mathematics

MDPI

Article

Efficient Evaluation of Matrix Polynomials beyond the Paterson–Stockmeyer Method

Jorge Sastre [1] **and Javier Ibáñez** [2,*]

[1] Instituto de Telecomunicaciones y Aplicaciones Multimedia, Universitat Politècnica de València, Camino de Vera s/n, 46022 Valencia, Spain; jsastrem@upv.es
[2] Instituto de Instrumentación para Imagen Molecular, Universitat Politècnica de València, Camino de Vera s/n, 46022 Valencia, Spain
* Correspondence: jjibanez@upv.es

Abstract: Recently, two general methods for evaluating matrix polynomials requiring one matrix product less than the Paterson–Stockmeyer method were proposed, where the cost of evaluating a matrix polynomial is given asymptotically by the total number of matrix product evaluations. An analysis of the stability of those methods was given and the methods have been applied to Taylor-based implementations for computing the exponential, the cosine and the hyperbolic tangent matrix functions. Moreover, a particular example for the evaluation of the matrix exponential Taylor approximation of degree 15 requiring four matrix products was given, whereas the maximum polynomial degree available using Paterson–Stockmeyer method with four matrix products is 9. Based on this example, a new family of methods for evaluating matrix polynomials more efficiently than the Paterson–Stockmeyer method was proposed, having the potential to achieve a much higher efficiency, i.e., requiring less matrix products for evaluating a matrix polynomial of certain degree, or increasing the available degree for the same cost. However, the difficulty of these family of methods lies in the calculation of the coefficients involved for the evaluation of general matrix polynomials and approximations. In this paper, we provide a general matrix polynomial evaluation method for evaluating matrix polynomials requiring two matrix products less than the Paterson-Stockmeyer method for degrees higher than 30. Moreover, we provide general methods for evaluating matrix polynomial approximations of degrees 15 and 21 with four and five matrix product evaluations, respectively, whereas the maximum available degrees for the same cost with the Paterson–Stockmeyer method are 9 and 12, respectively. Finally, practical examples for evaluating Taylor approximations of the matrix cosine and the matrix logarithm accurately and efficiently with these new methods are given.

Keywords: efficient; matrix polynomial evaluation; matrix function; Taylor approximation; cosine; logarithm

Citation: Sastre, J.; Ibáñez, J. Efficient Evaluation of Matrix Polynomials beyond the Paterson–Stockmeyer Method. *Mathematics* **2021**, *9*, 1600. https://doi.org/10.3390/math9141600

Academic Editors: Luca Gemignani

Received: 14 May 2021
Accepted: 1 July 2021
Published: 7 July 2021

1. Introduction

The authors of [1] presented a new family of methods for the evaluation of matrix polynomials more efficiently than the state-of-the-art method from [2] by Paterson and Stockmeyer (see [3], Section 4.2). These methods are based on the multiplication of matrix polynomials to get a new matrix polynomial with degree given by the sum of the degrees of the original matrix polynomials. The main difficulty in these methods lies in obtaining the coefficients involved for the evaluation of general matrix polynomials. In this sense, the authors of [1] (Section 3) gave two concrete general methods for evaluating matrix polynomials requiring one less matrix product than the Paterson–Stockmeyer method. Regarding the cost of evaluating matrix polynomials, since the cost of a matrix product, denoted by M, is $O(n^3)$ for $n \times n$ matrices, and both the cost of the sum of two matrices and the cost of a product of a matrix by a scalar are $O(n^2)$, similarly to [3] (Section 4.2) the

overall cost of the evaluation of matrix polynomials will be approximated asymptotically by the total number of matrix products.

The stability of the two methods from [1] (Section 3) was also analyzed and applications to the Taylor approximation of the exponential and cosine matrix functions were given.

The two general polynomial evaluation methods from [1] (Section 3) named above were applied in [4,5] and for the evaluation of Taylor polynomial approximations of the matrix exponential and the matrix cosine more efficiently than the state-of-the-art Padé methods from [6,7].

Moreover, the authors of [1] (Example 5.1) provided a polynomial evaluation formula for computing the matrix exponential Taylor approximation of degree 15 with cost $4M$, whereas the maximum available degree for that cost using the two general methods from [1] (Section 3) named above is 12. Based on this example, the authors of [5] (Section 3) proposed other new particular evaluation formulae for computing the matrix exponential Taylor polynomial approximations of degrees 15, 21, 24, and 30 that had cost $4M$, $5M$, $6M$, and $7M$, respectively. In [8], the authors proposed a particular method for evaluating the matrix exponential Taylor approximations with the lower degrees 12, 18, and 22 with cost $4M$, $5M$, and $6M$, respectively (see [8], Table 3). Note that general methods for the evaluation of matrix polynomials more efficiently than Paterson–Stockmeyer method are provided in [1], whereas [8] deals with the concrete case of the matrix exponential Taylor approximation. Moreover, (7) from [9] is equivalent to the particular case of taking $s = 1$ in (62)–(65) from [1]. This work was submitted in October 2016, and evaluation Formulas (62)–(65) were first introduced as (25)–(28) of the early unpublished version [10] of February 2016, whereas [8] is an updated version of the unpublished reference [9] released more than one year later, i.e., October 2017.

In this paper, we generalize the results from [5] (Section 3) given there for the particular case of the matrix exponential Taylor approximation of degrees 15, 21, 24, and 30. These generalizations consist of giving general procedures for:

- Evaluating polynomial approximations of matrix functions of degrees 15 and 21 with cost $4M$ and $5M$, respectively.
- Evaluating matrix polynomials of degrees $6s$ with $s = 3, 4, \ldots$ with cost $(s+2)M$.
- Evaluating matrix polynomials of degrees greater than 30 with two matrix products less than the Paterson–Stockmeyer method.

Finally, examples for computing Taylor approximations of the matrix cosine and the matrix logarithm efficiently and accurately using those evaluation formulae are given.

Regarding Taylor approximations, if

$$f(X) = \sum_{i \geq 0} a_i X^i,$$

is the Taylor series of the matrix function $f(\cdot)$, where $X \in \mathbb{C}^{n \times n}$, then

$$T_m(X) = \sum_{i=0}^{m} a_i X^i,$$

is its Taylor approximation of order m (for the convergence of matrix Taylor series, see Theorem 4.7 of [3], p. 76).

From [11] (Section 1), a matrix $X \in \mathbb{C}^{n \times n}$ is a logarithm of $B \in \mathbb{C}^{n \times n}$ if $e^X = B$. Therefore, any nonsingular matrix has infinitely many logarithms and we will focus on the principal logarithm, denoted by $\log(B)$. For a matrix $B \in \mathbb{C}^{n \times n}$ with no eigenvalues on R^- the principal logarithm is the unique logarithm whose eigenvalues have imaginary parts

lying in the interval $(-\pi, \pi)$. Therefore, in the given examples, we will assume that B has no eigenvalues on R^- and we will take the logarithm Taylor series

$$\log(B) = \log(I - A) = -\sum_{i>1} A^i/i, \text{ where } \mathrm{A} = \mathrm{I} - \mathrm{B}. \tag{1}$$

The exponential matrix has been studied in numerous papers (see [3] (Chap. 10), and [5,6,8,12] and the references therein). This matrix function can be defined by

$$\exp(X) = \sum_{i\geq 0} \frac{X^i}{i!}. \tag{2}$$

The matrix cosine has received attention recently (see [4,7] and the references therein). This matrix function can be defined by

$$\cos(X) = \sum_{i\geqslant 0} (-1)^i \frac{X^{2i}}{(2i)!} = \sum_{i\geqslant 0} (-1)^i \frac{Y^i}{(2i)!}, \quad Y = X^2. \tag{3}$$

Note that if we truncate the Taylor series on the right-hand side of (3) by the term $i = m$, then the order of the corresponding cosine Taylor approximation is $2m$.

Regarding the cost in matrix rational approximations, note that the multiplication by the corresponding matrix inverse is calculated by solving a multiple right-hand side linear system. From [13] (Appendix C), it follows that the cost of the solution of multiple right-hand side linear systems $AX = B$, where matrices A and B are $n \times n$, denoted by D (see [14], p. 11940) is

$$D \approx 4/3M. \tag{4}$$

Therefore, using (4), the cost of computing rational approximations will be also given in terms of M.

