published in:
Linear and Multilinear Algebra
Vol. 63, No. 3, January 2015, 472–489

RESEARCH ARTICLE

Numerical approximations of second-order matrix differential equations using higher-degree splines

Emilio Defez[a], Michael M. Tung[a*] Francisco J. Solis[b] and Javier Ibáñez[c]

[a]*Instituto de Matemática Multidisciplinar, Universitat Politècnica de València, 46022 Valencia, Spain*; [b]*CIMAT, Guanajuato Gto. México 36000, Mexico*; [c]*Instituto de Instrumentación para Imagen Molecular, Universitat Politècnica de València, 46022 Valencia, Spain*;

(*v2.0 released August 2013*)

Many studies of mechanical systems in engineering are based on second-order matrix models. This work discusses the second-order generalization of previous research on matrix differential equations dealing with the construction of approximate solutions for non-stiff initial problems $Y''(x) = f(x, Y(x), Y'(x))$ using higher-degree matrix splines without any dimensional increase. An estimation of the approximation error for some illustrative examples are presented by using Mathematica. Several MatLab functions have also been developed, comparing, under equal conditions, accuracy and execution times with built-in MatLab functions. Experimental results show the advantages of solving the above initial problem by using the implemented MatLab functions.

Keywords: matrix models; second-order matrix differential equations; matrix splines; approximations for non-stiff initial problems; extended MatLab functions

AMS Subject Classification: 15A99; 65D07; 65F30; 65Y04; 34L99

1. Introduction

Splines are an important tool to solve scalar first-order differential equations [16], obtaining approximations that, among other advantages, are of class $\mathcal{C}^1$ 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 [1, 2, 15], for vector problems [19], linear matrix problems [6] and for first-order matrix differential equations [7]. Recent work in this field can be found in Refs. [3, 4, 18]. 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 [22]. 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 degree of the spline. By adapting the step size h and the degree m of the spline to a particular problem, in principle, any desired accuracy can be reached. Unfortunately, as detected by

*Corresponding author. Email: mtung@mat.upv.es

Loscalzo and Talbot, their scalar procedure is divergent when higher-degree spline functions with $m > 3$ are used [16, p. 444–445]. They have explicitly shown by numerical computations that the equation $y' = y, y(0) = 1$ contains noticeable divergences for splines of degree $m > 3$. For matrix differential equations of first order, we already presented in Ref. [8] a method which avoids these problems with divergences for splines $S(x)$ of degree 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 degree m but only of differentiability class $\mathcal{C}^2$.

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. Therefore, $\mathbb{C}^{p\times q}$ will in general denote the set of rectangular $p \times q$ complex matrices. For matrix $A \in \mathbb{C}^{r\times s}$ its 2-norm is defined by

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

and where for a vector $z \in \mathbb{C}^s$ the conventional Euclidean norm is $\|z\| = \left(z^t z\right)^{\frac{1}{2}}$. As in Ref. [11, p. 56] for $A = (a_{ij}) \in \mathbb{C}^{m\times n}$ we assume that

$$\max_{ij} |a_{ij}| \le \|A\| \le \sqrt{s\,r}\,\max_{ij} |a_{ij}|\,. \tag{1}$$

Then, the Frobenius norm of A is given by

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

and the 2-norm and Frobenius norm are related in the following manner (see Ref. [11]):

$$\|A\|_2 \le \|A\|_F \le \sqrt{n}\,\|A\|_2\,. \tag{3}$$

Employing this notation, 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 [12, p.62 and 81]:

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

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 [12, p.84]:

$$\frac{\partial XY}{\partial Z} = \frac{\partial X}{\partial Z}\left[I_q \otimes Y\right] + \left[I_p \otimes X\right]\frac{\partial Y}{\partial Z} , \tag{4}$$

where I_q and I_p denote the identity matrices of dimensions q and p, respectively. If $X \in \mathbb{C}^{m\times n}, Y \in \mathbb{C}^{u x v}, Z \in \mathbb{C}^{p\times q}$, the following chain rule [12, p.88] is valid:

$$\frac{\partial Z}{\partial X} = \left[\frac{\partial\left[vec(Y)\right]^t}{\partial X} \otimes I_p\right]\left[I_n \otimes \frac{\partial Z}{\partial\left[vec(Y)\right]}\right] . \tag{5}$$

2. Higher-Degree Matrix Splines

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

$$\left.\begin{aligned} Y''(x) &= f(x, Y(x), Y'(x)) \\ Y(a) = Y_0 \ , \ Y'(a) &= Y_1 \end{aligned}\right\} \ a \le x \le b, \tag{6}$$

Note that Eq. (6) 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 [5, 10, 17, 20, 21, 24].

