Numerical solutions of matrix differential models using higher-order matrix splines

Emilio Defez, Antonio Hervás, J. Ibañez and Michael M. Tung

Abstract. This paper deals with the construction of approximate solution of first-order matrix linear differential equations using higher-order matrix splines. An estimation of the approximation error, an algorithm for its implementation and some illustrative examples are included.

Keywords. Matrix linear differential equations, higher-order matrix splines.

1. Introduction

Matrix differential equations emerge frequently in a great variety of models in physics and engineering [3, 10, 21]. 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 [23], shooting methods for scalar or vectorial problems with boundary values conditions [19], lines method for the numerical integration of partial differential equations [22] or homotopic methods to solve non-linear systems equations [5].

The vectorization techniques to transform a matrix problem into a set of scalar equations has several drawbacks [13]. 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

$$\left.\begin{array}{rcl} Y'(x) & = & A(x)Y(x) + B(x),\ a \leq x \leq b \\ Y(a) & = & Y_a \end{array}\right\}. \tag{1}$$

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 $A, B \in \mathcal{C}^s([a,b])$, $s \geq 1$, which guarantees the existence of a unique and continuously differentiable solution $Y(x)$ of (1), see [11, p.99].

Problem (1) is not only used in the mathematical modelling of many different technological applications [2], but also permits to deal with nonlinear problems, such as Riccati equations [17, 15, 14], after employing some linearization techniques. Numerical methods for the calculation of approximate solutions of problems of the type (1) by means of linear multi-step methods with constant steps have been studied in [16]. 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, [16]. Other methods, based on the developments of Magnus and Fer [4], 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 [9] and references therein).

In the scalar case, cubic splines were used in for the resolution of ordinary differential equations [18], 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 [1]. For example, Ref. [20] 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. [7, 8]. Unfortunately, as already detected by Loscalzo and Talbot in [18], their scalar procedure is divergent when higher-order spline functions are used [18, 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 $\mathcal{C}^1$.

In this paper, we propose a method using higher-order matrix splines for the numerical approximation to the solution of (1). The present work extends all important advantages already obtained in [18] for the scalar case to the matrix framework.

This paper is organized as follows. 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 $g \in \mathcal{C}^k([a, b])$. 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 $\mathcal{C}^n([a,b])$ as

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

For $m = 3, n = 2$ these matrix splines are called *matrix cubic splines* [6].

2. Description of the method

Let us consider the following first-order matrix problem

$$\left.\begin{array}{rcl} Y'(x) & = & A(x)Y(x) + B(x) \\ Y(a) & = & Y_a \end{array}\right\}\ a \leq x \leq b, \tag{2}$$

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 $A, B \in \mathcal{C}^s([a,b])$, $s \geq 1$, guarantees the uniqueness of solution $Y(x)$ of problem (1), which is continuously differentiable [11, p.99].

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

$$\Delta_{[a,b]} = \{a = x_0 < x_1 < \ldots < x_n = b\},\ x_k = a + kh,\ k = 0,1,\ldots,n, \tag{3}$$

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 (2) can be approximated by the matrix spline $S(x) \in \mathcal{C}^1([a,b])$.

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

$$S_{\big|_{[a,a+h]}}(x) = \sum_{j=0}^{m-1} \frac{1}{j!} Y^{(j)}(a)(x-a)^j + \frac{1}{m!}\alpha_0 (x-\alpha)^m, \tag{4}$$

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

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

and therefore the spline satisfies the differential equation Eq. (2) at $x = a$.

In order to determine the matrix spline (4), we still must obtain the values $Y''(a), Y^{(3)}(a), \ldots, Y^{(m-1)}(a)$, and A_0. For the second-order derivative $Y''(x)$, we proceed to compute

$$\begin{array}{rcl} Y''(x) & = & A'(x)Y(x) + A(x)Y'(x) + B'(x) \\ & = & g_1(x, Y(x)), \end{array} \tag{5}$$

where $g_1 \in \mathcal{C}^{s-1}([a,b])$. Using (5), we now can evaluate $Y''(a) = g_1(a, Y(a))$.