In this article, the following notation will be used: $\lceil x \rceil$ denotes the smallest integer greater than or equal to x, and $\lfloor x \rfloor$ the largest integer less than or equal to x. u denotes the unit roundoff in IEEE double precision arithmetic (see [15], Section 2.2). The set of positive integers is denoted as $\mathbb{N}$. The set of real and complex matrices of size $n \times n$ are denoted, respectively, by $\mathbb{R}^{n\times n}$ and $\mathbb{C}^{n\times n}$. The identity matrix for both sets is denoted as I. The dependence of a variable y on the variables

$$x_1, x_2, \ldots, x_n$$

is denoted by

$$y = y(x_1, x_2, \ldots, x_n).$$

In Section 2, we recall some results for computing matrix polynomials using the Paterson–Stockmeyer method and summarize the matrix polynomial evaluation methods from [1]. In Section 3, we describe the general methods for computing polynomial approximations of degrees 15, 21, and $6s$ with $s = 3, 4, \ldots$ and give examples for the Taylor approximation of the cosine and logarithm matrix functions. Finally, in Section 4, we give some conclusions. In this paper, we provide a method to evaluate matrix polynomials with two matrix products less than the Paterson–Stockmeyer method and one matrix product less than the methods from [1] (Section 3). Moreover, in this paper, we provide methods to evaluate polynomial approximations of matrix functions of degrees 15 and 21 with cost $3M$ and $4M$. These methods are interesting because the maximum available degrees using the other method proposed in this paper are 12 and 18, respectively. All of the new methods proposed can be used in the applications for computing approximations of matrix functions or evaluating matrix polynomials more efficiently than using the state-of-the-art methods.

2. Efficient Evaluation of Matrix Polynomials

2.1. Paterson–Stockmeyer Method

The Paterson–Stockmeyer method [2] for computing a matrix polynomial

$$P_m(A) = \sum_{i=0}^{m} c_i A^i, \tag{5}$$

consists of calculating $P_m(A)$ as

$$\begin{aligned} PS_m(A) = & \Big(\Big(\cdots\Big(c_m A^s + c_{m-1}A^{s-1} + \ldots + c_{m-s+1}A + c_{m-s}I\Big) \\ & \times A^s + c_{m-s-1}A^{s-1} + c_{m-s-2}A^{s-2} + \ldots + c_{m-2s+1}A + c_{m-2s}I\Big) \\ & \times A^s + c_{m-2s-1}A^{s-1} + c_{m-2s-2}A^{s-2} + \ldots + c_{m-3s+1}A + c_{m-3s}I\Big) \\ & \vdots \\ & \times A^s + c_{s-1}A^{s-1} + c_{s-2}A^{s-2} + \ldots + c_1 A + c_0 I, \end{aligned} \tag{6}$$

where $PS_m(A)$ denotes the Paterson–Stockmeyer evaluation Formula (6) and $s > 0$ is an integer that divides m. Given a number of matrix products, the maximum degrees of $P_m(A)$ that are available using the Paterson–Stockmeyer method are the following:

$$m = s^2, \text{ and } m = s(s+1), \tag{7}$$

where $s \in \mathbb{N}$, denoted by m^*, $m^* = \{1, 2, 4, 6, 9, 12, \ldots\}$ [14] (Section 2.1). The cost C_{PS} for computing (6) for the values of m^* are given by [14] (Equation (5)), which appear in [14] (Table 1). In [16], the optimality of the rule $m^* = (C_{PS} - s + 2)s$, where $s = \lfloor C_{PS}/2 \rfloor + 1$, was demonstrated. This rule gives the same results as (7), since if C_{PS} is even then $C_{PS} = 2s - 1$, and in that case $m^* = s(s+1)$, and if C_{PS} is odd then $C_{PS} = 2s$, and then $m^* = s^2$. Note that, for positive integers $m \notin m^*$, $P_m(A) = PS_{m_0}(A)$ can be evaluated using (6) taking $m_0 = \min\{m_1 \in m^*, m_1 > m\}$ and setting some coefficients as zero [1] (Section 2.1).

2.2. General Polynomial Evaluation Methods beyond the Paterson–Stockmeyer Method

The authors of [1] (Example 3.1) give a method to compute $P_8(A)$ from (5) with a cost of $3M$ with the following evaluation formulae

$$y_{02}(A) = A^2(q_4 A^2 + q_3 A), \tag{8}$$

$$\begin{aligned} y_{12}(A) = & (y_{02}(A) + r_2 A^2 + r_1 A)(y_{02}(A) + s_2 A^2) \\ & + s_0 y_{02}(A) + t_2 A^2 + t_1 A + t_0 I, \end{aligned} \tag{9}$$

where q_4, q_3, r_2, r_1, s_2, s_0, t_2, t_1, and t_0 are complex numbers. In order to compute (5) with $m = 8$, if we equate $y_{12}(A) = P_m(A)$ from (5), then the system of eight equations with eight coefficients from (16)–(24) from [1] arises. In this system, some coefficients can be obtained directly from the polynomial $P_m(A)$ coefficients as

$$q_4 = \pm\sqrt{c_8}, \tag{10}$$

$$q_3 = c_7/(2q_4), \tag{11}$$

$$t_2 = c_2, \tag{12}$$

$$t_1 = c_1, \tag{13}$$

$$t_0 = c_0, \tag{14}$$

and the remaining equations can be reduced by variable substitution to a quadratic equation on s_2. This equation gives two solutions for $q_4 = \sqrt{c_8}$ and two more solutions for $q_4 =$

$-\sqrt{c_8}$. The remaining coefficients can be obtained from s_2, q_4, and q_3. From (11), one gets $q_4 \neq 0$ giving condition

$$c_8 \neq 0, \tag{15}$$

for coefficient c_8 in $P_m(A)$ from (5).

In order to check the stability of the solutions of q_i, r_i, and s_i rounded to IEEE double precision arithmetic, the authors of [1] (Example 3.1) proposed to compute the relative error for each coefficient c_i, for $i = 3, 4, \ldots, 8$ substituting those solutions into the original system of Equations (16)–(24) from [1]. For instance, from (10), it follows that the relative error for c_8 using q_4 rounded to IEEE double precision arithmetic is

$$|c_8 - \tilde{q}_4^2|/|c_8|,$$

where $\tilde{q}_4$ is the value $q_4 = \pm\sqrt{c_8}$ rounded to IEEE double precision arithmetic. Then, if the relative errors for all the expressions of the coefficients c_i are of order the unit roundoff in IEEE double precision arithmetic, i.e., $u = 2^{-53} \approx 1.11 \times 10^{-16}$, then the solution is stable.

In [1] (Table 4), one of the solutions rounded to IEEE double precision arithmetic for evaluating the Taylor polynomial of the exponential and cosine functions is shown. These solutions were substituted into the original system of equations to calculate the relative error for c_i, for $i = 3, 4, \ldots, 8$ (see [1], Example 3.1), giving a relative error of order u, turning out to be stable solutions. Moreover, the numerical tests from [1] (Example 3.2) and [4,5] also show that if the relative error for each coefficient is $O(u)$, then the polynomial evaluation formulae are accurate, and if the relative errors are $O(10u)$ or greater, then the polynomial evaluation formulae are not so accurate.

The authors of [1] (Section 3) also provided a more general method for computing matrix polynomials $P_m(A)$ from (5) of degree $m = 4s$ based on the evaluation formulae

$$y_{0s}(A) = A^s \sum_{i=1}^{s} q_{s+i} A^i, \tag{16}$$

$$\begin{aligned} y_{1s}(A) = &\left(y_{0s}(A) + \sum_{i=1}^{s} r_i A^i \right) \left(y_{0s}(A) + \sum_{i=2}^{s} s_i A^i \right) \\ &+ s_0 y_{0s}(A) + \sum_{i=0}^{s} t_i A^i, \end{aligned} \tag{17}$$

where $s \geq 2$, q_{s+i}, r_i, s_i and t_i are scalar coefficients, $q_{2s} = \pm\sqrt{c_{4s}} \neq 0$ and then $c_{4s} \neq 0$ for coefficient c_{4s} from $P_m(A)$. Note that A^i, $i = 2, 3, \ldots, s$ are computed only once. The degree and computing cost of $y_{1s}(A)$ are given by (36) of [1], i.e., $d_{y_{1s}} = 4s$ and $C_{y_{1s}} = s + 1$, $s = 2, 3, \ldots$, respectively. A general solution for the coefficients in (16) and (17) is given in [1] (Section 3), with the condition

$$c_{4s} \neq 0. \tag{18}$$

Given a cost $C(M)$, the maximum orders that can be reached when using the Formulae (16) and (17) and the Paterson–Stockmeyer method are shown in [1] (Table 5).

Proposition 1 from [1] (Section 3) shows a method for computing matrix polynomials combining the Paterson–Stockmeyer method with (17) as

$$\begin{aligned} z_{kps}(x) = &\Big(\Big(\cdots \Big(y_{ks}(x) x^s + a_{p-1} x^{s-1} + a_{p-2} x^{s-2} + \ldots + a_{p-s+1} x + a_{p-s} \Big) \\ &\times x^s + a_{p-s-1} x^{s-1} + a_{p-s-2} x^{s-2} + \ldots + a_{p-2s+1} x + a_{p-2s} \Big) \\ &\times x^s + a_{p-2s-1} x^{s-1} + a_{p-2s-2} x^{s-2} + \ldots + a_{p-3s+1} x + a_{p-3s} \Big) \\ &\vdots \\ &\times x^s + a_{s-1} x^{s-1} + a_{s-2} x^{s-2} + \ldots + a_1 x + a_0, \end{aligned} \tag{19}$$

where $k = 1$, p is a multiple of s and $y_{ks}(x) = y_{1s}(x)$ is evaluated using (16) and (17). This allows one to increase the degree of the polynomial to be evaluated. The degree of $z_{1ps}(A)$ and its computational cost are given by (53) of [1], i.e., $d_{z_{1ps}} = 4s + p$, $C_{z_{1ps}} = (1 + s + p/s)M$, respectively. Ref. [1] (Table 6) shows that evaluating a matrix polynomial using (19) requires one less product than using the Paterson–Stockmeyer Formula (6).