Let us consider the initial matrix value problem Eq. (6) where $Y_0, Y_1, Y(t) \in \mathbb{R}^{r\times q}$, $f : [a, b] \times \mathbb{R}^{r\times q} \times \mathbb{R}^{r\times q} \longrightarrow \mathbb{R}^{r\times q}$, $f \in \mathcal{C}^s(T)$, with

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

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,

$$\left.\begin{aligned} \|f(x, Y_1, Y) - f(x, Y_2, Y)\| &\le L_1 \|Y_1 - Y_2\| \ , \ a \le x \le b \ , Y_1, Y_2, Y \in \mathbb{R}^{r\times q} \\ \|f(x, Y, Y_1) - f(x, Y, Y_2)\| &\le L_2 \|Y_1 - Y_2\| \ , \ a \le x \le b \ , Y_1, Y_2, Y \in \mathbb{R}^{r\times q} \end{aligned}\right\} , \tag{7}$$

guarantee the existence and uniqueness of the continuously differentiable solution $Y(x)$ for the set of equations (6), see *e.g.* [9, p.99].

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

$$\Delta_{[a,b]} = \{a = x_0 < x_1 < \ldots < x_n = b\}\,,\ x_k = a + kh,\ k = 0,1,\ldots,n, \tag{8}$$

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 2.1 : *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 degree 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. (7).

Proof:
We will construct in each subinterval I_k a matrix spline $S(x)$ of degree $m \in \mathbb{N}$ with $m = s+3$, where s is the order of the differentiability class of f. This will approximate the solution of problem (6) so that $S(x) \in \mathcal{C}^2([a,b])$.
In the first interval I_0, we define the matrix spline as

$$\begin{aligned} S_{|_{I_0}}(x) &= Y(a) + Y'(a)(x-a) + \frac{1}{2!}Y''(a)(x-a)^2 + \frac{1}{3!}Y^{(3)}(a)(x-a)^3 \\ &+ \cdots + \frac{1}{(m-1)!}Y^{(m-1)}(a)(x-a)^{m-1} + \frac{1}{m!}A_0(x-a)^m. \end{aligned} \tag{9}$$

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 (6) is fulfilled by the spline at point $x = a$.

For the construction of matrix spline (9), 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

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

where $Y(x)$ is the theoretical solution of (6) 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

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

Thus, ϕ is a real variable function of x, and applying theorem 8.9.2 of [12, p.170] its derivative takes the form:

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

where the partial derivatives of f, $D_1(f)$, $D_2(f)$, $D_3(f)$ exist and are continuous since it is assumed that $f \in \mathcal{C}^s(T)$. By (6) it is clear that

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

Next, applying the chain rule for matrix functions (4) and then taking the derivative of a matrix with respect to a matrix, (5), one obtains

$$\begin{aligned} Y^{(3)}(x) &= \frac{\partial f(x,Y(x),Y'(x))}{\partial x} + \left[\left[vec\ Y'(x)\right]^T \otimes I_r\right] \frac{\partial f(x,Y(x),Y'(x))}{\partial\,(vec\ Y(x))} \\ &+ \left[\left[vec\ f(x,Y(x),Y'(x))\right]^T \otimes I_r\right] \frac{\partial f(x,Y(x),Y'(x))}{\partial\,(vec\ Y'(x))} \\ &= g_1\left(x, Y(x), Y'(x)\right), \end{aligned} \tag{10}$$

where $g_1 \in \mathcal{C}^{s-1}(T)$. Using (10), we are now in position to evaluate $Y^{(3)}(a) = g_1(a, Y(a), Y'(a))$ It is safe to take $f \in \mathcal{C}^s(T)$ 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

$$\left.\begin{aligned} Y^{(4)}(x) &= g_2(x, Y(x), Y'(x)) \in \mathcal{C}^{s-2}(T) \\ &\vdots \\ Y^{(m-1)}(x) &= g_{m-3}(x, Y(x), Y'(x)) \in \mathcal{C}^{s-(m-3)}(T) \end{aligned}\right\} . \tag{11}$$

Apart from the matrix coefficient A_0, all other parameters of the spline are already known. To find A_0, we assume that (9) is a solution of equation (6) at $x = a + h$, and thus yields

$$S''_{|_{I_0}}(a+h) = f\left(a+h, S_{|_{I_0}}(a+h), S'_{|_{I_0}}(a+h)\right). \tag{12}$$

