B(x) + C(x) \\ Y(a) = Y_a \end{array}\right\} \; a \le x \le b \;,\; Y(x), A(x), B(x), C(x) \in \mathbb{C}^{r\times r} \;, \quad (1.5.28)$$

arise in many fields of science and engineering. In the case of constant coefficients has been studied by several authors (see for example [11]). However, the variable-coefficient case has so far received little numerical treatment in the literature. We can observe that the proposed method require the matrix functions $A(x), B(x)$ and $C(x)$ to be differentiable, while, for example, in the method proposed in [12], it is necessary that $A(x), B(x)$ have continuous second-order derivatives and $C(x)$ continuous in the domain $a \le x \le b$.

As an example, here let us consider the Sylvester problem (1.5.28) with

$$\left.\begin{array}{l} A(x) = \begin{pmatrix} 0 & xe^{-x} \\ x & 0 \end{pmatrix}, \quad B(x) = \begin{pmatrix} 0 & x \\ 0 & 0 \end{pmatrix}, \quad C(x) = \begin{pmatrix} -e^{-x}(1+x^2) & -2e^{-x}x \\ 1 - e^{-x}x & -x^2 \end{pmatrix} \\ Y(0) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad Y(x) \in \mathbb{C}^{2\times 2} \;,\; 0 \le x \le 1 \end{array}\right\} \quad (1.5.29)$$

This problem has an exact solution $Y(x) = \begin{pmatrix} e^{-x} & 0 \\ x & 1 \end{pmatrix}$, so in this particular case we will be able to obtain the exact error of our numerical estimates.

As we have $\max_{x\in[0,1]} \left(\left\| \begin{pmatrix} 0 & xe^{-x} \\ x & 0 \end{pmatrix} \right\| + \left\| \begin{pmatrix} 0 & x \\ 0 & 0 \end{pmatrix} \right\| \right) \leq 1.69443$, one can take the constant L given for (1.2.8) as $L = 2$.

Taking the derivative of $Y'(x) = A(x)Y(x) + Y(x)B(x) + C(x)$, gives:

$$Y''(x) = \left(A'(x) + (A(x))^2\right) Y(x) + Y(x) \left((B(x))^2 + B'(x)\right) + 2A(x)Y(x)B(x) + A(x)C(x) + C(x)B(x) + C'(x) \,. \tag{1.5.30}$$

We see that $Y'(0) = \begin{pmatrix} -1 & 0 \\ 1 & 0 \end{pmatrix}$, and by applying (1.5.30) it is $Y''(0) = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}$.

In this numerical example, we take $n = 10$ such that $n > L(b-a)/3$ and $h = 0.1 = (b-a)/n$. The results are generated with *Mathematica* using the Bartels-Stewart algorithm (see for example [7]) to solve the emerging algebraic equations, and are summarized in Table 2, where the numerical estimates have been rounded to the fourth relevant digit. In each interval, we evaluated the difference between the estimates of our numerical approach and the exact solution, and then take the Frobenius norm of this difference. The maximum of these errors are indicated in the third column for each subinterval.

Interval	Approximation	Max. Error
$[0, 0.1]$	$\begin{pmatrix} 1 - x + 0.5x^2 - 0.1612x^3 & 0 \\ x & 1 \end{pmatrix}$	1.33472×10^{-6}
$[0.1, 0.2]$	$\begin{pmatrix} 1 - 0.9994x + 0.4938x^2 - 0.1406x^3 & 0 \\ x & 1 \end{pmatrix}$	1.33472×10^{-6}
$[0.2, 0.3]$	$\begin{pmatrix} 1 - 0.9984x + 0.4890x^2 - 0.1325x^3 & 0 \\ x & 1 \end{pmatrix}$	1.2445×10^{-6}
$[0.3, 0.4]$	$\begin{pmatrix} 0.9994 - 0.9936x + 0.4728x^2 - 0.1146x^3 & 0 \\ x & 1 \end{pmatrix}$	1.2445×10^{-6}
$[0.4, 0.5]$	$\begin{pmatrix} 0.9991 - 0.9909x + 0.4661x^2 - 0.1090x^3 & -0.0001x^2 \\ 1.0001x - 0.0001x^2 & 1 \end{pmatrix}$	1.17402×10^{-6}
$[0.5, 0.6]$	$\begin{pmatrix} 0.9971 - 0.9791x + 0.4426x^2 - 0.0933x^3 & -0.0001x + 0.0002x^2 - 0.0001x^3 \\ 0.9999x + 0.0002x^2 - 0.0001x^3 & 0.9999 \end{pmatrix}$	1.17402×10^{-6}
$[0.6, 0.7]$	$\begin{pmatrix} 0.9963 - 0.9732x + 0.4361x^2 - 0.0898x^3 & 0.0002x - 0.0004x^2 + 0.0002x^3 \\ 1.0002x - 0.0004x^2 + 0.0002x^3 & 1 \end{pmatrix}$	1.12331×10^{-6}
$[0.7, 0.8]$	$\begin{pmatrix} 0.9916 - 0.9549x + 0.4071x^2 - 0.07591x^3 & 0.0001 - 0.0004x + 0.0006x^2 - 0.0003x^3 \\ 0.0001 + 0.9996x + 0.0006x^2 - 0.0003x^3 & 0.9999 \end{pmatrix}$	1.12331×10^{-6}
$[0.8, 0.9]$	$\begin{pmatrix} 0.9906 - 0.9512x + 0.4025x^2 - 0.0739x^3 & 0.0002 + 0.0007x - 0.0009x^2 + 0.0003x^3 \\ -0.0002 + 1.0007x - 0.0009x^2 + 0.0003x^3 & 1 + 0.0001x^2 \end{pmatrix}$	1.09412×10^{-6}
$[0.9, 1.0]$	$\begin{pmatrix} 0.9816 - 0.9212x + 0.3691x^2 - 0.0616x^3 & 0.0004 - 0.0011x + 0.0012x^2 - 0.0004x^3 \\ 0.0004 + 0.0.9989x + 0.0012x^2 - 0.0004x^3 & 0.9999 + 0.0002x - 0.0002x^2 \end{pmatrix}$	1.09412×10^{-6}

Table 1.2. Approximation for Sylvester matrix differential equation (1.5.29) in the interval $[0, 1]$ with step size $h = 0.1$.

6 Example: Riccati matrix differential equation

Rectangular non-symmetric Riccati matrix-differential equation of the type

$$\left.\begin{aligned} Y'(x) &= C(x) - D(x)Y(x) - Y(x)A(x) - Y(x)B(x)Y(x) \\ Y(0) &= Y_0 \end{aligned}\right\} \quad 0 \leq x \leq c, \tag{1.6.31}$$

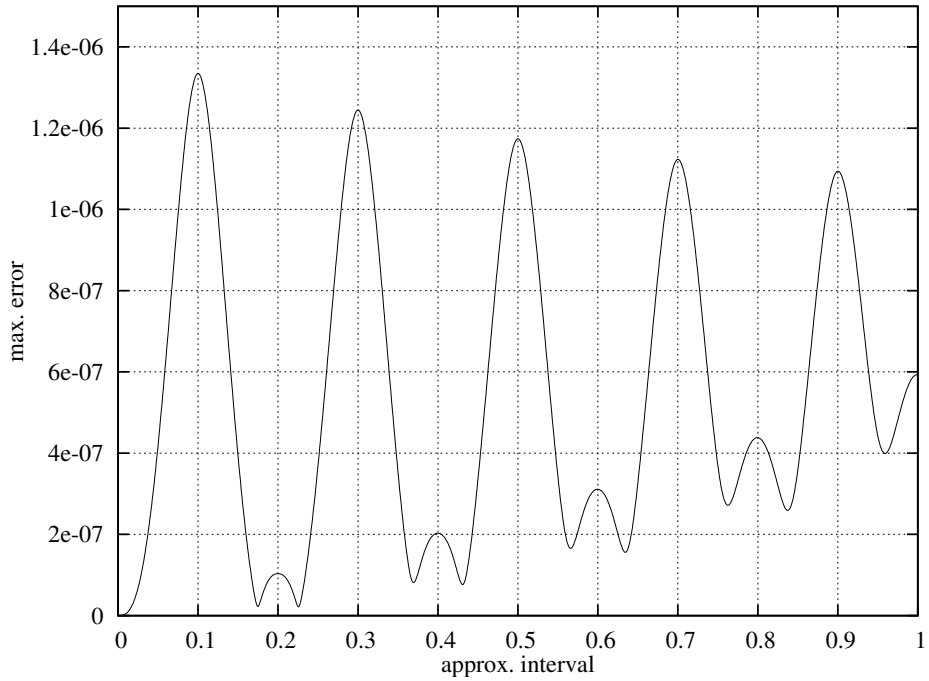

Fig. 1.2. Representing the absolute error margins for the Sylvester matrix differential equation (1.5.29) in the interval $[0,1]$ with step size $h = 0.1$.

where the unknown $Y(x) \in \mathbb{C}^{p\times q}$ and coefficients $A(t) \in \mathbb{C}^{q\times q}, B(x) \in \mathbb{C}^{q\times p}, C(x) \in \mathbb{C}^{p\times q}, D(x) \in \mathbb{C}^{p\times p}$ are differentiable matrix-valued functions arise frequently in important applications to classical control theory [13] and as decoupling techniques for both the analytic and numerical study of boundary value problems [2]. The Riccati equation (1.6.31) has been studied extensively, and different resolution techniques have been introduced (see [14] and references therein).

The study of the Riccati equation (1.6.31) is closely related to the underlying linear system

$$\left.\begin{array}{l} X'(x) = S(x)X(x) \\ X(0) = \begin{bmatrix} I_q \\ Y_0 \end{bmatrix} \end{array}\right\} \text{ where } X(x) = \begin{bmatrix} U(x) \\ V(x) \end{bmatrix}, \; S(x) = \begin{bmatrix} A(x) & B(x) \\ C(x) & -D(x) \end{bmatrix}. \tag{1.6.32}$$

Specifying the solution of (1.6.31) is given by

$$Y(x) = V(x)U^{-1}(x) \tag{1.6.33}$$

where $Y(x)$ is defined in the interval where $U(x)$ is invertible, see [15].

Taking into account lemma 1 and 2 of [16], $U(x)$ is invertible in the interval $[0,\delta]$ and the solution $Y(x)$ of problem (1.6.31) satisfies

$$\|Y(x)\| \leq M \;,\; M = \left(1 - \delta q_0 \exp(\delta k_0) w_0\right)^{-1} w_0 \exp(\delta k_0), \tag{1.6.34}$$

where δ is a positive number satisfying

$$\delta k_0 + \log(\delta) < -\log(q_0\, w_0)\;, \tag{1.6.35}$$

and

$$\left.\begin{aligned} k_0 &= \max\left\{\left\|\begin{bmatrix} A(x) & B(x) \\ C(x) & -D(x)\end{bmatrix}\right\| \;;\; 0 \le x \le c\right\} \\ q_0 &= \max\left\{\|A(x)\; B(x)\| \;;\; 0 \le x \le c\right\} \\ w_0 &= \left\|\begin{matrix} I_q \\ Y_0 \end{matrix}\right\| \end{aligned}\right\} \tag{1.6.36}$$

In accordance with [17, p. 1064], we consider the matrix-valued function

$$F(x,Y) \;=\; C(x) - D(x)Y - YA(x) - YB(x)Y \;, \tag{1.6.37}$$

then, if we define

$$\left.\begin{aligned} a &= \sup\{\|A(x)\|\,; 0 \le x \le \delta\} \\ b &= \sup\{\|B(x)\|\,; 0 \le x \le \delta\} \\ c &= \sup\{\|C(x)\|\,; 0 \le x \le \delta\} \\ d &= \sup\{\|D(x)\|\,; 0 \le x \le \delta\} \end{aligned}\right\} \tag{1.6.38}$$

and $\|Y\| \le M, \left\|\widetilde{Y}\right\| \le M$, with M gives by (1.6.34), the following local Lipschitz condition holds

$$\left\|F(x,Y) - F(x,\widetilde{Y})\right\| \le L\left\|Y - \widetilde{Y}\right\| \;,\; L \;=\; a + d + 2bM \;. \tag{1.6.39}$$

In addition , if $\|Y\| \le N$,

$$\|F(x,Y)\| \le c + N(a + d + bN) \;. \tag{1.6.40}$$

Using the proposed spline method, the only one solution of the matrix equations (1.2.13) and (1.2.18) for $k = 1,\ldots,n-1$ is guaranteed using a fixed-point argument and the global Lipschitz's condition (1.2.8). In our case, we need to prove the only one solution of the matrix equations (1.2.13) and (1.2.18) using a fixed point argument and the local Lipschitz's condition (1.6.39).

We start with the matrix equation (1.2.13). Let us suppose that $\|T\| \le N_1$. Taking into account (1.6.40), we take

$$\begin{cases} N_2 = \|Y(a)\| + h\,\|Y'(a)\| + \frac{h^2}{2}\,\|Y''(a)\| + \frac{h^3}{6}N_1 \\ N_3 = c + N_2(a + d + bN_2) \\ N_4 = \frac{2}{h^2}\left(N_3 + \|Y'(a)\| + h\,\|Y''(a)\|\right) \end{cases} \tag{1.6.41}$$

with a, b, c given by (1.6.38), and let be $N = \max\{N_1, N_2, N_3, N_4, M\}$ with M gives by (1.6.34). Let be $\mathscr{A} = \{Y \in \mathbb{C}^{r\times q}; \|Y\| \le N\}$ and we consider the continuous matrix-valued function of matrix variable $g : \mathbb{C}^{r\times q} \mapsto \mathbb{C}^{r\times q}$ defined by (1.2.19) for $k = 0$. It is simple to verify that if $T \in \mathscr{A}$, by (1.6.41) and (1.6.40) then $g(T) \in \mathscr{A}$. Thus, $g : \mathscr{A} \mapsto \mathscr{A}$ and A_0 is a fixed point of g. In addition, if $T_1, T_2 \in \mathscr{A}$, $\|T_1\| \le M, \|T_2\| \le M$, has then that for f defined by (1.6.37), f fulfills the local Lipschitz's condition (1.6.39) and taking $h < 3/L$, $g(T)$ yields a contractive matrix function, which guarantees that equation (1.2.13) has unique solutions A_0. Hence, the matrix-cubic spline is completely determined in $[a, a + h]$.

For a fixed value of $k = 1, \ldots, n-1$, we construct the cubic-matrix spline $S(x)$ by taking $[a+(k-1)h, a+kh]$ as the last subinterval, for the next subinterval $[a+kh, a+(k+1)h]$. To define the corresponding spline, we need determine $A_k \in \mathbb{C}^{r\times q}$ as the only one solution of the matrix equation (1.2.18). Let us suppose that $\|T\| \leq \widetilde{N}_1$. Taking into account (1.6.40), we take

$$\begin{cases} \widetilde{N}_2 = \left\| S_{|_{[a+(k-1)h,a+kh]}}(a+kh) \right\| + h \left\| S'_{|_{[a+(k-1)h,a+kh]}}(a+kh) \right\| + \frac{h^2}{2} \left\| S''_{|_{[a+(k-1)h,a+kh]}}(a+kh) \right\| + \frac{h^3}{6}\widetilde{N}_1 \\ \widetilde{N}_3 = c + \widetilde{N}_2(a + d + b\widetilde{N}_2) \\ \widetilde{N}_4 = \frac{2}{h^2}\left(\widetilde{N}_3 + \left\| S'_{|_{[a+(k-1)h,a+kh]}}(a+kh) \right\| + h \left\| S''_{|_{[a+(k-1)h,a+kh]}}(a+kh) \right\| \right) \end{cases} \tag{1.6.42}$$

with a, b, c given by (1.6.38), and let be $\widetilde{N} = \max\left\{\widetilde{N}_1, \widetilde{N}_2, \widetilde{N}_3, \widetilde{N}_4, M\right\}$ with M gives by (1.6.34). Let be $\mathscr{A} = \left\{Y \in \mathbb{C}^{r\times q}; \|Y\| \leq \widetilde{N}\right\}$ and we consider the continuous matrix-valued function of matrix variable $g : \mathbb{C}^{r\times q} \mapsto \mathbb{C}^{r\times q}$ defined by (1.2.19).

It is simple to verify that if $T \in \mathscr{A}$, by (1.6.42) and (1.6.40) then $g(T) \in \mathscr{A}$. Thus, $g : \mathscr{A} \mapsto \mathscr{A}$ and A_k is a fixed point of g. In addition, if $T_1, T_2 \in \mathscr{A}$, $\|T_1\| \leq M, \|T_2\| \leq M$, has then that for f defined by (1.6.37), f fulfills the local Lipschitz's condition (1.6.39) and taking $h < 3/L$, $g(T)$ yields a contractive matrix function, which guarantees that equation (1.2.18) has unique solutions A_k. Hence, the matrix-cubic spline is completely determined.

As an additional example for our proposed method, we consider the Riccati matrix differential equation (1.6.31) with

$$A(x) = \begin{pmatrix} -x & 0 \\ -x & x \end{pmatrix}, \; B(x) = \begin{pmatrix} -x^2 & -2 \\ 0 & 1 \end{pmatrix}, \; D(x) = \begin{pmatrix} -1 & -x^2 \\ x & x \end{pmatrix},$$

$$C(x) = \begin{pmatrix} x\left(-e^x + e^x x - x^3\right) & x\left(2e^x - x^2\right) \\ (1-x)x(2+x+2x^2) & 1 + (3-2x)\,x^2 + e^x\left(x - x^4\right) \end{pmatrix}, \; Y(0) = \begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix}. \tag{1.6.43}$$

In this case, the problem has an exact solution given by $Y(x) = \begin{pmatrix} 0 & e^x \\ x^2 & x \end{pmatrix}$, which will permit us to obtain the total error for all our numerical estimates. A short computation using expressions (1.6.34)–(1.6.39) yields the following constants

$$\left.\begin{array}{ll} k_0 = 6.13866 & q_0 = 3 \\ w_0 = \sqrt{2} & \delta = 0.115758 \\ M = 12.0883 & a = 0.173205 \\ b = 2.23609 & c = 1.17928 \\ d = 1.01 & L = 55.2443 \end{array}\right\} \tag{1.6.44}$$

which are necessary for the spline approximation in the interval $[0, 0.1]$, where $\delta = 0.1$ is taken for convenience. Therefore, we need to take $h < 3/L = 0.0543042$ and thus $h = 0.01$. The results are generated with *Mathematica* using `FindRoot` function to solve the emerging algebraic equations, and are summarized in Table 3, where the numerical estimates have been rounded to the fourth relevant digit. In each interval, we evaluated the difference between the estimates of our numerical approach and the exact solution, and then take the Frobenius norm of this difference. The maximum of these errors are indicated in the third column for each subinterval.

Interval	Approximation	Max. Error
$[0, 0.01]$	$\begin{pmatrix} 0 & 1 + x + 0.5\,x^2 + 0.167224\,x^3 \\ x^2 & x \end{pmatrix}$	1.39903×10^{-10}
$[0.01, 0.02]$	$\begin{pmatrix} 0 & 1. + x + 0.499933\,x^2 + 0.169461\,x^3 \\ x^2 & x \end{pmatrix}$	1.39903×10^{-10}
$[0.02, 0.03]$	$\begin{pmatrix} 0 & 1 + x + 0.499864\,x^2 + 0.17061\,x^3 \\ x^2 & x \end{pmatrix}$	1.41977×10^{-10}
$[0.03, 0.04]$	$\begin{pmatrix} 0 & 1 + 1.00001\,x + 0.49966\,x^2 + 0.172877\,x^3 \\ x^2 & x \end{pmatrix}$	1.41977×10^{-10}
$[0.04, 0.05]$	$\begin{pmatrix} 0 & 1 + 1.00001\,x + 0.499518\,x^2 + 0.174063\,x^3 \\ x^2 & x \end{pmatrix}$	1.44084×10^{-10}
$[0.05, 0.06]$	$\begin{pmatrix} 0 & 1 + 1.00003\,x + 0.499173\,x^2 + 0.176362\,x^3 \\ x^2 & x \end{pmatrix}$	1.44084×10^{-10}
$[0.06, 0.07]$	$\begin{pmatrix} 0 & 0.999999 + 1.00004\,x + 0.498952\,x^2 + 0.177587\,x^3 \\ x^2 & x \end{pmatrix}$	1.46223×10^{-10}
$[0.07, 0.08]$	$\begin{pmatrix} 0 & 0.999998 + 1.00008\,x + 0.498463\,x^2 + 0.179918\,x^3 \\ x^2 & x \end{pmatrix}$	1.46223×10^{-10}
$[0.08, 0.09]$	$\begin{pmatrix} 0 & 0.999998 + 1.0001\,x + 0.49816\,x^2 + 0.181181\,x^3 \\ x^2 & x \end{pmatrix}$	1.48391×10^{-10}
$[0.09, 0.1]$	$\begin{pmatrix} 0 & 0.999996 + 1.00016\,x + 0.497521\,x^2 + 0.183546\,x^3 \\ x^2 & x \end{pmatrix}$	1.48391×10^{-10}

Table 1.3. Approximation for Riccati matrix differential equation (1.6.43) in the interval $[0, 0.1]$ with step size $h = 0.01$.

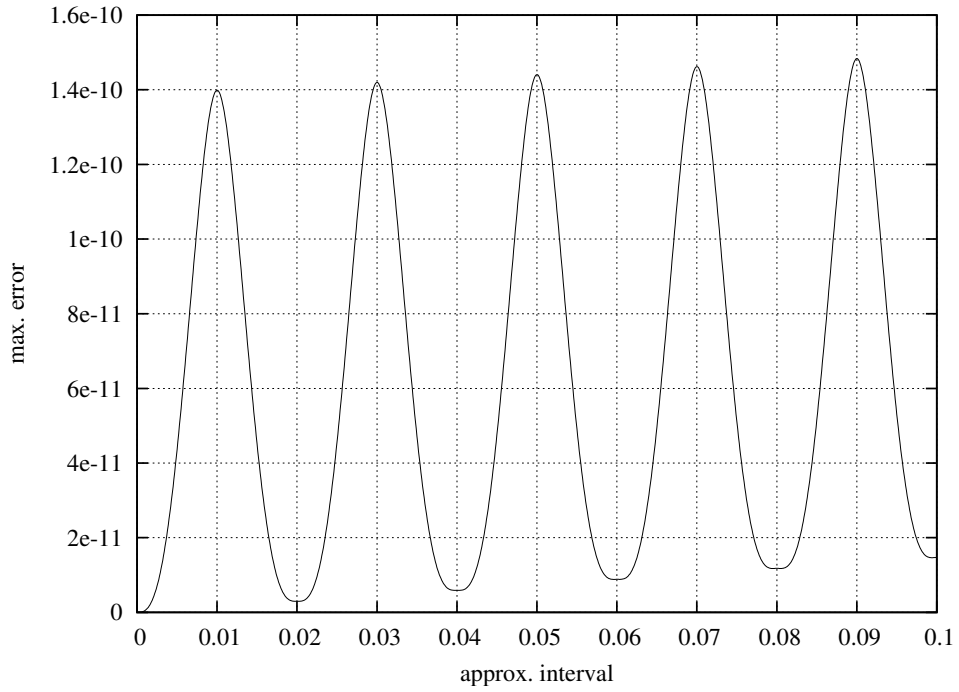

Fig. 1.3. Representing the absolute error margins for the Riccati matrix differential equation (1.6.43) in the interval $[0, 0.1]$ with step size $h = 0.01$.

7 Conclusions

This chapter developed a new method for the numerical integration of first-order matrix differential equations of the non-linear type $Y'(x) = f(x, Y(x)), x \in [a, b]$ using matrix-cubic splines, and thereby generalizing the approach for the linear case in previous work [1]. An important advantage of the proposed method is that the approximated solution is continuous in the interval under consideration, is easy to evaluate, and has an error of the order $O(h^4)$.
Our method is well-suited for implementation on numerical and/or symbolical computer systems (*Mathematica*, MATLAB, etc.) as we have shown in Section 3 giving the explicit algorithm.

For a full demonstration of our approach and its advantages, we conclude with two numerical examples for the Sylvester and Riccati matrix differential equations.

References

1. E. Defez, L. Soler, A. Hervás, and C. Santamaría, *Numerical solutions of matrix differential models using cubic matrix splines*, Comput. Math. Appl. **50** (2005), 693–699.
2. U. M. Ascher, R. M. M. Mattheij, and R. D. Russell, *Numerical solutions of boundary value problems for ordinary differential equations*, Prentice Hall, New Jersey, 1988.
3. L. Jódar and E. Ponsoda, *Continuous numerical solutions and error bounds for matrix differential equations*, Int. Proc. First Int. Colloq. Num. Anal. (Utrecht, The Netherlands) (D. Bainov and V. Covachev, eds.), VSP, 1993, pp. 73–88.
4. F. R. Loscalzo and T. D. Talbot, *Spline function approximations for solutions of ordinary differential equations*, SIAM J. Numer. Anal. **4** (1967), no. 3, 433–445.
5. E. A. Al-Said and M. A. Noor, *Cubic splines method for a system of third-order boundary value problems*, Appl. Math. Comput. **142** (2003), 195–204.
6. A. Graham, *Kronecker products and matrix calculus with applications*, John Wiley, New York, 1981.
7. G. H. Golub and C. F. Van Loan, *Matrix computations*, second ed., The Johns Hopkins University Press, Baltimore, MD, USA, 1989.
8. T. M. Flett, *Differential analysis*, Cambridge University Press, Cambridge, UK, 1980.
9. P. Lancaster, *Explicit solutions of linear matrix equations*, SIAM Review **12** (1970), 544–566.
10. J. M. Ortega and W. C. Rheinboldt, *Iterative solution of nonlinear equations in several variables*, Academic Press, New York, 1970.
11. A. Y. Barraud, *Nouveaux développements sur la résolution numérique de* $x' = ax + xb + c$; $x(0) = c$, RAIRO, Autom. Syst. Anal. Control **16** (1982), no. 4, 341–356.
12. L. Jódar and E. Ponsoda, *Computing continuous numerical solutions of matrix differential equations*, Comput. Math. Appl. **29** (1995), no. 4, 73–84.
13. J. L. Casti, *Dynamical systems and their applications: Linear theory*, Academic Press, New York, 1977.
14. L Jódar and J. C. Cortés López, *Rational matrix approximation with a priori error bounds for non-symmetric matrix riccati equation with analytic coefficients*, IMA J. Numer. Anal. **18** (1998), no. 4, 545–561.
15. W. T. Reid, *Riccati differential equations*, Mathematics in Science and Engineering, Academic Press, New York, 1972.
16. L. Jódar and E. Ponsoda, *Non-autonomous Riccati-type matrix differential equations: Existence interval, construction of continuous numerical solutions and error bounds*, IMA J. Numer. Anal. **15** (1995), 61–74.
17. J. Camacho, J. C. Cortés, E. Navarro, and A. E. Posso, *Chebyshev rational matrix approximation with a priori error bounds for linear and Riccati matrix equations*, Math. Comp. Modelling **35** (2002), 1061–1076.

CHAPTER 2

Numerical solutions of second-order matrix differential equations using cubic splines

1 Introduction

Matrix initial value problems of the form:

$$\left.\begin{aligned} Y''(x) &= f(x, Y(x), Y'(x)) \\ Y(a) = Y_0\ ,\ Y'(a) &= Y_1 \end{aligned}\right\}\ a \le x \le b\ ,\ [a,b] \subset \mathbb{R}, \tag{2.1.1}$$

with matrices $Y_0, Y_1, Y(t) \in \mathbb{C}^{r\times q}$ and matrix function $f : [a,b] \times \mathbb{C}^{r\times q} \times \mathbb{C}^{r\times q} \longmapsto \mathbb{C}^{r\times q}$, are frequent in different fields in physics and engineering. Note that Eq. (2.1.1) could *e.g.* be the statement of Newton's second law of motion for a coupled mechanical system. Models of this kind frequently appear in molecular dynamics, quantum mechanics and for scattering methods, where one solves scalar or vectorial problems with boundary value conditions [1–6].

Numerical methods for the calculation of approximate solutions of simpler matrix problems $Y'' = f(x, Y(x))$ have been studied before using linear multi-step matrix methods with constant steps [7]. In this case, there arise *a priori* error bounds which depend on the data given. As these error bounds display exponential behavior with respect to the integration step h, it will in practice have to take very small values. This unfortunately comes with an increase in computational cost. Similarly, the standard transformation [8] of the differential matrix problem (2.1.1) into an extended first-order system induces a growth in CPU usage. Furthermore, these methods require some additional interpolation techniques in order to obtain a continuous solution, see Ref. [9].

Cubic splines were used in the scalar case to solve first-order differential equations [10], obtaining approximations that, among other advantages, were of class C^1 in a given interval $[a,b]$ and are easy to compute. For these first-order differential scalar problems, the associated approximation errors were of $O(h^4)$, see [10, p. 440], although this must not necessarily hold for scalar problems of different class [11]. Moreover, this method has been used in the resolution of various scalar problems as discussed in Refs. [12, 13], for vector problems [14], linear matrix problems [15], first-order matrix differential equations [16], and also for a simpler case of problem (2.1.1), given in Ref. [17]. In the present work, we extend previous work [15–17] to

the resolution of second-order matrix problems, Eq. (2.1.1) without any additional increase in dimensionality of the problem.

2 Notation and terminology

In this work, $\mathbb{C}^{p\times q}$ will in general denote the set of rectangular $p \times q$ complex matrices. If the matrix satisfies $A \in \mathbb{C}^{r\times s}$, then $\|A\|$ will be its 2-norm defined by

$$\|A\| = \sup_{z\neq 0} \frac{\|Az\|}{\|z\|}.$$

As usual, for a vector $z \in \mathbb{C}^s$ the Euclidean norm is $\|z\| = (z^t z)^{\frac{1}{2}}$ and the maximum norm $\|z\|_\infty = \max\limits_{i=1,\ldots,s} |z_i|$. Following the procedure given in Ref. [18, p. 56], we assume that

$$\max_{ij} |a_{ij}| \leq \|A\| \leq \sqrt{r\,s} \max_{ij} |a_{ij}| . \tag{2.2.2}$$

If $A = (a_{ij}) \in \mathbb{C}^{m\times n}$, the Frobenius norm of A is given by

$$\|A\|_F = \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} |a_{ij}|^2}, \tag{2.2.3}$$

and the following relationship between the 2-norm and Frobenius norm holds (see Ref. [18])

$$\|A\| \leq \|A\|_F \leq \sqrt{n}\,\|A\| . \tag{2.2.4}$$

We will denote by $P_n[x]$ the set of matrix polynomials with degree n and real variable x. Furthermore, we will say that a matrix function $g : [a,b] \mapsto \mathbb{C}^{r\times q}$ is of class $k \geq 0$, written as $g \in C^k([a,b])$, if g is k-times differentiable and its kth derivative is continuous in $[a,b]$. Let $[a,b] \subset \mathbb{R}$ and consider the following partition of $[a,b]$:

$$\Delta = \{a = x_0 < x_1 < \ldots < x_n = b\} .$$

Given m, an integer greater or equal than zero, we define the set of matrix splines of order m as

$$M\_\mathbb{C}^{r\times r}(\Delta)_{m-1}^{m} = \left\{ Q : [a,b] \longmapsto \mathbb{C}^{r\times q} \;;\; Q \in C^{m-1}([a,b]) \,,\; Q_{\big|_{[x_{i-1},x_i]}}(x) \in P_m[x] \;\forall i \in \{1,\ldots,n\} \right\} .$$

For $m = 3$ the matrix splines are called *matrix-cubic splines* as in [19].

The chapter is organized as follows. Section 3 outlines the proposed method and its algorithm given in Section 4. Finally, in Section 5, some practical examples are discussed.

3 Construction of the method

As a starting point, let us consider the initial value problem

$$\left.\begin{array}{l} Y''(x) = f(x, Y(x), Y'(x)) \\ Y(a) = Y_0 \;,\; Y'(a) = Y_1 \end{array}\right\} \; a \leq x \leq b, \tag{2.3.5}$$

where $Y_0, Y_1, Y(t) \in \mathbb{C}^{r\times q}$, $f : [a,b] \times \mathbb{C}^{r\times q} \times \mathbb{C}^{r\times q} \longmapsto \mathbb{C}^{r\times q}$, $f \in C^0(T)$, with

$$T = \{(x, Y, Z) \; ; \; a \le x \le b \; , \; Y, Z \in \mathbb{C}^{r\times q}\} \, . \tag{2.3.6}$$

As usual, the Lipschitz condition on function f, *i.e.*

$$\left.\begin{array}{l} \left\| f(x, Y_1, Y) - f(x, Y_2, Y) \right\| \le L_1 \left\| Y_1 - Y_2 \right\| \; , \; a \le x \le b \, , Y_1, Y_2, Y \in \mathbb{C}^{r\times q} \\ \left\| f(x, Y, Y_1) - f(x, Y, Y_2) \right\| \le L_2 \left\| Y_1 - Y_2 \right\| \; , \; a \le x \le b \, , Y_1, Y_2, Y \in \mathbb{C}^{r\times q} \end{array}\right\} , \tag{2.3.7}$$

guarantees the existence and uniqueness of the continuously differentiable solution $Y(x)$ for the set of equations (2.3.5), see *e.g.* [20, p. 99]. For general procedures and algorithms to estimate the Lipschitz constant for a scalar function, we refer to [21, p. 151], and [22] and references therein. For the matrix case see *e.g.* [23].

In the following, we divide the interval $[a,b]$ into subintervals according to the partition

$$\Delta_{[a,b]} = \{a = x_0 \; < \; x_1 \; < \; \ldots \; < \; x_n = b\} \, , \quad x_k = a + kh, \quad k = 0, 1, \ldots, n, \tag{2.3.8}$$

with step size $h = (b-a)/n$ and n being a positive integer.