Proposition 2 from [1] (Section 5) gives general formulae more efficient than the formulae of the previous methods, whenever at least one solution for the coefficients in (62)–(65) from [1] (Prop. 2) exists so that $y_{ks}(x)$ is equal to the polynomial P_m to evaluate. The maximum polynomial degree and the computing cost if $x = A$, $A \in \mathbb{C}^{n\times n}$, are given by (66) of [1], i.e., $d_{y_{ks}} = 2^{k+1}s$, $C_{y_{ks}} = (s + k)$ where $d_{y_{ks}}$ increases exponentially while $C_{y_{ks}}$ increases linearly. (17) is a particular case of (65) from [1] where $k = 1$.

3. Three General Expressions for $y_{2s}(A)$

This section gives general procedures to obtain the coefficients of $y_{2s}(A)$ from (65) from [1] with $k = 2$, generalizing the results from [5] (Section 3) for the evaluation of the matrix exponential Taylor approximations of degrees 15, 21, 24, and 30, also giving formulae for evaluating matrix polynomials of orders $6s$, where $s = 2, 3, \ldots$

3.1. Evaluation of Matrix Polynomial Approximations of Order 15 with $y_{2s}(A)$, $s = 2$.

The following proposition allows to compute polynomial approximations of order 15 with cost $4M$. Note that from [1] (Table 8), the maximum available order with cost $4M$ is 9 for the Paterson–Stockmeyer method and 12 for the method given by (16) and (17).

Proposition 1. *Let $y_{12}(A)$ and $y_{22}(A)$ be*

$$y_{12}(A) = \sum_{i=2}^{8} c_i A^i, \tag{20}$$

$$\begin{aligned} y_{22}(A) = {} & (y_{12}(A) + d_2A^2 + d_1A)(y_{12}(A) + e_0y_{02}(A) + e_1A) \\ & + f_0y_{12}(A) + g_0y_{02}(A) + h_2A^2 + h_1A + h_0I, \end{aligned} \tag{21}$$

and let $P_{15}(A)$ be a polynomial of degree 15 with coefficients b_i

$$P_{15}(A) = \sum_{i=0}^{15} b_i A^i. \tag{22}$$

Then,

$$y_{22}(A) = \sum_{i=0}^{16} a_i A^i, \tag{23}$$

where coefficients a_i are functions of the following variables

$$a_i = a_i(c_8, c_7, \ldots, c_2, d_2, d_1, e_1, e_0, f_0, g_0, h_2, h_1, h_0), \quad i = 0, 1, \ldots, 16,$$

and there exist at least one set of values of the 16 coefficients c_8, c_7, …, c_2, d_2, d_1, e_1, e_0, f_0, g_0, h_2, h_1, h_0 so that

$$a_i = b_i, \quad i = 0, 1, \ldots, 15, \tag{24}$$

and

$$a_{16} = c_8^2, \tag{25}$$

provided the following conditions are fulfilled:

$$c_8 \neq 0, \tag{26}$$

$$3b_{15}^2 \neq 8b_{14}c_8^2, \tag{27}$$

$$27b_{15}^6c_8^{45/2} + 576b_{14}^2b_{15}^2c_8^{53/2} \neq 512b_{14}^3c_8^{57/2} + 216b_{14}b_{15}^4c_8^{49/2}. \tag{28}$$

Proof of Proposition 1. Note that $y_{12}(A)$ from (20) is a matrix polynomial of degree 8. Therefore, if condition (26) holds, then Example [1] (Example 3.1) gives four possible solutions for evaluating $y_{12}(A)$ using the evaluation Formulas (8) and (9) with cost $3M$. Similarly to [5] (Section 3.2), we will denote these four solutions as nested solutions.

Using (10) and (11), one gets that $y_{02}(A)$ from (8) can be written as

$$y_{02}(A) = \pm A^2(\sqrt{c_8}A^2 + c_7/(2\sqrt{c_8})A). \tag{29}$$

Then, taking the positive solution in (29), if we equate $y_{22}(A)$ from (23) to $P_{15}(A)$ from (22), we obtain the following nonlinear system with 16 coefficients c_i, $i = 2, 3, \ldots, 8$, d_2, d_1, e_1, e_0, f_0, g_0, h_2, h_1, h_0:

$$\begin{aligned}
a_{15} &= 2c_7c_8 = b_{15},\\
a_{14} &= {c_7}^2 + 2c_6c_8 = b_{14},\\
a_{13} &= 2c_5c_8 + 2c_6c_7 = b_{13},\\
a_{12} &= {c_6}^2 + c_8(c_4 + \sqrt{c_8}e_0) + c_4c_8 + 2c_5c_7 = b_{12},\\
a_{11} &= c_7(c_4 + \sqrt{c_8}e_0) + c_3c_8 + c_4c_7 + 2c_5c_6 + c_8\left(c_3 + \frac{c_7e_0}{2\sqrt{c_8}}\right) = b_{11},\\
a_{10} &= {c_5}^2 + c_6(c_4 + \sqrt{c_8}e_0) + \sum_{i=0}^{2} c_{2+i}c_{8-i} + c_7\left(c_3 + \frac{c_7e_0}{2\sqrt{c_8}}\right) + c_8(c_2 + d_2) = b_{10},\\
a_9 &= c_5(c_4 + \sqrt{c_8}e_0) + \sum_{i=0}^{2} c_{2+i}c_{7-i} + c_6\left(c_3 + \frac{c_7e_0}{2\sqrt{c_8}}\right) + c_7(c_2 + d_2) + c_8(d_1 + e_1) = b_9,\\
a_8 &= c_4(c_4 + \sqrt{c_8}e_0) + c_2c_6 + c_3c_5 + c_5\left(c_3 + \frac{c_7e_0}{2\sqrt{c_8}}\right)\\
&\quad + c_6(c_2 + d_2) + c_7(d_1 + e_1) + c_8f_0 = b_8,\\
a_7 &= c_3(c_4 + \sqrt{c_8}e_0) + c_2c_5 + c_4\left(c_3 + \frac{c_7e_0}{2\sqrt{c_8}}\right) + c_5(c_2 + d_2) + c_6(d_1 + e_1) + c_7f_0 = b_7,\\
a_6 &= c_2c_4 + c_3\left(c_3 + \frac{c_7e_0}{2\sqrt{c_8}}\right) + (c_2 + d_2)(c_4 + \sqrt{c_8}e_0) + c_5(d_1 + e_1) + c_6f_0 = b_6,\\
a_5 &= d_1\sqrt{c_8}e_0 + c_2c_3 + (c_2 + d_2)\left(c_3 + \frac{c_7e_0}{2\sqrt{c_8}}\right) + c_4(d_1 + e_1) + c_5f_0 = b_5,\\
a_4 &= d_1\frac{c_7e_0}{2\sqrt{c_8}} + c_2(c_2 + d_2) + c_3(d_1 + e_1) + c_4f_0 + \sqrt{c_8}g_0 = b_4,\\
a_3 &= e_1d_2 + c_2(d_1 + e_1) + c_3f_0 + \frac{c_7g_0}{2\sqrt{c_8}} = b_3,\\
a_2 &= d_1e_1 + c_2f_0 + h_2 = b_2,\\
a_1 &= h_1 = 1,\\
a_0 &= h_0 = 1.
\end{aligned} \tag{30}$$