The only unknown parameter A_0 is now given by the following implicit matrix equation obtained from (12):

$$\begin{aligned} A_0 = \frac{(m-2)!}{h^{m-2}} &\left[f\left(a+h, Y(a) + Y'(a)h + \cdots + \frac{h^{m-1}}{(m-1)!}Y^{(m-1)}(a) + \frac{h^m}{m!}A_0, \right.\right. \\ &\left. Y'(a) + Y''(a)h + \cdots + \frac{h^{m-2}}{(m-2)!}Y^{(m-1)}(a) + \frac{h^{m-1}}{(m-1)!}A_0 \right) \\ &\left. - Y''(a) - \cdots - \frac{h^{m-3}}{(m-3)!}Y^{(m-1)}(a) \right]. \end{aligned} \tag{13}$$

The matrix equation (13) has only one solution A_0, which we shall see in the following. After finding the solution for A_0, the matrix spline (9) is totally determined within the first interval I_0. For the second interval I_1 we define

$$\begin{aligned} &S_{|_{I_1}}(x) = S_{|_{I_0}}(a+h) + S'_{|_{I_0}}(a+h)(x-(a+h)) + \\ &\tfrac{1}{2!}\overline{Y''(a+h)}(x-(a+h))^2 + \cdots + \tfrac{1}{(m-1)!}\overline{Y^{(m-1)}(a+h)}(x-(a+h))^{m-1} \\ &+\tfrac{1}{m!}A_1(x-(a+h))^m, \end{aligned} \tag{14}$$

and we use the following shorthand notation

$$\begin{aligned}
\overline{Y''(a+h)} &= f\left(a+h, S_{|_{I_0}}(a+h), S'_{|_{I_0}}(a+h)\right), \\
\overline{Y^{(3)}(a+h)} &= g_1\left(a+h, S_{|_{I_0}}(a+h), S'_{|_{I_0}}(a+h)\right), \\
&\vdots \\
\overline{Y^{(m-1)}(a+h)} &= g_{m-3}\left(a+h, S_{|_{I_0}}(a+h), S'_{|_{I_0}}(a+h)\right).
\end{aligned} \tag{15}$$

The splines introduced by Loscalzo and Talbot [16] and the ones employed in [22] were of class $\mathcal{C}^{m-1}(I_0 \cup I_1)$. In contrast, here the matrix spline $S(x)$ defined by (9) and (14) is of differentiability class $\mathcal{C}^2(I_0 \cup I_1)$. All of the coefficients in (14) are completely determined except for the parameter $A_1 \in \mathbb{R}^{r\times q}$. By definition, spline (14) satisfies the differential equation (6) at point $x = a+h$. To find the value of A_1 we also assume that the spline (14) satisfies (6) at point $x = a+2h$:

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

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

$$\begin{aligned}
A_1 = \frac{(m-2)!}{h^{m-2}}\Bigg[f\Bigg(& a+2h, S_{|_{I_0}}(a+h) + S'_{|_{I_0}}(a+h)h + \frac{h^2}{2!}\overline{Y''(a+h)} + \cdots + \\
+ & \frac{h^{m-1}}{(m-1)!}\overline{Y^{(m-1)}(a+h)} + \frac{h^m}{m!}A_1, S'_{|_{I_0}}(a+h) + \overline{Y''(a+h)}h + \cdots \\
+ & \frac{h^{m-2}}{(m-2)!}\overline{Y^{(m-1)}(a+h)} + \frac{h^{m-1}}{(m-1)!}A_1\Bigg) - \overline{Y''(a+h)} \\
- & \overline{Y^{(3)}(a+h)}h - \cdots - \frac{h^{m-3}}{(m-3)!}\overline{Y^{(m-1)}(a+h)}
\end{aligned} \tag{16}$$

Now we proceed in exactly similar manner as before. Namely, let us assume that the implicit matrix equation (16) 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

$$\begin{aligned}
& S_{|_{I_k}}(x) = S_{|_{I_{k-1}}}(a+kh) + S'_{|_{I_{k-1}}}(a+kh)(x-(a+kh)) \\
& + \tfrac{1}{2!}\overline{Y''(a+kh)}(x-(a+kh))^2 + \cdots + \\
& \tfrac{1}{(m-1)!}\overline{Y^{(m-1)}(a+kh)}(x-(a+kh))^{m-1} + \tfrac{1}{m!}A_k(x-(a+kh))^m,
\end{aligned} \tag{17}$$

where again

$$\begin{aligned}
\overline{Y''(a+kh)} &= f\left(a+kh, S_{|_{I_{k-1}}}(a+kh), S'_{|_{I_{k-1}}}(a+kh)\right), \\
\overline{Y^{(3)}(a+kh)} &= g_1\left(a+kh, S_{|_{I_{k-1}}}(a+kh), S'_{|_{I_{k-1}}}(a+kh)\right), \\
&\vdots \\
\overline{Y^{(m-1)}(a+kh)} &= g_{m-3}\left(a+kh, S_{|_{I_{k-1}}}(a+kh), S'_{|_{I_{k-1}}}(a+kh)\right).
\end{aligned} \tag{18}$$