For each subinterval $[a+kh, a+(k+1)h]$, we will construct a matrix-cubic spline approximating the solution of Eqs. (2.3.5). Starting off with the first interval $[a, a+h]$, we assume that the spline is given by

$$S\big|_{[a,a+h]}(x) = Y(a) + Y'(a)(x-a) + \frac{1}{2!}Y''(a)(x-a)^2 + \frac{1}{3!}A_0(x-a)^3, \tag{2.3.9}$$

where the matrix $A_0 \in \mathbb{C}^{r\times q}$ is an unknown to be determined. Given this definition of $S\big|_{[a,a+h]}(x)$, it is straightforward to check:

$$S\big|_{[a,a+h]}(a) = Y(a) \; , \quad S'\big|_{[a,a+h]}(a) = Y'(a) \; , \quad S''\big|_{[a,a+h]}(a) = Y''(a) = f(a, S\big|_{[a,a+h]}(a), S'\big|_{[a,a+h]}(a)).$$

By construction, Eq. (2.3.9) satisfies Eqs. (2.3.5) for $x = a$. However, for a complete determination of the spline in $[a, a+h]$, we still have to find A_0. This is readily done by imposing that (2.3.9) is a solution of problem (2.3.5) at $x = a+h$. Hence, we obtain

$$S''\big|_{[a,a+h]}(a+h) = f\left(a+h, S\big|_{[a,a+h]}(a+h), S'\big|_{[a,a+h]}(a+h)\right) . \tag{2.3.10}$$

Applying Eq. (2.3.10), we then find the matrix equation with the remaining unknown matrix A_0:

$$A_0 = \frac{1}{h}\left[f\left(a+h, Y(a) + Y'(a)h + \frac{1}{2}Y''(a)h^2 + \frac{1}{6}A_0h^3, Y'(a) + Y''(a)h + \frac{1}{2}A_0h^2\right) - Y''(a)\right] . \tag{2.3.11}$$

Finally, by imposing again that the matrix equation Eq. (2.3.11) has one solution A_0, the spline is fully determined in the first subinterval $[a, a+h]$.

Now, we move on to the next subinterval $[a+h, a+2h]$. Here, the matrix-cubic spline takes the following form

$$S_{|_{[a+h,a+2h]}}(x) = S_{|_{[a,a+h]}}(a+h)+S'_{|_{[a,a+h]}}(a+h)(x-(a+h)) + \frac{1}{2!}S''_{|_{[a,a+h]}}(a+h)(x-(a+h))^2+\frac{1}{3!}A_1(x-(a+h))^3. \tag{2.3.12}$$

Note that $S(x)$ is defined in such a way that it is of class $C^2([a,a+h]\cup[a+h,a+2h])$, and again all of the coefficients of the spline $S_{|_{[a+h,a+2h]}}(x)$ are determined with the exception of matrix $A_1 \in \mathbb{C}^{r\times q}$.

Repeating the previous procedure in the first subinterval, we use a spline of the form Eq. (2.3.12) which satisfies the differential equation (2.3.5) at $x = a+h$. Then, we are in the position to calculate A_1 assuming that the differential equation (2.3.5) also holds at point $x = a+2h$:

$$S''_{|_{[a+h,a+2h]}}(a+2h) \;=\; f\left(a+2h, S_{|_{[a+h,a+2h]}}(a+2h), S'_{|_{[a+h,a+2h]}}(a+2h)\right) .$$

After further simplification, we can isolate the matrix equation for the only unknown quantity A_1:

$$A_1 = \frac{1}{h}\left[f\left(a+2h, S_{|_{[a,a+h]}}(a+h) + S'_{|_{[a,a+h]}}(a+h)h + \frac{1}{2}S''_{|_{[a,a+h]}}(a+h)h^2 + \frac{1}{6}A_1h^3, S'_{|_{[a,a+h]}}(a+h) + S''_{|_{[a,a+h]}}(a+h)h + \frac{1}{2}A_1h^2\right) - S''_{|_{[a,a+h]}}(a+h)\right] . \tag{2.3.13}$$

Hence, assuming that the matrix equation (2.3.13) has only the solution A_1, the spline of subinterval $[a+h,a+2h]$ is totally determined.

From the previous explanation, it should be clear how to generalize this iteration process for all subsequent subintervals up to the last interval of the partition. For illustration, let us consider without any loss of generality the matrix-cubic spline in any subinterval $[a+(k-1)h,a+kh]$. Then, for the next subinterval $[a+kh,a+(k+1)h]$, we can define the corresponding spline in a similar fashion as seen before:

$$S_{|_{[a+kh,a+(k+1)h]}}(x) = \beta_k(x) \;+\; \frac{1}{3!}A_k(x-(a+kh))^3, \tag{2.3.14}$$

where

$$\beta_k(x) \;=\; \sum_{l=0}^{2}\frac{1}{l!}S^{(l)}_{|_{[a+(k-1)h,a+kh]}}(a+kh)(x-(a+kh))^l. \tag{2.3.15}$$

With this definition, the matrix-cubic spline is of class

$$S(x)\in C^2\left(\bigcup_{j=0}^{k}[a+jh,a+(j+1)h]\right)$$

and fulfills the differential equation (2.3.5) at point $x = a+kh$. As an additional requirement, we assume that $S(x)$ satisfies the differential equation (2.3.5) at point $x = a+(k+1)h$:

$$S''_{|_{[a+kh,a+(k+1)h]}}(a+(k+1)h) \;=\; f\left(a+(k+1)h, S_{|_{[a+kh,a+(k+1)h]}}(a+(k+1)h), S'_{|_{[a+kh,a+(k+1)h]}}(a+(k+1)h)\right),$$

and the equation with the unknown matrix A_k is

$$A_k = \frac{1}{h}\left[f\left(a+(k+1)h, \beta_k(a+(k+1)h)+\frac{1}{6}A_k h^3, \beta_k'(a+(k+1)h)+\frac{1}{2}A_k h^2\right)\right.$$
$$\left.-\beta_k''(a+(k+1)h)\right]. \tag{2.3.16}$$

Observe that this general result for A_k reduces to Eqs. (2.3.11) and (2.3.13), when taking $k=0$ and $k=1$, respectively.

It remains to show that expression Eq. (2.3.16) for A_k in fact provides a unique solution of the problem. In the following, we will demonstrate uniqueness by using a fixed-point argument (see Ref. [16] for the first-order case).

Given a fixed step size h, we consider the matrix function $g:\mathbb{C}^{r\times q}\to\mathbb{C}^{r\times q}$ defined by

$$g(T) = \frac{1}{h}\left[f\left(a+(k+1)h, \beta_k(a+(k+1)h)+\frac{1}{6}Th^3, \beta_k'(a+(k+1)h)+\frac{1}{2}Th^2\right)\right.$$
$$\left.-\beta_k''(a+(k+1)h)\right]. \tag{2.3.17}$$

It is clear that Eq. (2.3.16) will only be valid if and only if $A_k = g(A_k)$, Thus, A_k is a fixed-point solution of function $g(T)$.

By adding and subtracting expression

$$f\left(a+(k+1)h, \beta_k(a+(k+1)h)+\frac{1}{6}T_2h^3, \beta_k'(a+(k+1)h)+\frac{1}{2}T_1h^2\right)$$

to $\|g(T_1)-g(T_2)\|$ using Eq. (2.3.17) and applying Lipschitz's conditions (2.3.7), it follows that

$$\begin{aligned}
&\|g(T_1)-g(T_2)\| \\
&= \tfrac{1}{h}\left\| f\left(a+(k+1)h, \beta_k\left(a+(k+1)h\right)+\tfrac{1}{3!}T_1h^3, \beta_k'\left(a+(k+1)h\right)+\tfrac{1}{2}T_1h^2\right)\right. \\
&\quad -f\left(a+(k+1)h, \beta_k\left(a+(k+1)h\right)+\tfrac{1}{3!}T_2h^3, \beta_k'\left(a+(k+1)h\right)+\tfrac{1}{2}T_1h^2\right) \\
&\quad +f\left(a+(k+1)h, \beta_k\left(a+(k+1)h\right)+\tfrac{1}{3!}T_2h^3, \beta_k'\left(a+(k+1)h\right)+\tfrac{1}{2}T_1h^2\right) \\
&\quad \left. -f\left(a+(k+1)h, \beta_k\left(a+(k+1)h\right)+\tfrac{1}{3!}T_2h^3, \beta_k'\left(a+(k+1)h\right)+\tfrac{1}{2}T_2h^2\right)\right\| \\
&\le \tfrac{1}{2}h\left(\tfrac{L_1h^2}{3}+L_2\right)\|T_1-T_2\|.
\end{aligned}$$

Taking

$$h < \left(\sqrt{24L_1+9L_2^2}-3L_2\right)/2L_1$$

it is

$$\frac{1}{2}h\left(L_1\frac{h}{3}+L_2\right) < 1.$$

Therefore, $g(T)$ is a contractive matrix function and consequently Eq. (2.3.16) has unique solutions A_k for $k=0,1,\ldots,n-1$. This completes our proof that the matrix-cubic spline of cubic order is completely determined.

Summing up, the following result can been established (for an analogous argument see also Ref. [10] by using relation (2.2.2)):

Theorem 2.1.

(i) Let L_1, L_2 be Lipschitz constants defined by Eq. (2.3.7). If step size $h < \left(\sqrt{24L_1 + 9L_2^2} - 3L_2\right)/2L_1$ is chosen, then there exists a matrix-cubic spline $S(x)$ for each subinterval $[a + kh, a + (k + 1)h]$, $k = 0, 1, \ldots, n - 1$ (following our previously outlined construction method).

(ii) If $f \in C^1(T)$, then $\|Y(x) - S(x)\|$ is at least of order $O(h^3)$ $\forall x \in [a, b]$, where $Y(x)$ is the theoretical solution of Eq. (2.3.5).

4 Algorithm

The algorithm used is a straightforward implementation of the procedure explained in Section 3. As explained before, it will compute the approximate solution of Eq. (2.3.5) by means of matrix-cubic splines for the interval $[a, b]$ with an error at least of order $O(h^3)$ for step size h of the partition $\Delta_{[a,\, b]}$. Diagram 2.1 schematically displays this algorithm in a flow diagram. It can be easily implemented in various computer languages.

5 Examples

For some particular cases, the matrix equations (2.3.11) and (2.3.16) can be solved analytically [24], otherwise they can be tackled with standard iterative methods (see *e.g.* [25, 26]) using

$$T_{l+1}^s = g(T_l^s), \text{where } T_0^s \text{ is an arbitrary matrix in } \mathbb{C}^{r\times q} \text{ and } s = 0, 1, \ldots, n - 1,$$

where $g(T)$ is given by Eq. (2.3.17).

Knowing the fully analytical results for some standard examples of matrix differential equation of second order enables us to study the quality and convergence of the approximate results generated by our algorithm. In the following two sections, we have chosen a non-linear vector system and an incomplete second-order differential for testing the proposed algorithm.

5.1 A non-linear differential vector system

As a toy example, let us consider the following trivial non-linear vector system:

$$\left.\begin{aligned} y_1''(x) &= 1 - \cos(x) + \sin\left(y_2'(x)\right) + \cos\left(y_2'(x)\right) \\ y_2''(x) &= \frac{1}{4 + y_1(x)^2} - \frac{1}{5 - \sin^2(x)} \\ y_1(0) &= 1, \qquad y_2(0) = 0, \\ y_1'(0) &= 0, \qquad y_2'(0) = \pi \end{aligned}\right\} \; 0 \le x \le 1. \tag{2.5.18}$$

It is easy to verify that this problem has the exact solution $y_1(x) = \cos(x), y_2(x) = \pi x$. Thus, we can compare our numerical estimates with this solution in order to obtain the exact errors of the approximation.

The system Eq. (2.5.18) can be recast in the more compact form

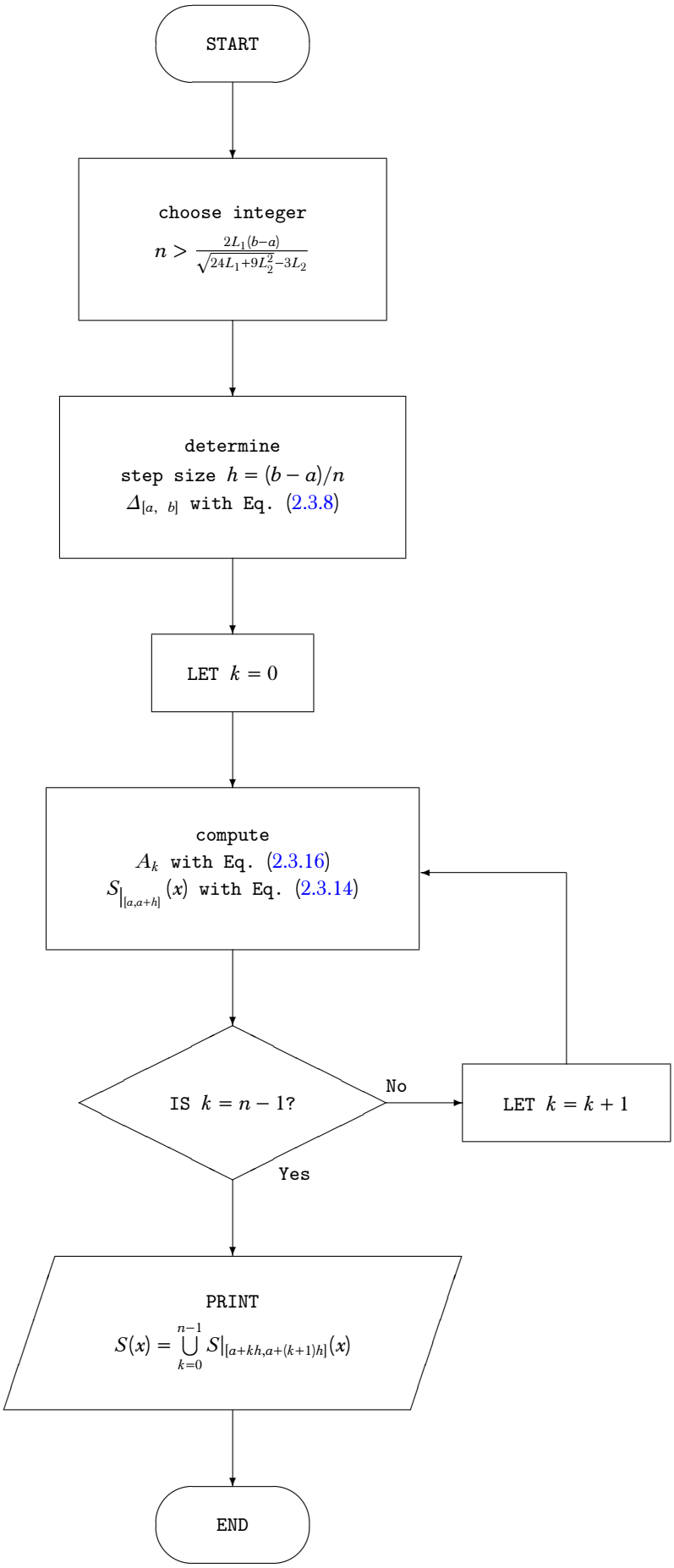

Diagram 2.1. Flow diagram representing the algorithm for matrix-cubic spline approximation of differential matrix systems of type $Y''(x) = f(x, Y(x), Y'(x))$ with constant $Y_0 = Y(a)$ and $Y_1 = Y'(a)$ at initial point $a \in \mathbb{R}$.

$$\begin{aligned} Y''(x) &= F(x, Y, Y'), \quad Y(x) = \begin{pmatrix} y_1(x) \\ y_2(x) \end{pmatrix}, \\ Y(0) &= \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad Y'(0) = \begin{pmatrix} 0 \\ \pi \end{pmatrix}, \end{aligned} \tag{2.5.19}$$

where

$$F(x, Y, Y') = \begin{pmatrix} 1 - \cos(x) + \sin\left(y_2'(x)\right) + \cos\left(y_2'(x)\right) \\ \dfrac{1}{4 + y_1(x)^2} - \dfrac{1}{5 - \sin^2(x)} \end{pmatrix}. \tag{2.5.20}$$

Thus, we obtain

$$Y''(0) = F(0, Y(0), Y'(0)) = \begin{pmatrix} -1 \\ 0 \end{pmatrix},$$

and it is not difficult to show that $F(x, Y, Y')$, given by Eq. (2.5.20), fulfills the global Lipschitz conditions:

$$\left.\begin{aligned} \|F(x, Y_1, Y) - F(x, Y_2, Y)\| &\le \|Y_1 - Y_2\| \\ \|F(x, Y, Y_1) - F(x, Y, Y_2)\| &\le 2\|Y_1 - Y_2\| \end{aligned}\right\}, \; 0 \le x \le 1, Y, Y_1, Y_2 \in \mathbb{R}^2. \tag{2.5.21}$$

Next, we determine L_1 and L_2 calculated by Eq. (2.3.7) as $L_1 = 1, L_2 = 2$. Theorem 2.1 implies that we need to take $h < 0.872983$, so here we choose $h = 0.1$. To attack the emerging algebraic equations, we have used *Mathematica* Most of the results are generated using its `FindRoot` function. Table 2.1 summarizes all the numerical estimates, which have been rounded to the fifth relevant digit.

Interval	Approximation	Error	Max. Error
$[0, 0.1]$	$\begin{pmatrix} 1 - 0.5x^2 + 0.00833x^3 \\ 3.14159x \end{pmatrix}$	4.16114×10^{-6}	4.16114×10^{-6}
$[0.1, 0.2]$	$\begin{pmatrix} 0.99998 + 0.00050x - 0.50497x^2 + 0.02490x^3 \\ 3.14159x \end{pmatrix}$	1.66032×10^{-5}	1.66032×10^{-5}
$[0.2, 0.3]$	$\begin{pmatrix} 0.99985 + 0.00246x - 0.51476x^2 + 0.04122x^3 \\ 3.14159x \end{pmatrix}$	3.72028×10^{-5}	3.72325×10^{-5}
$[0.3, 0.4]$	$\begin{pmatrix} 0.99942 + 0.00675x - 0.52908x^2 + 0.05713x^3 \\ 3.14159x \end{pmatrix}$	6.57658×10^{-5}	6.57551×10^{-5}
$[0.4, 0.5]$	$\begin{pmatrix} 0.99844 + 0.01412x - 0.54749x^2 + 0.07247x^3 \\ 3.14158x + 0.00002x^2 - 0.00001x^3 \end{pmatrix}$	1.02012×10^{-4}	1.01985×10^{-4}
$[0.5, 0.6]$	$\begin{pmatrix} 0.99662 + 0.02507x - 0.56941x^2 + 0.08708x^3 \\ 3.14159x \end{pmatrix}$	1.45563×10^{-4}	1.45550×10^{-4}
$[0.6, 0.7]$	$\begin{pmatrix} 0.99365 + 0.03992x - 0.59415x^2 + 0.10082x^3 \\ 3.14156x + 0.00004x^2 - 0.00002x^3 \end{pmatrix}$	1.96167×10^{-4}	1.96024×10^{-4}
$[0.7, 0.8]$	$\begin{pmatrix} 0.98928 + 0.0586x - 0.62090x^2 + 0.11356x^3 \\ 3.14159x \end{pmatrix}$	2.52912×10^{-4}	2.52907×10^{-4}
$[0.8, 0.9]$	$\begin{pmatrix} 0.98334 + 0.08091x - 0.64873x^2 + 0.12516x^3 \\ 0.00002 + 3.14153x + 0.00008x^2 - 0.00003x^3 \end{pmatrix}$	3.15643×10^{-4}	3.15637×10^{-4}
$[0.9, 1.0]$	$\begin{pmatrix} 0.97579 + 0.10608x - 0.67670x^2 + 0.13552x^3 \\ 3.1416x \end{pmatrix}$	3.83638×10^{-4}	3.83579×10^{-4}

Table 2.1. Approximation for the vector differential system Eq. (2.5.18) in the interval $[0, 1]$ with step size $h = 0.1$.

In each interval, we evaluated the difference between the estimates of our numerical approach and the exact solution, and then took the Frobenius norm of this difference. The corresponding errors are indicated in the third column. In Figure 2.1 these errors are plotted for each subinterval. Note that although the error is growing almost linearly, the error margin is

very small, namely $[0, 0.0004]$, and well within the bounds dictated by Theorem 2.1. To avoid any averaging, as occurs using the Frobenius norm, we also add in the last column the error corresponding to the maximum norm evaluated at the final point of each interval.

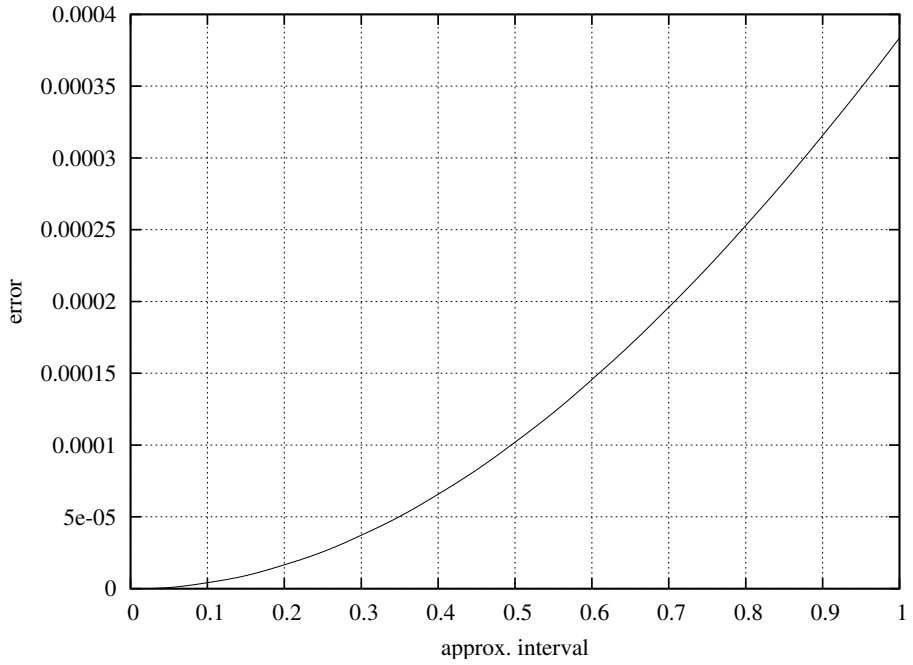

Figure 2.1. Error margins for the vector differential system Eq. (2.5.18) in the interval $[0, 1]$ with step size $h = 0.1$.

5.2 Incomplete second-order differential system

The problem

$$Y''(t) + A\,Y(t) = 0, \tag{2.5.22}$$

with $Y(0) = Y_0$, $Y'(0) = Y_1$, has the exact solution

$$Y(t) = \cos\left(\sqrt{A}t\right)Y_0 + \left(\sqrt{A}\right)^{-1}\sin\left(\sqrt{A}t\right)Y_1, \tag{2.5.23}$$

where $\sqrt{A}$ denotes any square root of a non-singular matrix A (see *e.g.* Ref. [27]). A major disadvantage of this formal solution is the non-trivial computation of $\sqrt{A}, \cos(\sqrt{A}t)$ and $\sin(\sqrt{A}t)$. Our proposed method aims at avoiding these difficulties.

For our example Eq. (2.5.22), we choose the parameters

$$A = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}, \quad Y_0 = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad Y_1 = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \quad t \in [0, 1]. \tag{2.5.24}$$

The corresponding exact solution, as given in Ref. [7], is

$$Y(t) = \sin\left[\begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix} t\right] = \begin{pmatrix} \sin(t) & 0 \\ t\cos(t) & \sin(t) \end{pmatrix}.$$

In this case, we have $L_2 = 0$ and $L_1 \approx 2.82843$. According to Theorem 2.1, we need to take $h < 1.45647$ as in Ref. [17], so we choose $h = 0.1$ for example. Again, we use *Mathematica* to

solve the algebraic equations which arise from the algorithm of Figure 2.1. Table 2.2 lists all numerical estimates (rounded to the fifth relevant digit). The maximum error for each subinterval is indicated in the third column of the table. Figure 2.2 gives a graphical representation for these errors. Note again that although the error is increasing within an error margin of $[0, 0.0007]$, and thus much below the bounds dictated by Theorem 2.1.

Interval	Approximation	Error
$[0, 0.1]$	$\begin{pmatrix} x - 0.1664x^3 & 0 \\ x - 0.4986x^3 & x - 0.1664x^3 \end{pmatrix}$	1.0072×10^{-6}
$[0.1, 0.2]$	$\begin{pmatrix} 1.00005x - 0.0005x^2 - 0.1647x^3 & 0 \\ 1.0002x - 0.0025x^2 - 0.4903x^3 & 1.0001x - 0.0005x^2 - 0.1647x^3 \end{pmatrix}$	6.3032×10^{-6}
$[0.2, 0.3]$	$\begin{pmatrix} 1.0005x - 0.0025x^2 - 0.1614x^3 & 0 \\ -0.0001 + 1.0022x - 0.0124x^2 - 0.4738x^3 & 1.0005x - 0.0025x^2 - 0.1614x^3 \end{pmatrix}$	2.0059×10^{-5}
$[0.3, 0.4]$	$\begin{pmatrix} -0.0002 + 1.0018x - 0.0069x^2 - 0.1565x^3 & 0 \\ -0.0008 + 1.0088x - 0.0344x^2 - 0.4494x^3 & -0.0002 + 1.0018x - 0.0069x^2 - 0.1565x^3 \end{pmatrix}$	4.6213×10^{-5}
$[0.4, 0.5]$	$\begin{pmatrix} -0.0006 + 1.0049x - 0.0147x^2 - 0.1500x^3 & 0 \\ -0.0028 + 1.0242x - 0.0728x^2 - 0.4174x^3 & -0.0006 + 1.0049x - 0.0147x^2 - 0.1500x^3 \end{pmatrix}$	8.8359×10^{-5}
$[0.5, 0.6]$	$\begin{pmatrix} -0.0016 + 1.0109x - 0.0266x^2 - 0.1420x^3 & 0 \\ -0.0077 + 1.0536x - 0.1316x^2 - 0.3782x^3 & -0.0016 + 1.0109x - 0.0266x^2 - 0.1420x^3 \end{pmatrix}$	1.4964×10^{-4}
$[0.6, 0.7]$	$\begin{pmatrix} -0.0036 + 1.0210x - 0.0436x^2 - 0.1327x^3 & 0 \\ -0.0176 + 1.1030x - 0.2140x^2 - 0.3324x^3 & -0.0036 + 1.0210x - 0.0436x^2 - 0.1327x^3 \end{pmatrix}$	2.3267×10^{-4}
$[0.7, 0.8]$	$\begin{pmatrix} -0.0073 + 1.0368x - 0.0661x^2 - 0.1219x^3 & 0 \\ -0.0354 + 1.1791x - 0.3227x^2 - 0.2807x^3 & -0.0073 + 1.0368x - 0.0661x^2 - 0.1219x^3 \end{pmatrix}$	3.3941×10^{-4}
$[0.8, 0.9]$	$\begin{pmatrix} -0.0134 + 1.0597x - 0.0947x^2 - 0.1100x^3 & 0 \\ -0.0646 + 1.2885x - 0.4595x^2 - 0.2237x^3 & -0.0134 + 1.0597x - 0.0947x^2 - 0.1100x^3 \end{pmatrix}$	4.7114×10^{-4}
$[0.9, 1]$	$\begin{pmatrix} -0.0229 + 1.0914x - 0.1299x^2 - 0.0970x^3 & 0 \\ -0.1093 + 1.4378x - 0.6253x^2 - 0.1623x^3 & -0.0229 + 1.0914x - 0.1299x^2 - 0.0970x^3 \end{pmatrix}$	6.2838×10^{-4}

Table 2.2. Approximation for the incomplete second-order differential system Eq. (2.5.22) in the interval $[0, 1]$ with step size $h = 0.1$ and parameters Eq. (2.5.24).

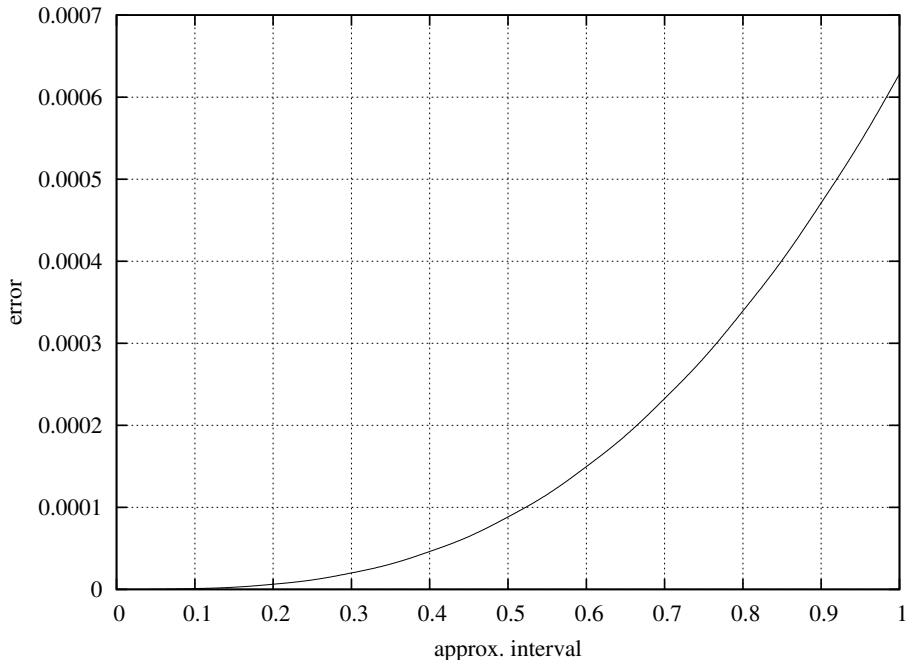

Figure 2.2. Error margins for the incomplete second-order differential system Eq. (2.5.22) in the interval $[0, 1]$ with step size $h = 0.1$.

5.3 Second-order polynomial matrix equation

Here, we consider the following problem in $\mathbb{C}^{2\times 2}$

$$Y''(t) + A_1\, Y'(t) + A_0 Y(t) = 0, \quad t \in [0,1], \tag{2.5.25}$$

where

$$A_1 = \begin{pmatrix} -1 & 1 \\ 0 & -2 \end{pmatrix}, \quad A_0 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix},$$

employing the method described in [28] with the general solution (2.5.25) given by

$$Y(t) = \begin{pmatrix} 1 \\ 0 \end{pmatrix} D_1 + \begin{pmatrix} 1 & -1 & 1 \\ 0 & -1 & 1 \end{pmatrix} \exp\left[\begin{pmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix} t \right] D_2, \quad D_1 \in \mathbb{C}^{1\times 2}, \quad D_2 \in \mathbb{C}^{3\times 2}\,. \tag{2.5.26}$$

With the initial conditions $Y(0) = Y'(0) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$, a simple calculation shows that

$$D_1 = \begin{pmatrix} 0 & -1 \end{pmatrix}, \quad D_2 = \begin{pmatrix} 1 & 0 \\ 0 & -1 \\ 0 & 0 \end{pmatrix},$$

and thus the solution of the problem (2.5.25) takes the simple form:

$$Y(t) = \begin{pmatrix} e^t & -1 + e^t - e^t t \\ 0 & e^t \end{pmatrix}, \quad t \in [0,1]. \tag{2.5.27}$$

Taking into account that (2.5.25), we obtain

$$f(t, Y, Z) = -A_0 Y - A_1 Z,$$

and therefore

$$\left.\begin{aligned} \left\| f\left(t, Y_1, Z\right) - f\left(t, Y_2, Z\right) \right\|_2 &\le L_1 \left\| Y_1 - Y_2 \right\|_2 \\ \left\| f\left(t, Y, Z_1\right) - f\left(t, Y, Z_2\right) \right\|_2 &\le L_2 \left\| Z_1 - Z_2 \right\|_2 \end{aligned}\right\}, \tag{2.5.28}$$

where $L_1 = \|A_0\|_2 = 1$ and $L_2 = \|A_1\|_2 = 2.28825$. Where we must take the step size given by the constraint $h < \left(\sqrt{24L_1 + 9L_2^2} - 3L_2\right)/2L_1 = 0.784401$.

In Table 2.3, we use a step size of $h = 0.1$ to represent the cubic-matrix approximation for the problem at hand, rounded to the fourth significant digit. Again, the maximum error (given by the Frobenius norm) is displayed in the fourth column of the table. Figure 2.3 displays a graphical representation for these errors. Observe that although the error is increasing up to a value of $[0, 0.0035]$, it remains well within the error bounds fixed by Theorem 2.1.

6 Conclusions

In this chapter, we have presented a novel method for the numerical treatment of second-order differential matrix systems of the type $Y''(x) = f(x, Y(x), Y'(x)), x \in [a,b]$, as they are frequently encountered in engineering modeling. Our approach is a generalization of previously developed methods employing matrix-cubic splines for lower-order equations.