This system of equations can be solved for a set of given variables b_i, $i = 1, 2, \ldots, 15$, using variable substitution with the MATLAB Symbolic Toolbox using the following MATLAB code fragment (we used MATLAB R2020a in all the computations):

```
% MATLAB code fragment 4.1:  solves coefficient c8 of
% the system of equations (30) for general coefficients bi
1 syms A c2 c3 c4 c5 c6 c7 c8 d1 d2 e0 e1 f0 g0 h2 h1 h0I
2 syms b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 b10 b11 b12 b13 b14 b15 b16
```

```
3 c=[c2;c3;c4;c5;c6;c7;c8];
4 b=[b16;b15;b14;b13;b12;b11;b10;b9;b8;b7;b6;b5;b4;b3;b2;b1;b0];
5 y0=A^2*(sqrt(c8)*A^2+c7/(2*sqrt(c8))*A); % y0≥ 0 from (29)
6 y1=sum(c.*A.^([2:8]'));
7 y2=(y1+d2*A^2+d1*A)*(y1+e0*y0+e1*A)+f0*y1+g0*y0+h2*A^2+h1*A+h0I;
8 [cy2,a1]=coeffs(y2,A);
9 cy2=cy2.';
10 v=[cy2 b a1.']%v shows the coefficients of each power of A
11 cy2=cy2(2:end)-b(2:end); %System of equations
12 c7s=solve(cy2(1),c7,'ReturnConditions',true); %c7s=f(c8,bi)
13 c7s.conditions %c8 ~= 0 condition for the existence of solutions
14 c7s=c7s.c7;
15 cy2=subs(cy2,c7,c7s);
16 c6s=solve(cy2(2), c6); %c6s depends on c8 bi
17 cy2=subs(cy2,c6,c6s);
18 c5s=solve(cy2(3), c5); %c5s depends on c8 bi
19 cy2=simplify(subs(cy2,c5,c5s));
20 symvar(cy2(4)) %cy2(4) depends on c8, c4, e0 bi
21 e0s=solve(cy2(4), e0);
22 cy2=simplify(subs(cy2,e0,e0s));
23 symvar(cy2(5)) %cy2(5) depends on c8, c3, c4, bi
24 c3s=solve(cy2(5), c3);
25 cy2=simplify(subs(cy2,c3,c3s));
26 symvar(cy2(6)) %cy2(6) depends only on c8, c2, d2, bi
27 d2s=solve(cy2(6), d2);
28 cy2=simplify(subs(cy2,d2,d2s));
29 symvar(cy2(7)) %cy2(7) depends only on c8, d1, e1, bi
30 d1s=solve(cy2(7), d1);
31 cy2=simplify(subs(cy2,d1,d1s));
32 symvar(cy2(8)) %cy2(8) depends only on c8, c4, f0, bi
33 f0s=solve(cy2(8), f0);
34 cy2=simplify(subs(cy2,f0,f0s));
35 symvar(cy2(9)) %cy2(9) depends only on c8, b7, b8,...,b15
36 c8s=solve(cy2(9), c8)
```

Since `cy2(9)` from the code fragment line 35 depends only on coefficients b_i, for $i = 7, 8, \ldots, 15$, and c_8, the solutions for c_8 are given by the zeros of equation `cy2(9)` with the condition given by MATLAB code fragment line 13, i.e., condition (26). `solve` function gives 16 solutions for c_8. They are the roots of a polynomial with coefficients depending on variables b_i, for $i = 7, 8, \ldots, 15$.

Once the 16 solutions for c_8 are obtained for concrete values of the coefficients b_i, $i = 0, 1, \ldots, 15$, the remaining variables can be obtained with the following MATLAB code fragment:

```
% MATLAB code fragment 4.2:  solves coefficient c2 of the
% system of equations (30) for general coefficients bi by
% using the solutions for coefficient c8 obtained using the
% MATLAB piece of code 4.1
1 symvar(cy2(10)) %cy2(10) depends on c8, c2, c4, bi
2 c4s=solve(cy2(10), c4) % two solutions depending on c8, c2, bi
3 cy2=simplify(subs(cy2,c4,c4s(1)))%change c4s(1) for c4s(2) for
  more solutions
4 symvar(cy2(11)) %cy2(11) depends on c8, c2, e1, bi
5 e1s=solve(cy2(11), e1)
```

```
6 cy2=simplify(subs(cy2,e1,e1s))
7 symvar(cy2(12)) %cy2(12) depends on c8, c2, g0, bi
8 g0s=solve(cy2(12), g0,'ReturnConditions',true)
9 g0s.conditions %conditions for the existence of solutions:
%3*b15^2 ~= 8*b14*c8^2 &
%27*b15^6*c8^(45/2) + 576*b14^2*b15^2*c8^(53/2) ~=
%512*b14^3*c8^(57/2) + 216*b14*b15^4*c8^(49/2) &
%c8 ~= 0
10 g0s=g0s.g0
11 cy2=simplify(subs(cy2,g0,g0s))
12 symvar(cy2(13)) %cy2(13) depends on c8, c2, bi
```

Since `cy2(13)` depends only on coefficients b_i, for $i = 3, 4, \ldots, 15$, c_8, and c_2, substituting the values obtained previously for c_8 in `cy2(13)`, the solutions for c_2 are given by the zeros of equation `cy2(13)` with the conditions given by line 9 from `MATLAB code fragment 4.2` when solving c_7 and g_0, given by (26)–(28). Both code fragments are available (http://personales.upv.es/jorsasma/Software/coeffspolm15plus.m (accessed on 24 June 2021)).

All of the coefficients c_7, c_6, ..., c_3, d_2, d_1, e_1, e_0, f_0, g_0, can be obtained from c_2, c_8 and b_i, $i = 0, 1, \ldots 15$, and then h_i can be obtained from the three last equations of system (30) as

$$h_2 = b_2 - d_1 e_1 - c_2 f_0, \tag{31}$$
$$h_1 = b_1, \tag{32}$$
$$h_0 = b_0. \tag{33}$$

Finally, using (20) and (21), coefficient a_{16} from (23) is given by (25).

Hence, for any values of the coefficients b_i, $i = 0, 1, \ldots, 15$, of the polynomial (22), then there exist at least one solution of system (30) giving a set of values of the coefficients from $y_{22}(A)$ from (20) and (21) so that (24) and (25) hold, provided conditions (26)–(28) are fulfilled. □

Given certain coefficients b_i, $i = 1, 2, \ldots, 15$ for $P_{15}(A)$ from (22), using MATLAB code fragments 4.1 and 4.2, one can get typically more than one solution of system (30). Moreover, if we take the negative sign in (29) another set of solutions fulfilling (24) can be obtained. For each of those solutions there are also different solutions for the nested solutions for evaluating (20) with the solutions from Example [1] (Example 3.1).

For each of those solutions, coefficient a_{16} from $y_{22}(A)$ in (23) is given by (25). For the particular case of the matrix exponential Taylor approximation from [5] (p. 209), there were two real solutions of c_8 giving

$$|c_8^2 - 1/16!|16! \approx 0.454, \tag{34}$$
$$|c_8^2 - 1/16!|16! \approx 2.510. \tag{35}$$

Therefore, we selected the first solution (34) since both solutions were stable according to the stability study from Section 2.2 (see [1], p. 243), but (34) had a lower error for a_{16} with respect to the corresponding Taylor coefficient $1/16!$. Then, considering exact arithmetic, one gets that the matrix exponential approximation from $y_{22}(A)$ in evaluation Formulas (10)–(12) from [5] (p. 209) with the coefficients from [5] (Table 3) is more accurate than the exponential Taylor approximation of order 15. For that reason, the corresponding Taylor approximation order was denoted by $m = 15+$ in [5] (Section 4).

Recently, in [17], an evaluation formula of the type given in Proposition 1 was used to evaluate a Taylor polynomial approximation of degree 15+ of the hyperbolic tangent. However, in this case, all the solutions obtained were complExample We tried different configurations of the evaluation formulae giving degree 15+, but all of them gave complex

solutions. Then, we proposed the similar evaluation Formula (11) from [17] (p. 6) with degree 14+ that did give real solutions. Similarly to (34), in the case of the hyperbolic tangent, the relative error of the coefficients a_i, $i = 15$, and 16 was also lower than 1—concretely, 0.38 and 0.85, respectively (see [17], p. 6). This method was compared to the Paterson–Stockmeyer method being noticeably more efficient without affecting the accuracy (see [17], Section 3) for details. Proposition 1 allows us to evaluate polynomial approximations of degree 15 not only for the matrix exponential or the hyperbolic tangent but also for other matrix functions. If all the given solutions were complex, we can modify the formula to evaluate approximation formulae with a lower degree, such as 14+, to check if they give real solutions.

Example 1. *In [4] (Section 2), we showed that the solutions for the coefficients of the polynomial evaluation method similar to [5] (Section 3.2) of the matrix cosine Taylor approximation of order $2m = 30+$ were not stable, giving poor accuracy results. Using Proposition 1, this example gives a stable solution for calculating a Taylor-based approximation of the matrix cosine with a combination of formula (21) with the Paterson–Stockmeyer method from (19). Setting $k = p = s = 2$ in (19) and $y_{ks} = y_{22}$ from (21), one gets*

$$z_{222}(B) = y_{22}(B)B^2 - B/2 + I = P_{17}(B) = \sum_{i=0}^{17} b_i B^i + a_{18}B^{18}, \tag{36}$$

where $B = A^2$ and $z_{222}(B)$ is a Taylor-based approximation of the matrix cosine (3) of order $2m = 34+$, i.e., $b_i = (-1)^i/(2i)!$ for $i = 0, 1, \ldots, 17$, coefficient a_{18} is given by $a_{18} = c_8^2$ (see (25)).