For the third derivative one continues in a similar manner:

$$\begin{array}{l} Y^{(3)}(x) = A''(x)Y(x) + 2A'(x)Y'(x) + A(x)Y''(x) + B''(x) \\ \quad = g_2(x, Y(x)) \in \mathcal{C}^{s-2}([a,b]), \end{array} \tag{6}$$

and evaluates $Y^{(3)}(a) = g_2(a, Y(a))$ using (6). For the next higher-order derivatives $Y^{(4)}(x), \ldots, Y^{(m-1)}(x)$ we proceed similarly and calculate

$$\left.\begin{array}{rcl} Y^{(4)}(x) & = & g_3(x, Y(x)) \in \mathcal{C}^{s-3}([a,b]) \\ & \vdots & \\ Y^{(m-1)}(x) & = & g_{m-2}(x, Y(x)) \in \mathcal{C}^{s-(m-2)}([a,b]) \end{array}\right\}. \tag{7}$$

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 (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 (4) is a solution of problem (2) at $x = a + h$, which gives

$$S'_{|_{[a,a+h]}}(a+h) = A(a+h)S_{|_{[a,a+h]}}(a+h) + B(a+h). \tag{8}$$

Next, we obtain from (8) the matrix equation with only one unknown α_0:

$$\left(I - \frac{h}{m}A(a+h)\right)\alpha_0 = \tag{9}$$
$$\frac{(m-1)!}{h^{m-1}}\left(A(a+h)\sum_{j=0}^{m-1}\frac{h^j}{j!}Y^{(j)}(a) - \sum_{j=0}^{m-2}\frac{h^j}{j!}Y^{(j+1)}(a) + B(a+h)\right)$$

Assuming uniqueness of the solution α_0 given by the matrix equation (9), the matrix spline introduced in Eq. (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

$$S_{|_{[a+h,a+2h]}}(x) = \tag{10}$$
$$S_{|_{[a,a+h]}}(a+h) + \sum_{j=1}^{m-1}\frac{1}{j!}\overline{Y^{(j)}(a+h)}(x-(a+h))^j + \frac{1}{m!}\alpha_1(x-(a+h))^m,$$

where

$$\overline{Y'(a+h)} = A(a+h)S_{|_{[a,a+h]}}(a+h) + B(a+h). \tag{11}$$

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 $S_{|_{[a,a+h]}}(a+h)$ in (5)–(7). In more compact form, we may write

$$\begin{array}{rcl} \overline{Y''(a+h)} & = & g_1\left(a+h, S_{|_{[a,a+h]}}(a+h)\right), \\ & \vdots & \\ \overline{Y^{(m-1)}(a+h)} & = & g_{m-2}\left(a+h, S_{|_{[a,a+h]}}(a+h)\right). \end{array} \tag{12}$$

Note that matrix spline $S(x)$ defined by (4) and (10) is of differentiability class $\mathcal{C}^1([a, a+2h])$, contrary to the splines introduced by Loscalzo and Talbot [18], which were of class $\mathcal{C}^{m-1}([a, a+2h])$. In Ref. [7], 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. (12).

By construction, the spline (10) satisfies the differential equation (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 (10) to be a unique solution of (2) at point $x = a + 2h$:

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

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

$$\begin{aligned}
&\left(I - \frac{h}{m}A(a+2h)\right)\alpha_1 = \\
&\frac{(m-1)!}{h^{m-1}}\left(A(a+2h)\left(S_{|_{[a,a+h]}}(a+h) + \sum_{j=1}^{m-1}\frac{h^j}{j!}\overline{Y^{(j)}(a+h)}\right)\right. \\
&\left. - \sum_{j=0}^{m-2}\frac{h^j}{j!}\overline{Y^{(j+1)}(a+h)} + B(a+h)\right).
\end{aligned} \tag{13}$$