Thus, the matrix spline $S(x) \in \mathcal{C}^2\left(\bigcup_{j=0}^{k} I_j\right)$ is solution of the differential equation (6) at point $x = a+kh$. In order to find A_k, we impose the additional requirement that $S_{|_{I_k}}(x)$ satisfies equation (6) 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

$$\begin{aligned}
A_k = \tfrac{(m-2)!}{h^{m-2}} \Bigg[f\Bigg(& a+(k+1)h, S_{|_{I_{k-1}}}(a+kh) + S'_{|_{I_{k-1}}}(a+kh)h + \cdots + \\
& + \tfrac{h^{m-1}}{(m-1)!}\overline{Y^{(m-1)}(a+kh)} + \tfrac{h^m}{m!}A_k, S'_{|_{I_{k-1}}}(a+kh) + \overline{Y''(a+kh)}h + \cdots \\
& + \tfrac{h^{m-2}}{(m-2)!}\overline{Y^{(m-1)}(a+kh)} + \tfrac{h^{m-1}}{(m-1)!}A_k \Bigg) - \overline{Y''(a+kh)} \\
& - \overline{Y^{(3)}(a+kh)}h - \cdots - \tfrac{h^{m-3}}{(m-3)!}\overline{Y^{(m-1)}(a+kh)} \Bigg].
\end{aligned} \tag{19}$$

Eqs. (13) and (16) are just a particular case of the final result (19), letting $k = 0$ and $k = 1$.

We are now in the position to demonstrate the uniqueness of equation (19) 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

$$\begin{aligned}
g(T) = \tfrac{(m-2)!}{h^{m-2}} \Bigg[f\Bigg(& a+(k+1)h, S_{|_{I_{k-1}}}(a+kh) + S'_{|_{I_{k-1}}}(a+kh)h + \cdots + \\
& + \tfrac{h^{m-1}}{(m-1)!}\overline{Y^{(m-1)}(a+kh)} + \tfrac{h^m}{m!}T, S'_{|_{I_{k-1}}}(a+kh) + \\
& + \overline{Y''(a+kh)}h + \cdots + \tfrac{h^{m-2}}{(m-2)!}\overline{Y^{(m-1)}(a+kh)} + \tfrac{h^{m-1}}{(m-1)!}T \Bigg) \\
& - \overline{Y''(a+kh)} - \overline{Y^{(3)}(a+kh)}h - \cdots - \tfrac{h^{m-3}}{(m-3)!}\overline{Y^{(m-1)}(a+kh)} \Bigg].
\end{aligned} \tag{20}$$

Observe that relation (19) 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 (20) of g in combination with the global Lipschitz's condition (7) 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 (19) 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 1 : 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 [16]. More precisely, 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. (6).

Remark 2 : For carrying out the derivatives in (10) and (11), one may make extensive use of standard symbolic software, such as Mathematica etc.

3. Algorithms and MatLab functions

For solving Equation (6), we consider the initial matrix value problem

$$\left.\begin{array}{l} Y''(x) = f(x, Y(x), Y'(x)) \\ Y(x_k) = Y_0 \ , \ Y'(x_k) \ = \ Y_1 \end{array}\right\} x_k \le x \le x_k + h, \tag{21}$$

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 degree 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

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

with all derivatives $Y^{(i)}$ computed by (10) and (11). If we substitute the above expressions in (21), we obtain

$$B_k^{(2)} + \frac{h^{m-2}}{(m-2)!} A_k = f\left(x, B_k^{(0)} + \frac{h^m}{m!} A_k, B_i^{(1)} + \frac{h^{m-1}}{(m-1)!} A_k\right). \tag{22}$$

Then matrix A_k can be obtained by solving (22).

The MatLab function **splin2order** computes the solution of (6) by a fixed-point method or another method, as for example the Newton method (for online availability of the software, see Ref. [14]). The function used for solving (22) 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

$$Y''(t) + A_1 Y'(t) + A_0 Y(t) = 0, \quad t \in [a, b], \tag{23}$$

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 [14]). If (23) 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 $Y''(t)+A_0Y(t) = 0$. 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-degree splines constructed by the method proposed in [22]. 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 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

$$\begin{cases} y' &= f(x, y),\ x \in [a, b], \\ y(a) &= y_0, \end{cases}$$

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

We have also implemented specific MatLab functions in order to solve Eq. (6) with fixed step size based on classical methods: Nyström of order 4 [13, p. 284], an extrapolation method [13, p. 294] and an explicit Störmer method [13, 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**. Throughout the calculations for the numerical error estimates the 2-norm has been used, except for Table 2 of Subsection 4.2 where the Frobenius norm is employed.