MATLAB code fragment 4.1 was used for obtaining all the real solutions of c_8. Then, MATLAB code fragment 4.2 was used with these solutions taking solution 1 for coefficient c_4 in line 3 of the MATLAB code fragment 4.2. Then, we obtain the equation `cy2(13)` *from the code fragment in line 12 depending on c_2 and c_8. This equation was solved for every real solution of c_8, using the MATLAB Symbolic Math Toolbox with variable precision arithmetic. Finally, we obtained the nested solutions for computing (20) with (8) and (9) with $q_4 > 0$ from (10).*

The real solutions of system (30) rounded to IEEE double precision arithmetic explored in [4] (Section 2) gave errors of order $\geq 10^{-14}$, greater than the unit roundoff in IEEE double precision arithmetic $u = 2^{-53} \approx 1.11 \times 10^{-16}$. Using MATLAB code fragments 4.1 and 4.2, we checked that there is no solution with a lower error. Then, according to the stability check from Section 2.2, the solutions are unstable, and we checked in [4] that they gave poor accuracy results. However, using Proposition 1, for $2m = 34+$, we could find two real solutions of system (30) giving a maximum error of order u. For those two solutions, a_{18} gave

$$|a_{18} - 1/36!|36! \approx 0.394, \tag{37}$$
$$|a_{18} - 1/36!|36! \approx 16.591, \tag{38}$$

respectively. Therefore, the solution (37) giving the lowest error was selected. Table 1 gives the corresponding coefficients in IEEE double precision arithmetic from (8) and (9) for computing (20) with three matrix products, and the rest of the needed coefficients for computing $y_{22}(B)$ from (21) with $s = 2$, given finally by

$$y_{02}(B) = B^2(q_4B^2 + q_3B), \tag{39}$$
$$y_{12}(B) = \left(y_{02}(B) + r_2B^2 + r_1B\right)\left(y_{02}(B) + s_2B^2\right) + s_0y_{02}(B) + t_2B^2, \tag{40}$$
$$y_{22}(B) = (y_{12}(B) + d_2B^2 + d_1B)(y_{12}(B) + e_0y_{02}(B) + e_1B) + f_0y_{12}(B) + g_0y_{02}(B) + h_2B^2 + h_1B + h_0I. \tag{41}$$

Using (39)–(41) with the coefficients from Table 1 and (36), a matrix cosine Taylor approximation of order $2m = 34+$ can be computed in IEEE double precision arithmetic with a cost of six

matrix products, i.e., $B = A^2$, B^2, *three for evaluating* (39)–(41), *and one more for evaluating* (36). *The maximum available and stable order given in [4] (Section 2) with six matrix products was* $2m = 30$. *The coefficients from Table 1 were computed with variable precision arithmetic with a precision of 32 and 250 decimal digits to check its correctness, giving the same results.*

Taking into account (3) *and the selection of the solution in* (37), *in exact arithmetic, one gets*

$$z_{222}(x) = \sum_{i=0}^{17} (-1)^i \frac{x^{2i}}{(2i)!} + a_{18} x^{36}. \tag{42}$$

where, using (39)–(41), *one gets* $a_{18} = q_4^4$.

Table 1. Coefficients of y_{02}, y_{12}, and y_{22} from (39)–(41) for computing the Taylor-based approximation $z_{222}(B)$ of order $2m = 34+$ from (36) of the matrix cosine.

q_4	$3.571998478323090 \times 10^{-11}$	d_1	$-2.645687940516643 \times 10^{-3}$
q_3	$-1.857982456862233 \times 10^{-8}$	e_1	$1.049722718717408 \times 10^{1}$
r_2	$3.278753597700932 \times 10^{-5}$	e_0	$8.965376033761624 \times 10^{-4}$
r_1	$-1.148774768780758 \times 10^{-2}$	f_0	$-1.859420533601965 \times 10^{0}$
s_2	$-2.008741312156575 \times 10^{-5}$	g_0	$1.493008139094410 \times 10^{1}$
s_0	$1.737292932136998 \times 10^{1}$	h_2	$1.570135323717639 \times 10^{-4}$
t_2	$6.982819862335600 \times 10^{-5}$	h_1	$-1/6!$
d_2	$-5.259287265295055 \times 10^{-5}$	h_0	$1/4!$

To check if the new evaluation formulae are accurate, we compared the results of computing the matrix cosine with function `cosm` *from [7] with a function using the coefficients from Table 1 in* (39)–(41) *and* (36) *with no scaling for simplicity. Since [7] used a relative backward error analysis, we used the values of* Θ *from [15] (Table 1) corresponding to the backward relative error analysis of the Taylor approximation of the matrix cosine, denoted by* E_b. *Then, if* $||B|| = ||A^2|| \leq \Theta$, *then* $||E_b|| \leq u$ *for the corresponding Taylor approximations. In [15] (Table 1),* Θ *for Taylor approximation of order 16 was 9.97 and* $\Theta_{20} = 10.18$, *showing two decimal digits. Then, for our test with order* $2m = 34+$, *we used a set of 48* 8×8 *matrices from the Matrix Computation Toolbox [18] divided by random numbers to give* $\|B\|$ *between 9 and 10. We compared the forward error* E_f *of both functions*

$$E_f = ||\cos(A) - f(A)||, \tag{43}$$

where function $f(A)$ *was* `cosm` *and the function using* $z_{222}(B)$. *The "exact value" of* $\cos(A)$ *was computed using the method in [19]. The total cost of the new matrix cosine computation function* z_{222} *summing up the number of matrix products over all the test matrices is denoted by* $\text{Cost}_{z_{222}}$. *Taking into account* (4), *the cost for the* `cosm` *Padé function summing up the number of matrix products and inversions over all the test matrices is denoted by* $\text{Cost}_{\text{cosm}}$. *Then, the following cost comparison was obtained for that set of test matrices*

$$100 \times \frac{\text{Cost}_{\text{cosm}} - \text{Cost}_{z_{222}}}{\text{Cost}_{z_{222}}} = 40.78\%,$$

i.e., the cost of z_{222} *is* 40.78% *lower than the cost of* `cosm`. *Moreover, the results were more accurate in* 76.60% *of the matrices. Therefore, the new formulae are efficient and accurate.*

3.2. Evaluation of Matrix Polynomial Approximations of Order 21

In this section, we generalize the results from [5] (Section 3.3) for evaluating polynomial approximations of order $m = 21$ with cost $5M$. Note that for that cost, from [1] (Table 8), the maximum available orders using the Paterson–Stockmeyer method and the evaluation Formulas (16) and (17) are 12 and 16, respectively. Applying a similar procedure to that in Section 3.1 to obtain the coefficients for evaluating a matrix polynomial approximation of order 21, in this case, a system of 22 equations with 22 unknown variables

arises. This system can be reduced to three equations with three unknowns using variable substitution with the MATLAB Symbolic Toolbox, provided that two of the variables are not zero. The following proposition summarizes the results

Proposition 2. *Let $y_{13}(A)$ and $y_{23}(A)$ be*

$$y_{13}(A) = \sum_{i=2}^{12} c_i A^i \tag{44}$$

$$y_{23}(A) = (y_{13}(A) + d_3A^3 + d_2A^2 + d_1A)(y_{13}(A) + e_0y_{03}(A) + e_1A) + f_0y_{13}(A) + g_0y_{03}(A) + h_3A^3 + h_2A^2 + h_1A + h_0I \tag{45}$$

and let $P_{21}(A)$ be a polynomial of degree 21 with coefficients b_i

$$P_{21}(A) = \sum_{i=0}^{21} b_i A^i. \tag{46}$$

Then,

$$y_{23}(A) = \sum_{i=0}^{24} a_i A^i, \tag{47}$$

where coefficients $a_i = a_i(c_{12}, c_{11}, \ldots, c_2, d_3, d_2, d_1, e_1, e_0, f_0, g_0, h_3, h_2, h_1, h_0)$, $i = 0, 1, \ldots, 24$, and the system of equations arising when equating

$$a_i = b_i, \quad i = 0, 1, \ldots, 21, \tag{48}$$

can be reduced to a system of three equations of variables c_{12}, c_{11} and c_{10}, provided

$$c_{12} \neq 0, \quad e_0 \neq 0, \tag{49}$$

and then variables a_i, $i = 22, 23$ and 24 are

$$\begin{aligned} a_{24} &= c_{12}^2, \\ a_{23} &= 2c_{11}c_{12}, \\ a_{22} &= c_{11}^2 + 2c_{10}c_{12}. \end{aligned} \tag{50}$$

Proof of Proposition 2. The proof of Proposition 2 is similar to the proof of Proposition 1. Analogously, if condition (18) is fulfilled with $s = 3$, i.e., $c_{12} \neq 0$, then polynomial $y_{13}(A)$ can be evaluated using (16) and (17) with $s = 3$ and cost $4M$, where y_{03} is given by (21) of [5] (Section 3.3), i.e.,

$$y_{03}(A) = \pm A^3(\sqrt{c_{12}}A^3 + c_{11}/(2\sqrt{c_{12}})A^2 + (4c_{10}c_{12} - c_{11}^2)/(8c_{12}^{3/2})A). \tag{51}$$

If we apply (48), we obtain a similar system to (30). Using variable substitution with the MATLAB Symbolic Toolbox, the MATLAB code `coeffspolm21plus.m` (http://personales.upv.es/jorsasma/Software/coeffspolm21plus.m (accessed on 24 June 2021)) similar to `MATLAB code fragments 4.1 and 4.2` is able to reduce the whole nonlinear system of 22 equations to a nonlinear system of three equations with three variables c_{10}, c_{11}, and c_{12}. The MATLAB code `coeffspolm21plus.m` returns conditions (49) (see the actual code for details.) □

If there is at least one solution for c_{10}, c_{11}, and c_{12} fulfilling condition (49), all of the other coefficients can be obtained using the values of c_{10}, c_{11}, c_{12}. Then, $y_{13}(A)$ from (44) can be evaluated using (16) and (17) giving several possible solutions. Finally, the solutions are rounded to the required precision. Then, according to the stability study from Section 2.2 (see [1], p. 243), the solution giving the least error should be selected.