Let us assume again that the matrix equation (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

$$\begin{aligned}
&S_{|_{[a+kh,a+(k+1)h]}}(x) = \\
&S_{|_{[a+(k-1)h,a+kh]}}(a+kh) + \sum_{j=1}^{m-1}\frac{1}{j!}\overline{Y^{(j)}(a+kh)}(x-(a+kh))^j \\
&+ \frac{1}{m!}\alpha_k(x-(a+kh))^m,
\end{aligned} \tag{14}$$

where

$$\overline{Y'(a+kh)} = A(a+kh)S_{|_{[a+(k-1)h,a+kh]}}(a+kh) + B(a+kh). \tag{15}$$

In a similar manner as before, one abbreviates

$$\begin{aligned}
\overline{Y''(a+kh)} &= g_1\left(a+kh, S_{|_{[a+(k-1)h,a+kh]}}(a+kh)\right), \\
&\vdots \\
\overline{Y^{(m-1)}(a+kh)} &= g_{m-2}\left(a+kh, S_{|_{[a+(k-1)h,a+kh]}}(a+kh)\right).
\end{aligned} \tag{16}$$

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 (2) at point $x = a + kh$. Recall that Eq. (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 (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

$$\left(I - \frac{h}{m}A(a+(k+1)h)\right)\alpha_k = \tag{17}$$

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

Observe that the final result (17) relates directly to equations (9) and (13), when setting $k = 0$ and $k = 1$. Note also that solubility of equation (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

$$M = \max\{\|A(x)\|\,;\ a \le x \le b\}, \tag{18}$$

where any induced norm applies. Then, one obtains

$$\left\|I - \left(I - \frac{h}{m}A(a+(k+1)h)\right)\right\| = \frac{h}{m}\|A(a+(k+1)h)\| \le \frac{h}{m}M. \tag{19}$$

If we take $h \le m/M$, according to Lemma 2.3.3 in [12], it follows that matrix $I - (h/m)A(a+(k+1)h)$ is invertible, and therefore equation (17) has a unique solution α_k, for each $k = 0, 1, \ldots, n-1$. In summary, we have proved the following theorem:

Theorem 2.1. *For the first-order matrix differential equation (2), assume that $A, B \in \mathcal{C}^s([a,b])$, $s \ge 1$. Let $h > 0$ so that $h \le m/M$, where M is given by (18) and $0 < m \le s+1$. We also consider the partition (3) with step size $h < m/L$. Then, a matrix spline $S(x)$ of order $m \in \mathbb{N}$ and differentiability class $\mathcal{C}^1[a,b]$ 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 [18].

The approximate solution of (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 2.1. The procedure is as follows:

- Compute the functions $g_1(x, Y(x)), \ldots, g_{m-2}(x, Y(x))$ given by Eqs. (5)–(7) to determine constants $Y''(a)$, ..., $Y^{(m-1)}(a)$. Compute constant M of Eq. (18). Choose $n > M(b-a)/m$ so that $h = (b-a)/n$, which produces the partition $\Delta_{[a,b]}$ defined by Eq. (3).

• Solve equation Eq. (9) to find α_0, and determine $S_{|_{[a,a+h]}}(x)$ of Eq. (4).

• Solve Eq. (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. (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. [7]. 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 (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 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 $AbsTol = 1.0 \cdot 10^{-14}$.

Example 3.1. *Let us consider the problem*

$$\left.\begin{aligned} Y'(x) &= \frac{1}{x^3-x-1}\begin{pmatrix} 2x^2-1 & x^2-2x-1 \\ -x-1 & x^3+x^2-x-1 \end{pmatrix} Y(x), \quad 0 \le x \le 1, \\ Y(0) &= \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad Y(x) \in \mathbb{C}^2. \end{aligned}\right\} \tag{20}$$

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.* [7], *produced the absolute errors listed in Table* 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. [7]*? For fourth-order splines* ($m = 4$), *we obtain the results given in Table* 2*(b). If we further increase the order of the splines, the result is worsening, as shown in part (c) of Table* 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 *together with their corresponding absolute errors. We also present the results for fifth-order splines* ($m = 5$) *in Table* 4. *Figures* 1 *and* 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* 5 *and* 6 *present*

the results of the proposed method, with $h = 0.1$ *and* $h = 0.01$*, 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* $x = 10$*.*