4.1. *A non-linear vector system*

Consider the following non-linear vector differential system

$$\left.\begin{array}{l} y_1''(x) = 1-\cos(x)+\sin(y_2'(x))+\cos(y_2'(x)) \\ y_2''(x) = \frac{1}{4+y_1(x)^2}-\frac{1}{5-\sin^2(x)} \\ y_1(0)=1, \qquad y_2(0)=0, \\ y_1'(0)=0, \qquad y_2'(0)=\pi \end{array}\right\} \; 0 \le x \le 1. \tag{24}$$

with exact solution $y_1(x)=\cos(x), y_2(x)=\pi x$. The system Eq. (24) can be written in the more compact form

$$Y''(x) = F\left(x,Y,Y'\right), \quad Y(x)=\begin{pmatrix} y_1(x) \\ y_2(x) \end{pmatrix}, \quad Y(0)=\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \quad Y'(0)=\begin{pmatrix} 0 \\ \pi \end{pmatrix}, \tag{25}$$

where

$$F(x,Y,Y') = \begin{pmatrix} 1-\cos(x)+\sin(y_2'(x))+\cos(y_2'(x)) \\ \frac{1}{4+y_1(x)^2}-\frac{1}{5-\sin^2(x)} \end{pmatrix}. \tag{26}$$

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

$$\left.\begin{array}{l} \|F(x,Y_1,Y) \; - \; F(x,Y_2,Y)\| \; \le \|Y_1-Y_2\| \\ \|F(x,Y,Y_1) \; - \; F(x,Y,Y_2)\| \; \le 2\,\|Y_1-Y_2\| \end{array}\right\}, \; 0\le x\le 1 \;, Y, Y_1, Y_2 \in \mathbb{R}^2. \tag{27}$$

It is not difficult to evaluate $Y''(0)$ using (25), thus $Y''(0)=F(0,Y(0),Y'(0)) = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$. We can calculate $Y^{(3)}(0)$ using (10). This will illustrate how the calculation of (10) can be done. In this case, one gets

$$vec\,Y(x)=Y(x), vec\,Y'(x)=\begin{pmatrix} y_1'(x) \\ y_2'(x) \end{pmatrix}, \frac{\partial F(x,Y(x),Y'(x))}{\partial x} = \begin{pmatrix} \sin(x) \\ \frac{-2\sin(x)\cos(x)}{(5-\sin^2(x))^2} \end{pmatrix}.$$

On the other hand, we have

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

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

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

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

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

Thus, using (10) we have

$$\begin{aligned}Y^{(3)}(x)&=\begin{pmatrix}\sin(x)\\ \frac{-2\sin(x)\cos(x)}{(5-\sin^2(x))^2}\end{pmatrix}+\begin{pmatrix}y_1'(x)&0&y_2'(x)&0\\0&y_1'(x)&0&y_2'(x)\end{pmatrix}\begin{pmatrix}0\\ \frac{-2y_1(x)}{(4+y_1^2(x))^2}\\0\\0\end{pmatrix}\\&+\begin{pmatrix}A&0&B&0\\0&A&0&B\end{pmatrix}\begin{pmatrix}0\\0\\ \cos(y_2'(x))-\sin(y_2'(x))\\0\end{pmatrix}\\&=\begin{pmatrix}\sin(x)+\left(\cos(y_2'(x))-\sin(y_2'(x))\right)\left(\frac{1}{4+y_1^2(x)}-\frac{1}{5-\sin^2(x)}\right)\\ \frac{-2\sin(x)\cos(x)}{(5-\sin^2(x))^2}-\frac{2y_1(x)y_1'(x)}{(4+y_1^2(x))^2}\end{pmatrix}.\end{aligned}\quad(28)$$

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 (28), finally one concludes that $Y^{(3)}(0)=\begin{pmatrix}0\\0\end{pmatrix}$.

If we consider matrix splines of degree $m=6$, Theorem 2.1 requires to take $h<2.12404$, and thus we select $h=0.1$. In Table 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 degree of the spline using the same technique as in [22], we obtain the results given in Table 3, where in the last two intervals the maximum error increases dramatically.

For another test, the MatLab function **splin2order** was used. Figure 1 displays the approximation behavior for splines of degree $m=6$ with different step sizes

$h = 0.1$, $h = 0.01$ and $h = 0.001$, respectively. Table 4 compares **splin2order** (spline of degree $m = 9$) 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.