Similarly to (34) and (35), the degree of $y_{23}(A)$ of (45) is 24, but with the proposed method, we can only set the polynomial approximation coefficients of (46) up to order $m = 21$. The coefficients of a_i of the power A^i, i =22, 23, and 24 are given by (50). The authors of [5] (Section 3.3) give one particular example of this method for calculating a matrix Taylor approximation of the exponential function, where in exact arithmetic

$$y_{23}(A) = T_{21}(A) + a_{22}A^{22} + a_{23}A^{23} + a_{24}A^{24}, \tag{52}$$

where T_{21} is the Taylor approximation of order $m = 21$ of the exponential function and

$$|a_{22} - 1/22!|22! \approx 0.437, \tag{53}$$
$$|a_{23} - 1/23!|23! \approx 0.270, \tag{54}$$
$$|a_{24} - 1/24!|24! \approx 0.130, \tag{55}$$

showing three decimal digits. Again, in exact arithmetic, the approximation $y_{23}(A)$ is more accurate than $T_{21}(A)$. Therefore, the order of that approximation was denoted as $m = 21+$ in [5] (Section 4). The experimental results from [5] showed that this method was more accurate and efficient than the Padé method from [6].

Recently, in [17], an evaluation formula similar to (45) was used to evaluate a Taylor polynomial approximation of the hyperbolic tangent. Similarly to (53), in the case of the hyperbolic tangent, the relative error of the coefficients a_i, $i = 22$, 23, and 24 was also lower than 1—concretely, 0.69, 0.69, and 0.70, respectively (see [17], p. 7). This method was compared to the Paterson–Stockmeyer method being noticeably more efficient without affecting the accuracy (see [17], Section 3 for details).

Proposition 2 allows us to evaluate polynomial approximations of degree 21 not only for the matrix exponential or the hyperbolic tangent but also for other matrix functions. In the following example, we show an application for the evaluation of the Taylor approximation of the matrix logarithm.

Example 2. *In this example, we give real coefficients for computing a Taylor-based approximation of the matrix logarithm of order $m = 21+$ in a stable manner based on the previous results. Evaluating (44) using (16) and (17) with $s = 3$, and using (45), the following formulae can be used to compute the approximation of order $m = 21+$ of the principal logarithm $\log(B)$ for a square matrix $B = I - A$ with no eigenvalues on $\mathbb{R}^-$*

$$\begin{aligned} y_{03}(A) &= A^3(c_1A^3 + c_2A^2 + c_3A), && (56)\\ y_{13}(A) &= (y_{03}(A) + c_4A^3 + c_5A^2 + c_6A)(y_{03}(A) + c_7A^3 + c_8A^2) \\ &\quad + c_9y_{03}(A) + c_{10}A^3 + c_{11}A^2, && (57)\\ y_{23}(A) &= (y_{13}(A) + c_{12}A^3 + c_{13}A^2 + c_{14}A)(y_{13}(A) + c_{15}y_{03}(A) + c_{16}A) \\ &\quad + c_{17}y_{13}(A) + c_{18}y_{03}(A) + c_{19}A^3 + c_{20}A^2 + A, && (58)\end{aligned}$$

where the coefficients are numbered correlatively, and using (1), we take

$$\log(B) = \log(I - A) = -\sum_{i>1} A^i/i \approx -y_{23}(A). \tag{59}$$

The coefficients can be obtained solving first the system of equations arising from (48) with $b_i = 1/i$ for $i = 1, 2, \ldots, 21$, $b_0 = 0$. We used `vpasolve` *(https://es.mathworks.com/help/symbolic/vpasolve.html (accessed on 24 June 2021)) function from the MATLAB Symbolic Computation Toolbox to solve those equations with variable precision arithmetic. We used the* `Random` *option of* `vpasolve`*, which allows to obtain different solutions for the coefficients, running it 100 times. The majority of the solutions were complex, but there were two real stable solutions. Then, we obtained the nested solutions for the coefficients of (16) and (17) with $s = 3$ for computing polynomial (44) with four matrix products (see [1], Section 3), giving also real and complex solutions.*

Again, we selected the real stable solution given in Table 2. This solution avoids complex arithmetic if the matrix A is real. The relative errors of the coefficients of A^{22}, A^{23} and A^{24} of $y_{23}(A)$ with respect to the corresponding Taylor approximation of order 24 of $-\log(I-A)$ function are:

$$a_{22} = 3.205116205918952 \times 10^{-2}, \quad |a_{22} - 1/22|22 \approx 0.295, \tag{60}$$
$$a_{23} = 1.480540983455180 \times 10^{-2}, \quad |a_{23} - 1/23|23 \approx 0.659, \tag{61}$$
$$a_{24} = 3.754613237786792 \times 10^{-3}, \quad |a_{24} - 1/24|24 \approx 0.910, \tag{62}$$

where a_{22}, a_{23}, and a_{24} are rounded to double precision arithmetic. Then, considering exact arithmetic, one gets

$$y_{23}(A) = \sum_{i=1}^{21} A^i/i + a_{22}A^{22} + a_{23}A^{23} + a_{24}A^{24}, \tag{63}$$

which is more accurate than the corresponding Taylor approximation of $\log(B)$ of order $m = 21$. Therefore, similarly to [5] (Section 4), the approximation order of (63) is denoted by $m = 21+$.

Table 2. Coefficients of y_{03}, y_{13}, and y_{23} from (56)–(58) for computing a Taylor-based approximation of function $\log(B) = \log(I-A)$ of order $m = 21+$.

c_1	$2.475376717210241 \times 10^{-1}$	c_{11}	$-1.035631527011582 \times 10^{-1}$
c_2	$2.440262449961976 \times 10^{-1}$	c_{12}	$-3.416046999733390 \times 10^{-1}$
c_3	$1.674278428631194 \times 10^{-1}$	c_{13}	$4.544910328432021 \times 10^{-2}$
c_4	$-9.742340743664729 \times 10^{-2}$	c_{14}	$2.741820014945195 \times 10^{-1}$
c_5	$-4.744919764579607 \times 10^{-2}$	c_{15}	$-1.601466804001392 \times 10^{0}$
c_6	$5.071515307996127 \times 10^{-1}$	c_{16}	$1.681067607322385 \times 10^{-1}$
c_7	$2.025389951302878 \times 10^{-1}$	c_{17}	$7.526271076306975 \times 10^{-1}$
c_8	$-4.809463272682823 \times 10^{-2}$	c_{18}	$4.282509402345739 \times 10^{-2}$
c_9	$6.574533191427105 \times 10^{-1}$	c_{19}	$1.462562712251202 \times 10^{-1}$
c_{10}	$3.236650728737168 \times 10^{-1}$	c_{20}	$5.318525879522635 \times 10^{-1}$

The θ values such that the relative backward errors for the Padé approximations are lower than u are shown in [11] (Table 2.1). The corresponding θ value for the Taylor approximation of $\log(I-A)$ of order $m = 21+$, denoted by θ_{21+}, can be computed similarly (see [11] for details), giving $\theta_{21+} = 0.211084493690929$, where the value is rounded to IEEE double precision arithmetic.