Example 3.2. *Consider the matrix problem*

$$\left.\begin{array}{rcl} Y'(x) & = & A(x)\,Y(x) + B(x) \\ Y(0) & = & \begin{pmatrix} 3 & 0 \\ 1 & 1 \end{pmatrix}, \quad x \in [0,1] \end{array}\right\} \tag{21}$$

where

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

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* [7]*, we obtain the errors given in Table* 7*(a). The absolute errors are calculated as in Example* 3.1*.*

What happens if we increase the order of the splines using the same technique as in Ref. [7]*? For fourth-order splines, we obtain the result given in Table* 7*(b). If we increase the order of the spline, the quality of the approximation gets worse, which is shown in Table* 7*(c) for splines of order* $m = 5$*.*

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* 8*(a) with absolute errors. Similarly, the results using fifth-order splines are shown in Table* 8*(b).*

Figures 3 *and* 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* 4 *we observe that* $h = 0.001$ *yields an accuracy very close to machine precision. Table* 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* (21)*. For* $x \geq 3$ *all solvers and splines presented convergence problems.*

4. Conclusions

This work 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 [7]) 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. [9]. A closer focus on the behavior of stiff problems should also be interesting.

5. Acknowledgments

The authors are grateful to the referees for their helpful comments.

References

[1] Al-Said, E.A., Noor, M.A.: Cubic splines method for a system of third-order boundary value problems. Appl. Math. Comput. **142**, 195–204 (2003)

[2] Ascher, U., Mattheij, R., Russell, R.: Numerical solutions of boundary value problems for ordinary differential equations. Prentice Hall, New Jersey, USA (1988)

[3] Barnett, S.: Matrices in Control Theory. Van Nostrand, Reinhold (1971)

[4] Blanes, S., Casas, F., Oteo, J.A., Ros, J.: Magnus and Fer expansion for matrix differential equations: the convergence problem. J. Phys. Appl. **31**, 259–268 (1998)

[5] Boggs, P.T.: The solution of nonlinear systems of equations by a-stable integration techniques. SIAM J. Numer. Anal. **8**(4), 767–785 (1971)

[6] Defez, E., Hervás, A., Law, A., Villanueva-Oller, J., Villanueva, R.: Matrix-cubic splines for progressive transmission of images. J. Math. Imaging Vision **17**(1), 41–53 (2002)

[7] Defez, E., Soler, L., Hervás, A., Santamaría, C.: Numerical solutions of matrix differential models using cubic matrix splines. Comput. Math. Appl. **50**, 693–699 (2005)

[8] Defez, E., Soler, L., Hervás, A., Tung, M.M.: Numerical solutions of matrix differential models using cubic matrix splines II. Mathematical and Computer Modelling **46**, 657–669 (2007)

[9] F. Mazzia, A.S., Trigiante, D.: B-spline linear multistep methods and their conitinuous extensions. SIAM J. Numer. Anal. **44**(5), 1954–1973 (2006)

[10] Faddeyev, L.D.: The inverse problem in the quantum theory of scattering. J. Math. Physics **4**(1), 72–104 (1963)

[11] Flett, T.M.: Differential Analysis. Cambridge University Press (1980)

[12] Golub, G.H., Loan, C.F.V.: Matrix Computations, second edn. The Johns Hopkins University Press, Baltimore, MD, USA (1989)

[13] Graham, A.: Kronecker products and matrix calculus with applications. John Wiley & Sons, New York, USA (1981)

[14] Jódar, L., Cortés, J.C.: Rational matrix approximation with a priori error bounds for non-symmetric matrix riccati equations with analytic coefficients. IMA J. Numer. Anal. **18**(4), 545–561 (1998)

[15] Jódar, L., Cortés, J.C., Morera, J.L.: Construction and computation of variable coefficient sylvester differential problems. Computers Maths. Appl. **32**(8), 41–50 (1996)

[16] Jódar, L., Ponsoda, E.: Continuous numerical solutions and error bounds for matrix differential equations. In: Int. Proc. First Int. Colloq. Num. Anal., pp. 73–88. VSP, Utrecht, The Netherlands (1993)