In conclusion, Table 5 illustrates the behaviour of the global error (using the 2-norm) produced by the function **splin2order** with varying step sizes and spline degrees $m = 3, 4, 5$. As expected, the global error decreases with smaller step size and with higher order for the splines. Also note that the actual global errors are well below the theoretically predicted error bounds.

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.

We consider the problem (23) 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

$$Y(t) = \begin{pmatrix} e^t & -1 + e^t - e^t t \\ 0 & e^t \end{pmatrix}, \quad t \in [0, b]. \tag{29}$$

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

$$\left.\begin{aligned} \|f(t, Y_1, Z) - f(t, Y_2, Z)\|_2 &\le \|Y_1 - Y_2\|_2 \\ \|f(t, Y, Z_1) - f(t, Y, Z_2)\|_2 &\le 2.28825\,\|Z_1 - Z_2\|_2 \end{aligned}\right\}. \tag{30}$$

If we consider again matrix splines of degree $m = 6$, 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 2.1. If we compare these results with those given by a sixth-degree spline using the same technique as in Ref. [22], we obtain comparable results for the first five intervals. However, our method truly improves the results in the last five intervals. Table 6 shows the results obtained in the last four intervals with the method from Ref. [22].

Figure 2 depicts the approximation behavior for splines of degree $m = 6$ with different step sizes $h = 0.1$, $h = 0.01$ and $h = 0.001$, respectively. Table 7 shows the results of **spline2linear** (spline of degree $m = 10$) 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.

4.3. *Incomplete linear second-order differential matrix equations*

As a second example, we study

$$Y''(t) + A\,Y(t) = 0, \tag{31}$$

where

$$A = \begin{pmatrix} 1 & 0 \\ 2 & 1 \end{pmatrix}, \quad Y(0) = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix} \quad \text{and} \quad Y'(0) = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}, \quad t \in [0, b]. \tag{32}$$

with the known analytical solution

$$Y(t) = \cos\left(\sqrt{A}t\right)Y(0) + \left(\sqrt{A}\right)^{-1}\sin\left(\sqrt{A}t\right)Y'(0). \tag{33}$$

As usual, $\sqrt{A}$ denotes the square root of the non-singular matrix A (see [11]). In this example, we choose $L_2 = 0$ and $L_1 \approx 2.82843$ as suggested in [22]. If we consider $b = 1$ and matrix splines of degree $m = 6$, according to Theorem 2.1, we need to take $h < 3.25678$ as in Ref. [23], so we selected $h = 0.1$. Table 8 provides all numerical results (rounded to the third relevant digit) for the maximum error in each interval.

If we consider high-degree splines, for example with $m = 6$, and if we use the same technique as in [22], 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 9), showing a bad performance for this method.

In the following test we consider $b = 5$. Figure 3 depicts the approximation behavior for splines of sixth degree with different step sizes $h = 0.1$, $h = 0.01$ and $h = 0.001$, respectively. Table 10 shows the results of function **splin2lineari** (spline of degree $m = 10$) 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 study 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 errors only of the order $O(h^{m-1})$, where m is the degree of the spline. By adjusting the step size h and selecting a corresponding higher-degree 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 paper. 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.

Acknowledgements

The authors wish to thank for financial support by the Universidad Politécnica de Valencia under grant PAID-06-11-2020. We also appreciate the valuable comments made by the anonymous referee to improve the original manuscript.