We compared the results of using (56)–(58) with the coefficient values from Table 2, with the results given by function `logm_iss_full` *from [20]. For that comparison, we used a matrix test set of 43 8×8 matrices of the Matrix Computation Toolbox [18]. We reduced their norms so that they are random with a uniform distribution in $[0.2, \theta_{21+}]$ in order to compare the Padé approximations of* `logm_iss_full` *with the Taylor-based evaluation Formulas (56)–(58) using no inverse scaling in none of the approximations (see [11]).*

The "exact" matrix logarithm was computed using the method from [19]. The error of the implementation using Formula (58) was lower than `logm_iss_full` *in 100% of the matrices with a 19.61% lower relative cost in flops. Therefore, evaluation Formulas (56)–(58) are efficient and accurate for a future Taylor-based implementation for computing the matrix logarithm.*

3.3. Evaluation of Matrix Polynomials of Degree $m = 6s$

The following proposition generalizes the particular cases of the evaluation of the matrix exponential Taylor approximation with degrees $m = 24$ and 30 from [5] (Section 3.4) for evaluating general matrix polynomials of degree $m = 6s$, $s = 2, 3, \ldots$

Proposition 3. *Let $y_{0s}(A)$, $y_{1s}(A)$, and $y_{2s}(A)$ be the polynomials*

$$y_{0s}(A) = A^s \sum_{i=1}^{s} e_{s+i} A^i, \tag{64}$$

$$y_{1s}(A) = \sum_{i=1}^{4s} c_i A^i, \tag{65}$$

$$y_{2s}(A) = y_{1s}(A)\left(y_{0s}(A) + \sum_{i=1}^{s} e_i A^i\right) + \sum_{i=0}^{s} f_i A^i, \tag{66}$$

and let $P_m(A)$ be the polynomial

$$P_m(A) = \sum_{i=0}^{6s} b_i A^i. \tag{67}$$

Then,

$$y_{2s}(A) = \sum_{i=0}^{6s} a_i A^i, \tag{68}$$

where coefficients $a_i = a_i(c_i, e_j, f_k)$, $i = 1, 2, \ldots, 4s$, $j = 1, 2, \ldots, 2s$, $k = 0, 1, \ldots, s$, and if

$$b_{6s} \neq 0, \tag{69}$$

then, if we equate $y_{2s}(A) = P_m(A)$, i.e.,

$$a_i = b_i, \;\; i = 0, 1, \ldots, 6s, \tag{70}$$

then the following relationships between the coefficients of the polynomials $y_{0s}(A)$, $y_{1s}(A)$, $y_{2s}(A)$, and $P_m(A)$ are fulfilled:

a.

$$\begin{aligned} c_{4s-k} &= c_{4s-k}(b_{6s}, b_{6s-1}, \ldots, b_{6s-k}), \text{ for } k = 0, 1, \ldots, s-1, \\ e_{2s-k} &= e_{2s-k}(b_{6s}, b_{6s-1}, \ldots, b_{6s-k}), \text{ for } k = 0, 1, \ldots, s-1. \end{aligned} \tag{71}$$

b.

$$c_{3s-k} = c_{3s-k}(b_{6s}, b_{6s-1}, \ldots, b_{5s-k}, e_s, \ldots, e_{s-k}), k = 0, \ldots, s-1. \tag{72}$$

c.

$$c_{2s-k} = c_{2s-k}(b_{6s}, \ldots, b_{4s-k}, e_s, \ldots, e_1), k = 0, \ldots, s-1. \tag{73}$$

d.

$$c_{s-k} = c_{s-k}(b_{6s}, \ldots, b_{3s-k}, e_s, \ldots, e_1), k = 0, \ldots, s-1. \tag{74}$$

Proof of Proposition 3. Polynomial $y_{1s}(A)$ from (65) can be computed using the general method from [1] (Section 3), reproduced here as (16) and (17), provided condition (18) is fulfilled, i.e., $c_{4s} \neq 0$.

a. In the following, we show that (71) holds. Taking (16) and (17) into account, one gets

$$y_{1s}(A) = y_{0s}^2(A) + q(A), \tag{75}$$

where $q(x)$ is a polynomial of degree lower than $3s + 1$, and equating the terms of degree $4s$ in (75), we obtain $e_{2s} = \pm\sqrt{c_{4s}}$. On the other hand, equating the terms of degree $6s$ in (66), taking condition (69) into account, we obtain

$$\begin{aligned} c_{4s} e_{2s} &= b_{6s}, \\ c_{4s}(\pm\sqrt{c_{4s}}) &= b_{6s}, \end{aligned}$$

$$c_{4s} = \sqrt[3]{b_{6s}^2} \neq 0, \tag{76}$$

$$e_{2s} = \pm\sqrt[3]{b_{6s}} \neq 0. \tag{77}$$

Since condition (69) is fulfilled, then by (76), one gets that condition (18) is also fulfilled. Then, polynomial $y_{1s}(A)$ from (65) can be effectively computed using (16) and (17), and by (76) and (77), one gets that c_{4s} and e_{4s} depend on b_{6s}, i.e.,

$$c_{4s} = c_{4s}(b_{6s}),\ e_{2s} = e_{2s}f(b_{6s}). \tag{78}$$

Equating the terms of degree $4s-1$ in (75), we obtain

$$c_{4s-1} = 2e_{2s}e_{2s-1}.$$

Therefore,

$$e_{2s-1} = \frac{c_{4s-1}}{2e_{2s}}. \tag{79}$$

Equating the terms of degree $4s-2$ in (75), we obtain

$$c_{4s-2} = e_{2s}e_{2s-2} + e_{2s-1}^2 + e_{2s-2}e_{2s},$$

then

$$e_{2s-2} = \frac{c_{4s-2} - e_{2s-1}^2}{2e_{2s}}. \tag{80}$$

Equating the terms of degree $4s-3$ in (75), we obtain

$$c_{4s-3} = e_{2s}e_{2s-3} + e_{2s-1}e_{2s-2} + e_{2s-2}e_{2s-1} + e_{2s-3}e_{2s},$$

then

$$e_{2s-3} = \frac{c_{4s-3} - (e_{2s-1}e_{2s-2} + e_{2s-2}e_{2s-1})}{2e_{2s}}.$$

Equating the terms of degree $4s-4$ in (75), we obtain

$$c_{4s-4} = e_{2s}e_{2s-4} + e_{2s-1}e_{2s-3} + e_{2s-2}e_{2s-2} + e_{2s-3}e_{2s-1} + e_{2s-4}e_{2s},$$

then

$$e_{2s-4} = \frac{c_{4s-4} - \sum_{i=1}^{3} e_{2s-i}e_{2s+i-4}}{2e_{2s}}.$$

Proceeding in an analogous way with e_{2s-k} for $k = 5, 6, \ldots, s-1$, we obtain

$$e_{2s-k} = \frac{c_{4s-k} - \sum_{i=1}^{k-1} e_{2s-i}e_{2s+i-k}}{2e_{2s}},\ k = 1, 2, \ldots, s-1. \tag{81}$$

On the other hand, equating the terms of degree $6s-1$ in (66), and taking (79) into account, we obtain

$$c_{4s}e_{2s-1} + c_{4s-1}e_{2s} = c_{4s-1}\left(\frac{c_{4s}}{2e_{2s}} + e_{2s}\right) = b_{6s-1},$$

Since

$$\frac{c_{4s}}{2e_{2s}} + e_{2s} = \frac{c_{4s} + 2e_{2s}^2}{2e_{2s}} = \frac{\sqrt[3]{b_{6s}^2} + 2\sqrt[3]{b_{6s}^2}}{2\sqrt[3]{b_{6s}}} = \frac{3\sqrt[3]{b_{6s}}}{2} \neq 0, \tag{82}$$

then

$$c_{4s-1} = \frac{2b_{6s-1}}{3\sqrt[3]{b_{6s}}}. \tag{83}$$

Taking into account (77), (79) and (83), we obtain that c_{4s-1} and e_{2s-1} depend on b_{6s} and b_{6s-1}, i.e.,

$$c_{4s-1} = c_{4s-1}(b_{6s}, b_{6s-1}),\ e_{2s-1} = e_{2s-1}(b_{6s}, b_{6s-1}). \tag{84}$$

Equating the terms of degree $6s-2$ in (66) and taking (82) into account, we obtain

$$c_{4s}e_{2s-2} + c_{4s-1}e_{2s-1} + c_{4s-2}e_{2s} = b_{6s-2},$$

and taking into account (80) and (82), it follows that

$$c_{4s-2} = \frac{b_{6s-2} - c_{4s-1}e_{2s-1} + \frac{e_{2s-1}^2}{2e_{2s}}}{\frac{3\sqrt[3]{b_{6s}}}{2}}. \tag{85}$$

On the other hand, from (77), (84) and (85), one gets that c_{4s-2} and e_{2s-2} can be computed explicitly depending on b_{6s}, b_{6s-1}, and b_{6s-2}, i.e.,

$$c_{4s-2} = c_{4s-2}(b_{6s}, b_{6s-1}, b_{6s-2}), e_{2s-2} = e_{2s-2}(b_{6s}, b_{6s-1}, b_{6s-2}). \tag{86}$$

Proceeding similarly when equating the terms of degrees $6s-3$, $6s-4 \ldots, 5s+1$ in (66), one gets (71).

b. In the following, we show that (72) holds. Equating the terms of degree $5s$ in (66) and taking condition $e_{2s} \neq 0$ from (77) into account, we obtain

$$\begin{aligned} &c_{4s}e_s + c_{4s-1}e_{s+1} + \ldots + c_{3s+1}e_{2s-1} + c_{3s}e_{2s} = b_{5s}, \\ &c_{3s} = \frac{b_{5s} - (c_{4s}e_s + c_{4s-1}e_{s+1} + \ldots + c_{3s+1}e_{2s-1})}{e_{2s}}. \end{aligned}$$