[17] Jódar, L., Ponsoda, E.: Non-autonomous riccati-type matrix differential equations: Existence interval, construction of continuous numerical solutions and error bounds. IMA J. Numer. Anal. **15**(1), 61–74 (1995)

[18] Loscalzo, F.R., Talbot, T.D.: Spline function approximations for solutions of ordinary differential equations. SIAM J. Numer. Anal. **4**(3), 433–445 (1967)

[19] Marzulli, P.: Global error estimates for the standard parallel shooting method. J. Comput. Appl. Math. **34**, 233–241 (1991)

[20] Micula, G., Revnic, A.: An implicit numerical spline method for systems for ode's. Appl. Math. Comput. **111**, 121–132 (2000)

[21] Reid, W.T.: Riccati Differential Equations. Academic Press (1972)

[22] Rektorys, K.: The method of discretization in time and partial differential equations. D. Reidel Pub. Co., Dordrecht (1982)

[23] Scott, M.: Invariant imbedding and its Applications to Ordinary Differential Equations. Addison-Wesley (1973)

Emilio Defez
Instituto de Matemática Multidisciplinar
Universidad Politécnica de Valencia
Camino de Vera s/n, 46022 Valencia (Spain)
e-mail: edefez@imm.upv.es

Antonio Hervás
Instituto de Matemática Multidisciplinar
Universidad Politécnica de Valencia
Camino de Vera s/n, 46022 Valencia (Spain)
e-mail: ahervas@imm.upv.es

J. Ibañez
Instituto de Instrumentación para Imagen Molecular
Universidad Politécnica de Valencia
Camino de Vera s/n, 46022 Valencia (Spain)
e-mail: jjibanez@dsic.upv.es

Michael M. Tung
Instituto de Matemática Multidisciplinar
Universidad Politécnica de Valencia
Camino de Vera s/n, 46022 Valencia (Spain)
e-mail: mtung@mat.upv.es

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 1. MATLAB solvers used in the tests.

$[x_i, x_{i+1}]$	ERRORS	$[x_i, x_{i+1}]$	ERRORS	$[x_i, x_{i+1}]$	ERRORS
$[0, 0.1]$	6.33721×10^{-6}	$[0, 0.1]$	1.14628×10^{-7}	$[0, 0.1]$	1.7956×10^{-9}
$[0.1, 0.2]$	6.05558×10^{-6}	$[0.1, 0.2]$	8.81776×10^{-7}	$[0.1, 0.2]$	5.7101×10^{-8}
$[0.2, 0.3]$	8.14626×10^{-6}	$[0.2, 0.3]$	2.2721×10^{-6}	$[0.2, 0.3]$	5.46782×10^{-7}
$[0.3, 0.4]$	7.81749×10^{-6}	$[0.3, 0.4]$	9.75288×10^{-6}	$[0.3, 0.4]$	5.32517×10^{-6}
$[0.4, 0.5]$	11.5296×10^{-6}	$[0.4, 0.5]$	0.000033	$[0.4, 0.5]$	0.000051
$[0.5, 0.6]$	11.6396×10^{-6}	$[0.5, 0.6]$	0.00012	$[0.5, 0.6]$	0.00049
$[0.6, 0.7]$	16.357×10^{-6}	$[0.6, 0.7]$	0.00045	$[0.6, 0.7]$	0.0048
$[0.7, 0.8]$	17.359×10^{-6}	$[0.7, 0.8]$	0.0016	$[0.7, 0.8]$	0.047
$[0.8, 0.9]$	23.29×10^{-6}	$[0.8, 0.9]$	0.0060	$[0.8, 0.9]$	0.45
$[0.9, 1]$	24.6909×10^{-6}	$[0.9, 1]$	0.022	$[0.9, 1]$	4.50

(a) *(b)* *(c)*

TABLE 2. Absolute errors using the matrix splines of order (a) $m = 3$, (b) $m = 4$ and (c) $m = 5$ with the method given in [7] with $n = 10$ and $h = 0.1$, for Example 3.1.