References

[1] 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), pp. 861–869.
[2] 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), pp. 195–204.
[3] U. Ascher, S. Pruess, and R.D. Russell, *On spline basis selection for solving differential equations*, SIAM J. Numer. Anal. 20 (1983), pp. 121–142.
[4] H. Brunner, *On the divergence of collocation solutions in smooth piecewise polynomial spaces for volterra integral equations*, BIT Numerical Mathematics 44 (2004), pp. 631–650.
[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), pp. 65–78.
[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), pp. 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), pp. 657–669.
[8] E. Defez, M.M. Tung, J. Ibáñez, and J. Sastre, *Approximating and computing nonlinear matrix differential models*, Math. Comput. Model. 55 (2012), pp. 2012–2022.
[9] T.M. Flett, *Differential Analysis*, Cambridge University Press, Cambridge, UK, 1980.
[10] C. Froese, *Numerical solutions of the hartree-fock equations*, Can. J. Phys. 41 (1963), pp. 1895–1910.
[11] G.H. Golub and C.F. Van Loan, *Matrix Computations*, 2nd ed., The Johns Hopkins University Press, Baltimore, MD, USA, 1989.
[12] A. Graham, *Kronecker products and matrix calculus with applications*, John Wiley & Sons, New York, USA, 1981.
[13] E. Hairer, S.P. Nørsett, and G. Wanner, Springer, Berlin 2000.
[14] J. Ibáñez, *MatLab Implementation for Matrix Splines (mims)*, URL `http://personales.upv.es/jjibanez/MIMS.html`.
[15] 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), pp. 299–320.
[16] F.R. Loscalzo and T.D. Talbot, *Spline function approximations for solutions of ordinary differential equations*, SIAM J. Numer. Anal. 4 (1967), pp. 433–445.
[17] P. Marzulli, *Global error estimates for the standard parallel shooting method*, J. Comput. Appl. Math. 34 (1991), pp. 233–241.
[18] G. Micula, *Approximate solutions of the differential equation $y'' = f(x, y)$ with spline functions*, Math. Comp. 27 (1973), pp. 807–816.
[19] G. Micula and A. Revnic, *An implicit numerical spline method for systems for ode's*, Appl. Math. Comput. 111 (2000), pp. 121–132.
[20] J.M. Ortega, *Numerical Analysis: A Second Course*, Academic Press, New York, 1972.
[21] B.W. Shore, *Comparison of matrix methods to the radii schrödinger eigenvalue equation: The morse potential*, J. Chemical Physics 59 (1971), pp. 6450–6463.
[22] M.M. Tung, E. Defez, and J. Sastre, *Numerical solutions of second-order matrix models using cubic-matrix splines*, Comput. Math. Appl. 56 (2008), pp. 2561–2571.
[23] M.M. Tung, L. Soler, E. Defez, and A. Hervás, *Cubic-matrix splines and second-order matrix model*, in *The 14th European Conference on Mathematics for Industry (ECMI 2006)*, Springer Verlag, Universidad Carlos III de Madrid, Spain, 2006.
[24] J.F. Zhang, *Optimal control for mechanical vibration systems based on second-order matrix equations*, Mechanical Systems and Signal Processing 16 (2002), pp. 61–67.

Table 1. MatLab solvers used in the tests.

MatLab Solver	Problem	Method
ode45	non-stiff differential equations	Runge-Kutta
ode45	non-stiff differential equations	Runge-Kutta
ode23	non-stiff differential equations	Runge-Kutta
ode113	non-stiff differential equations	Adams

Table 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 $m = 6$. The maximum error is calculated by using the Frobenius norm.

interval max. 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 3. Maximum 2-norm error using method [22] with $m = 6$ for the the test problem of Subsection 4.1.

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

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. List of 2-norm errors for the function **splin2order** used in numerical example 4.1, varying the step size h and the degree m of the spline.

	$m = 3$	$m = 4$	$m = 5$
$h = 0.1$	$3.051299 \cdot 10^{-5}$	$3.054623 \cdot 10^{-7}$	$3.054663 \cdot 10^{-9}$
$h = 0.01$	$4.652307 \cdot 10^{-6}$	$4.657409 \cdot 10^{-9}$	$4.623465 \cdot 10^{-12}$
$h = 0.001$	$8.471262 \cdot 10^{-9}$	$7.555910 \cdot 10^{-13}$	$5.520485 \cdot 10^{-14}$

Table 6. Maximum 2-norm error using method [22] with $m = 6$ for the test problem of Subsection 4.2.

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

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 8. 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 $m = 6$.

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 9. Maximum 2-norm error using method [22] for $m = 6$ for the incomplete second-order differential system Eq. (31).

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 10. Relative errors for the test problem of Subsection 4.3.