Interval	Approximation	Error
$[0,0.1]$	$\begin{pmatrix} 1+x+0.5x^2+0.1754x^3 & -0.5x^2-0.3601x^3 \\ 0 & 1+x+0.5x^2+0.1756x^3 \end{pmatrix}$	1.53895×10^{-5}
$[0.1,0.2]$	$\begin{pmatrix} 1+1.0006x+0.4945x^2+0.1939x^3 & 0.0001-0.0017x-0.4828x^2-0.4175x^3 \\ 0 & 1+1.0006x+0.4944x^2+0.1941x^3 \end{pmatrix}$	6.67523×10^{-5}
$[0.2,0.3]$	$\begin{pmatrix} 0.9998+1.0030x+0.4822x^2+0.2143x^3 & 0.0006-0.0096x-0.4435x^2-0.4829x^3 \\ 0 & 0.9998+1.0030x+0.4822x^2+0.2146x^3 \end{pmatrix}$	1.63924×10^{-4}
$[0.3,0.4]$	$\begin{pmatrix} 0.9992+1.0091x+0.4619x^2+0.2369x^3 & 0.0026-0.0297x-0.3764x^2-0.5576x^3 \\ 0 & 0.9992+1.0091x+0.4618x^2+0.2372x^3 \end{pmatrix}$	3.18789×10^{-4}
$[0.4,0.5]$	$\begin{pmatrix} 0.9976+1.0211x+0.4320x^2+0.2618x^3 & 0.0080-0.0705x-0.2744x^2-0.6425x^3 \\ 0 & 0.9976+1.0211x+0.4318x^2+0.2623x^3 \end{pmatrix}$	5.45654×10^{-4}
$[0.5,0.6]$	$\begin{pmatrix} 0.9942+1.0417x+0.3907x^2+0.2894x^3 & 0.0201-0.1430x-0.1293x^2-0.7392x^3 \\ 0 & 0.9942+1.0419x+0.3903x^2+0.2899x^3 \end{pmatrix}$	8.61682×10^{-4}
$[0.6,0.7]$	$\begin{pmatrix} 0.9876+1.0746x+0.3358x^2+0.3198x^3 & 0.0439-0.2618x+0.0686x^2-0.8492x^3 \\ 0 & 0.9875+1.0749x+0.3352x^2+0.3205x^3 \end{pmatrix}$	1.28740×10^{-3}
$[0.7,0.8]$	$\begin{pmatrix} 0.9760+1.1241x+0.2651x^2+0.3535x^3 & 0.0867-0.4454x+0.3309x^2-0.9741x^3 \\ 0 & 0.9759+1.1246x+0.2642x^2+0.3543x^3 \end{pmatrix}$	1.84731×10^{-3}
$[0.8,0.9]$	$\begin{pmatrix} 0.9570+1.1956x+0.1758x^2+0.3907x^3 & 0.1593-0.7176x+0.6712x^2-1.1159x^3 \\ 0 & -0.9568+1.1964x+0.1745x^2+0.3917x^3 \end{pmatrix}$	2.57055×10^{-3}
$[0.9,1]$	$\begin{pmatrix} 0.9270+1.2955x+0.0648x^2+0.4318x^3 & 0.2766-1.1086x+1.1055x^2-1.2768x^3 \\ 0 & 0.9267+1.2969x+0.0628x^2+0.4331x^3 \end{pmatrix}$	3.49171×10^{-3}

Table 2.3. Approximation for the second-order polynomial matrix equation Eq. (2.5.25) in the interval $[0,1]$ with step size $h=0.1$.

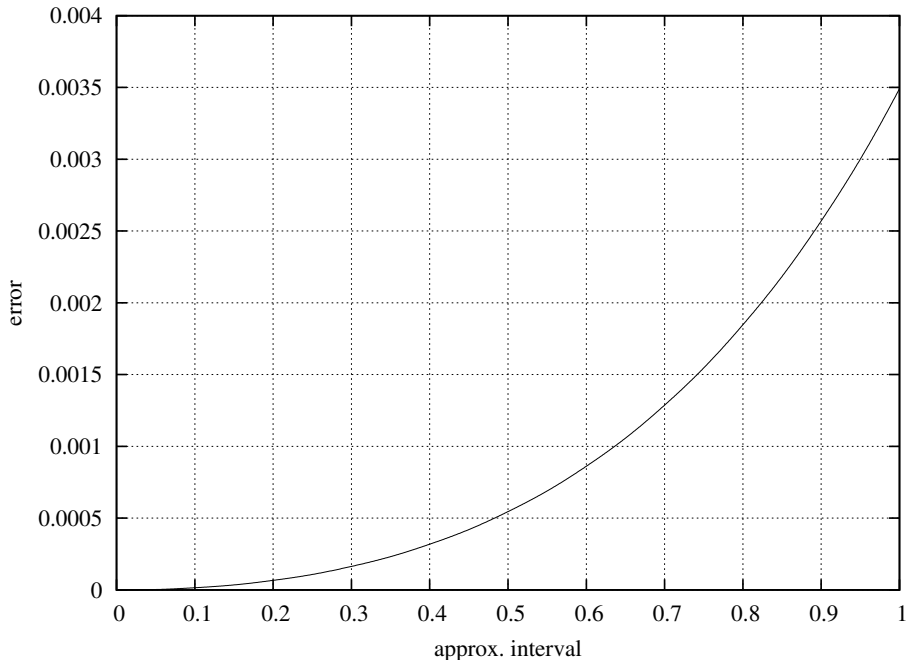

Figure 2.3. Error margins for the second-order polynomial matrix equation Eq. (2.5.25) in the interval $[0,1]$ with step size $h=0.1$.

There are several important advantages of our proposed method. Firstly, the algorithm is straightforward to implement on numerical and symbolical computer systems or by even using suitable low-level programming languages. Secondly, in the case of second-order differential matrix systems, our method does not require to disentangle the system at hand and reduce it to a higher dimensional system of lower order, commonly practice in problems of this kind. This reduction would only come at the price of increasing computational cost. Thirdly, all spline solutions are by construction already continuous in the interval under consideration. Three explicit numerical examples have tested the method and have shown that errors are at most of the order $O(h^3)$. By adapting the step size to a particular problem, in principle, any desired

accuracy can be reached.

With these benefits, it is hoped that our approach provides an alternative method to existing ones and may open up new avenues to the numerical integration of second-order models in practical applications. Especially, if work in progress will show that it may be a second option to yet existing boundary-value integrators, such as MATLAB's BVP routines.

References

1. P. Marzulli, *Global error estimates for the standard parallel shooting method*, J. Comput. Appl. Math. **34** (1991), 233–241.
2. J. M. Ortega, *Numerical analysis: A second course*, Academic Press, New York, 1972.
3. B. W. Shore, *Comparison of matrix methods to the radii Schrödinger eigenvalue equation: The Morse potential*, J. Chemical Physics **59** (1971), no. 12, 6450–6463.
4. C. Froese, *Numerical solutions of the hartree-fock equations*, Can. J. Phys. **41** (1963), 1895–1910.
5. J. R. Claeyssen, G. Canahualpa, and C. Jung, *A direct approach to second-order matrix non-classical vibrating equations*, Appl. Numer. Math. **30** (1999), 65–78.
6. J. F. Zhang, *Optimal control for mechanical vibration systems based on second-order matrix equations*, Mechanical Systems and Signal Processing **16** (2002), no. 1, 61–67.
7. L. Jódar, J. L. Morera, and R.J. Villanueva, *Numerical multistep matrix methods for $y'' = f(t, y)$*, Appl. Math. Comput. **59** (1993), 257–274.
8. E. A. Coddington and N. Levinson, *Theory of ordinary differential equations*, McGraw-Hill, New York, 1955.
9. L. Jódar and E. Ponsoda, *Continuous numerical solutions and error bounds for matrix differential equations*, Int. Proc. First Int. Colloq. Num. Anal. (Utrecht, The Netherlands), VSP, 1993, pp. 73–88.
10. F. R. Loscalzo and T. D. Talbot, *Spline function approximations for solutions of ordinary differential equations*, SIAM J. Numer. Anal. **4** (1967), no. 3, 433–445.
11. E. A. Al-Said, *The use of cubic splines in the numerical solution of a system of second-order boundary value problems*, Comput. Math. Appl. **42** (2001), 861–869.
12. M. K. Kadalbajoo and K. C. Patidar, *Numerical solution of singularly perturbed two-point boundary value problems by spline in tension*, Appl. Math. Comput. **131** (2002), 299–320.
13. E. A. Al-Said and M. A. Noor, *Cubic splines method for a system of third-order boundary value problems*, Appl. Math. Comput. **142** (2003), 195–204.
14. G. Micula and A. Revnic, *An implicit numerical spline method for systems for ODE's*, Appl. Math. Comput. **111** (2000), 121–132.
15. E. Defez, L. Soler, A. Hervás, and C. Santamaría, *Numerical solutions of matrix differential models using cubic matrix splines*, Comput. Math. Appl. **50** (2005), 693–699.
16. E. Defez, L. Soler, A. Hervás, and M. M. Tung, *Numerical solutions of matrix differential models using cubic matrix splines II*, Mathematical and Computer Modelling **46** (2007), 657–669.
17. M. M. Tung, L. Soler, E. Defez, and A. Hervás, *Cubic-matrix splines and second-order matrix model*, The 14th European Conference on Mathematics for Industry (ECMI 2006) (Universidad Carlos III de Madrid, Spain), 2006.
18. G. H. Golub and C. F. Van Loan, *Matrix computations*, second ed., The Johns Hopkins University Press, Baltimore, MD, USA, 1989.
19. E. Defez, A. Hervás, A. Law, J. Villanueva-Oller, and R. Villanueva, *Matrix-cubic splines for progressive transmission of images*, J. Math. Imaging Vision **17** (2002), no. 1, 41–53.
20. T. M. Flett, *Differential analysis*, Cambridge University Press, 1980.
21. C. T. Kelley, *Iterative methods for optimization*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 1999.
22. G. Wood and B. Zhang, *Estimation of the Lipschitz constant of a function*, J. Global Optim. **8** (1996), no. 1, 91–103.
23. L. Jódar and H. Abou-Kandil, *A resolution method for Riccati differential systems coupled in their quadratic terms*, SIAM Journal on Mathematical Analysis **19** (1988), no. 6, 1425–1430.
24. P. Lancaster, *Explicit solutions of linear matrix equations*, SIAM Review **12** (1970), 544–566.
25. P. T. Boggs, *The solution of nonlinear systems of equations by A-stable integration techniques*, SIAM J. Numer. Anal. **8** (1971), no. 4, 767–785.
26. J. M. Ortega and W. C. Rheinboldt, *Iterative solution of nonlinear equations in several variables*, Academic Press, New York, 1970.
27. G. I. Hargreaves and N. J. Higham, *Efficient algorithms for the matrix cosine and sine*, Numerical Algorithms **40** (2005), 383–400.
28. L. Jódar and E. Navarro, *Rectangular co-solutions of polynomial matrix equations and applications*, Appl. Math. Lett. **4** (1991), no. 2, 13–16.

CHAPTER 3

Approximate solutions of linear matrix differential equations with higher-order splines

1 Introduction

Matrix differential equations emerge frequently in a great variety of models in physics and engineering [1–3]. Apart from problems where the mathematical framework is cast in matrix form, they also appear when special techniques to solve scalar or vectorial problems are used. Examples of such situations are the embedding methods for the study of linear boundary value problems [4], shooting methods for scalar or vectorial problems with boundary values conditions [5], lines method for the numerical integration of partial differential equations [6] or homotopic methods to solve non-linear systems equations [7].

The vectorization techniques to transform a matrix problem into a set of scalar equations has several drawbacks [8]. Firstly, the physical sense of the magnitudes is lost with vectorization techniques. Secondly, the computational cost increases. Moreover, these vectorization techniques interfere with the advantages of symbolic languages especially adapted to deal with matrix expressions.

In this work we will develop a method for the numerical integration of first-order matrix differential linear equations given by

$$\left.\begin{aligned} Y'(x) &= A(x)Y(x) + B(x),\ a \le x \le b \\ Y(a) &= Y_a \end{aligned}\right\}. \tag{3.1.1}$$

Here, we assume $Y_a, Y \in \mathbb{C}^{r\times q}$, $A : [a,b] \to \mathbb{C}^{r\times r}$, $B : [a,b] \to \mathbb{C}^{r\times q}$ with $A, B \in \mathscr{C}^s([a,b])$, $s \ge 1$, which guarantees the existence of a unique and continuously differentiable solution $Y(x)$ of (3.1.1), see [9, p. 99].

Problem (3.1.1) is not only used in the mathematical modelling of many different technological applications [10], but also permits to deal with nonlinear problems, such as Riccati equations [11–13], after employing some linearization techniques. Numerical methods for the calculation of approximate solutions of problems of the type (3.1.1) by means of linear multi-

step methods with constant steps have been studied in [14]. Although for these methods exist *a priori* errors bounds as function of the problem data, these error bounds will be given in terms of an exponential depending on the integration step h, and thus require in practice a very small value for h. Therefore, these methods will involve some interpolation techniques in order to obtain a continuous solution, [14]. Other methods, based on the developments of Magnus and Fer [15], require the calculation of the matrix exponential at high computational cost. Another alternative method would be the so-called B-splines method, which combines linear multi-step methods and B-splines interpolation (see [16] and references therein).

In the scalar case, cubic splines were used in for the resolution of ordinary differential equations [17], obtaining approximations that, among other advantages, were of class C^1 in the interval $[a,b]$, and easily to evaluate with an error of the order $O(h^4)$. Recently, splines have also been used in the resolution of other scalar problems [18]. For example, Ref. [19] develops an implicit spline method by means of Hermite interpolation techniques to tackle vector problems.

The corresponding generalizations of the Loscalzo-Talbot method to the matrix framework have been carried out in Refs. [20, 21]. Unfortunately, as already detected by Loscalzo and Talbot in [17], their scalar procedure is divergent when higher-order spline functions are used [17, p. 444–445]. Their numerical computations have explicitly shown that the system $y' = y,\ y(0) = 1$, contains significant divergences for splines of order $m > 3$. However, our new method avoids these problems with divergences for splines $S(x)$ of order m, provided they are of differentiability class $\mathscr{C}^1$.

In this chapter, we propose a method using higher-order matrix splines for the numerical approximation to the solution of (3.1.1). The present work extends all important advantages already obtained in [17] for the scalar case to the matrix framework.

We adopt the following organization. In section 2 we develop the proposed method including the study of the approximation error and formulate a constructive algorithm. Finally, in section 3 we conclude with some illustrative examples of the new method.

Along this work we will denote by $\mathbb{C}^{p\times q}$ the set of rectangular $p \times q$ complex matrices, and $\|A\|$ denotes any induced norm of matrix $A \in \mathbb{C}^{p\times q}$. Further, we will denote by $P_n[x]$ the set of matrix polynomials of degree n for the real variable x. If a matrix function $g : [a,b] \to \mathbb{C}^{r\times q}$ is k-times differentiable, and its kth derivative is continuous in $[a,b]$, we will say that it is of class $k \geq 0$. We will represent it as $g \in \mathscr{C}^k([a,b])$. For the interval $[a,b] \subset \mathbb{R}$ consider the partition

$$\Delta = \{a = x_0 \ < \ x_1 \ < \ \ldots \ < \ x_n = b\}\,.$$

Given an integer $m \geq 0$, we proceed to define the set of matrix splines of order m and class $\mathscr{C}^n([a,b])$ as

$$M\_\mathbb{C}^{r\times r}(\Delta)_1^m = \left\{ Q : [a,b] \longrightarrow \mathbb{C}^{r\times q};\ \left\{ \begin{array}{l} Q\big|_{[x_{i-1},x_i]}(x) \in P_m[x], i \in \{1,\ldots,n\}\,, \\ \\ Q \in \mathscr{C}^n([a,b]) \end{array} \right\} \right.\,.$$

For $m = 3, n = 2$ these matrix splines are called *matrix cubic splines* [22].

2 Description of the method

Let us consider the following first-order matrix problem

$$\left.\begin{array}{l} Y'(x) = A(x)Y(x) + B(x) \\ Y(a) = Y_a \end{array}\right\}\ a \leq x \leq b, \tag{3.2.2}$$

where the unknown matrix is $Y(x) \in \mathbb{R}^{r\times q}$, and $Y_a \in \mathbb{R}^{r\times q}$ is constant. The matrix coefficients depend on the parameter $x \in [a,b]$ such that $A : [a,b] \to \mathbb{R}^{r\times r}$, $B : [a,b] \to \mathbb{R}^{r\times q}$. The condition $A, B \in \mathscr{C}^s([a,b])$, $s \geq 1$, guarantees the uniqueness of solution $Y(x)$ of problem (3.1.1), which is continuously differentiable [9, p. 99].

The partition of the interval $[a,b]$ shall be given by

$$\Delta_{[a,b]} = \{a = x_0 < x_1 < \ldots < x_n = b\}\,,\ x_k = a + kh,\ k = 0,1,\ldots,n, \tag{3.2.3}$$

where n is a positive integer with step size $h = (b-a)/n$. For each subinterval $[a+kh, a+(k+1)h]$ we will construct a matrix spline $S(x)$ of order $m \in \mathbb{N}$ with $1 \leq m \leq s$, where s is the order of differentiability. Then, the solution for problem (3.2.2) can be approximated by the matrix spline $S(x) \in \mathscr{C}^1([a,b])$.

In the first interval $[a, a+h]$, we define the matrix spline as

$$S_{|_{[a,a+h]}}(x) = \sum_{j=0}^{m-1} \frac{1}{j!} Y^{(j)}(a)(x-a)^j + \frac{1}{m!}\alpha_0 (x-\alpha)^m, \tag{3.2.4}$$

where $\alpha_0 \in \mathbb{R}^{r\times q}$ is a matrix parameter to be determined. It is straightforward to check

$$S_{|_{[a,a+h]}}(a) = Y(a), \quad S'_{|_{[a,a+h]}}(a) = Y'(a) = A(a)Y(a) + B(a),$$

and therefore the spline satisfies the differential equation Eq. (3.2.2) at $x = a$.

In order to determine the matrix spline (3.2.4), we still must obtain the values for the matrices $Y''(a), Y^{(3)}(a), \ldots, Y^{(m-1)}(a)$, and A_0. For the second-order derivative $Y''(x)$, we proceed to compute

$$\begin{aligned} Y''(x) &= A'(x)Y(x) + A(x)Y'(x) + B'(x) \\ &= g_1(x, Y(x))\,, \end{aligned} \tag{3.2.5}$$

where $g_1 \in \mathscr{C}^{s-1}([a,b])$. Using (3.2.5), we now can evaluate $Y''(a) = g_1(a, Y(a))$.

For the third derivative, one continues in a similar manner:

$$\begin{aligned} Y^{(3)}(x) &= A''(x)Y(x) + 2A'(x)Y'(x) + A(x)Y''(x) + B''(x) \\ &= g_2(x, Y(x)) \in \mathscr{C}^{s-2}([a,b])\,, \end{aligned} \tag{3.2.6}$$

and one calculates $Y^{(3)}(a) = g_2(a, Y(a))$ using (3.2.6). For the next higher-order derivatives $Y^{(4)}(x), \ldots, Y^{(m-1)}(x)$, we proceed similarly and finds

$$\left.\begin{aligned} Y^{(4)}(x) &= g_3(x, Y(x)) \in \mathscr{C}^{s-3}([a,b]) \\ &\vdots \\ Y^{(m-1)}(x) &= g_{m-2}(x, Y(x)) \in \mathscr{C}^{s-(m-2)}([a,b]) \end{aligned}\right\}. \tag{3.2.7}$$

Note that it is fairly easy to create a table summarizing all such derivatives by using automatized programs on standard computer algebra systems. Substituting $x = a$ in (3.2.7), one obtains $Y^{(4)}(a), \ldots, Y^{(m-1)}(a)$. All matrix parameters of the spline which were to be determined are now known, except for α_0. To determine α_0, we suppose that (3.2.4) is a solution of problem (3.2.2) at $x = a + h$, which gives

$$S'_{|_{[a,a+h]}}(a+h) = A(a+h)S_{|_{[a,a+h]}}(a+h) + B(a+h). \tag{3.2.8}$$

Next, we obtain from (3.2.8) the matrix equation with only one unknown α_0:

$$\left(I - \frac{h}{m}A(a+h)\right)\alpha_0 = \tag{3.2.9}$$

$$\frac{(m-1)!}{h^{m-1}}\left(A(a+h)\sum_{j=0}^{m-1}\frac{h^j}{j!}Y^{(j)}(a) - \sum_{j=0}^{m-2}\frac{h^j}{j!}Y^{(j+1)}(a) + B(a+h)\right)$$

Assuming uniqueness of the solution α_0 given by the matrix equation (3.2.9), the matrix spline introduced in Eq. (3.2.4) is then totally determined in the interval $[a, a+h]$.

In the subsequent interval $[a+h, a+2h]$, the matrix spline takes the form

$$S_{\big|_{[a+h,a+2h]}}(x) = \tag{3.2.10}$$
$$S_{\big|_{[a,a+h]}}(a+h) + \sum_{j=1}^{m-1}\frac{1}{j!}\overline{Y^{(j)}(a+h)}(x-(a+h))^j + \frac{1}{m!}\alpha_1(x-(a+h))^m,$$

where

$$\overline{Y'(a+h)} = A(a+h)S_{\big|_{[a,a+h]}}(a+h) + B(a+h). \tag{3.2.11}$$

The expressions

$$\overline{Y''(a+h)}, \ldots, \overline{Y^{(m-1)}(a+h)}$$

are similar to the previous results, obtained after evaluating the respective derivatives of $Y(x)$ using $S_{\big|_{[a,a+h]}}(a+h)$ in (3.2.5)–(3.2.7). In more compact form, we may write

$$\begin{aligned}\overline{Y''(a+h)} &= g_1\left(a+h, S_{\big|_{[a,a+h]}}(a+h)\right),\\ &\vdots\\ \overline{Y^{(m-1)}(a+h)} &= g_{m-2}\left(a+h, S_{\big|_{[a,a+h]}}(a+h)\right).\end{aligned} \tag{3.2.12}$$

Note: the spline $S(x)$ defined by (3.2.4) and (3.2.10) is of differentiability class $\mathscr{C}^1([a, a+2h])$, contrary to the splines introduced by Loscalzo and Talbot [17], which were of differentiability class $\mathscr{C}^{m-1}([a, a+2h])$. In Ref. [20], our approach to obtain the coefficients of the approximation $\overline{Y^{(k)}(a+h)}(x-(a+h))$, for $k > 2$ was based on the derivatives for each spline in the previous interval. Now our approach to obtain an estimate for theses coefficients consists in employing the functions defined in Eq. (3.2.12).

By construction, the spline (3.2.10) satisfies the differential equation (3.2.2) at $x = a+h$. All of its coefficients are determined with the exception of $\alpha_1 \in \mathbb{R}^{r\times q}$. To obtain the value of α_1 we only require the spline (3.2.10) to be a unique solution of (3.2.2) at point $x = a+2h$:

$$S'_{\big|_{[a+h,a+2h]}}(a+2h) = A(a+2h)S_{\big|_{[a+h,a+2h]}}(a+2h) + B(a+2h).$$

An expansion yields the matrix equation with the only unknown A_1:

$$\left(I - \frac{h}{m}A(a+2h)\right)\alpha_1 = \tag{3.2.13}$$

$$\frac{(m-1)!}{h^{m-1}}\left(A(a+2h)\left(S_{|_{[a,a+h]}}(a+h) + \sum_{j=1}^{m-1}\frac{h^j}{j!}\overline{Y^{(j)}(a+h)}\right)\right.$$

$$\left. - \sum_{j=0}^{m-2}\frac{h^j}{j!}\overline{Y^{(j+1)}(a+h)} + B(a+h)\right).$$

Let us assume again that the matrix equation (3.2.13) has only one solution α_1. This way the spline is totally determined in the interval $[a+h, a+2h]$.

Iterating this process, we proceed to construct the matrix spline consecutively up to the last subinterval $[a+(n-1)h, b]$. For example, the general subinterval $[a+kh, a+(k+1)h]$ will contain the matrix spline

$$S_{|_{[a+kh,a+(k+1)h]}}(x) = \tag{3.2.14}$$

$$S_{|_{[a+(k-1)h,a+kh]}}(a+kh) + \sum_{j=1}^{m-1}\frac{1}{j!}\overline{Y^{(j)}(a+kh)}(x-(a+kh))^j$$

$$+ \frac{1}{m!}\alpha_k(x-(a+kh))^m,$$

where

$$\overline{Y'(a+kh)} = A(a+kh)S_{|_{[a+(k-1)h,a+kh]}}(a+kh) + B(a+kh). \tag{3.2.15}$$

In a similar manner as before, one abbreviates

$$\begin{aligned}\overline{Y''(a+kh)} &= g_1\left(a+kh, S_{|_{[a+(k-1)h,a+kh]}}(a+kh)\right),\\ &\vdots\\ \overline{Y^{(m-1)}(a+kh)} &= g_{m-2}\left(a+kh, S_{|_{[a+(k-1)h,a+kh]}}(a+kh)\right).\end{aligned} \tag{3.2.16}$$

With this definition, the matrix spline $S(x) \in \mathcal{C}^1\left(\bigcup_{j=0}^{k}[a+jh, a+(j+1)h]\right)$ fulfills the differential equation (3.2.2) at point $x = a+kh$. Recall that Eq. (3.2.16) was necessary to obtain the spline coefficients $\overline{Y^{(k)}}$ by using the known derivatives of the solution of the previous spline. Now we assume that $S_{|_{[a+kh,a+(k+1)h]}}(x)$ satisfies (3.2.2) at point $x = a+(k+1)h$, *i.e.*

$$S'_{|_{[a+kh,a+(k+1)h]}}\big(a+(k+1)\big) =$$

$$A\big(a+(k+1)\big)S_{|_{[a+kh,a+(k+1)h]}}\big(a+(k+1)\big) + B\big(a+(k+1)h\big).$$

Expanding this expression yields

$$\left(I - \frac{h}{m}A(a+(k+1)h)\right)\alpha_k = \tag{3.2.17}$$
$$\frac{(m-1)!}{h^{m-1}}\left[A(a+(k+1)h)\left(S_{|_{[a+(k-1)h,a+kh]}}(a+kh) + \sum_{j=1}^{m-1}\frac{h^j}{j!}\overline{Y^{(j)}(a+kh)}\right)\right.$$
$$\left. - \sum_{j=0}^{m-2}\frac{h^j}{j!}\overline{Y^{(j+1)}(a+kh)} + B(a+(k+1)h)\right].$$

Observe that the final result (3.2.17) relates directly to equations (3.2.9) and (3.2.13), when setting $k = 0$ and $k = 1$. Note also that solubility of equation (3.2.17) is guaranteed by showing that the matrix $\left(I - \frac{h}{m}A(a+(k+1)h)\right)$ is invertible, for $k = 0, 1, \ldots, n-1$. To see this, let us denote

$$M = \max\{\|A(x)\|\,;\ a \le x \le b\}\,, \tag{3.2.18}$$

where any induced norm applies. Then, one obtains

$$\left\| I - \left(I - \frac{h}{m}A(a+(k+1)h)\right)\right\| = \frac{h}{m}\|A(a+(k+1)h)\| \le \frac{h}{m}M. \tag{3.2.19}$$

If we take $h \le m/M$, according to Lemma 2.3.3 in [23], it follows that matrix $I - (h/m)A(a+(k+1)h)$ is invertible, and therefore equation (3.2.17) has a unique solution α_k, for each $k = 0, 1, \ldots, n-1$. In summary, we have proved the following theorem:

Theorem 3.1. *For the first-order matrix differential equation (3.2.2), assume that $A, B \in \mathscr{C}^s([a,b])$, $s \ge 1$. Let $h > 0$ so that $h \le m/M$, where M is given by (3.2.18) and $0 < m \le s+1$. We also consider the partition (3.2.3) with step size $h < m/L$. Then, a matrix spline $S(x)$ of order $m \in \mathbb{N}$ and differentiability class $\mathscr{C}^1[a,b]$ exists for each subinterval $[a+kh, a+(k+1)h]$, $k = 0, 1, \ldots, n-1$, following the method of construction detailed before.*

It is important to observe that these splines have a local error of $O(h^m)$. This is a consequence of an analysis similar to Loscalzo and Talbot's work [17].

The approximate solution of (3.2.2) can be computed by means of matrix splines of order m in the interval $[a,b]$ with a local error of the order $O(h^m)$ under the conditions of Theorem 3.1. The procedure is as follows:

- Compute the functions $g_1(x, Y(x)), \ldots, g_{m-2}(x, Y(x))$ given by Eqs. (3.2.5)–(3.2.7) to determine constants $Y''(a)$, ..., $Y^{(m-1)}(a)$. Compute constant M of Eq. (3.2.18). Choose $n > M(b-a)/m$ so that $h = (b-a)/n$, which produces the partition $\Delta_{[a,b]}$ defined by Eq. (3.2.3).

- Solve equation Eq. (3.2.9) to find α_0, and determine $S_{|_{[a,a+h]}}(x)$ of Eq. (3.2.4).

- Solve Eq. (3.2.17) iteratively for $k = 1, \ldots, n-1$ to determine all α_k. Then compute splines $S_{|_{[a+kh,a+(k+1)h]}}(x)$ according to Eq. (3.2.14).

3 Examples

In this section, we test our MATLAB implementations for the proposed spline method with problems where the exact solution is known, using the same examples as in Ref. [20]. All tests

have been carried out on an *Intel Core 2 Duo T5600* with 2 GB main memory, using MATLAB version 7.9. For our programs we have developed symbolic as well as numerical algorithms. The symbolic algorithm uses the *Symbolic Math Toolbox* of MATLAB for computing the derivatives of matrices A and B and for solving the implicit equations (3.2.17). In the numerical algorithm, the derivatives are provided by a function that calculates the derivatives of the matrices A and B for any value of x. The newly implemented algorithms based on our method have been compared with the results produced by the corresponding MATLAB functions solving ordinary differential equations (see Table 3.1). The values of *RelTol* and *AbsTol* for these functions have been chosen such to obtain the maximum precision with minimum execution time. These values are $RelTol = 2.22045 \cdot 10^{-14}$ and $AbsTol = 1.0 \cdot 10^{-14}$.

SOLVER	PROBLEM	METHOD
ode45	non-stiff differential equations	Runge-Kutta
ode23	non-stiff differential equations	Runge-Kutta
ode113	non-stiff differential equations	Adams
ode15s	stiff differential equations	NDFs (BDFs)
ode23s	stiff differential equations	Rosenbrock
ode23t	moderately stiff differential equations	Trapezoidal rule
ode23tb	stiff differential equations	TR-BDF2

Table 3.1. MATLAB solvers used in the tests.

3.1 Example 1

Let us consider the problem

$$\left.\begin{aligned} Y'(x) &= \frac{1}{x^3 - x - 1} \begin{pmatrix} 2x^2 - 1 & x^2 - 2x - 1 \\ -x - 1 & x^3 + x^2 - x - 1 \end{pmatrix} Y(x), \quad 0 \le x \le 1, \\ Y(0) &= \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad Y(x) \in \mathbb{C}^2. \end{aligned}\right\} \tag{3.3.20}$$

This problem has the exact solution $Y(x) = \begin{pmatrix} e^x \\ x\,e^x \end{pmatrix}$, so that we will be able to calculate the approximation error. Since $\max_{x \in [0,1]} \|A(x)\| \le 3$, we take $m = 3$ and choose $h \le m/3$. Conventional matrix cubic splines ($m = 3$), as introduced in Ref. [20], produced the absolute errors listed in Table 3.2(a). The values in the error column correspond to the maximum of the 2-norm for each subinterval.

What happens if we increase the order of the splines using the same technique as in Ref. [20]? For fourth-order splines ($m = 4$), we obtain the results given in Table 3.2(b). If we further increase the order of the splines, the result is worsening, as shown in part (c) of Table 3.2 for spline order $m = 5$.

For the same problem, we now use the new algorithm proposed in this work with fourth-order splines ($m = 4$). The results, obtained with *Mathematica* version 7.0, are shown in Table 3.3 together with their corresponding absolute errors. We also present the results for fifth-order splines ($m = 5$) in Table 3.4. Figures 3.1 and 3.2 depict the approximation behavior for splines of fourth-order and fifth-order with different step sizes $h = 0.01$ and $h = 0.001$, respectively. Tables 3.5 and 3.7 present the results of the proposed method, with $h = 0.1$ and $h = 0.01$,

$[x_i, x_{i+1}]$	ERRORS
$[0, 0.1]$	6.33721×10^{-6}
$[0.1, 0.2]$	6.05558×10^{-6}
$[0.2, 0.3]$	8.14626×10^{-6}
$[0.3, 0.4]$	7.81749×10^{-6}
$[0.4, 0.5]$	11.5296×10^{-6}
$[0.5, 0.6]$	11.6396×10^{-6}
$[0.6, 0.7]$	16.357×10^{-6}
$[0.7, 0.8]$	17.359×10^{-6}
$[0.8, 0.9]$	23.29×10^{-6}
$[0.9, 1]$	24.6909×10^{-6}

(a)

$[x_i, x_{i+1}]$	ERRORS
$[0, 0.1]$	1.14628×10^{-7}
$[0.1, 0.2]$	8.81776×10^{-7}
$[0.2, 0.3]$	2.2721×10^{-6}
$[0.3, 0.4]$	9.75288×10^{-6}
$[0.4, 0.5]$	0.000033
$[0.5, 0.6]$	0.00012
$[0.6, 0.7]$	0.00045
$[0.7, 0.8]$	0.0016
$[0.8, 0.9]$	0.0060
$[0.9, 1]$	0.022

(b)

$[x_i, x_{i+1}]$	ERRORS
$[0, 0.1]$	1.7956×10^{-9}
$[0.1, 0.2]$	5.7101×10^{-8}
$[0.2, 0.3]$	5.46782×10^{-7}
$[0.3, 0.4]$	5.32517×10^{-6}
$[0.4, 0.5]$	0.000051
$[0.5, 0.6]$	0.00049
$[0.6, 0.7]$	0.0048
$[0.7, 0.8]$	0.047
$[0.8, 0.9]$	0.45
$[0.9, 1]$	4.50

(c)

Table 3.2. Absolute errors using the matrix splines of order (a) $m = 3$, (b) $m = 4$ and (c) $m = 5$ with the method given in [20] with $n = 10$ and $h = 0.1$, for Example 3.1.

compared to the results produced by MATLAB functions. The second column indicates the execution time in seconds and the third column the relative errors at $x = 10$.