$[x_i, x_{i+1}]$	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. Absolute errors using the spline algorithm for problem (20) with fourth-order splines ($m = 4$).

$[x_i, x_{i+1}]$	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 4. Absolute errors using the spline algorithm for problem (20) with fifth-order splines ($m = 5$).

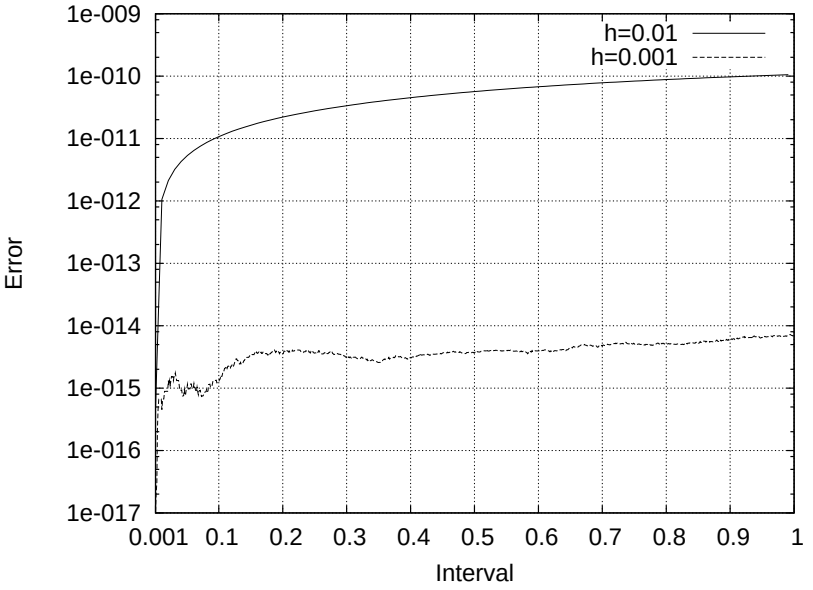

FIGURE 1. Relative errors for the test problem (20) with fourth-order splines ($m = 4$) using our proposed method with $h = 0.01$ and $h = 0.001$, respectively.

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 5. Relative errors for the test problem (20) with splines ($h = 0.1$) and MATLAB solvers.

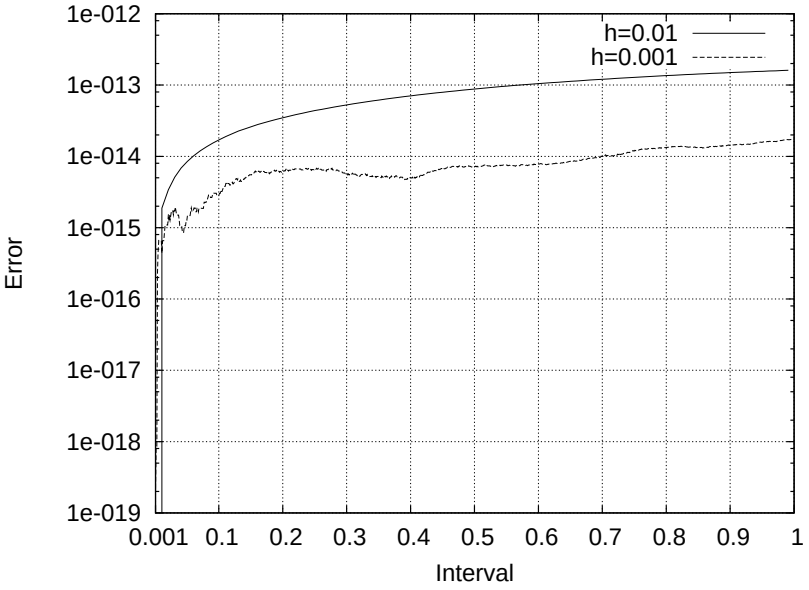

FIGURE 2. Approximation error for problem (20) with fifth-order splines ($m = 5$) using our proposed method with $h = 0.01$ and $h = 0.001$, respectively.