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	$2.026853 \cdot 10^{-15}$

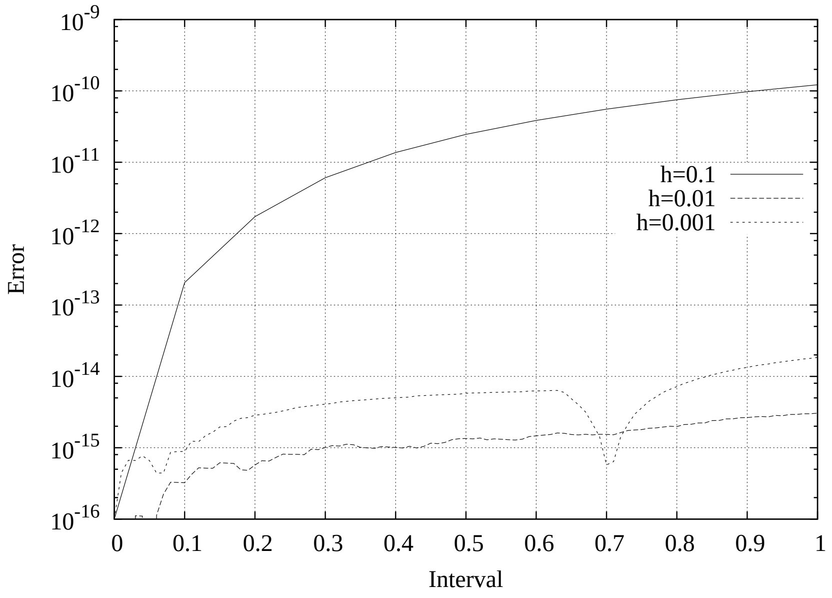

Figure 1. Relative errors for the test problem of Subsection 4.1 with fourth-order splines ($m = 6$), using MatLab function **splin2order** with $h = 0.1$, $h = 0.01$ and $h = 0.001$, respectively.

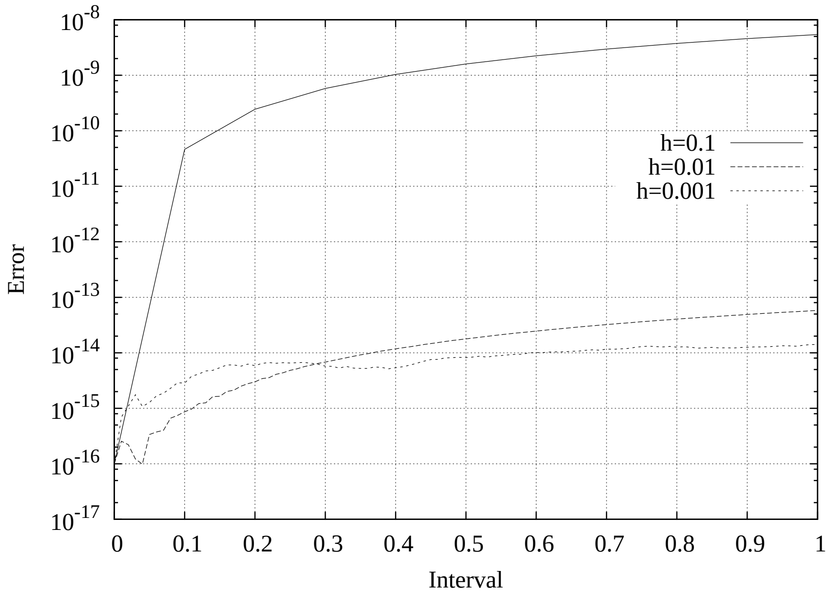

Figure 2. Relative errors for the test problem of Subsection 4.2 with fifth-order splines ($m = 6$), using MatLab function with **splin2linear** $h = 0.1$, $h = 0.01$ and $h = 0.001$, respectively.

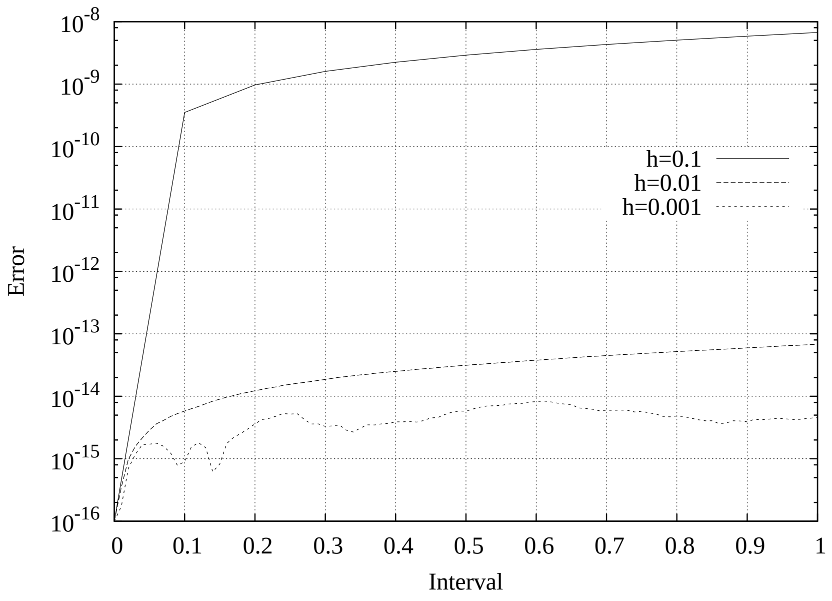

Figure 3. Relative errors for the test problem of Subsection 4.3 with fourth-order splines ($m = 6$) , using MatLab function **splin2lineari** with $h = 0.1$, $h = 0.01$ and $h = 0.001$, respectively.