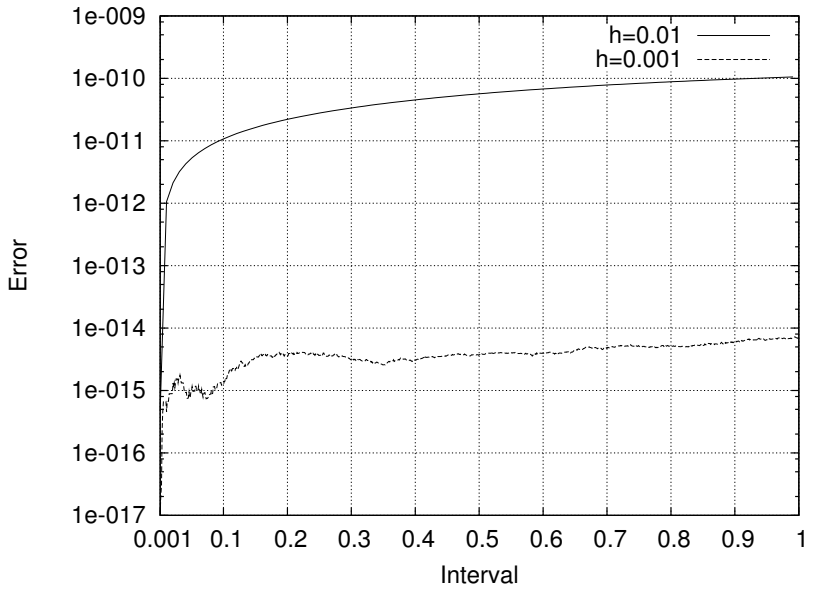

Fig. 3.1. Relative errors for the test problem (3.3.20) with fourth-order splines ($m = 4$) using our proposed method with $h = 0.01$ and $h = 0.001$, respectively.

3.2 Example 2

Consider the matrix problem

$$\left.\begin{aligned} Y'(x) &= A(x)\,Y(x) + B(x) \\ Y(0) &= \begin{pmatrix} 3 & 0 \\ 1 & 1 \end{pmatrix}, \quad x \in [0, 1] \end{aligned}\right\} \tag{3.3.21}$$

where

$[x_i, x_{i+1}]$	APPROXIMATION	ERRORS
$[0, 0.1]$	$\begin{pmatrix} 1 + x + 0.5x^2 + 0.1667x^3 + 0.0428x^4 \\ x + x^2 + 0.5x^3 + 0.1720x^4 \end{pmatrix}$	1.14×10^{-7}
$[0.1, 0.2]$	$\begin{pmatrix} 1 + 0.9991x + 0.5002x^2 + 0.1653x^3 + 0.0473x^4 \\ 0.99995x + 1.0008x^2 + 0.4931x^3 + 0.1949x^4 \end{pmatrix}$	2.62×10^{-7}
$[0.2, 0.3]$	$\begin{pmatrix} 1.0000 + 0.9999x + 0.5011x^2 + 0.1618x^3 + 0.0522x^4 \\ 0.9994x + 1.0056x^2 + 0.4750x^3 + 0.2206x^4 \end{pmatrix}$	4.51×10^{-7}
$[0.3, 0.4]$	$\begin{pmatrix} 1.0000 + 0.9994x + 0.5036x^2 + 0.1557x^3 + 0.0577x^4 \\ 0.0002 + 0.9969x + 1.0189x^2 + 0.4430x^3 + 0.24953x^4 \end{pmatrix}$	6.89×10^{-7}
$[0.4, 0.5]$	$\begin{pmatrix} 1.0002 + 0.9981x + 0.5088x^2 + 0.1465x^3 + 0.0638x^4 \\ 0.0009 + 0.98995x + 1.0466x^2 + 0.3939x^3 + 0.2821x^4 \end{pmatrix}$	9.89×10^{-7}
$[0.5, 0.6]$	$\begin{pmatrix} 1.0005 + 0.9952x + 0.5180x^2 + 0.1338x^3 + 0.0705x^4 \\ 0.0028 + 0.9741x + 1.0966x^2 + 0.3240x^3 + 0.3189x^4 \end{pmatrix}$	1.36×10^{-6}
$[0.6, 0.7]$	$\begin{pmatrix} 1.0013 + 0.9895x + 0.5328x^2 + 0.1166x^3 + 0.07794x^4 \\ 0.0073 + 0.9424x + 1.1788x^2 + 0.2289x^3 + 0.3602x^4 \end{pmatrix}$	1.82×10^{-6}
$[0.7, 0.8]$	$\begin{pmatrix} 1.0031 + 0.9793x + 0.5553x^2 + 0.0944x^3 + 0.0861x^4 \\ 0.0171 + 0.8849x + 1.3063x^2 + 0.1032x^3 + 0.4067x^4 \end{pmatrix}$	2.37×10^{-6}
$[0.8, 0.9]$	$\begin{pmatrix} 1.0064 + 0.9623x + 0.5882x^2 + 0.0663x^3 + 0.0952x^4 \\ 0.0360 + 0.7871x + 1.4952x^2 - 0.0590x^3 + 0.4589x^4 \end{pmatrix}$	3.05×10^{-6}
$[0.9, 1]$	$\begin{pmatrix} 1.0123 + 0.9352x + 0.6344x^2 + 0.0311x^3 + 0.1052x^4 \\ 0.0707 + 0.6291x + 1.7657x^2 - 0.2649x^3 + 0.5177x^4 \end{pmatrix}$	3.86×10^{-6}

Table 3.3. Absolute errors using the spline algorithm for problem (3.3.20) with fourth-order splines ($m = 4$).

$$A(x) = \begin{pmatrix} 1 & -1 \\ 1 & e^x \end{pmatrix}, \; B(x) = \begin{pmatrix} -3e^{-x} - 1 & 2 - 2e^{-x} \\ -3e^{-x} - 2 & 1 - 2\cosh(x) \end{pmatrix},$$

which has the exact solution

$$Y(x) = \begin{pmatrix} 2e^{-x} + 1 & e^{-x} - 1 \\ e^{-x} & 1 \end{pmatrix}.$$

which asymptotically converges to

$$\lim_{x \to \infty} Y(x) = \begin{pmatrix} 1 & -1 \\ 0 & 1 \end{pmatrix}.$$

Since $\max_{x \in [0,1]} \|A(x)\| \leq 6$, we take $M = 6$ and choose $h \leq M/6$. Using conventional matrix cubic splines [20], we obtain the errors given in Table 3.6(a). The absolute errors are calculated as in Example 3.1.

$[x_i, x_{i+1}]$	APPROXIMATION	ERRORS
$[0, 0.1]$	$\begin{pmatrix} 1 + x + 0.5x^2 + 0.1667x^3 + 0.0417x^4 + 0.0085x^5 \\ x + x^2 + 0.5x^3 + 0.1667x^4 + 0.0427x^5 \end{pmatrix}$	1.80×10^{-9}
$[0.1, 0.2]$	$\begin{pmatrix} 1 + x + 0.4996x^2 + 0.1667x^3 + 0.0413x^4 + 0.0094x^5 \\ x + 0.99997x^2 + 0.5003x^3 + 0.1647x^4 + 0.0481x^5 \end{pmatrix}$	4.09×10^{-9}
$[0.2, 0.3]$	$\begin{pmatrix} 1 + x + 0.4999x^2 + 0.1670x^3 + 0.0405x^4 + 0.0104x^5 \\ 1.0000x + 0.9997x^2 + 0.5021x^3 + 0.1595x^4 + 0.0542x^5 \end{pmatrix}$	7.00×10^{-9}
$[0.3, 0.4]$	$\begin{pmatrix} 0.99998 + 1.0000x + 0.4997x^2 + 0.1678x^3 + 0.0390x^4 + 0.0115x^5 \\ 1.0002x + 0.9983x^2 + 0.5072x^3 + 0.1502x^4 + 0.0611x^5 \end{pmatrix}$	1.07×10^{-8}
$[0.4, 0.5]$	$\begin{pmatrix} 0.99998 + 1.0002x + 0.4991x^2 + 0.1695x^3 + 0.0368x^4 + 0.0127x^5 \\ -0.0001 + 1.0010x + 0.9943x^2 + 0.5178x^3 + 0.1360x^4 + 0.0688x^5 \end{pmatrix}$	1.53×10^{-8}
$[0.5, 0.6]$	$\begin{pmatrix} 0.99996 + 1.0005x + 0.4977x^2 + 0.1725x^3 + 0.0336x^4 + 0.0140x^5 \\ -0.0003 + 1.0031x + 0.9852x^2 + 0.5370x^3 + 0.1157x^4 + 0.0774x^5 \end{pmatrix}$	2.10×10^{-8}
$[0.6, 0.7]$	$\begin{pmatrix} 0.9999 + 1.0013x + 0.4949x^2 + 0.1773x^3 + 0.0294x^4 + 0.0155x^5 \\ -0.0009 + 1.0083x + 0.9671x^2 + 0.5686x^3 + 0.0880x^4 + 0.0871x^5 \end{pmatrix}$	2.80×10^{-8}
$[0.7, 0.8]$	$\begin{pmatrix} 0.9996 + 1.0030x + 0.4900x^2 + 0.1846x^3 + 0.0239x^4 + 0.0171x^5 \\ -0.0024 + 1.0194x + 0.9342x^2 + 0.6176x^3 + 0.0515x^4 + 0.0980x^5 \end{pmatrix}$	3.65×10^{-8}
$[0.8, 0.9]$	$\begin{pmatrix} 0.9991 + 1.0062x + 0.4817x^2 + 0.1954x^3 + 0.0170x^4 + 0.0189x^5 \\ -0.0057 + 1.0410x + 0.8782x^2 + 0.6901x^3 + 0.0045x^4 + 0.1101x^5 \end{pmatrix}$	4.67×10^{-8}
$[0.9, 1]$	$\begin{pmatrix} 0.9981 + 1.0119x + 0.4685x^2 + 0.2105x^3 + 0.0083x^4 + 0.0209x^5 \\ -0.0126 + 1.0805x + 0.7877x^2 + 0.7939x^3 - 0.0550x^4 + 0.1238x^5 \end{pmatrix}$	5.90×10^{-8}

Table 3.4. Absolute errors using the spline algorithm for problem (3.3.20) with fifth-order splines ($m = 5$).

What happens if we increase the order of the splines using the same technique as in Ref. [20]? For fourth-order splines, we obtain the result given in Table 3.6(b). If we increase the order of the spline, the quality of the approximation gets worse, which is shown in Table 3.6(c) for splines of order $m = 5$.

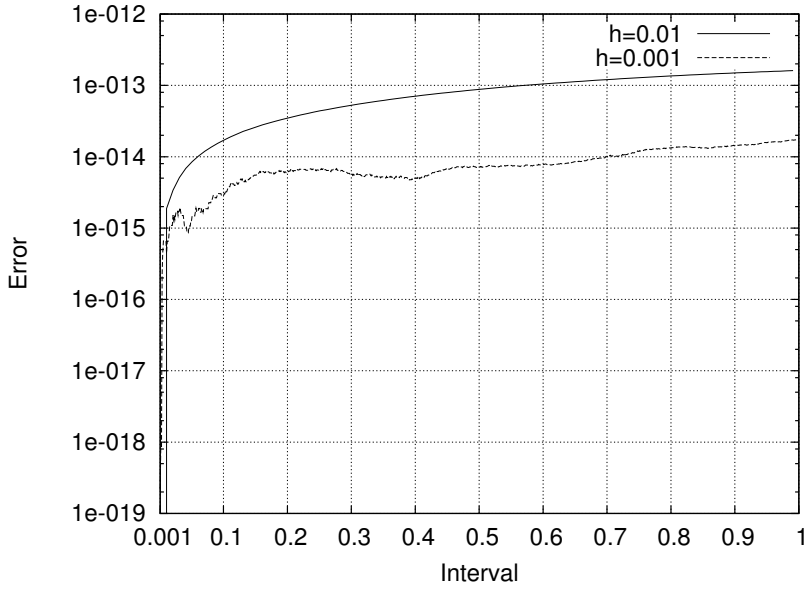

Fig. 3.2. Approximation error for problem (3.3.20) with fifth-order splines ($m = 5$) using our proposed method with $h = 0.01$ and $h = 0.001$, respectively.

Method	Time [s]	Error
Spline of order $m = 4$	0.006679	$6.825762e-008$
Spline of order $m = 5$	0.008287	$8.749450e-010$
Spline of order $m = 6$	0.011020	$1.015738e-011$
ode45	0.428922	$3.438694e-007$
ode23	15.370957	$4.448549e-006$
ode113	0.033385	$6.488040e-013$
ode15s	0.548005	$1.041483e-011$
ode23s	70.147383	$1.737104e-001$
ode23t	64.297144	$1.446478e-009$
ode23tb	291.413931	$2.441626e-007$

Table 3.5. Relative errors for the test problem (3.3.20) with splines ($h = 0.1$) and MATLAB solvers.

$[x_i, x_{i+1}]$	ERRORS
$[0, 0.1]$	3.3824×10^{-6}
$[0.1, 0.2]$	3.3824×10^{-6}
$[0.2, 0.3]$	3.3704×10^{-6}
$[0.3, 0.4]$	3.3704×10^{-6}
$[0.4, 0.5]$	3.4512×10^{-6}
$[0.5, 0.6]$	3.4512×10^{-6}
$[0.6, 0.7]$	3.8211×10^{-6}
$[0.7, 0.8]$	3.8211×10^{-6}
$[0.8, 0.9]$	4.9777×10^{-6}
$[0.9, 1]$	6.3207×10^{-6}

(a)

$[x_i, x_{i+1}]$	ERRORS
$[0, 0.1]$	5.0639×10^{-8}
$[0.1, 0.2]$	3.9495×10^{-7}
$[0.2, 0.3]$	1.0951×10^{-6}
$[0.3, 0.4]$	4.4842×10^{-6}
$[0.4, 0.5]$	0.000015
$[0.5, 0.6]$	0.000057
$[0.6, 0.7]$	0.00021
$[0.7, 0.8]$	0.00076
$[0.8, 0.9]$	0.0028
$[0.9, 1]$	0.01

(b)

$[x_i, x_{i+1}]$	ERRORS
$[0, 0.1]$	6.7494×10^{-10}
$[0.1, 0.2]$	2.1233×10^{-8}
$[0.2, 0.3]$	2.0815×10^{-7}
$[0.3, 0.4]$	2.0325×10^{-6}
$[0.4, 0.5]$	0.00002
$[0.5, 0.6]$	0.00019
$[0.6, 0.7]$	0.0018
$[0.7, 0.8]$	0.018
$[0.8, 0.9]$	0.17
$[0.9, 1]$	1.68

(c)

Table 3.6. Absolute errors for Example 3.2 using the matrix splines of order (a) $m = 3$, (b) $m = 4$ and (c) $m = 5$ with the method given in [20] with $n = 10$ and $h = 0.1$.

Method	Time [s]	Error
Spline of order $m = 4$	0.055846	$9.994253e-013$
Spline of order $m = 5$	0.075211	$1.944154e-013$
Spline of order $m = 6$	0.097185	$1.848712e-013$
ode45	0.428922	$3.438694e-007$
ode23	15.370957	$4.448549e-006$
ode113	0.033385	$6.488040e-013$
ode15s	0.548005	$1.041483e-011$
ode23s	70.147383	$1.737104e-001$
ode23t	64.297144	$1.446478e-009$
ode23tb	291.413931	$2.441626e-007$

Table 3.7. Relative errors for the test problem (3.3.20) with splines ($h = 0.01$) and MATLAB solvers.

We now use the algorithm proposed in this work for the same problem using fourth-order splines. The results, obtained with *Mathematica 7*, are shown in Table 3.8(a) with absolute errors. Similarly, the results using fifth-order splines are shown in Table 3.8(b).

Figures 3.3 and 3.4 illustrate the approximation behavior for splines of fourth and fifth order for step sizes $h = 0.01$ and $h = 0.001$, respectively. In Figure 3.4 we observe that $h = 0.001$ yields an accuracy very close to machine precision. Table 3.9 presents the results of the proposed method in the interval $[0, 3]$ with step size $h = 0.01$ compared with the results produced by the MATLAB functions. The second column indicates the execution time in seconds and the third column the relative errors at $x = 3$. For the evaluation using MATLAB functions it was necessary to vectorize problem (3.3.21). For $x \geq 3$ all solvers and splines presented convergence problems.

4 Conclusions

This chapter proposes a method for the numerical integration of first-order matrix linear differential equations of the type $Y'(x) = A(x)\,Y(x) + B(x)$, $x \in [a, b]$, using higher-order matrix splines. Contrary to existing spline methods in the literature, this new algorithm provides continuous spline approximations of the global order $O(h^{m-1})$ by requiring only first-order derivatives–a significant advantage over existing approaches. Additionally, our method is well-suited for implementation on numerical and/or symbolical computer systems.

For an explicit demonstration of our proposed method and its advantages over existing conventional methods, we discussed two numerical test (the same examples as chosen in [20]) with excellent results and considerable improvements compared to the different methods implemented in MATLAB. Our approach excels not only in speed but also in accuracy. One has to take into account that MATLAB's solvers strongly rely on adaptive algorithms, which still have to be included in the method we propose and will certainly lead to further performance boosts.

In future works, we hope to develop a complete analysis of the stability for B-splines following the scheme outlined in Ref. [16]. A closer focus on the behavior of stiff problems should also be interesting.

$[x_i, x_{i+1}]$	ERRORS
$[0, 0.1]$	5.0639×10^{-8}
$[0.1, 0.2]$	1.01878×10^{-7}
$[0.2, 0.3]$	1.5456×10^{-7}
$[0.3, 0.4]$	2.0995×10^{-7}
$[0.4, 0.5]$	2.7002×10^{-7}
$[0.5, 0.6]$	3.3797×10^{-7}
$[0.6, 0.7]$	4.1898×10^{-7}
$[0.7, 0.8]$	5.2140×10^{-7}
$[0.8, 0.9]$	6.5853×10^{-7}
$[0.9, 1]$	8.5131×10^{-7}

(a)

$[x_i, x_{i+1}]$	ERRORS
$[0, 0.1]$	6.7494×10^{-10}
$[0.1, 0.2]$	1.3578×10^{-9}
$[0.2, 0.3]$	2.0596×10^{-9}
$[0.3, 0.4]$	2.7970×10^{-9}
$[0.4, 0.5]$	3.5963×10^{-9}
$[0.5, 0.6]$	4.4994×10^{-9}
$[0.6, 0.7]$	5.5749×10^{-9}
$[0.7, 0.8]$	6.9335×10^{-9}
$[0.8, 0.9]$	8.7516×10^{-9}
$[0.9, 1]$	1.1307×10^{-8}

(b)

Table 3.8. Absolute errors for problem (3.3.21) using the matrix splines method of order (a) $m = 4$ and (b) $m = 5$, with $n = 10$ and $h = 0.1$.

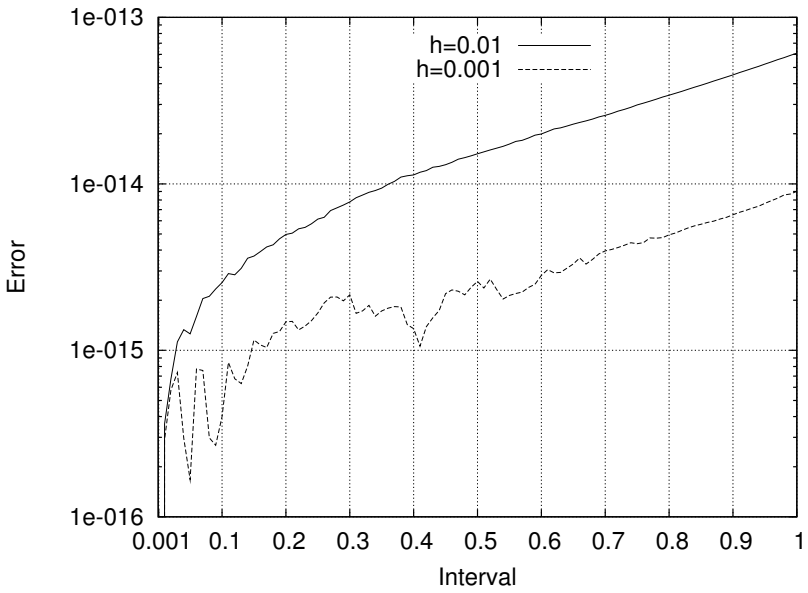

Fig. 3.3. Approximation errors for problem (3.3.21) with fourth-order splines ($m = 4$) using our proposed method with $h = 0.01$ and $h = 0.001$, respectively.

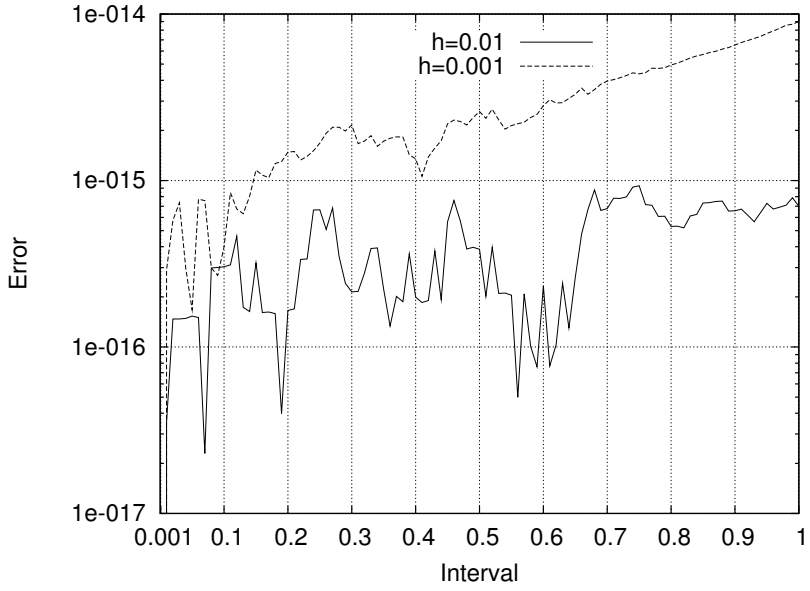

Fig. 3.4. Approximation errors for problem (3.3.21) with fifth-order splines ($m = 5$) using our proposed method with $h = 0.01$ and $h = 0.001$, respectively.

Method	Time [s]	Error
Spline of order $m = 4$	0.007799	$4.093852e-12$
Spline of order $m = 5$	0.009094	$1.539909e-14$
Spline of order $m = 6$	0.010771	$3.070086e-14$
ode45	0.122250	$6.402663e-14$
ode23	2.412271	$1.610360e-12$
ode113	0.013974	$8.550309e-14$
ode15s	0.098997	$7.226276e-12$
ode23s	34.756489	$2.618326e-09$
ode23t	7.692395	$9.432798e-10$
ode23tb	7.580115	$9.772905e-10$

Table 3.9. Approximation errors for problem (3.3.21) with splines of several orders using MATLAB solvers and taking $h = 0.02$.

References

1. S. Barnett, *Matrices in control theory*, Van Nostrand-Reinhold, New York, 1971.
2. L. D. Faddeyev, *The inverse problem in the quantum theory of scattering*, J. Math. Physics **4** (1963), no. 1, 72–104.
3. W. T. Reid, *Riccati differential equations*, Academic Press, New York, 1972.
4. M. Scott, *Invariant imbedding and its applications to ordinary differential equations*, Addison-Wesley Pub. Co., Reading, MA, 1973.
5. P. Marzulli, *Global error estimates for the standard parallel shooting method*, J. Comput. Appl. Math. **34** (1991), 233–241.
6. K. Rektorys, *The method of discretization in time and partial differential equations*, D. Reidel Pub. Co., Dordrecht, The Netherlands, 1982.
7. P. T. Boggs, *The solution of nonlinear systems of equations by α-stable integration techniques*, SIAM J. Numer. Anal. **8** (1971), no. 4, 767–785.
8. A. Graham, *Kronecker products and matrix calculus with applications*, John Wiley & Sons, New York, USA, 1981.
9. T. M. Flett, *Differential analysis*, Cambridge University Press, Cambridge, UK, 1980.
10. U.M. Ascher, R.M.M. Mattheij, and R.D. Russell, *Numerical solutions of boundary value problems for ordinary differential equations*, Prentice Hall, New Jersey, USA, 1988.

11. L. Jódar and E. Ponsoda, *Non-autonomous riccati-type matrix differential equations: Existence interval, construction of continuous numerical solutions and error bounds*, IMA J. Numer. Anal. **15** (1995), no. 1, 61–74.
12. L. Jódar, J. C. Cortés, and J. L. Morera, *Construction and computation of variable coefficient sylvester differential problems*, Computers Maths. Appl. **32** (1996), no. 8, 41–50.
13. Jódar and J. C. Cortés, *Rational matrix approximation with a priori error bounds for non-symmetric matrix Riccati equations with analytic coefficients*, IMA J. Numer. Anal. **18** (1998), no. 4, 545–561.
14. L. Jódar and E. Ponsoda, *Continuous numerical solutions and error bounds for matrix differential equations*, Int. Proc. First Int. Colloq. Num. Anal. (Utrecht, The Netherlands), VSP, 1993, pp. 73–88.
15. S. Blanes, F. Casas, J. A. Oteo, and J. Ros, *Magnus and Fer expansion for matrix differential equations: the convergence problem*, J. Phys. Appl. **31** (1998), 259–268.
16. A. Sestini F. Mazzia and D. Trigiante, *B-spline linear multistep methods and their conitinuous extensions*, SIAM J. Numer. Anal. **44** (2006), no. 5, 1954–1973.
17. F. R. Loscalzo and T. D. Talbot, *Spline function approximations for solutions of ordinary differential equations*, SIAM J. Numer. Anal. **4** (1967), no. 3, 433–445.
18. E. A. Al-Said and M. A. Noor, *Cubic splines method for a system of third-order boundary value problems*, Appl. Math. Comput. **142** (2003), 195–204.
19. G. Micula and A. Revnic, *An implicit numerical spline method for systems for ODE's*, Appl. Math. Comput. **111** (2000), 121–132.
20. E. Defez, L. Soler, A. Hervás, and C. Santamaría, *Numerical solutions of matrix differential models using cubic matrix splines*, Comput. Math. Appl. **50** (2005), 693–699.
21. E. Defez, L. Soler, A. Hervás, and M. M. Tung, *Numerical solutions of matrix differential models using cubic matrix splines II*, Math. Comput. Modelling **46** (2007), 657–669.
22. E. Defez, A. Hervás, A. Law, J. Villanueva-Oller, and R. Villanueva, *Matrix-cubic splines for progressive transmission of images*, J. Math. Imaging Vision **17** (2002), no. 1, 41–53.
23. G. H. Golub and C. F. Van Loan, *Matrix computations*, second ed., The Johns Hopkins University Press, Baltimore, MD, USA, 1989.

CHAPTER 4

Approximate solutions of first-order matrix differential equations with higher-order splines

1 Introduction

In this chapter, we propose a novel algorithm to tackle matrix differential equations of the first order. Matrix differential models are relevant for the description of many phenomena in physics and engineering, ranging from such diverse applications as control theory to game theory [1]. In particular, we will develop in this work a method for the numerical integration of first-order matrix differential equations with initial conditions. For different examples of this class of problems, we also refer to Ref. [2].

In their seminal work, Loscalzo and Talbot introduce spline function approximations for solutions of scalar differential equations [3]. These spline solutions $S(x)$ are of degree $m = 2, 3$ and continuity class $\mathscr{C}^{m-1}$. Recently, this method has been used in the resolution of other scalar problems as discussed in Ref. [4]. The corresponding generalizations to the matrix framework have been carried out in Refs. [5, 6].

Unfortunately, as detected by Loscalzo and Talbot, their scalar procedure is divergent when higher-order spline functions are used [3, p. 444–445]. They have explicitly shown by numerical computations that the equation $y' = y, y(0) = 1$ contains noticeable divergences for splines of order $m > 3$. However, our new method avoids these problems with divergences for splines $S(x)$ of order m but only require them to be of differentiability class $\mathscr{C}^1$.

Throughout this work, we will adopt the notation for norms and matrix cubic splines as in the previous work [5] and common in matrix calculus. Following this nomenclature, we define the Kronecker product of $A = (a_{ij}) \in \mathbb{C}^{m\times n}$ and $B \in \mathbb{C}^{r\times s}$, denoted by $A \otimes B$, as the block matrix

$$A \otimes B = \begin{pmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{pmatrix}.$$

The column-vector operator on a matrix $A \in \mathbb{C}^{m\times n}$ is given by

$$\mathbf{vec}(A) = \begin{pmatrix} \mathbf{A}_{\bullet 1} \\ \vdots \\ \mathbf{A}_{\bullet n} \end{pmatrix}, \text{ where } \mathbf{A}_{\bullet k} = \begin{pmatrix} a_{1k} \\ \vdots \\ a_{mk} \end{pmatrix}.$$

Here and in the following, we denote vectors and vector-valued functions by bold-face characters.

If $Y = (y_{ij}) \in \mathbb{C}^{p\times q}$ and $X = (x_{ij}) \in \mathbb{C}^{m\times n}$, then the derivative of a matrix with respect to a matrix is defined by [7, p. 62 and 81]:

$$\frac{\partial Y}{\partial X} = \begin{pmatrix} \frac{\partial Y}{\partial x_{11}} & \cdots & \frac{\partial Y}{\partial x_{1n}} \\ \vdots & & \vdots \\ \frac{\partial Y}{\partial x_{m1}} & \cdots & \frac{\partial Y}{\partial x_{mn}} \end{pmatrix}, \text{ where } \frac{\partial Y}{\partial x_{rs}} = \begin{pmatrix} \frac{\partial y_{11}}{\partial x_{rs}} & \cdots & \frac{\partial y_{1q}}{\partial x_{rs}} \\ \vdots & & \vdots \\ \frac{\partial y_{p1}}{\partial x_{rs}} & \cdots & \frac{\partial y_{pq}}{\partial x_{rs}} \end{pmatrix}.$$

If $X \in \mathbb{C}^{m\times n}, Y \in \mathbb{C}^{n\times v}, Z \in \mathbb{C}^{p\times q}$, then the following rule for the derivative of a matrix product with respect to another matrix applies [7, p. 84]:

$$\frac{\partial XY}{\partial Z} = \frac{\partial X}{\partial Z}\left[I_q \otimes Y\right] + \left[I_p \otimes X\right]\frac{\partial Y}{\partial Z}, \tag{4.1.1}$$

where I_q and I_p denote the identity matrices of dimensions q and p, respectively. If $X \in \mathbb{C}^{m\times n}, Y \in \mathbb{C}^{u\times v}, Z \in \mathbb{C}^{p\times q}$, the following chain rule [7, p. 88] is valid :

$$\frac{\partial Z}{\partial X} = \left[\frac{\partial \left[\mathbf{vec}(Y)\right]^t}{\partial X} \otimes I_p\right]\left[I_n \otimes \frac{\partial Z}{\partial \left[\mathbf{vec}(Y)\right]}\right]. \tag{4.1.2}$$

We proceed as follows. In Section 2, we give a description of the proposed method and give details of the corresponding procedure. Section 3 concludes the discussion with some numerical examples for the scalar, vector and matrix cases, respectively.

2 Description of the method

As usual, let us consider the following first-order matrix problem

$$\left.\begin{aligned} Y'(x) &= f(x, Y(x)) \\ Y(a) &= Y_a \end{aligned}\right\}, \quad a \le x \le b, \tag{4.2.3}$$

where the unknown matrix is $Y(x) \in \mathbb{R}^{r\times q}$ with initial condition $Y_a \in \mathbb{R}^{r\times q}$. The matrix-valued function $f : [a,b] \times \mathbb{R}^{r\times q} \to \mathbb{R}^{r\times q}$ is of differentiability class $f \in \mathscr{C}^s(T)$, $s \ge 1$, with

$$T = \{(x, Y);\ a \le x \le b,\ Y \in \mathbb{R}^{r\times q}\}, \tag{4.2.4}$$

and f fulfills the global Lipschitz's condition

$$\left\| f(x, Y_1) - f(x, Y_2) \right\| \le L \left\| Y_1 - Y_2 \right\|, \ a \le x \le b, Y_1, Y_2 \in \mathbb{R}^{r\times q} \tag{4.2.5}$$

to guarantee the existence and uniqueness of the continuously differentiable solution $Y(x)$ of problem (4.2.3), see Ref. [8, p. 99].

The partition of the interval $[a,b]$ shall be given by

$$\Delta_{[a,b]} = \{a = x_0 < x_1 < \ldots < x_n = b\}\,,\ x_k = a + kh,\ k = 0, 1, \ldots, n, \tag{4.2.6}$$

where n is a positive integer with the corresponding step size $h = (b-a)/n$. We will construct in each subinterval $[a + kh, a + (k+1)h]$ a matrix spline $S(x)$ of order $m \in \mathbb{N}$ with $1 \le m \le s$, where s is the order of the differentiability class of f. This will approximate the solution of problem (4.2.3) so that $S(x) \in C^1([a,b])$.