Method	Time [s]	Error
Spline of order $m = 4$	0.055846	$9.994253e-013$
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 6. Relative errors for the test problem (20) with splines ($h = 0.01$) 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 7. 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 [7] with $n = 10$ and $h = 0.1$.

$[x_i, x_{i+1}]$	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 8. Absolute errors for problem (21) using the matrix splines method of order (a) $m = 4$ and (b) $m = 5$, with $n = 10$ and $h = 0.1$.

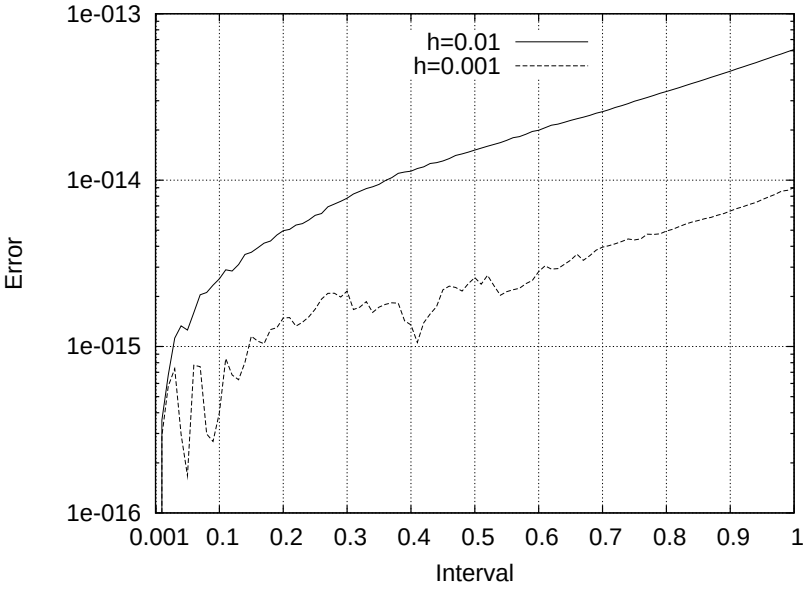

FIGURE 3. Approximation errors for problem (21) with fourth-order splines ($m = 4$) using our proposed method with $h = 0.01$ and $h = 0.001$, respectively.

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 9. Approximation errors for problem (21) with splines of several orders using MATLAB solvers and taking $h = 0.02$.

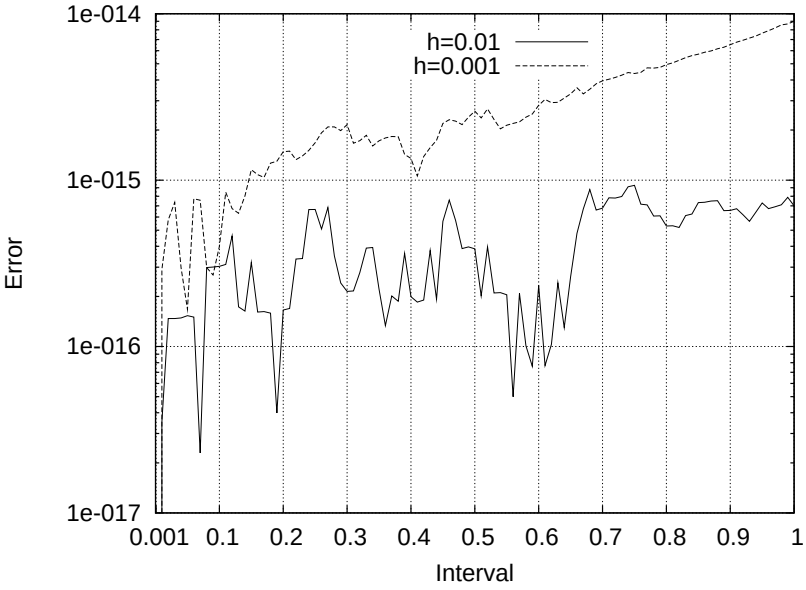

FIGURE 4. Approximation errors for problem (21) with fifth-order splines ($m = 5$) using our proposed method with $h = 0.01$ and $h = 0.001$, respectively.