In the first interval $[a, a+h]$, we define the matrix spline as

$$\begin{aligned} S_{|_{[a,a+h]}}(x) &= Y(a) + Y'(a)(x-a) + \frac{1}{2!}Y''(a)(x-a)^2 + \frac{1}{3!}Y^{(3)}(a)(x-a)^3 \\ &\quad + \cdots + \frac{1}{(m-1)!}Y^{(m-1)}(a)(x-a)^{m-1} + \frac{1}{m!}A_0(x-a)^m, \end{aligned} \tag{4.2.7}$$

where $A_0 \in \mathbb{R}^{r\times q}$ is a matrix parameter to be determined. It is straightforward to check

$$S_{|_{[a,a+h]}}(a) = Y(a), \quad S'_{|_{[a,a+h]}}(a) = Y'(a) = f(a, Y(a)),$$

and therefore the spline satisfies the differential equation Eq. (4.2.3) at $x = a$.

We must obtain the values $Y''(a), Y^{(3)}(a), \ldots, Y^{(m-1)}(a)$, and A_0 in order to determine the matrix spline (4.2.7). To compute the second-order derivative $Y''(x)$, we follow the procedure given in Ref. [6] and use the nomenclature as already outlined in the introduction. We then obtain

$$\begin{aligned} Y''(x) &= \frac{\partial f(x, Y(x))}{\partial x} + \left[\left[\mathbf{vec}\ f(x, Y(x))\right]^T \otimes I_r\right] \frac{\partial f(x, Y(x))}{\partial \mathbf{vec}\ Y(x)} \\ &= g_1(x, Y(x))\,, \end{aligned} \tag{4.2.8}$$

where $g_1 \in \mathscr{C}^{s-1}(T)$. We are now in the position to evaluate $Y''(a) = g_1(a, Y(a))$ using (4.2.8). Similarly, we can assume that $f \in \mathscr{C}^s(T)$ for $s \ge 2$. Then, the second partial derivatives of f exist and are continuous. This yields the third derivative:

$$\begin{aligned} Y^{(3)}(x) &= \frac{\partial^2 f(x, Y(x))}{\partial x^2} + \left(\left[\mathbf{vec}\ f(x, Y(x))\right]^T \otimes I_r\right) \frac{\partial}{\partial x}\left(\frac{\partial f(x, Y(x))}{\partial \mathbf{vec}\ Y(x)}\right) \\ &+ \left(\frac{\partial \left[\mathbf{vec}\ f(x, Y(x))\right]^T}{\partial x} \otimes I_r\right) \frac{\partial f(x, Y(x))}{\partial \mathbf{vec}\ Y(x)} \\ &+ \left(\left[\mathbf{vec}\ f(x, Y(x))\right]^T \otimes I_r\right) \frac{\partial}{\partial \mathbf{vec}\ Y(x)}\left(\frac{\partial f(x, Y(x))}{\partial x}\right) \\ &+ \left(\left[\mathbf{vec}\ f(x, Y(x))\right]^T \otimes I_r\right) \left(\frac{\partial \left[\mathbf{vec}\ f(x, Y(x))\right]^T}{\partial \mathbf{vec}\ Y(x)} \otimes I_r\right) \frac{\partial f(x, Y(x))}{\partial \mathbf{vec}\ Y(x)} \\ &+ \left(\left[\mathbf{vec}\ f(x, Y(x))\right]^T \otimes I_r\right) \left(\left[\mathbf{vec}\ f(x, Y(x))\right]^T \otimes I_{r^2 q}\right) \frac{\partial^2 f(x, Y(x))}{\left(\partial \mathbf{vec}\ Y(x)\right)^2} \\ &= g_2(x, Y(x)) \in \mathscr{C}^{s-2}(T)\,. \end{aligned} \tag{4.2.9}$$

Now we can evaluate $Y^{(3)}(a) = g_2(a, Y(a))$ using (4.2.9). For all higher-order derivatives $Y^{(4)}(x), \ldots, Y^{(m-1)}(x)$ we proceed in like manner and calculate

$$\left.\begin{array}{c} Y^{(4)}(x) = g_3\left(x, Y(x)\right) \in \mathscr{C}^{s-3}\left(T\right) \\ \vdots \\ Y^{(m-1)}(x) = g_{m-2}\left(x, Y(x)\right) \in \mathscr{C}^{s-(m-2)}\left(T\right) \end{array}\right\}. \tag{4.2.10}$$

A list of all these derivatives can be easily established by employing standard computer algebra systems. Substituting $x = a$ in (4.2.10), one gets $Y^{(4)}(a)$, ..., $Y^{(m-1)}(a)$. In summary, all matrix parameters of the spline which were to be determined are known, except for A_0. To determine A_0, we suppose that (4.2.7) is a solution of problem (4.2.3) at $x = a + h$, which gives

$$S'_{\big|_{[a,a+h]}}(a+h) = f\left(a+h, S_{\big|_{[a,a+h]}}(a+h)\right). \tag{4.2.11}$$

Next, we obtain from (4.2.11) the matrix equation with only one unknown A_0:

$$A_0 = \tfrac{(m-1)!}{h^{m-1}}\left[f\left(a+h, Y(a) + Y'(a)h + \cdots + \tfrac{h^{m-1}}{(m-1)!}Y^{(m-1)}(a) + \tfrac{h^m}{m!}A_0\right)\right.$$

$$\left. - Y'(a) - Y''(a)h - \tfrac{1}{2}Y^{(3)}(a)h^2 + \cdots + \tfrac{1}{(m-2)!}Y^{(m-1)}(a)h^{m-2}\right]. \tag{4.2.12}$$

Assuming that the implicit matrix equation (4.2.12) has only one solution A_0, the matrix spline (4.2.7) is totally determined in the interval $[a, a+h]$.

In the following interval $[a+h, a+2h]$, the matrix spline takes the form

$$S_{\big|_{[a+h,a+2h]}}(x) = S_{\big|_{[a,a+h]}}(a+h) + \overline{Y'(a+h)}(x-(a+h)) +$$

$$\tfrac{1}{2!}\overline{Y''(a+h)}(x-(a+h))^2 + \cdots + \tfrac{1}{(m-1)!}\overline{Y^{(m-1)}(a+h)}(x-(a+h))^{m-1}$$

$$+\tfrac{1}{m!}A_1(x-(a+h))^m, \tag{4.2.13}$$

where

$$\overline{Y'(a+h)} = f\left(a+h, S_{\big|_{[a,a+h]}}(a+h)\right), \tag{4.2.14}$$

and $\overline{Y''(a+h)}, \ldots, \overline{Y^{(m-1)}(a+h)}$ are the similar results obtained after evaluating the respective derivatives of $Y(x)$ using $S_{\big|_{[a,a+h]}}(a+h)$ in (4.2.8)–(4.2.10). In more compact form, we may write

$$\begin{array}{rcl} \overline{Y''(a+h)} & = & g_1\left(a+h, S_{\big|_{[a,a+h]}}(a+h)\right), \\ & \vdots & \\ \overline{Y^{(m-1)}(a+h)} & = & g_{m-2}\left(a+h, S_{\big|_{[a,a+h]}}(a+h)\right). \end{array} \tag{4.2.15}$$

Observe that the matrix spline $S(x)$ defined by (4.2.7) and (4.2.13) is of differentiability class $\mathscr{C}^1\left([a, a+h] \cup [a+h, a+2h]\right)$, contrary to the splines introduced by Loscalzo and Talbot [3], which were of class $\mathscr{C}^{m-1}\left([a, a+h] \cup [a+h, a+2h]\right)$. By construction, spline (4.2.13) satisfies the differential equation (4.2.3) at $x = a + h$. and all of its coefficients are determined with the exception of $A_1 \in \mathbb{R}^{r\times q}$.

The value of A_1 can be found by taking the spline (4.2.13) as a solution of (4.2.3) at point $x = a + 2h$:

$$S'_{\big|_{[a+h,a+2h]}}(a+2h) = f\left(a+2h, S_{\big|_{[a+h,a+2h]}}(a+2h)\right).$$

An expansion yields the matrix equation with the only unknown A_1:

$$A_1 = \frac{(m-1)!}{h^{m-1}}\left[f\left(a+2h, S_{|_{[a,a+h]}}(a+h) + \overline{Y'(a+h)}h + \frac{h^2}{2!}\overline{Y''(a+h)} + \right.\right.$$
$$+ \cdots + \frac{h^{m-1}}{(m-1)!}\overline{Y^{(m-1)}(a+h)} + \frac{h^m}{m!}A_1\Big) - \overline{Y'(a+h)} - \overline{Y''(a+h)}h$$
$$\left. - \cdots - \frac{1}{(m-2)!}\overline{Y^{(m-1)}(a+h)}h^{m-2}\right]. \tag{4.2.16}$$

Let us assume that the matrix equation (4.2.16) has only one solution A_1. This way the spline is totally determined in the interval $[a+h, a+2h]$.

Iterating this process, we can construct the matrix spline approximation by taking the interval $[a+(k-1)h, a+kh]$ as the last subinterval. For the succeeding interval $[a+kh, a+(k+1)h]$, we define the corresponding matrix spline as

$$S_{|_{[a+kh,a+(k+1)h]}}(x) = S_{|_{[a+(k-1)h,a+kh]}}(a+kh) + \overline{Y'(a+kh)}(x-(a+kh))$$
$$+\tfrac{1}{2!}\overline{Y''(a+kh)}(x-(a+kh))^2 + \cdots +$$
$$\tfrac{1}{(m-1)!}\overline{Y^{(m-1)}(a+kh)}(x-(a+kh))^{m-1} + \tfrac{1}{m!}A_k(x-(a+kh))^m, \tag{4.2.17}$$

where

$$\overline{Y'(a+kh)} = f\left(a+kh, S_{|_{[a+(k-1)h,a+kh]}}(a+kh)\right), \tag{4.2.18}$$

and in a similar manner one abbreviates

$$\begin{aligned} \overline{Y''(a+kh)} &= g_1\left(a+kh, S_{|_{[a+(k-1)h,a+kh]}}(a+kh)\right), \\ &\vdots \\ \overline{Y^{(m-1)}(a+kh)} &= g_{m-2}\left(a+kh, S_{|_{[a+(k-1)h,a+kh]}}(a+kh)\right). \end{aligned} \tag{4.2.19}$$

With this definition, the matrix spline $S(x) \in \mathscr{C}^1\left(\bigcup_{j=0}^{k}[a+jh, a+(j+1)h]\right)$ fulfills the differential equation (4.2.3) at point $x = a+kh$. As an additional requirement, we assume that $S_{|_{[a+kh,a+(k+1)h]}}(x)$ satisfies (4.2.3) at point $x = a+(k+1)h$:

$$S'_{|_{[a+kh,a+(k+1)h]}}(a+(k+1)h) = f\left(a+(k+1)h, S_{|_{[a+kh,a+(k+1)h]}}(a+(k+1)h)\right),$$

and expanding this expression gives

$$A_k = \frac{(m-1)!}{h^{m-1}}\left[f\left(a+(k+1)h, S_{|_{[a+kh,a+(k+1)h]}}(a+(k+1)h) + \overline{Y'(a+kh)}h\right.\right.$$
$$+ \cdots + \frac{h^{m-1}}{(m-1)!}\overline{Y^{(m-1)}(a+kh)} + \frac{h^m}{m!}A_1\Big) - \overline{Y'(a+kh)} - \overline{Y''(a+kh)}h$$
$$\left. - \cdots - \frac{h^{m-2}}{(m-2)!}\overline{Y^{(m-1)}(a+kh)}\right]. \tag{4.2.20}$$

Observe that the final result (4.2.20) relates directly to equations (4.2.12) and (4.2.16), when setting $k = 0$ and $k = 1$. We will demonstrate that these equations have a unique solution using a fixed-point argument.

For a fixed h and k, we consider the matrix function $g : \mathbb{R}^{r\times q} \to \mathbb{R}^{r\times q}$ defined by

$$g(T) = \frac{(m-1)!}{h^{m-1}}\left[f\left(a+(k+1)h, S_{|_{[a+kh,a+(k+1)h]}}(a+(k+1)h) + \overline{Y'(a+kh)}h\right.\right.$$
$$+ \cdots + \frac{h^{m-1}}{(m-1)!}\overline{Y^{(m-1)}(a+kh)} + \frac{h^m}{m!}T\Big) - \overline{Y'(a+kh)} - \overline{Y''(a+kh)}h$$
$$\left. - \cdots - \frac{h^{m-2}}{(m-2)!}\overline{Y^{(m-1)}(a+kh)}\right]. \tag{4.2.21}$$

Relation (4.2.20) holds if and only if $A_k = g(A_k)$, that is, if A_k is a fixed point for function $g(T)$. By using the definition (4.2.21) of g and applying the global Lipschitz's condition (4.2.5) for f, it immediately follows that

$$\|g(T_1) - g(T_2)\| \le \frac{Lh}{m}\|T_1 - T_2\|.$$

Taking $h < m/L$, the matrix function g is contractive. Therefore equation (4.2.20) has unique solutions A_k for $k = 0, 1, \ldots, n-1$, and the matrix spline is completely determined. In summary, we have proved the following theorem:

Theorem 4.1. *For the first-order matrix differential equation (4.2.3), let L be the corresponding Lipschitz constant defined by (4.2.5). We also consider the partition (4.2.6) with step size $h < m/L$. Then, the matrix spline $S(x)$ of order $m \in \mathbb{N}$ exists in each subinterval $[a+kh, a+(k+1)h]$, $k = 0, 1, \ldots, n-1$, as defined in the previous construction and is of class $\mathscr{C}^1[a,b]$.*

Observe that the so constructed splines have a global error of $O(h^{m-1})$, which follows from an analysis similar to Loscalzo and Talbot's work [3].

The approximate solution of (4.2.3) can be computed by means of matrix splines of order m in the interval $[a,b]$ with an error of the order $O(h^{m-1})$ under the conditions of Theorem 4.1. The procedure is as follows:

- Compute the functions $g_1(x, Y(x)), \ldots, g_{m-2}(x, Y(x))$ given by (4.2.8)–(4.2.10) to determine the constant $Y''(a), \ldots, Y^{(m-1)}(a)$. Choose $n > L(b-a)/m$ so that $h = (b-a)/n$ with the partition $\Delta_{[a,b]}$ defined by Eq. (4.2.6).
- Solve equation (4.2.12) to find A_0, and determine $S_{|_{[a,a+h]}}(x)$ of Eq. (4.2.7).
- Solve equations (4.2.20) iteratively for $k = 1, \ldots, n-1$ to find all A_k, and then compute the splines $S_{|_{[a+kh,a+(k+1)h]}}(x)$ according to Eq. (4.2.17).

In order to find A_k for $k = 0, 1, \ldots, n-1$, one may solve equations (4.2.12) and (4.2.20) either explicitly [9], or by employing an iterative method [10]. For example, we can consider the recursion relation $T_{l+1}^s = g(T_l^s)$. Here, T_0^s is an arbitrary matrix in $\mathbb{R}^{r\times q}$ for $s = 0, 1, \ldots, n-1$, and $g(T)$ is given by (4.2.21).

3 Numerical Examples

3.1 A scalar test problem

This simple test problem is motivated by Loscalzo and Talbot's seminal work on scalar spline function approximation for ordinary differential equations [3]. Unfortunately, their otherwise very efficient method had the drawback to be divergent for higher degree spline functions ($m > 3$). Here, we will compare our procedure with their test case for the spline solution of $y' = y$ with initial condition $y(0) = 1$.

Figure 4.1 depicts the error of fourth-order spline solutions for the Loscalzo-Talbot problem which were constructed by our proposed method. Observe that for $h = 0.01$ the results already reach the accuracy of 10^{-14}, compared to the serious error of the conventional Loscalzo-Talbot method [3]. It also becomes clear that a further reduction in step size h does not necessarily improve the approximation.

It may be interesting to study the increasing quality of the approximation with higher-order splines. Figure 4.2 shows how the solutions improve by taking $m = 4, 5, 6$, respectively, with a constant step size $h = 0.1$.

3.2 A non-linear vector system

As a second example of our method, we choose the following vector differential system for the interval $x \in [0, 1]$, which is clearly non-linear:

$$\left.\begin{aligned} y_1'(x) &= -1 + e^x - \sin x + \sin\left(y_2(x)\right) \\ y_2'(x) &= \frac{1}{4 + y_1^2(x)} - \frac{1}{5 + e^{2x} + 2\,e^x \cos x - \sin^2 x} \end{aligned}\right\} \tag{4.3.22}$$

with the initial values

$$\left.\begin{aligned} y_1(0) &= 2 \\ y_2(0) &= \pi/2 \end{aligned}\right\}.$$

We can then rewrite the problem using vector notation $\mathbf{y}(x) = \begin{pmatrix} y_1(x) \\ y_2(x) \end{pmatrix}$ with $\mathbf{y}(0) = \begin{pmatrix} 2 \\ \pi/2 \end{pmatrix}$ to obtain the nonlinear vector problem $\mathbf{y}'(x) = \mathbf{f}(x, \mathbf{y}(x))$, where

$$\mathbf{f}\left(x, \mathbf{y}(x)\right) = \begin{pmatrix} -1 + e^x - \sin x + \sin\left(y_2(x)\right) \\ \dfrac{1}{4 + y_1^2(x)} - \dfrac{1}{5 + e^{2x} + 2\,e^x \cos x - \sin^2 x} \end{pmatrix}. \tag{4.3.23}$$

According to Ref. [6] this problem has the exact solution $y_1(x) = e^x + \cos x$ and $y_2(x) = \pi/2$, and hence for this test case we will be able to assess the exact error of our numerical estimates. Our proposed method serves to construct the splines of fifth order for the problem given in Eq. (4.3.22). For this we require to calculate $\mathbf{y}''(x)$, $\mathbf{y}^{(3)}(x)$ and $\mathbf{y}^{(4)}(x)$, which in general is straightforward. We may derive $\mathbf{y}''(x) = \begin{pmatrix} y_1''(x) \\ y_2''(x) \end{pmatrix}$ using a computer algebra system such as *Mathematica*, which readily produces:

$$\left.\begin{aligned} y_1''(x) &= e^x - \cos(x) + \cos\left(y_2(x)\right) y_2'(x) \\ y_2''(x) &= \frac{2e^{2x} + 2e^x\cos(x) - 2e^x\sin(x) - 2\cos(x)\sin(x)}{\left(5 + e^{2x} + 2e^x\cos(x) - \sin(x)^2\right)^2} - \frac{2y_1(x)y_1'(x)}{\left(4 + y_1(x)^2\right)^2} \end{aligned}\right\}. \tag{4.3.24}$$

Taking into account that $y_1(0) = 2, y_1'(0) = 1, y_2(0) = \frac{\pi}{2}$, and $y_2'(0) = 0$, it follows by Eq. (4.3.24) that $\mathbf{y}''(0) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$. We similarly calculate the third-order derivative $\mathbf{y}^{(3)}(x) = \begin{pmatrix} y_1^{(3)}(x) \\ y_2^{(3)}(x) \end{pmatrix}$ with components:

$$\left.\begin{aligned} y_1^{(3)}(x) &= e^x + \sin(x) - \sin\left(y_2(x)\right)\left(y_2'(x)\right)^2 + \cos\left(y_2(x)\right) y_2''(x) \\ y_2^{(3)}(x) &= -\frac{8\left(\cos(2x) - 2e^x\left(e^x - \sin(x)\right)\right)}{\left(9 + 2e^{2x} + 4e^x\cos(x) + \cos(2x)\right)^2} \\ &\quad - \frac{64\left(e^x + \cos(x)\right)^2\left(e^x - \sin(x)\right)^2}{\left(9 + 2e^{2x} + 4e^x\cos(x) + \cos(2x)\right)^3} \\ &\quad + \frac{8\left(y_1(x)\right)^2\left(y_1'(x)\right)^2}{\left(4 + \left(y_1(x)\right)^2\right)^3} - \frac{2\left(y_1'(x)\right)^2}{\left(4 + \left(y_1(x)\right)^2\right)^2} - \frac{2y_1(x)y_1''(x)}{\left(4 + \left(y_1(x)\right)^2\right)^2} \end{aligned}\right\} \tag{4.3.25}$$

In like manner as before, we consider $y_1(0) = 2, y_1'(0) = 1, y_1''(0) = 0, y_2(0) = \pi/2, y_2'(0) = 0$, and $y_2''(0) = 0$ with (4.3.25) to deduce $\mathbf{y}^{(3)}(0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$. Similarly, we may then derive the explicit results for the components of $\mathbf{y}^{(4)}(x) = \begin{pmatrix} y_1^{(4)}(x) \\ y_2^{(4)}(x) \end{pmatrix}$. In the final step, it remains to substitute the known values $y_1(0) = 2, y_1'(0) = 1, y_1''(0) = 0, y_1'''(0) = 1, y_2(0) = \pi/2, y_2'(0) = 0, y_2''(0) = 0, y_2'''(0) = 0$, into the last expression to obtain $\mathbf{y}^{(4)} = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$.

Also, it is not difficult to see that f, defined by (4.3.23), fulfills the global Lipschitz's condition

$$\left\|\mathbf{f}(x,\mathbf{y}) - \mathbf{f}(x,\mathbf{z})\right\| \le \left\|\mathbf{y} - \mathbf{z}\right\|, \quad 0 \le x \le 1, \quad \mathbf{y},\mathbf{z} \in \mathbb{R}^2. \tag{4.3.26}$$

Comparing with the general form (4.2.5), we note that $L = 1$. Therefore, by Theorem 4.1 we need to take $h < 5$. In the following, for example we choose $h = 0.1$ and summarize the numerical results in Table 4.1. In each interval, we evaluated the difference between the estimates of our numerical approach and the exact solution, and then take the Fröbenius norm of this difference, following the procedure explained in Ref. [6]. Table 4.2 lists the maximum of these errors for each subinterval.

For the solution of the vector differential system (4.3.22), Figure 4.3 illustrates the approximation behavior of various splines of the fourth order ($m = 4$) with the different step sizes $h = 0.1, 0.01$, and $h = 0.001$. All vector splines lie well in the predicted range of Theorem 4.1 and provide excellent approximations for the problem at hand with the benefit of very low computational cost. Observe that at step size $h = 0.001$ the limit of machine precision is practically reached and explains the random fluctuations around 10^{-15}. Hence, it obviously is of lesser interest to obtain more accurate approximations for $m = 4$ and $h = 0.001$.

3.3 Sylvester matrix differential equation

In many areas of science and engineering linear matrix differential equations appear of the type

$$\left.\begin{aligned} Y'(x) &= A(x)Y(x) + Y(x)B(x) + C(x) \\ Y(a) &= Y_a \end{aligned}\right\} \quad a \le x \le b, \tag{4.3.27}$$

where $Y(x), A(x), B(x), C(x) \in \mathbb{R}^{r\times r}$. The case of constant coefficients has been studied by several authors [11], whereas the variable-coefficient case has so far received little numerical treatment in the literature.

Following Ref. [6], we choose the following Sylvester problem (4.3.27) as a final example:

$$\begin{aligned} A(x) &= \begin{pmatrix} 0 & xe^{-x} \\ x & 0 \end{pmatrix}, \quad B(x) = \begin{pmatrix} 0 & x \\ 0 & 0 \end{pmatrix}, \\ C(x) &= \begin{pmatrix} -e^{-x}(1+x^2) & -2e^{-x}x \\ 1 - e^{-x}x & -x^2 \end{pmatrix}, \\ Y(0) &= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad Y(x) \in \mathbb{R}^{2\times 2},\ 0 \le x \le 1. \end{aligned} \tag{4.3.28}$$

According to [6] we know that this problem has the exact solution

$$Y(x) = \begin{pmatrix} e^{-x} & 0 \\ x & 1 \end{pmatrix}$$

with the Lipschitz constant $L = 2$. The higher-order derivatives of $Y(x)$ are required for the construction of the spline approximation and can be readily obtained.

For splines of the fifth order ($m = 5$), we take $n = 10$ partitions and $h = 0.1$. The results are summarized in Table 4.3, where the numerical estimates have been rounded to the sixth relevant digit. In Table 4.4, we evaluated the difference between the estimates of our numerical approach and the exact solution, and then take the Fröbenius norm of this difference. The maximum of these errors are indicated for each subinterval.

For the solution of the Sylvester matrix problem (4.3.27), Figure 4.4 depicts the approximation behavior of various splines of the fifth order ($m = 5$) with the different step sizes $h = 0.1, 0.01$, and $h = 0.001$. As before, all matrix splines lie well in the predicted range of Theorem 4.1. It becomes evident that the splines for step sizes $h = 0.01$ and $h = 0.001$ are almost indistinguishable and reach the same precision of almost 10^{-14}.

We also carried out the computations for the sixth order matrix splines ($m = 6$) with the step sizes $h = 0.1, 0.01$, and $h = 0.001$, and as expected, we could observe that $h = 0.01$ yields an accuracy close to machine precision. Interestingly, higher step sizes do not improve these approximations–the quality of approximation indeed deteriorates due to the accumulation of rounding errors.

3.4 The Hénon-Heiles system

The Hénon-Heiles equation [12] is a nonlinear nonintegrable Hamiltonian system defined by

$$\left.\begin{aligned} x'' &= -\frac{\partial V(x,y)}{\partial x} \\ y'' &= -\frac{\partial V(x,y)}{\partial y} \end{aligned}\right\}, \quad a \le t \le b, \tag{4.3.29}$$

where the potential-energy function is conserved during motion and given by the following expression

$$V(x,y) = \frac{1}{2}\left(x^2 + y^2 + 2x^2 - \frac{2}{3}y^3\right).$$

The differential system (4.3.29) can be recast in vectorial form $\mathbf{u}'(t) = \mathbf{f}(t, \mathbf{u}(t))$, where $\mathbf{u}(t) = (u_1(t)\ u_2(t)\ u_3(t)\ u_4(t))^T \in \mathbb{C}^4$ and

$$\mathbf{f}(t,\mathbf{u}) = \begin{pmatrix} u_2 \\ -u_1 - 2u_1u_3 \\ u_4 \\ -u_3 - u_1^2 + u_4^2 \end{pmatrix}.$$

For these benchmark tests, we have taken $t \in [0,1]$ with the initial conditions $x(0) = 1, \dot{x}(0) = 0.5, y(0) = 1$, and $\dot{y}(0) = 0.5$. Since the solution is unknown, we have considered as reference values the results generated by MATLAB ODE solver *ode45*. The parameters *RelTol* and *AbsTol* were chosen to obtain the maximum precision ($RelTol = 2.22045 \cdot 10^{-14}$, $AbsTol = 1.0 \cdot 10^{-14}$). The ODE solver *ode45* allows to solve non-stiff differential equations and is based on the Runge-Kutta method.

The numerical estimates are shown in Figures 4.5 and 4.6. Figure 4.5 depicts the errors for splines of order $m = 4$ with variable step size $h = 0.1, 0.01, 0.001$, whereas in Figure 4.6 the step size $h = 0.1$ is fixed and the spline order varies $m = 4, 5, 6$.

As can be seen in Figure 4.5, the error is situated well within the expected margins improving with each lower value of h. On the other hand, in Figure 4.6 with $h = 0.1$ the error is not exceeding the predicted maximum estimate $O(h^{m-1})$ for $m = 4, 5, 6$.

4 Conclusions

The present chapter focused on the presentation of a new method for the numerical integration of first-order matrix differential equations of the type $Y'(x) = f(x, Y(x))$ in the interval $[a,b]$ using higher-order matrix splines ($m > 3$). Contrary to existing spline methods in the literature, this new method only requires first-order derivatives for the construction of the splines to provide a continuous approximation of order $O(h^{m-1})$. Additionally, our method is well-suited for implementation on numerical and/or symbolical computer systems.

For an explicit demonstration of our proposed method and its advantages over existing conventional methods, we discussed three numerical test cases with excellent results. It is hoped that this new approach to approximating matrix differential models will motivate and open up alternative avenues to tackle different related problems in science and engineering.

References

1. G. Freiling and A Hochhaus, *On a class of rational matrix differential equations arising in stochastic control*, Linear Algebra Appl. **379** (2004), 43–68.
2. U. M. Ascher, R. M. M. Mattheij, and R. D. Russell, *Numerical solutions of boundary value problems for ordinary differential equations*, Prentice Hall, New Jersey, 1988.
3. F. R. Loscalzo and T. D. Talbot, *Spline function approximations for solutions of ordinary differential equations*, SIAM J. Numer. Anal. **4** (1967), no. 3, 433–445.
4. E. A. Al-Said and M. A. Noor, *Cubic splines method for a system of third-order boundary value problems*, Appl. Math. Comput. **142** (2003), 195–204.
5. E. Defez, L. Soler, A. Hervás, and C. Santamaría, *Numerical solutions of matrix differential models using cubic matrix splines*, Comput. Math. Appl. **50** (2005), 693–699.
6. E. Defez, A. Hervás, L. Soler, and M. M. Tung, *Numerical solutions of matrix differential models using cubic matrix splines II*, Math. Comput. Modelling **46** (2007), 657–669.

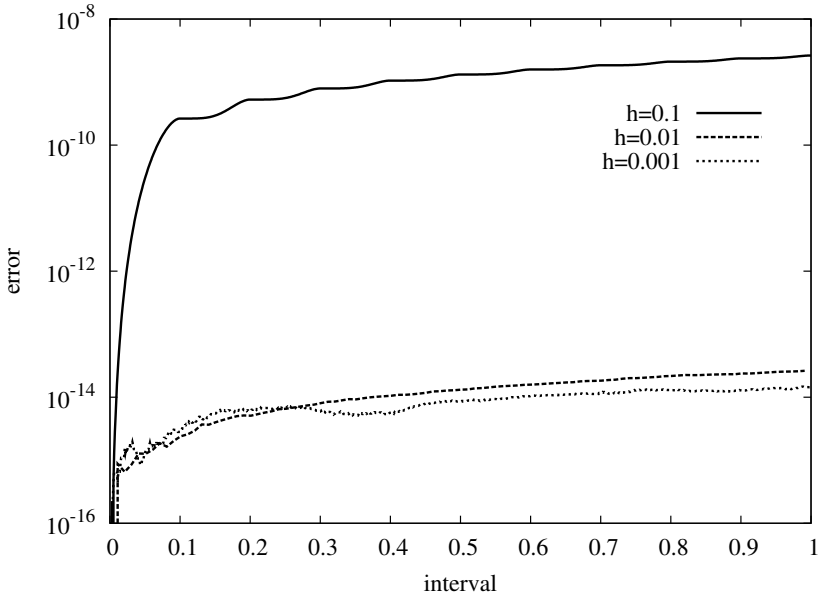

Fig. 4.1. Error for the Loscalzo-Talbot problem with splines of fourth order ($m = 4$) using our proposed method for various step sizes.

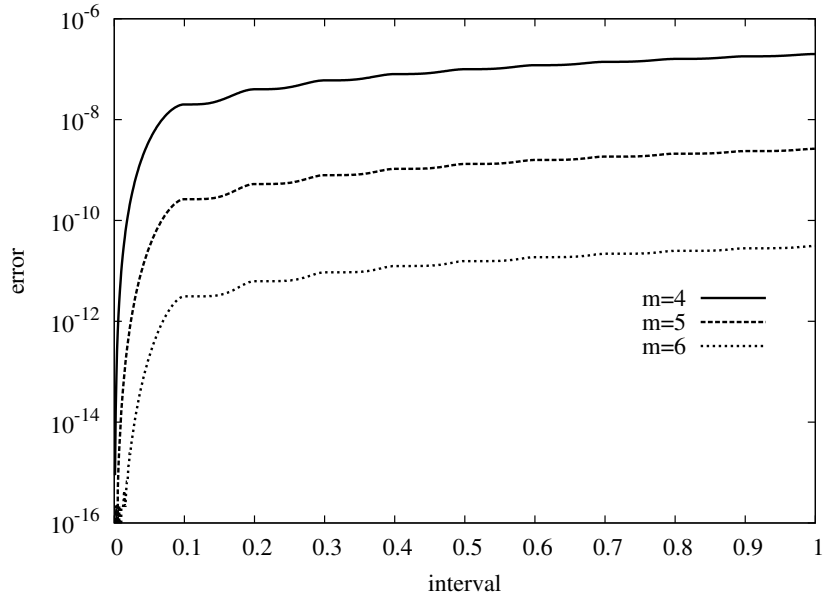

Fig. 4.2. Errors for increasing spline orders ($m = 4, 5, 6$) solving the Loscalzo-Talbot problem. The step size is constant ($h = 0.1$).

7. A. Graham, *Kronecker products and matrix calculus with applications*, John Wiley, New York, 1981.
8. T. M. Flett, *Differential analysis*, Cambridge University Press, Cambridge, UK, 1980.
9. P. Lancaster, *Explicit solutions of linear matrix equations*, SIAM Review **12** (1970), 544–566.
10. J. M. Ortega and W. C. Rheinboldt, *Iterative solution of nonlinear equations in several variables*, Academic Press, New York, 1972.
11. A. Y. Barraud, *Nouveaux développements sur la résolution numérique de* $x' = ax + xb + c; x(0) = c$, R.A.I.R.O., Automatique/Systems Analysis and Control **16** (1982), no. 4, 341–356.
12. M. Hénon and C. Heiles, *The applicability of the third integral of motion: Some numerical experiments*, Astron. J. **69** (1964), 73–79.

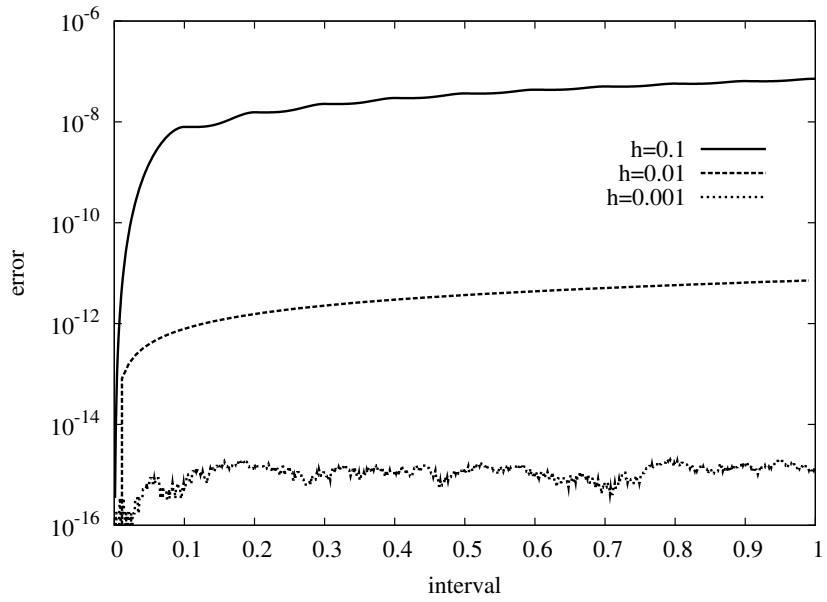

Fig. 4.3. Representing the 2-norm error for the vector differential system (4.3.22) using splines of fourth order ($m = 4$).

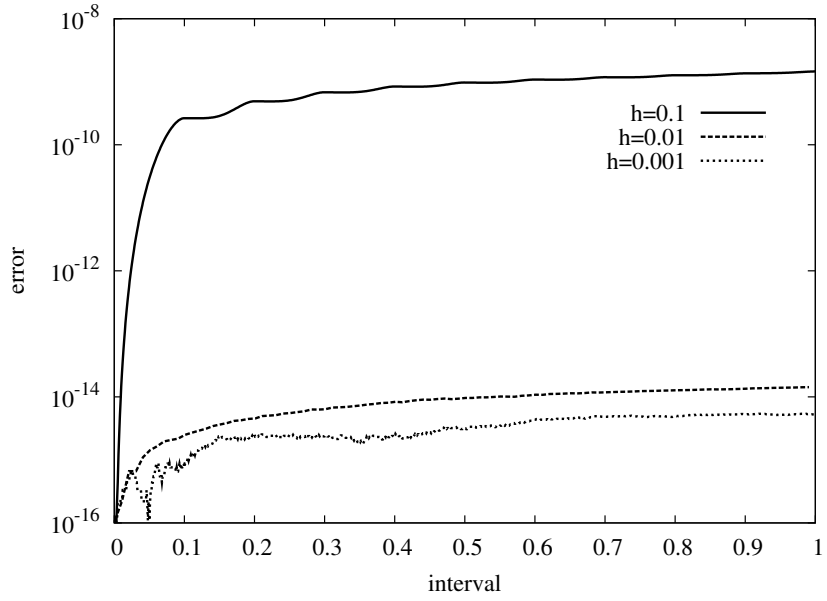

Fig. 4.4. Representing the 2-norm error for the Sylvester matrix differential equation (4.3.27) using splines of fourth order ($m = 4$).

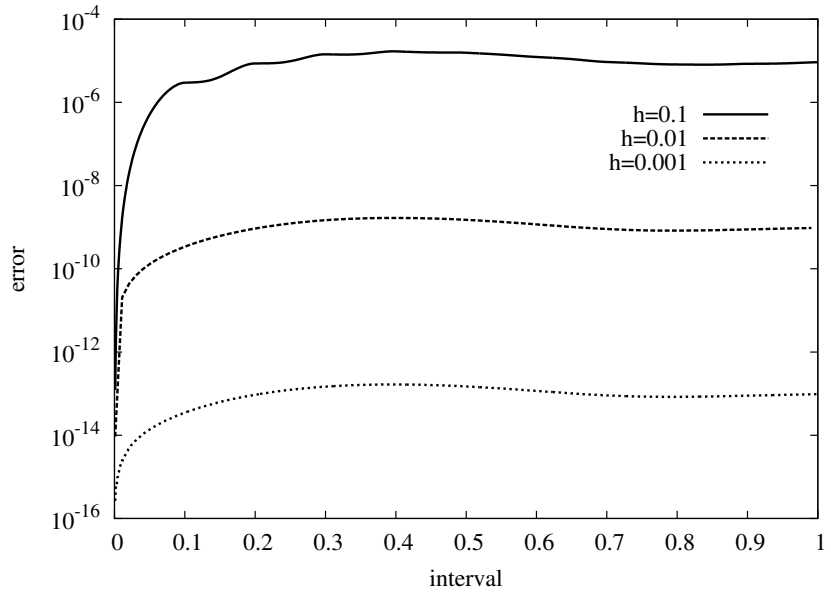

Fig. 4.5. Error for the Henon-Heiles problem with splines of fourth order ($m = 4$) using our proposed method for various step sizes.

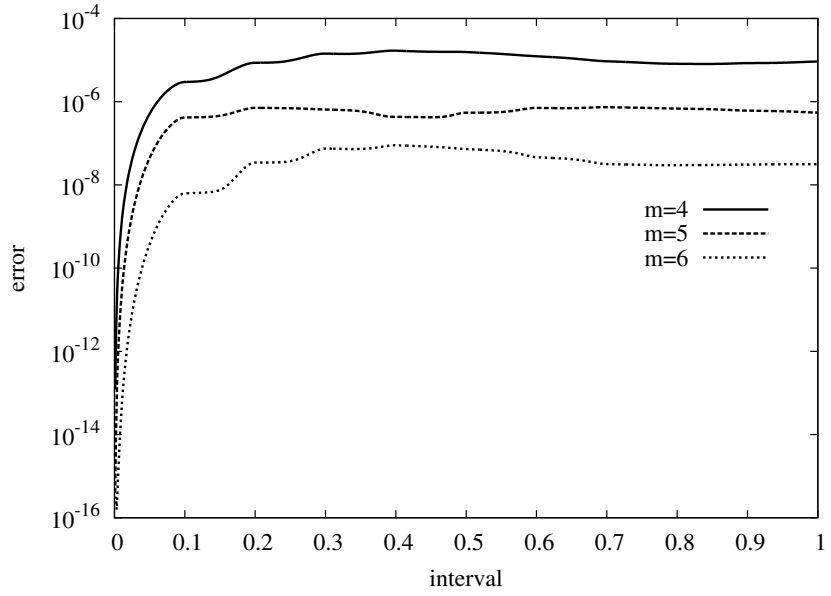

Fig. 4.6. Errors for increasing spline orders ($m = 4, 5, 6$) solving the Henon-Heiles problem. The step size is constant ($h = 0.1$).

Interval	Approximation
$[0,0.1]$	$\begin{pmatrix} 2.+x+0.166667x^3+0.0833333x^4+0.00833619x^5 \\ 1.5708 \end{pmatrix}$
$[0.1,0.2]$	$\begin{pmatrix} 2.+1.x-3.98676\times10^{-7}x^2+0.166671x^3+0.0833075x^4+0.0083996x^5 \\ 1.5708+1.02341\times10^{-9}x^2-9.53254\times10^{-9}x^3+4.27272\times10^{-8}x^4-7.27159\times10^{-8}x^5 \end{pmatrix}$
$[0.2,0.3]$	$\begin{pmatrix} 2.+1.x-9.78808\times10^{-6}x^2+0.166723x^3+0.0831609x^4+0.00856703x^5 \\ 1.5708-2.80891\times10^{-9}x+2.68447\times10^{-8}x^2-1.2696\times10^{-7}x^3+2.96285\times10^{-7}x^4-2.72203\times10^{-7}x^5 \end{pmatrix}$
$[0.3,0.4]$	$\begin{pmatrix} 2.+1.00001x-0.000070073x^2+0.166941x^3+0.0827649x^4+0.00885657x^5 \\ 1.5708-2.3641\times10^{-8}x+1.51773\times10^{-7}x^2-4.84484\times10^{-7}x^3+7.68203\times10^{-7}x^4-4.83576\times10^{-7}x^5 \end{pmatrix}$
$[0.4,0.5]$	$\begin{pmatrix} 2.+1.00005x-0.000295117x^2+0.167541x^3+0.0819626x^4+0.00928717x^5 \\ 1.5708-1.04291\times10^{-7}x+5.05234\times10^{-7}x^2-1.21958\times10^{-6}x^3+1.46618\times10^{-6}x^4-7.01984\times10^{-7}x^5 \end{pmatrix}$
$[0.5,0.6]$	$\begin{pmatrix} 1.99998+1.0002x-0.000921692x^2+0.168862x^3+0.080566x^4+0.00987867x^5 \\ 1.5708-3.25859\times10^{-7}x+1.26869\times10^{-6}x^2-2.46395\times10^{-6}x^3+2.3864\times10^{-6}x^4-9.21882\times10^{-7}x^5 \end{pmatrix}$
$[0.6,0.7]$	$\begin{pmatrix} 1.99993+1.00062x-0.00237386x^2+0.171395x^3+0.0783551x^4+0.0106518x^5 \\ 1.5708-8.18293\times10^{-7}x+2.66421\times10^{-6}x^2-4.32971\times10^{-6}x^3+3.51164\times10^{-6}x^4-1.13697\times10^{-6}x^5 \end{pmatrix}$
$[0.7,0.8]$	$\begin{pmatrix} 1.9998+1.00162x-0.00534205x^2+0.175805x^3+0.0750753x^4+0.0116284x^5 \\ 1.5708-1.76297\times10^{-6}x+4.93332\times10^{-6}x^2-6.89355\times10^{-6}x^3+4.80962\times10^{-6}x^4-1.34027\times10^{-6}x^5 \end{pmatrix}$
$[0.8,0.9]$	$\begin{pmatrix} 1.99947+1.00376x-0.0108784x^2+0.18297x^3+0.0704351x^4+0.0128313x^5 \\ 1.5708-3.38486\times10^{-6}x+8.30591\times10^{-6}x^2-0.0000101804x^3+6.23218\times10^{-6}x^4-1.52432\times10^{-6}x^5 \end{pmatrix}$
$[0.9,1.0]$	$\begin{pmatrix} 1.99873+1.00796x-0.0205098x^2+0.19401x^3+0.0641039x^4+0.0142844x^5 \\ 1.5708-5.93162\times10^{-6}x+0.000012961x^2-0.0000141487x^3+7.71598\times10^{-6}x^4-1.68162\times10^{-6}x^5 \end{pmatrix}$

Table 4.1. Vector approximation for system (4.3.22) in the interval $[0,1]$.

Interval	$[0,0.1]$	$[0.1,0.2]$	$[0.2,0.3]$	$[0.3,0.4]$	$[0.4,0.5]$
Max. error	8.2362×10^{-12}	4.8717×10^{-11}	1.27357×10^{-10}	2.50353×10^{-10}	4.24194×10^{-10}
Interval	$[0.5,0.6]$	$[0.6,0.7]$	$[0.7,0.8]$	$[0.8,0.9]$	$[0.9,1.0]$
Max. error	6.55672×10^{-10}	9.51896×10^{-10}	1.32033×10^{-9}	1.7688×10^{-9}	2.30555×10^{-9}

Table 4.2. Approximation error for vector problem (4.3.22).

Interval	Approximation
$[0,0.1]$	$\begin{pmatrix} 1.-1.x+0.5x^2-0.166667x^3+0.0416667x^4-0.00816941x^5 & 0. \\ x. & 1. \end{pmatrix}$
$[0.1,0.2]$	$\begin{pmatrix} 1.-1.x+0.499997x^2-0.166626x^3+0.0413976x^4-0.00739198x^5 & 0. \\ 1.x & 1. \end{pmatrix}$
$[0.2,0.3]$	$\begin{pmatrix} 1.-0.999997x+0.499961x^2-0.166422x^3+0.0408023x^4-0.00668854x^5 & 0. \\ 1.x & 1. \end{pmatrix}$
$[0.3,0.4]$	$\begin{pmatrix} 0.999999-0.999979x+0.499834x^2-0.165957x^3+0.0399455x^4-0.00605204x^5 & 0. \\ 1.x & 1. \end{pmatrix}$
$[0.4,0.5]$	$\begin{pmatrix} 0.999995-0.999925x+0.499542x^2-0.16517x^3+0.0388822x^4-0.00547612x^5 & 0. \\ 1.x & 1. \end{pmatrix}$
$[0.5,0.6]$	$\begin{pmatrix} 0.999983-0.999797x+0.499x^2-0.16402x^3+0.0376596x^4-0.00495499x^5 & 0 \\ 1.x & 1. \end{pmatrix}$
$[0.6,0.7]$	$\begin{pmatrix} 0.999954-0.999547x+0.498127x^2-0.16249x^3+0.0363175x^4-0.00448346x^5 & 0 \\ 1.x & 1. \end{pmatrix}$
$[0.7,0.8]$	$\begin{pmatrix} 0.999896-0.999117x+0.496844x^2-0.160578x^3+0.0348899x^4-0.00405681x^5 & 0 \\ 1.x & 1. \end{pmatrix}$
$[0.8,0.9]$	$\begin{pmatrix} 0.999792-0.998438x+0.495083x^2-0.158291x^3+0.033405x^4-0.00367075x^5 & 0 \\ 1.x & 1. \end{pmatrix}$
$[0.9,1.0]$	$\begin{pmatrix} 0.999617-0.997437x+0.492785x^2-0.155651x^3+0.0318868x^4-0.00332143x^5 & 0 \\ 1.x & 1. \end{pmatrix}$

Table 4.3. Approximation for the Sylvester matrix problem (4.3.27).

Interval	$[0,0.1]$	$[0.1,0.2]$	$[0.2,0.3]$	$[0.3,0.4]$	$[0.4,0.5]$
Max. error	2.6999×10^{-10}	5.1438×10^{-10}	7.36134×10^{-10}	9.38797×10^{-10}	1.1268×10^{-9}
Interval	$[0.5,0.6]$	$[0.6,0.7]$	$[0.7,0.8]$	$[0.8,0.9]$	$[0.9,1.0]$
Max. error	1.30572×10^{-9}	1.48252×10^{-9}	1.66579×10^{-9}	1.86603×10^{-9}	2.09601×10^{-9}

Table 4.4. Approximation error for the Sylvester matrix problem (4.3.27).

CHAPTER 5

Approximate solutions of second-order matrix differential equations with higher-order splines

1 Introduction

Splines are an important tool to solve scalar first-order differential equations [1], obtaining approximations that, among other advantages, are of class $\mathscr{C}^1$ in a given interval $[a,b]$. Splines are easy to compute and the associated approximation errors are only of $O(h^4)$. They also have been used in the resolution of other scalar problems, as discussed in [2–4], for vector problems [5], linear matrix problems [6] and for first-order matrix differential equations [7]. Recent work in this field can be found in Refs. [8–10]. Recently, numerical schemes with cubic splines were extended to the resolution of second-order matrix problems without any additional increase in dimensionality of the problem [11]. To achieve this goal, these schemes do not require the reduction of the system to a higher dimensional system of lower order–a common practice in problems of this kind. All spline approximations are by construction already continuous in the interval under consideration. Some explicit numerical examples have been used to test the methods and have shown that errors are only of the order $O(h^{m-1})$, where m is the order of the spline. By adapting the step size h and the order m of the spline to a particular problem, in principle, any desired accuracy can be reached. Unfortunately, as detected by Loscalzo and Talbot, their scalar procedure is divergent when higher-order spline functions with $m > 3$ are used [1, p. 444–445]. They have explicitly shown by numerical computations that the equation $y' = y, y(0) = 1$ contains noticeable divergences for splines of order $m > 3$. For matrix differential equations of first order, we already presented in Ref. [12] a method which avoids these problems with divergences for splines $S(x)$ of order m. Our goal in this work is to extend this method to second-order matrix equations without increasing the dimension of the problem, using splines $S(x)$ of order m but only of differentiability class $\mathscr{C}^2$.

With these benefits, it is hoped that our approach provides an alternative method to existing ones and may open up new avenues to the numerical integration of second-order models in practical applications.

Throughout this work, we will adopt the notation for norms and matrix cubic splines as in the previous work [6] and common in matrix calculus. Following this nomenclature, we define

the Kronecker product of $A = (a_{ij}) \in \mathbb{C}^{m\times n}$ and $B \in \mathbb{C}^{r\times s}$, denoted by $A \otimes B$, as the block matrix

$$A \otimes B = \begin{bmatrix} a_{11}B & \dots & a_{1n}B \\ \vdots & & \vdots \\ a_{m1}B & \dots & a_{mn}B \end{bmatrix}.$$

The column-vector operator on a matrix $A \in \mathbb{C}^{m\times n}$ is given by

$$vec(A) = \begin{bmatrix} A_{\bullet 1} \\ \vdots \\ A_{\bullet n} \end{bmatrix}, \text{ where } A_{\bullet k} = \begin{bmatrix} a_{1k} \\ \vdots \\ a_{mk} \end{bmatrix}.$$

If $Y = (y_{ij}) \in \mathbb{C}^{p\times q}$ and $X = (x_{ij}) \in \mathbb{C}^{m\times n}$, then the derivative of a matrix with respect to a matrix is defined by [13, p. 62 and 81]:

$$\frac{\partial Y}{\partial X} = \begin{bmatrix} \frac{\partial Y}{\partial x_{11}} & \cdots & \frac{\partial Y}{\partial x_{1n}} \\ \vdots & & \vdots \\ \frac{\partial Y}{\partial x_{m1}} & \cdots & \frac{\partial Y}{\partial x_{mn}} \end{bmatrix}, \text{ where } \quad \frac{\partial Y}{\partial x_{rs}} = \begin{bmatrix} \frac{\partial y_{11}}{\partial x_{rs}} & \cdots & \frac{\partial y_{1q}}{\partial x_{rs}} \\ \vdots & & \vdots \\ \frac{\partial y_{p1}}{\partial x_{rs}} & \cdots & \frac{\partial y_{pq}}{\partial x_{rs}} \end{bmatrix}.$$

If $X \in \mathbb{C}^{m\times n}, Y \in \mathbb{C}^{n\times v}, Z \in \mathbb{C}^{p\times q}$, then the following rule for the derivative of a matrix product with respect to another matrix applies [13, p. 84]:

$$\frac{\partial XY}{\partial Z} = \frac{\partial X}{\partial Z}\left[I_q \otimes Y\right] + \left[I_p \otimes X\right]\frac{\partial Y}{\partial Z}, \tag{5.1.1}$$

where I_q and I_p denote the identity matrices of dimensions q and p, respectively. If $X \in \mathbb{C}^{m\times n}, Y \in \mathbb{C}^{u\times v}, Z \in \mathbb{C}^{p\times q}$, the following chain rule [13, p. 88] is valid:

$$\frac{\partial Z}{\partial X} = \left[\frac{\partial \left[vec(Y)\right]^t}{\partial X} \otimes I_p\right]\left[I_n \otimes \frac{\partial Z}{\partial \left[vec(Y)\right]}\right]. \tag{5.1.2}$$

2 Higher-Order Matrix Splines

Frequent in different fields of physics and engineering are matrix initial value problems of the form:

$$\left.\begin{aligned} Y''(x) &= f(x, Y(x), Y'(x)) \\ Y(a) = Y_0\ ,\ Y'(a) &= Y_1 \end{aligned}\right\}\ a \le x \le b, \tag{5.2.3}$$

Note that Eq. (5.2.3) could *e.g.* be the statement of Newton's second law of motion for a coupled mechanical system. Moreover, models of this kind often appear in molecular dynamics, quantum mechanics and for scattering methods, where one solves scalar or vectorial problems with boundary values conditions [14–19].

Let us consider the initial matrix value problem Eq. (5.2.3) where $Y_0, Y_1, Y(t) \in \mathbb{R}^{r\times q}$, $f : [a,b] \times \mathbb{R}^{r\times q} \times \mathbb{R}^{r\times q} \longrightarrow \mathbb{R}^{r\times q}$, $f \in \mathscr{C}^s(T)$, with

$$T = \{(x, Y, Z)\ ;\ a \le x \le b\ ,\ Y, Z \in \mathbb{R}^{r\times q}\}$$

and $\mathbb{R}^{r\times q}$ will in general denote the set of $r \times q$ rectangular real matrices.

The Lipschitz conditions on the function f

$$\left.\begin{array}{l} \left\| f\left(x, Y_1, Y\right) - f\left(x, Y_2, Y\right)\right\| \leq L_1 \left\| Y_1 - Y_2\right\| \ , \ a \leq x \leq b \ , Y_1, Y_2, Y \in \mathbb{R}^{r\times q} \\ \left\| f\left(x, Y, Y_1\right) - f\left(x, Y, Y_2\right)\right\| \leq L_2 \left\| Y_1 - Y_2\right\| \ , \ a \leq x \leq b \ , Y_1, Y_2, Y \in \mathbb{R}^{r\times q} \end{array}\right\} , \tag{5.2.4}$$

guarantees the existence and uniqueness of the continuously differentiable solution $Y(x)$ for the set of equations (5.2.3), see *e.g.* [20, p. 99].

The partition of the interval $[a, b]$ shall be given by

$$\Delta_{[a,b]} = \{a = x_0 < x_1 < \ldots < x_n = b\} \ , \ x_k = a + kh, \ k = 0, 1, \ldots, n, \tag{5.2.5}$$

where n is a positive integer with the corresponding step size $h = (b-a)/n$. To be brief, we will denote $I_k = [a + kh, a + (k + 1)h)], k = 0, 1, 2, ...$

Theorem 5.1.

(i) Let $f \in C^s(T)$ and $m = s + 3$, then, there exists a matrix spline $S(x) \in C^2([a, b])$, $S(x)$ of order m for each subinterval I_k, $k = 0, 1, \ldots, n-1$, if the step size h is chosen as

$$h < \left(\sqrt{L_2^2 m^2 + 4m(m-1)L_1} - mL_2\right)/2L_1$$

where L_1, L_2 are Lipschitz constants defined by Eq. (5.2.4).

(ii) If $f \in C^s(T)$ with $m = s + 3$, then $\|Y(x) - S(x)\|$ is at least of order $O(h^{m-1})$ $\forall x \in [a, b]$, where $Y(x)$ is the solution of Eq. (5.2.3).

Proof. We will construct in each subinterval I_k a matrix spline $S(x)$ of order $m \in \mathbb{N}$ with $m = s + 3$, where s is the order of the differentiability class of f. This will approximate the solution of problem (5.2.3) so that $S(x) \in \mathscr{C}^2([a, b])$.

In the first interval I_0, we define the matrix spline as

$$\begin{aligned} S_{|_{I_0}}(x) = Y(a) + Y'(a)(x-a) + \frac{1}{2!}Y''(a)(x-a)^2 + \frac{1}{3!}Y^{(3)}(a)(x-a)^3 \\ + \cdots + \frac{1}{(m-1)!}Y^{(m-1)}(a)(x-a)^{m-1} + \frac{1}{m!}A_0(x-a)^m. \end{aligned} \tag{5.2.6}$$

Here the matrix coefficient $A_0 \in \mathbb{R}^{r\times q}$ is a parameter still to be determined. Observe that $S_{|_{I_0}}(a) = Y(a) = Y_0$, $S'_{|_{I_0}}(a) = Y'(a) = Y_1$ and $S''_{|_{I_0}}(a) = Y''(a) = f(a, Y(a), Y'(a))$, and thus equation (5.2.3) is fulfilled by the spline at point $x = a$.

For the construction of matrix spline (5.2.6), we first must find the values of $Y^{(3)}(a)$, $Y^{(4)}(a)$, ..., $Y^{(m-1)}(a)$. To compute the third-order derivative $Y^{(3)}(x)$, we consider the functions h_1, h_2 and h_3 defined by

$$\begin{array}{ccc} h_1 : [a, b] \mapsto [a, b] & h_2 : [a, b] \mapsto \mathbb{C}^{r\times q} & h_3 : [a, b] \mapsto \mathbb{C}^{r\times q} \\ h_1(x) = x \ , & h_2(x) = Y(x) \ , & h_3(x) = Z(x) \end{array}$$

where $Y(x)$ is the theoretical solution of (5.2.3) and $Z(x) = Y'(x)$. We describe now $f(x, Y(x), Y'(x))$ as a composition of functions f and (h_1, h_2, h_3), that is, let $\phi : [a, b] \mapsto \mathbb{C}^{r\times q}$ be defined by

$$\phi(x) \;=\; \left[f \circ (h_1, h_2, h_3)\right](x) \;=\; f\left(h_1(x), h_2(x), h_3(x)\right) \;=\; f(x, Y(x), Z(x)) \,.$$

Thus, ϕ is a real variable function of x, and applying theorem 8.9.2 of [13, p. 170] its derivative takes the form:

$$\begin{aligned} D\phi &= D\left(f \circ (h_1, h_2, h_3)\right) \\ &= \left((D_1 f)(h_1, h_2, h_3)\right)\cdot Dh_1 + \left((D_2 f)(h_1, h_2, h_3)\right)\cdot Dh_2 + \left((D_3 f)(h_1, h_2, h_3)\right)\cdot Dh_3, \end{aligned}$$

where the partial derivatives of f, $D_1(f)$, $D_2(f)$, $D_3(f)$ exist and are continuous since it is assumed that $f \in \mathscr{C}^s(T)$. By (5.2.3) it is clear that

$$\frac{d\left(vec\, Y'(x)\right)^T}{dx} \;=\; \left[vec\, f(x, Y(x), Y'(x))\right]^T \,.$$

Next, applying the chain rule for matrix functions (5.1.1) and then taking the derivative of a matrix with respect to a matrix, (5.1.2), one obtains

$$\begin{aligned} Y^{(3)}(x) &= \frac{\partial f(x, Y(x), Y'(x))}{\partial x} + \left[\left[vec\, Y'(x)\right]^T \otimes I_r\right] \frac{\partial f(x, Y(x), Y'(x))}{\partial\left(vec\, Y(x)\right)} \\ &\quad + \left[\left[vec\, f(x, Y(x), Y'(x))\right]^T \otimes I_r\right] \frac{\partial f(x, Y(x), Y'(x))}{\partial\left(vec\, Y'(x)\right)} \\ &= g_1\left(x, Y(x), Y'(x)\right), \end{aligned} \tag{5.2.7}$$

where $g_1 \in \mathscr{C}^{s-1}(T)$. Using (5.2.7), we are now in position to evaluate $Y^{(3)}(a) = g_1\left(a, Y(a), Y'(a)\right)$ It is safe to take $f \in \mathscr{C}^s(T)$ for $s \geq 2$. For all higher-order derivatives $Y^{(4)}(x), \ldots, Y^{(m-1)}(x)$, we proceed in a similar way and calculate

$$\left.\begin{aligned} Y^{(4)}(x) &= g_2\left(x, Y(x), Y'(x)\right) \in \mathscr{C}^{s-2}(T) \\ &\vdots \\ Y^{(m-1)}(x) &= g_{m-3}\left(x, Y(x), Y'(x)\right) \in \mathscr{C}^{s-(m-3)}(T) \end{aligned}\right\}. \tag{5.2.8}$$

Apart from the matrix coefficient A_0, all other parameters of the spline are already known. To find A_0, we assume that (5.2.6) is a solution of equation (5.2.3) at $x = a + h$, and thus yields

$$S''_{\big|_{I_0}}(a+h) = f\left(a+h, S_{\big|_{I_0}}(a+h), S'_{\big|_{I_0}}(a+h)\right). \tag{5.2.9}$$

The only unknown parameter A_0 is now given by the following implicit matrix equation obtained from (5.2.9):

$$\begin{aligned} A_0 = \frac{(m-2)!}{h^{m-2}} \Bigg[& f\left(a+h, Y(a) + Y'(a)h + \cdots + \frac{h^{m-1}}{(m-1)!}Y^{(m-1)}(a) + \frac{h^m}{m!}A_0,\right. \\ & \left. Y'(a) + Y''(a)h + \cdots + \frac{h^{m-2}}{(m-2)!}Y^{(m-1)}(a) + \frac{h^{m-1}}{(m-1)!}A_0\right) \\ & - Y''(a) - \cdots - \frac{h^{m-3}}{(m-3)!}Y^{(m-1)}(a)\Bigg]. \end{aligned} \tag{5.2.10}$$

The matrix equation (5.2.10) has only one solution A_0, which we shall see in the following. After finding the solution for A_0, the matrix spline (5.2.6) is totally determined within the first interval I_0. For the second interval I_1 we define

$$\begin{aligned} S_{\big|_{I_1}}(x) &= S_{\big|_{I_0}}(a+h) + S'_{\big|_{I_0}}(a+h)(x-(a+h)) + \\ &\tfrac{1}{2!}\overline{Y''(a+h)}(x-(a+h))^2 + \cdots + \tfrac{1}{(m-1)!}\overline{Y^{(m-1)}(a+h)}(x-(a+h))^{m-1} \\ &+\tfrac{1}{m!}A_1(x-(a+h))^m, \end{aligned} \tag{5.2.11}$$

and we use the following shorthand notation

$$\begin{aligned} \overline{Y''(a+h)} &= f\left(a+h, S_{\big|_{I_0}}(a+h), S'_{\big|_{I_0}}(a+h)\right), \\ \overline{Y^{(3)}(a+h)} &= g_1\left(a+h, S_{\big|_{I_0}}(a+h), S'_{\big|_{I_0}}(a+h)\right), \\ &\vdots \\ \overline{Y^{(m-1)}(a+h)} &= g_{m-3}\left(a+h, S_{\big|_{I_0}}(a+h), S'_{\big|_{I_0}}(a+h)\right). \end{aligned} \tag{5.2.12}$$

The splines introduced by Loscalzo and Talbot [1] and the ones employed in [11] were of class $\mathscr{C}^{m-1}(I_0 \cup I_1)$. In contrast, here the matrix spline $S(x)$ defined by (5.2.6) and (5.2.11) is of differentiability class $\mathscr{C}^2(I_0 \cup I_1)$. All of the coefficients in (5.2.11) are completely determined except for the parameter $A_1 \in \mathbb{R}^{r\times q}$. By definition, spline (5.2.11) satisfies the differential equation (5.2.3) at point $x = a+h$. To find the value of A_1 we also assume that the spline (5.2.11) satisfies (5.2.3) at point $x = a+2h$:

$$S''_{\big|_{I_1}}(a+2h) = f\left(a+2h, S_{\big|_{I_1}}(a+2h), S'_{\big|_{I_1}}(a+2h)\right),$$

which readily may be recast in the following form to provide a matrix equation for the only unknown A_1:

$$\begin{aligned} A_1 = \frac{(m-2)!}{h^{m-2}}\Bigg[& f\left(a+2h, S_{\big|_{I_0}}(a+h) + S'_{\big|_{I_0}}(a+h)h + \frac{h^2}{2!}\overline{Y''(a+h)} + \cdots + \right. \\ & + \frac{h^{m-1}}{(m-1)!}\overline{Y^{(m-1)}(a+h)} + \frac{h^m}{m!}A_1, S'_{\big|_{I_0}}(a+h) + \overline{Y''(a+h)}h + \cdots \\ & \left. + \frac{h^{m-2}}{(m-2)!}\overline{Y^{(m-1)}(a+h)} + \frac{h^{m-1}}{(m-1)!}A_1\right) - \overline{Y''(a+h)} \\ & - \overline{Y^{(3)}(a+h)}h - \cdots - \frac{h^{m-3}}{(m-3)!}\overline{Y^{(m-1)}(a+h)} \end{aligned} \tag{5.2.13}$$

Now we proceed in exactly similar manner as before. Namely, let us assume that the implicit matrix equation (5.2.13) has a unique solution A_1, so that the spline is totally determined in the interval I_1. By iteration, we may construct the matrix spline taking I_{k-1} as the last subinterval. For the next subinterval I_k, we define the corresponding matrix spline as

$$
\begin{aligned}
S_{|_{I_k}}(x) &= S_{|_{I_{k-1}}}(a+kh) + S'_{|_{I_{k-1}}}(a+kh)(x-(a+kh)) \\
&+\tfrac{1}{2!}\overline{Y''(a+kh)}(x-(a+kh))^2 + \cdots + \\
&\tfrac{1}{(m-1)!}\overline{Y^{(m-1)}(a+kh)}(x-(a+kh))^{m-1} + \tfrac{1}{m!}A_k(x-(a+kh))^m,
\end{aligned} \tag{5.2.14}
$$

where again

$$
\begin{aligned}
\overline{Y''(a+kh)} &= f\left(a+kh, S_{|_{I_{k-1}}}(a+kh), S'_{|_{I_{k-1}}}(a+kh)\right), \\
\overline{Y^{(3)}(a+kh)} &= g_1\left(a+kh, S_{|_{I_{k-1}}}(a+kh), S'_{|_{I_{k-1}}}(a+kh)\right), \\
&\vdots \\
\overline{Y^{(m-1)}(a+kh)} &= g_{m-3}\left(a+kh, S_{|_{I_{k-1}}}(a+kh), S'_{|_{I_{k-1}}}(a+kh)\right).
\end{aligned} \tag{5.2.15}
$$

Thus, the matrix spline $S(x) \in \mathscr{C}^2\left(\bigcup_{j=0}^{k} I_j\right)$ is solution of the differential equation (5.2.3) at point $x = a+kh$. In order to find A_k, we impose the additional requirement that $S_{|_{I_k}}(x)$ satisfies equation (5.2.3) at point $x = a+(k+1)h$:

$$
S''_{|_{I_k}}(a+(k+1)h) = f\left(a+(k+1)h, S_{|_{I_k}}(a+(k+1)h), S'_{|_{I_k}}(a+(k+1)h)\right),
$$

which can be rewritten as

$$
\begin{aligned}
A_k = \tfrac{(m-2)!}{h^{m-2}}\Bigg[& f\Bigg(a+(k+1)h, S_{|_{I_{k-1}}}(a+kh) + S'_{|_{I_{k-1}}}(a+kh)h + \cdots + \\
& + \tfrac{h^{m-1}}{(m-1)!}\overline{Y^{(m-1)}(a+kh)} + \tfrac{h^m}{m!}A_k, S'_{|_{I_{k-1}}}(a+kh) + \overline{Y''(a+kh)}h + \cdots \\
& + \tfrac{h^{m-2}}{(m-2)!}\overline{Y^{(m-1)}(a+kh)} + \tfrac{h^{m-1}}{(m-1)!}A_k\Bigg) - \overline{Y''(a+kh)} \\
& - \overline{Y^{(3)}(a+kh)}h - \cdots - \tfrac{h^{m-3}}{(m-3)!}\overline{Y^{(m-1)}(a+kh)}\Bigg].
\end{aligned} \tag{5.2.16}
$$

Eqs. (5.2.10) and (5.2.13) are just a particular case of the final result (5.2.16), letting $k = 0$ and $k = 1$.

We are now in the position to demonstrate the uniqueness of equation (5.2.16) by using a fixed-point argument. For any choice of step size h and partition number k, we look at the matrix function $g : \mathbb{R}^{r\times q} \to \mathbb{R}^{r\times q}$ defined by

$$
\begin{aligned}
g(T) = \tfrac{(m-2)!}{h^{m-2}}\Bigg[& f\Bigg(a+(k+1)h, S_{|_{I_{k-1}}}(a+kh) + S'_{|_{I_{k-1}}}(a+kh)h + \cdots + \\
& + \tfrac{h^{m-1}}{(m-1)!}\overline{Y^{(m-1)}(a+kh)} + \tfrac{h^m}{m!}T, S'_{|_{I_{k-1}}}(a+kh) + \\
& + \overline{Y''(a+kh)}h + \cdots + \tfrac{h^{m-2}}{(m-2)!}\overline{Y^{(m-1)}(a+kh)} + \tfrac{h^{m-1}}{(m-1)!}T\Bigg) \\
& - \overline{Y''(a+kh)} - \overline{Y^{(3)}(a+kh)}h - \cdots - \tfrac{h^{m-3}}{(m-3)!}\overline{Y^{(m-1)}(a+kh)}\Bigg].
\end{aligned} \tag{5.2.17}
$$

Observe that relation (5.2.16) holds if and only if $A_k = g(A_k)$, which means that A_k is a fixed point for function $g(T)$. The definition (5.2.17) of g in combination with the global Lipschitz's condition (5.2.4) for f, implies immediately that

$$\|g(T_1) - g(T_2)\| \leq \left(\frac{L_1 h^2}{m(m-1)} + \frac{L_2 h}{m-1}\right) \|T_1 - T_2\| .$$

Selecting $h < \left(\sqrt{L_2^2 m^2 + 4m(m-1)L_1} - mL_2\right)/2L_1$ gives $\left(\frac{L_1 h^2}{m(m-1)} + \frac{L_2 h}{m-1}\right) < 1$ and the matrix function g is contractive. Therefore, equation (5.2.16) has the unique solutions A_k for each $k = 0, 1, \ldots, n-1$. Thus, the matrix spline is completely determined, which concludes this proof.∎

Remark 5.2. Observe that the constructed splines have a global error at least of order $O(h^{m-1})$, which follows from an analysis similar to Loscalzo and Talbot's work [1].

Remark 5.3. For carrying out the derivatives in (5.2.7) and (5.2.8), one may make extensive use of standard symbolic software, such as *Mathematica* etc.

3 Algorithms and MATLAB functions

For solving Equation (5.2.3), we consider the initial matrix value problem

$$\left.\begin{aligned} Y''(x) &= f(x, Y(x), Y'(x)) \\ Y(x_k) = Y_0 \ , \ Y'(x_k) &= Y_1 \end{aligned}\right\} \ x_k \leq x \leq x_k + h, \tag{5.3.18}$$

where h is the step size, and Y_0 and Y_1 are the values of Y and Y' obtained in the above step at x_k, respectively. If we denote by $S_k(x)$ the spline of order m in the interval $[x_k, x_k + h]$, then

$$\begin{aligned} S_k(x_k + h) &= B_k^{(0)} + \frac{h^m}{m!} A_k, \\ S_k'(x_k + h) &= B_k^{(1)} + \frac{h^{m-1}}{(m-1)!} A_k, \\ S_i''(x_k + h) &= B_k^{(2)} + \frac{h^{m-2}}{(m-2)!} A_k, \end{aligned}$$

where

$$B_k^{(0)} = \sum_{i=0}^{m-1} \frac{Y^{(i)}(x_k) h^i}{i!}, \ B_i^{(1)} = \sum_{i=1}^{m-1} \frac{Y^{(i)}(x_k) h^{i-1}}{(i-1)!}, \ B_i^{(2)} = \sum_{i=2}^{m-1} \frac{Y^{(i)}(x_k) h^{i-2}}{(i-2)!}.$$

If we substitute the above expressions in (5.3.18), we obtain

$$B_k^{(2)} + \frac{h^{m-2}}{(m-2)!} A_k = f\left(x, B_k^{(0)} + \frac{h^m}{m!} A_k, B_i^{(1)} + \frac{h^{m-1}}{(m-1)!} A_k\right). \tag{5.3.19}$$

Then matrix A_k can be obtained by solving (5.3.19).

The MATLAB function **splin2order** computes the solution of (5.2.3) by a fixed-point method or another method, as for example the Newton method (for online availability of the software, see Ref. [21]). The function used for solving (5.3.19) by the fixed-point method is

$$F(A_k) = \frac{(m-2)!}{h^{m-2}} \left[f\left(x, B_k^{(0)} + \frac{h^m}{m!} A_k, B_i^{(1)} + \frac{h^{m-1}}{(m-1)!} A_k\right) - B_k^{(2)} \right].$$

Hence, the values of Y and Y' at $x_k + h$ are

$$Y(x_k + h) = B_k^{(0)} + \frac{h^m}{m!} A_k,$$
$$Y_k'(x_k + h) = B_k^{(1)} + \frac{h^{m-1}}{(m-1)!} A_k.$$

The computer storage required for the MATLAB function implies the use of seven internal matrices. However, it can be further optimized for some special classes of second-order differential equations. Consider, for example, the following second-order linear differential system

$$Y''(t) + A_1 Y'(t) + A_0 Y(t) = 0, \quad t \in [a, b], \tag{5.3.20}$$

where $A_0, A_1 \in \mathbb{R}^{n \times n}$ are constant matrix coefficients. In this case, the MATLAB function which solves the above equation is called **splin2linear** (and available online [21]). If (5.3.20) is incomplete, *i.e.* $A_1 = 0$, the computational costs can be cut down considerably. We have also developed the MATLAB function **spline2lineari**, which is an optimized version of **spline2linear** for the incomplete differential linear equation $Y''(t) + A_0 Y(t) = 0$. In both cases, the successive derivatives are computed within the corresponding function. The memory requirements for these functions are eleven and ten matrices, respectively.

4 Numerical Examples

The goal of this section is to show the effectiveness of our method by testing MATLAB and symbolic implementations of it. We will use some standard benchmarks examples to compare its numerical estimates with those obtained from higher-order splines constructed by the method proposed in [11]. It is important to remark that our method is not only a viable alternative to existing approaches, but has been applied successfully in several other practical examples.

The MATLAB benchmark tests have been carried out on an *Intel Core 2 Duo T5600* with 2 GB main memory and using *Mathematica* version 7.0. and MATLAB version 7.9. We test the MATLAB implementations for each of our proposed spline methods with problems where the exact solution is known. The symbolic implementations employ the *Symbolic Math Toolbox* of MATLAB for calculating derivatives of functions, such that the derivatives are provided by a function which calculates their values. In particular, these new implementations based on our proposed methods have been compared with the results produced by MATLAB solver functions with *RelTol* and *AbsTol* parameters equal to 10^{-14}. Table 5.1 lists all MATLAB functions used in these tests. The functions are based on adaptive methods to solve non-stiff, ordinary differential equations of the type

$$y' = f(x, y),\ x \in [a, b],$$
$$y(a) = y_0,$$

where $y \in \mathbb{R}^r$.

MATLAB SOLVER	PROBLEM	METHOD
ode45	non-stiff differential equations	Runge-Kutta
ode23	non-stiff differential equations	Runge-Kutta
ode113	non-stiff differential equations	Adams

Table 5.1. MATLAB solvers used in the tests.

We have also implemented specific MATLAB functions in order to solve Eq. (5.2.3) with fixed step size based on classical methods: Nyström of order 4 [22, p. 284], an extrapolation method [22, p. 294] and an explicit Störmer method [22, p. 462], but we only provide the results of the first one, because it gives much better results than the others. The implemented function based on this method is named **nystrom2order**.

4.1 A non-linear vector system

Consider the following non-linear vector differential system

$$\left.\begin{array}{l} y_1''(x) = 1 - \cos(x) + \sin\left(y_2'(x)\right) + \cos\left(y_2'(x)\right) \\ y_2''(x) = \dfrac{1}{4 + y_1(x)^2} - \dfrac{1}{5 - \sin^2(x)} \\ y_1(0) = 1, \qquad y_2(0) = 0, \\ y_1'(0) = 0, \qquad y_2'(0) = \pi \end{array}\right\} \; 0 \le x \le 1. \tag{5.4.21}$$

with exact solution $y_1(x) = \cos(x), y_2(x) = \pi x$. The system Eq. (5.4.21) can be written in the more compact form

$$Y''(x) = F(x, Y, Y'), \quad Y(x) = \begin{pmatrix} y_1(x) \\ y_2(x) \end{pmatrix}, \quad Y(0) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad Y'(0) = \begin{pmatrix} 0 \\ \pi \end{pmatrix}, \tag{5.4.22}$$

where

$$F(x, Y, Y') = \begin{pmatrix} 1 - \cos(x) + \sin\left(y_2'(x)\right) + \cos\left(y_2'(x)\right) \\ \dfrac{1}{4 + y_1(x)^2} - \dfrac{1}{5 - \sin^2(x)} \end{pmatrix}. \tag{5.4.23}$$

It is easy to check that $F(x, Y, Y')$ satisfies the global Lipschitz conditions:

$$\left.\begin{array}{l} \|F(x, Y_1, Y) - F(x, Y_2, Y)\| \le \|Y_1 - Y_2\| \\ \|F(x, Y, Y_1) - F(x, Y, Y_2)\| \le 2\|Y_1 - Y_2\| \end{array}\right\}, \; 0 \le x \le 1, Y, Y_1, Y_2 \in \mathbb{R}^2. \tag{5.4.24}$$

It is not difficult to evaluate $Y''(0)$ using (5.4.22), thus $Y''(0) = F(0, Y(0), Y'(0)) = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$. We can calculate $Y^{(3)}(0)$ using (5.2.7). This will illustrate how the calculation of (5.2.7) can be done. In this case, one gets

$$vec\, Y(x) = Y(x), vec\, Y'(x) = \begin{pmatrix} y_1'(x) \\ y_2'(x) \end{pmatrix},$$

$$\frac{\partial F(x, Y(x), Y'(x))}{\partial x} = \begin{pmatrix} \sin(x) \\ \dfrac{-2\sin(x)\cos(x)}{\left(5 - \sin^2(x)\right)^2} \end{pmatrix}.$$

On the other hand, we have

$$\frac{\partial F(x,Y(x),Y'(x))}{\partial\,(vec\;Y(x))}=\begin{pmatrix}\dfrac{\partial F(x,Y(x),Y'(x))}{\partial y_1(x)}\\ \hline \dfrac{\partial F(x,Y(x),Y'(x))}{\partial y_2(x)}\end{pmatrix}=\begin{pmatrix}0\\ \dfrac{-2y_1(x)}{\left(4+y_1^2(x)\right)^2}\\ \hline 0\\ 0\end{pmatrix},$$

$$\frac{\partial F(x,Y(x),Y'(x))}{\partial\,(vec\;Y'(x))}=\begin{pmatrix}\dfrac{\partial F(x,Y(x),Y'(x))}{\partial y_1'(x)}\\ \hline \dfrac{\partial F(x,Y(x),Y'(x))}{\partial y_2'(x)}\end{pmatrix}=\begin{pmatrix}0\\ 0\\ \hline \cos\left(y_2'(x)\right)-\sin\left(y_2'(x)\right)\\ 0\end{pmatrix},$$

and, if we denote by $A=1-\cos(x)+\sin\left(y_2'(x)\right)+\cos\left(y_2'(x)\right)$ and $B=1/\left(4+y_1(x)^2\right)-1/\left(5-\sin^2(x)\right)$, one obtains

$$[vec\;Y'(x)]^T\otimes I_2=\left(\begin{array}{cc|cc}y_1'(x)&0&y_2'(x)&0\\0&y_1'(x)&0&y_2'(x)\end{array}\right),$$

$$\left[vec\;f(x,Y(x),Y'(x))\right]^T\otimes I_2=\left(\begin{array}{cc|cc}A&0&B&0\\0&A&0&B\end{array}\right).$$

Thus, using (5.2.7) we have

$$\begin{aligned}Y^{(3)}(x)&=\begin{pmatrix}\sin(x)\\ \frac{-2\sin(x)\cos(x)}{\left(5-\sin^2(x)\right)^2}\end{pmatrix}+\begin{pmatrix}y_1'(x)&0&y_2'(x)&0\\0&y_1'(x)&0&y_2'(x)\end{pmatrix}\begin{pmatrix}0\\ \frac{-2y_1(x)}{\left(4+y_1^2(x)\right)^2}\\0\\0\end{pmatrix}\\&+\begin{pmatrix}A&0&B&0\\0&A&0&B\end{pmatrix}\begin{pmatrix}0\\0\\ \cos\left(y_2'(x)\right)-\sin\left(y_2'(x)\right)\\0\end{pmatrix}\\&=\begin{pmatrix}\sin(x)+\left(\cos\left(y_2'(x)\right)-\sin\left(y_2'(x)\right)\right)\left(\frac{1}{4+y_1^2(x)}-\frac{1}{5-\sin^2(x)}\right)\\ \frac{-2\sin(x)\cos(x)}{\left(5-\sin^2(x)\right)^2}-\frac{2y_1(x)y_1'(x)}{\left(4+y_1^2(x)\right)^2}\end{pmatrix}.\end{aligned}\tag{5.4.25}$$

Taking into account that $y_1(0)=1, y_2(0)=0, y_1'(0)=0, y_2'(0)=\pi$, and evaluating $Y^{(3)}(x)$ when $x=0$ from (5.4.25), finally one concludes that $Y^{(3)}(0)=\begin{pmatrix}0\\0\end{pmatrix}$.

If we consider matrix splines of order $m=6$, Theorem 5.1 requires to take $h<2.12404$, and thus we select $h=0.1$. In Table 5.2, we provide the numerical estimates, which have been rounded to the fourth relevant digit. We also give the Frobenius norm of the difference between the estimates of our numerical approach and the exact solution. Their maximum errors are shown also in the first column. Note that as we increase the order of the spline using the same technique as in [11], we obtain the results given in Table 5.3, where in the last two intervals the maximum error increases dramatically.

interval error	Spline
$[0,0.1]$ 2.14828×10^{-13}	$\begin{pmatrix} 1.-0.5x^2+0.0417x^4-0.0014x^6 \\ 3.1416x \end{pmatrix}$
$[0.1,0.2]$ 2.01417×10^{-12}	$\begin{pmatrix} 1.-0.5x^2+0.0417x^4-0.0014x^6 \\ 3.1416x \end{pmatrix}$
$[0.2,0.3]$ 8.15548×10^{-12}	$\begin{pmatrix} 1.-0.5x^2+0.0417x^4-0.0014x^6 \\ 3.1416x \end{pmatrix}$
$[0.3,0.4]$ 2.13535×10^{-11}	$\begin{pmatrix} 1.-0.5x^2+0.0417x^4-0.0001x^5-0.0013x^6 \\ 3.1416x \end{pmatrix}$
$[0.4,0.5]$ 4.42526×10^{-11}	$\begin{pmatrix} 1.-0.5x^2-0.0001x^3+0.0418x^4-0.0002x^5-0.0013x^6 \\ 3.1416x \end{pmatrix}$
$[0.5,0.6]$ 7.94035×10^{-11}	$\begin{pmatrix} 1.-0.4999x^2-0.0002x^3+0.042x^4-0.0004x^5-0.0012x^6 \\ 3.1416x \end{pmatrix}$
$[0.6,0.7]$ 1.29235×10^{-10}	$\begin{pmatrix} 1.-0.4998x^2-0.0005x^3+0.0424x^4-0.0006x^5-0.0011x^6 \\ 3.1416x \end{pmatrix}$
$[0.7,0.8]$ 1.96032×10^{-10}	$\begin{pmatrix} 1.-0.0001x-0.4996x^2-0.001x^3+0.043x^4-0.001x^5-0.0010x^6 \\ 3.1416x \end{pmatrix}$
$[0.8,0.9]$ 2.81915×10^{-10}	$\begin{pmatrix} 1.-0.0003x-0.4990x^2-0.0019x^3+0.0438x^4-0.0014x^5-0.0009x^6 \\ 3.1416x \end{pmatrix}$
$[0.9,1]$ 3.88818×10^{-10}	$\begin{pmatrix} 1.0001-0.0006x-0.4981x^2-0.0033x^3+0.0451x^4-0.002x^5-0.0008x^6 \\ 3.1416x \end{pmatrix}$

Table 5.2. Approximation for the test problem of Subsection 4.1 in the interval $[0,1]$ with step size $h = 0.1$ and matrix splines of order $m = 6$.

interval	$[0,0.1]$	$[0.1,0.2]$	$[0.2,0.3]$	$[0.3,0.4]$
max. error	2.14828×10^{-13}	1.16367×10^{-11}	7.7013×10^{-11}	8.46736×10^{-10}
interval	$[0.4,0.5]$	$[0.5,0.6]$	$[0.6,0.7]$	$[0.7,0.8]$
max. error	8.23407×10^{-9}	8.17369×10^{-8}	8.08793×10^{-7}	8.0067×10^{-6}
interval	$[0.8,0.9]$	$[0.9,1]$		
max. error	0.0000792582	0.0014899		

Table 5.3. Maximum 2-norm error using method [11] with $m = 6$ for the the test problem of Subsection 4.1.

For another test, the MATLAB function **splin2order** was used. Figure 5.1 displays the approximation behavior for splines of order $m = 6$ with different step sizes $h = 0.1$, $h = 0.01$ and $h = 0.001$, respectively. Table 5.4 compares **splin2order** (spline of order $m = 9$) with the other MATLAB functions, taking $h = 0.1$ and $b = 5$. The second column indicates the execution time in seconds and the third column the relative errors. For the fixed-step codes, we have chosen the optimal step size, *i.e.* the step size giving the lowest error possible with the best execution time.

4.2 Linear second-order differential matrix equations

Linear second-order differential matrix equations present another interesting testing ground for efficient matrix spline algorithms. In fact, it is possible to develop efficient algorithms for solving this type of problems. The MATLAB function **spline2linear** is an implementation of the method described in the previous section.

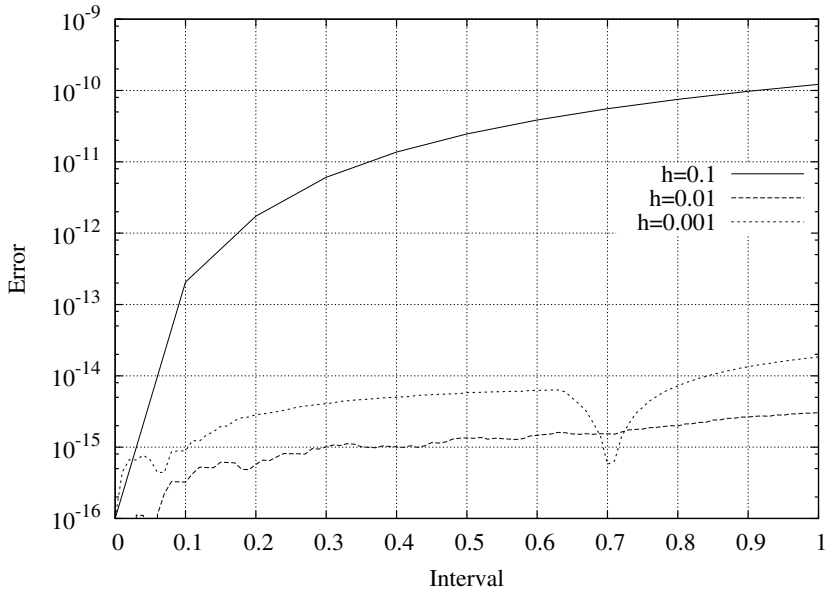

Fig. 5.1. Relative errors for the test problem of Subsection 4.1 with fourth-order splines ($m = 6$), using MATLAB function **splin2order** with $h = 0.1$, $h = 0.01$ and $h = 0.001$, respectively.

MATLAB function	Time [s]	Error
splin2order $(m = 9, h = 0.1)$	0.091408	$3.457835 \cdot 10^{-16}$
nystrom2order $(h = 2 \cdot 10^{-3})$	0.246637	$3.622907 \cdot 10^{-14}$
ode45	0.090288	$1.289173 \cdot 10^{-15}$
ode23	6.683251	$2.280581 \cdot 10^{-15}$
ode113	0.014599	$9.203065 \cdot 10^{-16}$

Table 5.4. Relative errors for the test problem of Subsection 4.1 for $b = 5$.

We consider the problem (5.3.20) with the following matrix coefficients

$$A_1 = \begin{pmatrix} -1 & 1 \\ 0 & -2 \end{pmatrix}, A_0 = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix},$$

and the initial conditions

$$Y(0) = Y'(0) = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

The analytical solution is known to be

$$Y(t) = \begin{pmatrix} e^t & -1 + e^t - e^t t \\ 0 & e^t \end{pmatrix}, \quad t \in [0, b]. \tag{5.4.26}$$

In this case, it is $f(t, Y, Z) = -A_0 Y - A_1 Z$, and therefore

$$\left.\begin{aligned} \left\| f(t, Y_1, Z) - f(t, Y_2, Z) \right\|_2 &\leq \|Y_1 - Y_2\|_2 \\ \left\| f(t, Y, Z_1) - f(t, Y, Z_2) \right\|_2 &\leq 2.28825 \, \|Z_1 - Z_2\|_2 \end{aligned}\right\}. \tag{5.4.27}$$

If we consider again matrix splines of order $m = 6$, we must take the step size given by the constraint $h < \left(\sqrt{36L_2^2 + 120L_1} - 6L_2\right)/2L_1 = 1.91732$. In this case, the errors are increasing

up to a value of $[0, 1.77112 \times 10^{-8}]$, but they remain well within the error bounds fixed by Theorem 5.1. If we compare these results with those given by a sixth-order spline using the same technique as in Ref. [11], we obtain comparable results for the first five intervals. However, our method truly improves the results in the last five intervals. Table 5.5 shows the results obtained in the last four intervals with the method from Ref. [11].

interval	$[0.6, 0.7]$	$[0.7, 0.8]$	$[0.8, 0.9]$	$[0.9, 1]$
max. error	0.00014554	0.00140688	0.0136004	0.131485

Table 5.5. Maximum 2-norm error using method [11] with $m = 6$ for the test problem of Subsection 4.2 .

Figure 5.2 depicts the approximation behavior for splines of order $m = 6$ with different step sizes $h = 0.1$, $h = 0.01$ and $h = 0.001$, respectively. Table 5.6 shows the results of **spline2linear** (spline of order $m = 10$) and the other MATLAB functions for $h = 0.1$ and $b = 5$. The second column gives the execution time in seconds and the third column the relative errors.

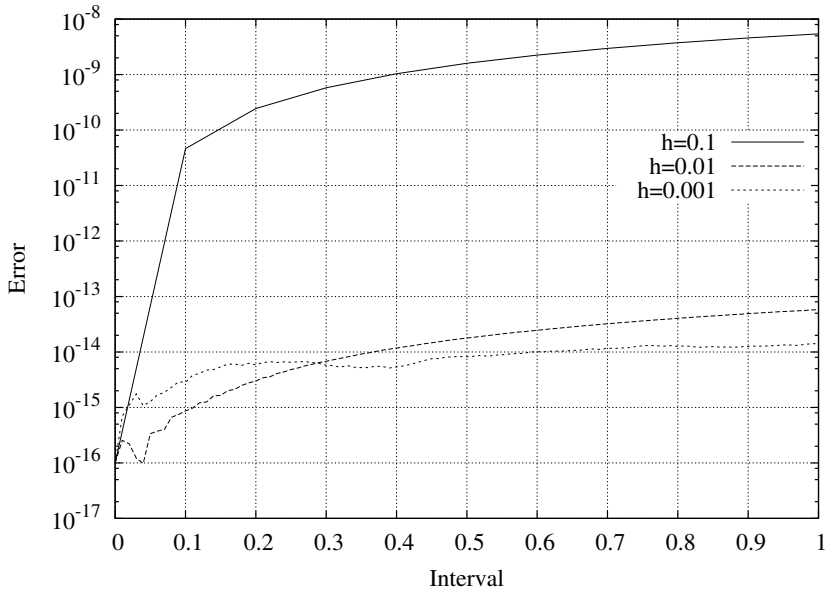

Fig. 5.2. Relative errors for the test problem of Subsection 4.2 with fifth-order splines ($m = 6$), using MATLAB function with **splin2linear** $h = 0.1$, $h = 0.01$ and $h = 0.001$, respectively.

Method	Time [s]	Error
splin2linear $(m = 10, h = 0.1)$	0.003051	$5.320190 \cdot 10^{-15}$
nystrom2order $(h = 5 \cdot 10^{-4})$	0.246637	$2.891663 \cdot 10^{-14}$
ode45	0.175877	$6.800561 \cdot 10^{-15}$
ode23	9.808279	$5.398916 \cdot 10^{-14}$
ode113	0.017892	$1.101141 \cdot 10^{-14}$

Table 5.6. Relative errors for the test problem of Subsection 4.2 for $b = 5$.

4.3 Incomplete linear second-order differential matrix equations

As a second example, we study

$$Y''(t) + A\,Y(t) = 0, \tag{5.4.28}$$

where

$$A = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}, \quad Y(0) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \quad \text{and} \quad Y'(0) = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \quad t \in [0, b]. \tag{5.4.29}$$

with the known analytical solution

$$Y(t) = \cos\left(\sqrt{A}t\right) Y(0) + \left(\sqrt{A}\right)^{-1} \sin\left(\sqrt{A}t\right) Y'(0). \tag{5.4.30}$$

As usual, $\sqrt{A}$ denotes the square root of the non-singular matrix A (see [23]). In this example, we choose $L_2 = 0$ and $L_1 \approx 2.82843$ as suggested in [11]. If we consider $b = 1$ and matrix splines of order $m = 6$, according to Theorem 5.1, we need to take $h < 3.25678$ as in Ref. [24], so we selected $h = 0.1$. Table 5.7 provides all numerical results (rounded to the third relevant digit) for the maximum error in each interval.

interval	$[0, 0.1]$	$[0.1, 0.2]$	$[0.2, 0.3]$	$[0.3, 0.4]$
max. error	5.66188×10^{-11}	3.09994×10^{-10}	7.54205×10^{-10}	1.37841×10^{-9}
interval	$[0.4, 0.5]$	$[0.5, 0.6]$	$[0.6, 0.7]$	$[0.7, 0.8]$
max. error	2.16706×10^{-9}	3.10015×10^{-9}	4.15361×10^{-9}	5.29975×10^{-9}
interval	$[0.8, 0.9]$	$[0.9, 1]$		
max. error	6.50774×10^{-9}	7.74422×10^{-9}		

Table 5.7. Maximum approximation error for the test problem of Subsection 4.2 in the interval $[0, 1]$ with step size $h = 0.1$ and splines of order $m = 6$.

If we consider high-order splines, for example with $m = 6$, and if we use the same technique as in [11], we obtain essentially the same results for the error in the first six intervals. For the last four intervals, the error increases considerably (see Table 5.8), showing a bad performance for this method.

interval	$[0.6, 0.7]$	$[0.7, 0.8]$	$[0.8, 0.9]$	$[0.9, 1]$
max. error	0.000189424	0.00187658	0.0185907	0.184173

Table 5.8. Maximum 2-norm error using method [11] for $m = 6$ for the incomplete second-order differential system Eq. (5.4.28).

In the following test we consider $b = 5$. Figure 5.3 depicts the approximation behavior for splines of six-order with different step sizes $h = 0.1$, $h = 0.01$ and $h = 0.001$, respectively. Table 5.9 shows the results of function **splin2lineari** (spline of order $m = 10$) compared to the results produced by the other functions, taking $h = 0.1$. The second column indicates the execution time in seconds and the third column the relative errors.

5 Conclusions

One goal of this chapter was to presented a generalized method for the numerical treatment of second order differential matrix systems. Our approach is a generalization of previously developed methods employing matrix-cubic splines. A second goal was to provide techniques to get reliable algorithms that are straightforward to implement.

All spline solutions are by construction already continuous in the interval under consideration. Several benchmark examples have been used to test our proposed method producing

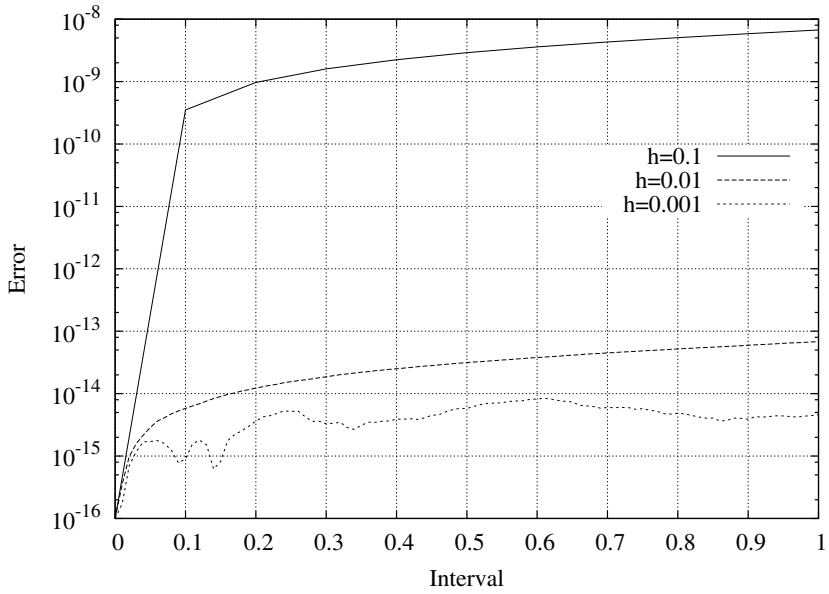

Fig. 5.3. Relative errors for the test problem of Subsection 4.3 with fourth-order splines ($m = 6$) , using MATLAB function **splin2lineari** with $h = 0.1$, $h = 0.01$ and $h = 0.001$, respectively.

MATLAB function	Time [s]	Error
splin2lineari ($m = 10$)	0.002224	$7.707535 \cdot 10^{-15}$
nystrom2order ($h = 1 \cdot 10^{-3}$)	0.502466	$5.538301 \cdot 10^{-14}$
ode45	0.197101	$1.836483 \cdot 10^{-14}$
ode23	11.563673	$7.383291 \cdot 10^{-14}$
ode113	0.019096	$2.026853 \cdot 10^{-15}$

Table 5.9. Relative errors for the test problem of Subsection 4.3.

errors only of the order $O(h^{m-1})$, where m is the order of the spline. By adjusting the step size h and selecting a corresponding higher-order spline, one can achieve a reliable estimate with high accuracy for any practical problem. Our method also makes clear the great precaution which is required when approximating second-order models in realistic applications.

Our method is well-suited for implementation on numerical and/or symbolical computer systems (*Mathematica*, MATLAB, etc). Three MATLAB implementations have been developed based on the spline method developed in this chapter. In order to assert the advantages of these implementations, extensive stress tests were made in three case studies by comparing these implementations under equal conditions with built-into MATLAB functions and an implementation of a Nyström method of fourth order. The results clearly demonstrated that the relative errors of our MATLAB implementations are lower than relative errors of the other, standard functions, and have in general lower execution times.

References

1. F. R. Loscalzo and T. D. Talbot, *Spline function approximations for solutions of ordinary differential equations*, SIAM J. Numer. Anal. **4** (1967), no. 3, 433–445.
2. E. A. Al-Said, *The use of cubic splines in the numerical solution of a system of second-order boundary value problems*, Comput. Math. Appl. **42** (2001), 861–869.
3. M. K. Kadalbajoo and K. C. Patidar, *Numerical solution of singularly perturbed two-point boundary value problems by spline in tension*, Appl. Math. Comput. **131** (2002), 299–320.
4. E. A. Al-Said and M. A. Noor, *Cubic splines method for a system of third-order boundary value problems*, Appl. Math. Comput. **142** (2003), 195–204.

5. G. Micula and A. Revnic, *An implicit numerical spline method for systems for ode's*, Appl. Math. Comput. **111** (2000), 121–132.
6. E. Defez, L. Soler, A. Hervás, and C. Santamaría, *Numerical solutions of matrix differential models using cubic matrix splines*, Comput. Math. Appl. **50** (2005), 693–699.
7. E. Defez, L. Soler, A. Hervás, and M. M. Tung, *Numerical solutions of matrix differential models using cubic matrix splines II*, Mathematical and Computer Modelling **46** (2007), 657–669.
8. Hermann Brunner, *On the divergence of collocation solutions in smooth piecewise polynomial spaces for volterra integral equations*, BIT Numerical Mathematics **44** (2004), 631–650.
9. G. Micula, *Approximate solutions of the differential equation $y'' = f(x, y)$ with spline functions*, Math. Comp. **27** (1973), 807–816.
10. Uri Ascher, S. Pruess, and Robert D. Russell, *On spline basis selection for solving differential equations*, SIAM J. Numer. Anal. **20** (1983), 121–142.
11. M. M. Tung, E. Defez, and J. Sastre, *Numerical solutions of second-order matrix models using cubic-matrix splines*, Comput. Math. Appl. **56** (2008), 2561–2571.
12. Emilio Defez, Michael M. Tung, Javier Ibáñez, and Jorge Sastre, *Approximating and computing nonlinear matrix differential models*, Math. Comput. Model. **55** (2012), no. 7–8, 2012–2022.
13. A. Graham, *Kronecker products and matrix calculus with applications*, John Wiley & Sons, New York, USA, 1981.
14. P. Marzulli, *Global error estimates for the standard parallel shooting method*, J. Comput. Appl. Math. **34** (1991), 233–241.
15. J. M. Ortega, *Numerical analysis: A second course*, Academic Press, New York, USA, 1972.
16. B. W. Shore, *Comparison of matrix methods to the radii Schrödinger eigenvalue equation: The Morse potential*, J. Chemical Physics **59** (1971), no. 12, 6450–6463.
17. C. Froese, *Numerical solutions of the hartree-fock equations*, Can. J. Phys. **41** (1963), 1895–1910.
18. J. R. Claeyssen, G. Canahualpa, and C. Jung, *A direct approach to second-order matrix non-classical vibrating equations*, Appl. Numer. Math. **30** (1999), 65–78.
19. J. F. Zhang, *Optimal control for mechanical vibration systems based on second-order matrix equations*, Mechanical Systems and Signal Processing **16** (2002), no. 1, 61–67.
20. T. M. Flett, *Differential analysis*, Cambridge University Press, New York, USA, 1980.
21. J. J. Ibáñez, MATLAB *Implementation for Matrix Splines (*MIMS*).*
22. E. Hairer, S. P. Nörsett, and G. Wanner, *Solving ordinary differential equations i: Nonstiff problems*, Springer-Verlag, Berlin, Germany, 2008.
23. G. H. Golub and C. F. Van Loan, *Matrix computations*, second ed., The Johns Hopkins University Press, Baltimore, MD, USA, 1989.
24. M. M. Tung, L. Soler, E. Defez, and A. Hervás, *Cubic-matrix splines and second-order matrix model*, The 14th European Conference on Mathematics for Industry (ECMI 2006) (Universidad Carlos III de Madrid, Spain), 2006.

APPENDIX

MATLAB Code

MATLAB [1–5] and its open-source companion Octave [6–9] are the most suitable numerical computing environments to carry out the matrix calculations at hand. Both are high-level interpreted 4GL language, primarily intended for numerical chores and strongly based on data in matrix format. Whereas MATLAB is commercially available since 1984 and maintained by MathWorks [1], Octave [7] is freely distributed under the GNU public license and maintained by a group of dedicated programmers and users. Octave may be used directly via an interactive command-line, which makes it also ideal for lengthier calculations on a UNIX based system by the usage of scripts [9]. Furthermore, most MATLAB code is readily available in Octave as well, due to its high level of compatibility.

In the following, we give a concise exemplary selection of various commented MATLAB code used for some of the computation and numerical data acquisition in the previous chapters.

The MATLAB code for second-order differential matrix equation $Y'' = f(x, Y(x), Y'(x))$ within the interval $[a,b]$ and with boundary conditions $Y(a) = Y_0$ and $Y'(a) = Y_1$. As usual h gives the step size and m represents the order of the spline.

Program 1: splin2linear.m

```
function [Y0,x]=splin2linear(A0,A1,Y0,Y1,a,b,h,m)
% This function compute the solution of a second order linear
% differential matriz equation y''+A0*Y+A1*Y'=0 at [a,b]
%                              Y(a)=Y0, Y'(a)=Y1
%Inputs:
%        A0 and A1 are the matrix coeficients
%   Y0 and Y1 are the initial condition matrices
%   [a,b] is the integration interval
%   h is the step size
%   m is the order of spline
%Outputs:
% Y0 and x: Y0 contains the solution at x (a<=x<a+h)
%
```

```
ph(1)=h;
fac=[1 2 6 24 120 720 5040 40320 362880 3628800];%factorials
for k=2:m%Compute the powers of h
    ph(k)=ph(k-1)*h;
end
n=size(A0,1);
I=eye(n);
nt=round((b-a)/h);%number of integrate subintervals
x=a;
for i=1:nt
    Aux0=Y0;
    Aux1=Y1;
    Aux2=-A1*Aux1-A0*Aux0;
    B0=Y0+Aux1*ph(1)+Aux2*ph(2)/fac(2);
    B1=Aux1+Aux2*ph(1);
    B2=Aux2;
    for k=2:m-2
        Aux0=Aux1;
        Aux1=Aux2;
        Aux2=-A1*Aux1-A0*Aux0;
        B0=B0+Aux2*ph(k+1)/fac(k+1);
        B1=B1+Aux2*ph(k)/fac(k);
        B2=B2+Aux2*ph(k-1)/fac(k-1);
    end
    x=x+h;
    X=(m*(m-1)*ph(m-2)*I+m*A1*ph(m-1)+A0*ph(m))\(-B2-A1*B1-A0*B0);
    Y0=B0+X*ph(m);
    Y1=B1+m*X*ph(m-1);
end
```

Program 1: splin2linear.m

Program 2: prsplin_second_order.m

```
function prsplin_second_order(i,m,b,h0)
%prsplin_second_order(1,9,5,0.1)
%prsplin_second_order(2,10,5,0.1)
%prsplin_second_order(3,10,5,0.1)
options=odeset('RelTol',2.22045e-14,'AbsTol',1e-14);

for k=1:20
    rand(1000)*rand(1000);
end
if i==1
        f='df_ej1';
    y0=[1;0];y1=[0;pi];a=0;%Ejemplo 3.1
    z0=[y0(:);y1(:)];

    solver='solver_g';
```

```
        tic
    [y,x]=splin2order(solver,f,y0,y1,a,b,h0,m);
    t0=toc;
    ys=ej1s(x);
    er0=norm(ys-y)/norm(ys);

    ej='ej1';
    fprintf('Spline General: Error=%e\tTiempo=%f\n',er0,t0)

elseif i==2
        F='df_ej2';
    [df,A0,A1]= feval(F,1,1,1,1);
    y0=eye(2);y1=y0;a=0;%Ejemplo 3.2

    tic
    [Y,x]=splin2linear(A0,A1,y0,y1,a,b,h0,m);
    t0=toc;
        ys=ej2s(x);
    er0=norm(ys-Y)/norm(ys);
        fprintf('Spline Lineal2: Error=%e\tTiempo=%f\n',er0,t0)


        z0=[y0(:);y1(:)];
    ej='ej2';

else
        F='df_ej3';
    [df,A0]= feval(F,1,1,1,1);
        y0=zeros(2);y1=[1 0;1 1];a=0; %Ejemplo 3.3

    tic
    [Y,x]=splin2lineari(A0,y0,y1,a,b,h0,m);
        t0=toc;
    ys=ej3s(x);
    er0=norm(ys-Y)/norm(ys);


    z0=[y0(:);y1(:)];
    ej='ej3';
    fprintf('Spline Lineal: Error=%e\tTiempo=%f\n',er0,t0)
end

if i==1
    tic
    [x,z]=ode45(ej,[0 b],z0,options);
    t2=toc;
    l=length(z);
    Z=reshape(z(l,1:2)',2,1);
    ys=ej1s(x(end));
    er2=norm(ys-Z)/norm(ys);
```

```
fprintf('ode45: Error=%e\tTiempo=%f\n',er2,t2)

tic
[x,z]=ode23(ej,[0 b],z0,options);
t2=toc;
l=length(z);
Z=reshape(z(l,1:2)',2,1);
ys=ej1s(x(end));
er2=norm(ys-Z)/norm(ys);
fprintf('ode23: Error=%e\tTiempo=%f\n',er2,t2)

tic
[x,z]=ode113(ej,[0 b],z0,options);
t2=toc;
l=length(z);
Z=reshape(z(l,1:2)',2,1);
ys=ej1s(x(end));
er2=norm(ys-Z)/norm(ys);
fprintf('ode113: Error=%e\tTiempo=%f\n',er2,t2)

tic
[x,z]=ode15s(ej,[0 b],z0,options);
t2=toc;
l=length(z);
Z=reshape(z(l,1:2)',2,1);
ys=ej1s(x(end));
er2=norm(ys-Z)/norm(ys);
fprintf('ode15s: Error=%e\tTiempo=%f\n',er2,t2)

tic
[x,z]=ode23s(ej,[0 b],z0,options);
t2=toc;
l=length(z);
Z=reshape(z(l,1:2)',2,1);
ys=ej1s(x(end));
er2=norm(ys-Z)/norm(ys);
fprintf('ode23s: Error=%e\tTiempo=%f\n',er2,t2)

tic
[x,z]=ode23t(ej,[0 b],z0,options);
t2=toc;
l=length(z);
Z=reshape(z(l,1:2)',2,1);
ys=ej1s(x(end));
er2=norm(ys-Z)/norm(ys);
fprintf('ode23t: Error=%e\tTiempo=%f\n',er2,t2)

tic
[x,z]=ode23tb(ej,[0 b],z0,options);
t2=toc;
```

```
    l=length(z);
    Z=reshape(z(l,1:2)',2,1);
    ys=ej1s(x(end));
    er2=norm(ys-Z)/norm(ys);
    fprintf('ode23tb: Error=%e\tTiempo=%f\n',er2,t2)
else
    tic
    [x,z]=ode45(ej,[0 b],z0,options);
    t2=toc;
    l=length(z);
    Z=reshape(z(l,1:4)',2,2);
    er2=norm(ys-Z)/norm(ys);
    fprintf('ode45: Error=%e\tTiempo=%f\n',er2,t2)

    tic
    [x,z]=ode23(ej,[0 b],z0,options);
    t2=toc;
    l=length(z);
    Z=reshape(z(l,1:4)',2,2);
    er2=norm(ys-Z)/norm(ys);
    fprintf('ode23: Error=%e\tTiempo=%f\n',er2,t2)

    tic
    [x,z]=ode113(ej,[0 b],z0,options);
    t2=toc;
    l=length(z);
    Z=reshape(z(l,1:4)',2,2);
    er2=norm(ys-Z)/norm(ys);
    fprintf('ode113: Error=%e\tTiempo=%f\n',er2,t2)

    tic
    [x,z]=ode15s(ej,[0 b],z0,options);
    t2=toc;
    l=length(z);
    Z=reshape(z(l,1:4)',2,2);
    er2=norm(ys-Z)/norm(ys);
    fprintf('ode15s: Error=%e\tTiempo=%f\n',er2,t2)

    tic
    [x,z]=ode23s(ej,[0 b],z0,options);
    t2=toc;
    l=length(z);
    Z=reshape(z(l,1:4)',2,2);
    er2=norm(ys-Z)/norm(ys);
    fprintf('ode23s: Error=%e\tTiempo=%f\n',er2,t2)

    tic
    [x,z]=ode23t(ej,[0 b],z0,options);
    t2=toc;
    l=length(z);
```

```
    Z=reshape(z(l,1:4)',2,2);
    er2=norm(ys-Z)/norm(ys);
    fprintf('ode23t: Error=%e\tTiempo=%f\n',er2,t2)

    tic
    [x,z]=ode23tb(ej,[0 b],z0,options);
    t2=toc;
    l=length(z);
    Z=reshape(z(l,1:4)',2,2);
    er2=norm(ys-Z)/norm(ys);
    fprintf('ode23tb: Error=%e\tTiempo=%f\n',er2,t2)
end
```

Program 2: prsplin_second_order.m

Program 3: splin2orderm.m

```
function [Y0,x]=splin2orderm(solver,derf,Y0,Y1,a,b,h,m)
% This function compute the solution of a second order
% differential matrix equation y''=f(x,y,y'), at [a,b]
%                              Y(a)=Y0, Y'(a)=Y1
%Inputs:
%        solver is a MATLAB function which solves Equation (21)
%   derf is a MATLAB function which allows to compute the function f and
%       its derivatives
%       problem and their derivatives
%   Y0 and Y1 are the initial condition matrices
%   [a,b] is the integration interval
%   h is the step size
%   m is the order of spline
%Outputs:
% Y0 and x: Y0 contains the solution at x (a<=x<a+h)


fac=[1 2 6 24 120 720 5040 40320 362880 3628800];%factorials
ph(1)=h;
for k=2:m%Compute the powers of h
    ph(k)=ph(k-1)*h;
end
nt=round((b-a)/h);%number of integrate subintervals
x=a;
Y2=feval(derf,1,x,Y0,Y1);
for i=1:nt
    %Computes B_i^(0), B_i^(1) and B_i^(2)
    B0=Y0+Y1*ph(1)+Y2*ph(2)/fac(2);
    B1=Y1+Y2*ph(1);
    B2=Y2;
    for k=2:m-2
        Aux=feval(derf,k,x,Y0,Y1);
```

```
        B0=B0+Aux*ph(k+1)/fac(k+1);
        B1=B1+Aux*ph(k)/fac(k);
        B2=B2+Aux*ph(k-1)/fac(k-1);
    end
    x=x+h;
    X=feval(solver,derf,x,B0,B1,B2,m,ph);%Solve the equation X=F(X)
    %Compute of values of y, y' and y'' at x=x+h
    Y0=B0+X*ph(m);
    Y1=B1+m*X*ph(m-1);
    Y2=B2+m*(m-1)*X*ph(m-2);
end
```

Program 3: splin2orderm.m

Program 4: prsplin_second_order_lineal_gnuplot.m

```
function prsplin_second_order_lineal_gnuplot(m)


%Ejemplo 1
        f='df_ej1';
    y0=[1;0];y1=[0;pi];a=0;%Ejemplo 3.1
    solver='solver_g';

        file_id= fopen('spline6_ej1_0.1','w');
    xb=0;h=0.1;
    fprintf(file_id,'0 1e-16\n');
    for i=1:10
        xb=xb+h;
        Ya=splin2order(solver,f,y0,y1,a,xb,h,m);
        Ys=ej1s(xb);
        norm_Ys=norm(Ys);
        er=norm(Ys-Ya)/norm_Ys;
        fprintf(file_id,'%f %e\n',xb,er);
    end
    fclose(file_id);

    file_id= fopen('spline6_ej1_0.01','w');
    xb=0;h=0.01;
    fprintf(file_id,'0 1e-16\n');
    for i=1:100
        xb=xb+h;
        Ya=splin2order(solver,f,y0,y1,a,xb,h,m);
        Ys=ej1s(xb);
        norm_Ys=norm(Ys);
        er=norm(Ys-Ya)/norm_Ys;
        fprintf(file_id,'%f %e\n',xb,er);
    end
```

```
    fclose(file_id);

    file_id= fopen('spline6_ej1_0.001','w');
    xb=0;h=0.001;
    fprintf(file_id,'0 1e-16\n');
    for i=1:100
        xb=xb+10*h;
        Ya=splin2order(solver,f,y0,y1,a,xb,h,m);
        Ys=ej1s(xb);
        norm_Ys=norm(Ys);
        er=norm(Ys-Ya)/norm_Ys;
        fprintf(file_id,'%f %e\n',xb,er);
    end
    fclose(file_id);

%Ejemplo 2
    A0=[0 0;0 1];A1=[-1 1;0 -2];y0=eye(2);y1=y0;a=0;fs='ej2s';

        file_id= fopen('spline6_ej2_0.1','w');
    xb=0;h=0.1;
    fprintf(file_id,'0 1e-16\n');
    for i=1:10
        xb=xb+h;
        Ya=splin2linear(A0,A1,y0,y1,a,xb,h,m);
        Ys=feval(fs,xb);
        norm_Ys=norm(Ys);
        er=norm(Ys-Ya)/norm_Ys;
        fprintf(file_id,'%f %e\n',xb,er);
    end
    fclose(file_id);

    file_id= fopen('spline6_ej2_0.01','w');
    xb=0;h=0.01;
    fprintf(file_id,'0 1e-16\n');
    for i=1:100
        xb=xb+h;
        Ya=splin2linear(A0,A1,y0,y1,a,xb,h,m);
        Ys=feval(fs,xb);
        norm_Ys=norm(Ys);
        er=norm(Ys-Ya)/norm_Ys;
        fprintf(file_id,'%f %e\n',xb,er);
    end
    fclose(file_id);

    file_id= fopen('spline6_ej2_0.001','w');
    xb=0;h=0.001;
    fprintf(file_id,'0 1e-16\n');
    for i=1:100
        xb=xb+10*h;
        Ya=splin2linear(A0,A1,y0,y1,a,xb,h,m);
```

```
        Ys=feval(fs,xb);
        norm_Ys=norm(Ys);
        er=norm(Ys-Ya)/norm_Ys;
        fprintf(file_id,'%f %e\n',xb,er);
    end
    fclose(file_id);

%Ejemplo 3
    A0=[1 0;2 1];y0=zeros(2);y1=[1 0;1 1];a=0;fs='ej3s';

     file_id= fopen('spline6_ej3_0.1','w');
    xb=0;h=0.1;
    fprintf(file_id,'0 1e-16\n');
    for i=1:10
        xb=xb+h;
        Ya=splin2lineari(A0,y0,y1,a,xb,h,m);
        Ys=feval(fs,xb);
        norm_Ys=norm(Ys);
        er=norm(Ys-Ya)/norm_Ys;
        fprintf(file_id,'%f %e\n',xb,er);
    end
    fclose(file_id);

    file_id= fopen('spline6_ej3_0.01','w');
    xb=0;h=0.01;
    fprintf(file_id,'0 1e-16\n');
    for i=1:100
        xb=xb+h;
        Ya=splin2lineari(A0,y0,y1,a,xb,h,m);
        Ys=feval(fs,xb);
        norm_Ys=norm(Ys);
        er=norm(Ys-Ya)/norm_Ys;
        fprintf(file_id,'%f %e\n',xb,er);
    end
    fclose(file_id);

    file_id= fopen('spline6_ej3_0.001','w');
    xb=0;h=0.001;
    fprintf(file_id,'0 1e-16\n');
    for i=1:100
        xb=xb+10*h;
        Ya=splin2lineari(A0,y0,y1,a,xb,h,m);
        Ys=feval(fs,xb);
        norm_Ys=norm(Ys);
        er=norm(Ys-Ya)/norm_Ys;
        fprintf(file_id,'%f %e\n',xb,er);
    end
    fclose(file_id);
```

Program 4: prsplin_second_order_lineal_gnuplot.m

The program splin2orderm.m for arbitrary order m of the matrix splines and the script for graphics generation prsplin_second_order_lineal_gnuplot.m, which passes the results to the *gnuplot* programming language [10], both rely on the external function call solver_g() provided by the following code:

Program 5: solver_g.m

```
function z= solver_g(f,x,dy0,dy1,dy2,m,h)
fac=[1 2 6 24 120 720 5040 40320 362880 3628800];
aux=fac(m-2)/h(m-2);
z=aux*(feval(f,1,x,dy0,dy1)-dy2);
ea=1;
while ea>eps
    z1=aux*(feval(f,1,x,dy0+z*h(m)/fac(m),dy1+z*h(m-1)/fac(m-1))-dy2);
    ea=norm(z-z1);
    z=z1;
end
```

Program 5: solver_g.m

Some of the examples for the numerical solutions of first-order matrix differential models using higher-order matrix splines were produced with the following MATLAB programs:

Program 6: pr_spline.m

```
function pr_spline(xb,h)
%pr_spline(1,0.02); definite example
%
%Example 1 for publication in the journal
%Mediterranean Journal of Mathematics

path(path,'G:\Docs\Documentos\Dropbox\Investigacin\Splines\matlab\numerico')
options=odeset('RelTol',2.22045e-14,'AbsTol',1e-14);
for i=1:20
    rand(1000)*rand(1000);
end

tic
[xb,Y0]=splin_AY('ej1',0,xb,ones(2),[1;0],h);
time=toc;
Ys=feval('ej1s',xb);
norm_Ys=norm(Ys);
if norm_Ys>0
    er=norm(Ys-Y0)/norm_Ys;
else
    er=norm(yas-ya);
end
fprintf('Spline4(h=%4f): Time=%f er=%e\n',h,time,er)
```

```
tic
[xb,Y0]=splin_AY4('ej14',0,xb,ones(2),[1;0],h);
time=toc;
Ys=feval('ej1s',xb);
norm_Ys=norm(Ys);
if norm_Ys>0
    er=norm(Ys-Y0)/norm_Ys;
else
    er=norm(yas-ya);
end
fprintf('Spline5(h=%4f): Time=%f er=%e\n',h,time,er)

tic
[xb,Y0]=splin_AY5('ej15',0,xb,ones(2),[1;0],h);
time=toc;
Ys=feval('ej1s',xb);
norm_Ys=norm(Ys);
if norm_Ys>0
    er=norm(Ys-Y0)/norm_Ys;
else
    er=norm(yas-ya);
end
fprintf('Spline6(h=%4f): Time=%f er=%e\n',h,time,er)


tic
[x,y]=ode45('ej1ode',[0 xb],[1;0],options);
time=toc;
l=length(x);
Ys=feval('ej1s',x(l));
norm_Ys=norm(Ys);
Y0=y(l,1:2)';
if norm_Ys>0
    er=norm(Ys-Y0)/norm_Ys;
else
    er=norm(yas-ya);
end
fprintf('ode45: Time=%f er=%e\n',time,er)

tic
[x,y]=ode23('ej1ode',[0 xb],[1;0],options);
time=toc;
l=length(x);
Ys=feval('ej1s',x(l));
norm_Ys=norm(Ys);
Y0=y(l,1:2)';
if norm_Ys>0
    er=norm(Ys-Y0)/norm_Ys;
else
```

```
    er=norm(yas-ya);
end
fprintf('ode23: Time=%f er=%e\n',time,er)

tic
[x,y]=ode113('ej1ode',[0 xb],[1;0],options);
time=toc;
l=length(x);
Ys=feval('ej1s',x(l));
norm_Ys=norm(Ys);
Y0=y(l,1:2)';
if norm_Ys>0
    er=norm(Ys-Y0)/norm_Ys;
else
    er=norm(yas-ya);
end
fprintf('ode113: Time=%f er=%e\n',time,er)

tic
[x,y]=ode15s('ej1ode',[0 xb],[1;0],options);
time=toc;
l=length(x);
Ys=feval('ej1s',x(l));
norm_Ys=norm(Ys);
Y0=y(l,1:2)';
if norm_Ys>0
    er=norm(Ys-Y0)/norm_Ys;
else
    er=norm(yas-ya);
end
fprintf('ode15s: Time=%f er=%e\n',time,er)

tic
[x,y]=ode23s('ej1ode',[0 xb],[1;0],options);
time=toc;
l=length(x);
Ys=feval('ej1s',x(l));
norm_Ys=norm(Ys);
Y0=y(l,1:2)';
if norm_Ys>0
    er=norm(Ys-Y0)/norm_Ys;
else
    er=norm(yas-ya);
end
fprintf('ode23s: Time=%f er=%e\n',time,er)

tic
[x,y]=ode23t('ej1ode',[0 xb],[1;0],options);
time=toc;
l=length(x);
```

```
Ys=feval('ej1s',x(l));
norm_Ys=norm(Ys);
Y0=y(l,1:2)';
if norm_Ys>0
    er=norm(Ys-Y0)/norm_Ys;
else
    er=norm(yas-ya);
end
fprintf('ode23t: Time=%f er=%e\n',time,er)

tic
[x,y]=ode23tb('ej1ode',[0 xb],[1;0],options);
time=toc;
l=length(x);
Ys=feval('ej1s',x(l));
norm_Ys=norm(Ys);
Y0=y(l,1:2)';
if norm_Ys>0
    er=norm(Ys-Y0)/norm_Ys;
else
    er=norm(yas-ya);
end
fprintf('ode23tb: Time=%f er=%e\n',time,er)
```

Program 6: pr_spline.m

Program 7: pr_spline_A_B.m

```
function pr_spline_A_B(xb,h,pr)
%pr_spline_A_B(2,0.01,1); definite example for article
path(path,'G:\Docs\Documentos\Dropbox\Investigacin\Splines\matlab\numerico')
options=odeset('RelTol',2.22045e-014,'AbsTol',1e-14);
%options=odeset('RelTol',1e-17);
%options=odeset('AbsTol',1e-14);
for i=1:20
    rand(1000)*rand(1000);
end

if pr==1
    %Example 2 first version for Mediterranean Journal of Mathematics
    f1='ej2';f2='ej24';f3='ej25';
    fs='ej2s';
    f0='ej2ode';fos='ej2sode';
    A0=[-1 0 0;1 0 1;0 -1 1];B0=[2 3;-1 4;1 -1];
    Y0=[1 1;0 -1;0 0];Yo0=[1;0;0;1;-1;0];
elseif pr==2
    %Example 2 second version for Mediterranean Journal of Mathematics
        f1='ej5';f2='ej54';f3='ej55';
    fs='ej5s';
```

```
    f0='ej5ode';fos='ej5sode';
    A0=[1 -1;1 1];B0=[-4 0;-5 -1];Y0=[3 0;1 1];Yo0=[3;1;0;1];
end

tic
[xb,Ya]=splin_AY_B(f1,0,xb,A0,B0,Y0,h);
time=toc;
Ys=feval(fs,xb);
norm_Ys=norm(Ys);
if norm_Ys>0
    er=norm(Ys-Ya)/norm_Ys;
else
    er=norm(Ys-Ya);
end
fprintf('Spline4: Time=%f-h=%4f-er=%e\n',time,h,er)

tic
[xb,Ya]=splin_AY_B4(f2,0,xb,A0,B0,Y0,h);
time=toc;
Ys=feval(fs,xb);
norm_Ys=norm(Ys);
if norm_Ys>0
    er=norm(Ys-Ya)/norm_Ys;
else
    er=norm(Ys-Ya);
end
fprintf('Spline5: Time=%f-h=%4f-er=%e\n',time,h,er)

tic
[xb,Ya]=splin_AY_B5(f3,0,xb,A0,B0,Y0,h);
time=toc;
Ys=feval(fs,xb);
norm_Ys=norm(Ys);
if norm_Ys>0
    er=norm(Ys-Ya)/norm_Ys;
else
    er=norm(Ys-Ya);
end
fprintf('Spline6: Time=%f-h=%4f-er=%e\n',time,h,er)

tic
[x,y]=ode45(f0,[0 xb],Yo0,options);
time=toc;
[m,n]=size(y);
Ys=feval(fos,x(m));
norm_Ys=norm(Ys);
Ya=y(m,1:n)';
if norm_Ys>0
    er=norm(Ys-Ya)/norm_Ys;
else
```

```
    er=norm(Ys-Ya);
end
fprintf('ode45: Time=%f er=%e\n',time,er)

tic
[x,y]=ode23(f0,[0 xb],Yo0,options);
time=toc;
[m,n]=size(y);
Ys=feval(fos,x(m));
norm_Ys=norm(Ys);
Ya=y(m,1:n)';
if norm_Ys>0
    er=norm(Ys-Ya)/norm_Ys;
else
    er=norm(Ys-Ya);
end
fprintf('ode23: Time=%f er=%e\n',time,er)

tic
[x,y]=ode113(f0,[0 xb],Yo0,options);
time=toc;
[m,n]=size(y);
Ys=feval(fos,x(m));
norm_Ys=norm(Ys);
Ya=y(m,1:n)';
if norm_Ys>0
    er=norm(Ys-Ya)/norm_Ys;
else
    er=norm(Ys-Ya);
end
fprintf('ode113: Time=%f er=%e\n',time,er)

tic
[x,y]=ode15s(f0,[0 xb],Yo0,options);
time=toc;
[m,n]=size(y);
Ys=feval(fos,x(m));
norm_Ys=norm(Ys);
Ya=y(m,1:n)';
if norm_Ys>0
    er=norm(Ys-Ya)/norm_Ys;
else
    er=norm(Ys-Ya);
end
fprintf('ode15s: Time=%f er=%e\n',time,er)

tic
[x,y]=ode23s(f0,[0 xb],Yo0,options);
time=toc;
[m,n]=size(y);
```

```
Ys=feval(fos,x(m));
norm_Ys=norm(Ys);
Ya=y(m,1:n)';
if norm_Ys>0
    er=norm(Ys-Ya)/norm_Ys;
else
    er=norm(Ys-Ya);
end
fprintf('ode23s: Time=%f er=%e\n',time,er)

tic
[x,y]=ode23t(f0,[0 xb],Yo0,options);
time=toc;
[m,n]=size(y);
Ys=feval(fos,x(m));
norm_Ys=norm(Ys);
Ya=y(m,1:n)';
if norm_Ys>0
    er=norm(Ys-Ya)/norm_Ys;
else
    er=norm(Ys-Ya);
end
fprintf('ode23t: Time=%f er=%e\n',time,er)

tic
[x,y]=ode23tb(f0,[0 xb],Yo0,options);
time=toc;
[m,n]=size(y);
Ys=feval(fos,x(m));
norm_Ys=norm(Ys);
Ya=y(m,1:n)';
if norm_Ys>0
    er=norm(Ys-Ya)/norm_Ys;
else
    er=norm(Ys-Ya);
end
fprintf('ode23tb: Time=%f er=%e\n',time,er)
```

Program 7: pr_spline_A_B.m

Program 8: pr_spline_A_B_gnuplot.m

```
function pr_spline_A_B_gnuplot(pr)
%
path(path,'G:\Docs\Documentos\Dropbox\Investigacin\Splines\matlab\numerico')


if pr==1
    %Example 2 first version for Mediterranean Journal of Mathematics
```

```
    f1='ej2';f2='ej24';f3='ej25';
    fs='ej2s';
    A0=[-1 0 0;1 0 1;0 -1 1];B0=[2 3;-1 4;1 -1];
    Y0=[1 1;0 -1;0 0];Yo0=[1;0;0;1;-1;0];
elseif pr==2
    %Example 2 second version for Mediterranean Journal of Mathematics
        f1='ej5';f2='ej54';f3='ej55';
    fs='ej5s';
    A0=[1 -1;1 1];B0=[-4 0;-5 -1];Y0=[3 0;1 1];Yo0=[3;1;0;1];
end

file_id= fopen('spline4_ej2_0.01','w');
xb=0;h=0.01;
fprintf(file_id,'0 0\n');
for i=1:100
    xb=xb+h;
    [x,Ya]=splin_AY_B(f1,0,xb,A0,B0,Y0,h);
    Ys=feval(fs,x);
    norm_Ys=norm(Ys);
    if norm_Ys>0
        er=norm(Ys-Ya)/norm_Ys;
    else
        er=norm(Ys-Ya);
    end
    fprintf(file_id,'%f %e\n',xb,er);
end
fclose(file_id);

file_id= fopen('spline4_ej2_0.001','w');
xb=0;h=0.001;
fprintf(file_id,'0 0\n');
for i=1:100
    xb=xb+10*h;
    [x,Ya]=splin_AY_B(f1,0,xb,A0,B0,Y0,h);
    Ys=feval(fs,x);
    norm_Ys=norm(Ys);
    if norm_Ys>0
        er=norm(Ys-Ya)/norm_Ys;
    else
        er=norm(Ys-Ya);
    end
    fprintf(file_id,'%f %e\n',xb,er);
end
fclose(file_id);

file_id= fopen('spline5_ej2_0.01','w');
xb=0;h=0.01;
fprintf(file_id,'0 0\n');
for i=1:100
    xb=xb+h;
```

```
    [x,Ya]=splin_AY_B4(f2,0,xb,A0,B0,Y0,h);
    Ys=feval(fs,x);
    norm_Ys=norm(Ys);
    if norm_Ys>0
        er=norm(Ys-Ya)/norm_Ys;
    else
        er=norm(Ys-Ya);
    end
    fprintf(file_id,'%f %e\n',xb,er);
end
fclose(file_id);

file_id= fopen('spline5_ej2_0.001','w');
xb=0;h=0.001;
fprintf(file_id,'0 0\n');
for i=1:100
    xb=xb+10*h;
    [x,Ya]=splin_AY_B4(f2,0,xb,A0,B0,Y0,h);
    Ys=feval(fs,x);
    norm_Ys=norm(Ys);
    if norm_Ys>0
        er=norm(Ys-Ya)/norm_Ys;
    else
        er=norm(Ys-Ya);
    end
    fprintf(file_id,'%f %e\n',xb,er);
end
fclose(file_id);
```

Program 8: pr_spline_A_B_gnuplot.m

References

1. The Mathworks Inc. Staff, MATLAB - *The Language of Technical Computing*, The Mathworks Inc., Natick, MA, 2004.
2. Hans Benker, *Ingenieurmathematik Kompakt - Problemlösungen mit* MATLAB*: Einstieg und Nachschlagewerk für Ingenieure und Naturwissenschaftler*, Springer, Berlin, 2010.
3. Jaan Kiusalaas, *Numerical Methods in Engineering with* MATLAB, Cambridge University Press, Cambridge, 2010.
4. Brian Hahn and Dan Valentine, *Essential* MATLAB *for Engineers and Scientists*, Academic Press, New York, 2007.
5. Aslak Tveito, Hans Petter Langtangen, Bjorn Frederik Nielsen, and Xing Cai, *Elements of Scientific Computing*, Springer, Berlin, 2010.
6. John W. Eaton, *GNU Octave Manual*, Network Theory Limited, Bristol, UK, 2002.
7. *GNU Octave*, http://www.gnu.org/software/octave/.
8. Alfio Quarteroni, Fausto Saleri, and Paola Gervasio, *Scientific Computing with* MATLAB *and Octave*, Texts in Computational Science and Engineering, Springer, Berlin, 2010.
9. Hans Petter Langtangen, *Scripting in Computational Science*, Springer, Berlin, 2004.
10. Philipp K. Janert, *gnuplot in Action: Understanding Data with Graphs*, Manning Publications Co., Greenwich, CT, 2009.