Hence, taking (71) into account, it follows that

$$c_{3s} = c_{3s}(b_{6s}, b_{6s-1}, \ldots, b_{5s+1}, b_{5s}, e_s). \tag{87}$$

Equating the terms $5s-1$ in (66) and taking condition $e_{2s} \neq 0$ from (77) into account, we obtain

$$\begin{aligned} &c_{4s}e_{s-1} + c_{4s-1}e_s + \ldots + c_{3s}e_{2s-1} + c_{3s-1}e_{2s} = b_{5s-1}, \\ &c_{3s-1} = \frac{b_{5s-1} - (c_{4s}e_{s-1} + c_{4s-1}e_s + \ldots + c_{3s}e_{2s-1})}{e_{2s}}. \end{aligned}$$

Hence, using (87), one gets

$$c_{3s-1} = c_{3s-1}(b_{6s}, b_{6s-1}, \ldots, b_{5s}, b_{5s-1}, e_s, e_{s-1}) \tag{88}$$

Proceeding similarly, equating the terms of degrees $5s-2$, $5s-3 \ldots$, $4s+1$ in (66), one gets (72).

c. In the following, we show that (73) holds. Equating the terms of degree $4s$ in (66) and taking condition $e_{2s} \neq 0$ from (77) into account, it follows that

$$\begin{aligned} &c_{4s-1}e_1 + c_{4s-2}e_2 + \ldots + c_{2s+1}e_{2s-1} + c_{2s}e_{2s} = b_{4s}, \\ &c_{2s} = \frac{b_{4s} - (c_{4s-1}e_1 + c_{4s-2}e_2 + \ldots + c_{2s+1}e_{2s-1})}{e_{2s}} \end{aligned}$$

Taking (71) and (72) into account, we obtain

$$c_{2s} = c_{2s}(b_{6s}, \ldots, b_{4s}, e_s, \ldots, e_1).$$

Equating the terms of degree $4s-1$ in (66) and condition $e_{2s} \neq 0$, one gets

$$c_{4s-2}e_1 + c_{4s-3}e_2 + \ldots + c_{2s}e_{2s-1} + c_{2s-1}e_{2s} = b_{4s-1},$$
$$c_{2s-1} = \frac{b_{4s-1} - (c_{4s-2}e_1 + c_{4s-3}e_2 + \ldots + c_{2s}e_{2s-1})}{e_{2s}}.$$

Taking (71) and (72) into account, we obtain

$$c_{2s-1} = c_{2s-1}(b_{6s}, \ldots, b_{4s-1}, e_s, \ldots, e_1).$$

Proceeding similarly, equating the terms of degrees $4s-2$, $4s-3, \ldots, 3s+1$ in (66) and taking (71), (72), and condition $e_{2s} \neq 0$ into account, one gets (73).

d. In the following, we show that (74) holds. Equating the terms of degree $3s$ in (66) and takingcondition $e_{2s} \neq 0$ into account, it follows that

$$c_{3s-1}e_1 + c_{3s-2}e_2 + \ldots + c_{s+1}e_{2s-1} + c_s e_{2s} = b_{3s},$$
$$c_s = \frac{b_{3s} - (c_{3s-1}e_1 + c_{3s-2}e_2 + \ldots + c_{s+1}e_{2s-1})}{e_{2s}}.$$

Hence, from (71)–(73), we obtain

$$c_s = c_s(b_{6s}, \ldots, b_{3s}, e_s, \ldots, e_1).$$

Equating the terms of degree $3s-1$ in (66) and condition $e_{2s} \neq 0$, we obtain

$$c_{3s-2}e_1 + c_{3s-3}e_2 + \ldots + c_s e_{2s-1} + c_{s-1}e_{2s} = b_{3s-1},$$
$$c_{s-1} = \frac{b_{3s-1} - (c_{3s-2}e_1 + c_{3s-3}e_2 + \ldots + c_s e_{2s-1})}{e_{2s}}.$$

Hence, from (71)–(73) we obtain

$$c_{s-1} = c_{s-1}(b_{6s}, \ldots, b_{3s-1}, e_s, \ldots, e_1).$$

Proceeding similarly, equating the terms of degrees $3s-2, 3s-3, \ldots, 2s+1$, in (66), one gets (74).

□

Corollary 1. *If condition* (69) *holds, then the system of* $6s+1$ *equations with* $7s+1$ *variables arising from* (70) *can be reduced using variable substitution to a system of* s *equations with* s *variables, and if there exist at least one solution for that system, then all the coefficients from* (64)–(66) *can be calculated using the solution of the system.*

Proof of Corollary 1. If we equate the terms of degree $2s, 2s-1, \ldots, s+1$ in (66), we obtain the following system of equations:

$$\begin{aligned}
&c_{2s-1}e_1 + c_{2s-2}e_2 + \ldots + c_2 e_{2s-2} + c_1 e_{2s-1} = b_{2s}\\
&c_{2s-2}e_1 + c_{2s-3}e_2 + \ldots + c_2 e_{2s-3} + c_1 e_{2s-2} = b_{2s-1}\\
&\ldots\\
&c_s e_1 + c_{s-1}e_2 + \ldots + c_2 e_{s-1} + c_1 e_s = b_{s+1}.
\end{aligned} \tag{89}$$

Taking (71), (73), and (74) into account, it follows that system (89) can be written as a system of s equations with a set of s unknown variables $\{e_1, e_2 \ldots, e_s\}$, where, in general, (89) is nonlinear system since the c_k coefficients depend on the e_k coefficients.

Equating the terms of degrees $s, s-1, \ldots, 0$, in (66), one gets

$$\begin{aligned} f_s &= b_s - c_{s-1}e_1 - c_{s-2}e_2 - \ldots - c_1 e_{s-1}, \\ f_{s-1} &= b_{s-1} - c_{s-2}e_1 - c_{s-3}e_2 - \ldots - c_1 e_{s-2}, \\ &\vdots \\ f_2 &= b_2 - c_1 e_1, \\ f_1 &= b_1, \\ f_0 &= b_0. \end{aligned} \tag{90}$$

Using (71) from Proposition 3, one gets that the values c_{4s-k} and e_{2s-k}, for $k = 0, \ldots, s-1$, can be calculated explicitly depending on the polynomial coefficients b_i for $i = 6s, 6s-2, \ldots, 5s+1$.

If there exist at least one solution of system (89), then the values c_{3s-k}, c_{2s-k} and c_{s-k} can be calculated for $k = 0, \ldots, s-1$ (see (72)–(74)), and coefficients f_{s-k} can be calculated for $k = 0, \ldots, s$, using (90), allowing one to obtain all the coefficients from (64)–(66). □

Using [1] (Table 6), in Table 3, we present the maximum available order for a cost $C(M)$ in the following cases:

- The Paterson–Stockmeyer evaluation formula.
- z_{kps} from (19) with $k = 1$, denoting the combination of (17) with the Paterson–Stockmeyer formula proposed in [1] (Section 3.1).
- z_{kps} from (19) with $k = 2$, denoting the combination of (66) with the Paterson–Stockmeyer formula, whenever a solution for the coefficients of z_{2ps} exist.

Table 3. Maximum available approximation order for a cost C using the Paterson–Stockmeyer method, order denoted by d_{PS}, maximum order using z_{1ps} from (19) combining (16) and (17) with the Paterson–Stockmeyer, denoted by $d_{z_{1s}}$, and maximum order using z_{2ps} from (19) combining (66) with the Paterson–Stockmeyer method, denoted by $d_{z_{2s}}$, whenever a solution for the coefficients of z_{2ps} exist. Parameters s and p for $z_{2ps}(x)$ such that s is minimum to obtain the required order giving a system (89) of s equations with minimum size.

C(M)	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
d_{PS}	6	9	12	16	20	25	30	36	42	49	56
$d_{z_{1s}}$	8	12	16	20	25	30	36	42	49	56	64
$d_{z_{2s}}$	-	12	18	24	30	36	42	49	56	64	72
$s_{z_{2s}}$	-	2	3	4	5	6	6	7	7	8	8
$p_{z_{2s}}$	-	0	0	0	0	0	6	7	14	16	24

Table 3 also shows the values of p and s for $z_{2ps}(A)$ such that s is minimum to obtain the required order, giving the minimum size of the system (89) to solve, i.e., s equations with s unknown variables. Note that it makes no sense to use (66) for $s = 1$ and cost $C = 3M$ since the order obtained is $m = 6s = 6$ and for that cost the Paterson–Stockmeyer method obtains the same order. Table 3 shows that evaluation formula z_{2ps} obtains a greater order than z_{1ps} for $d_{z_{1s}} > 12$. Concretely, for $s_{z_{2s}} \geq 5$, where the available order with z_{2s} is $d_{z_{2s}} = 30, 36, 42, \ldots$, z_{2ps} allows increments 10, 11, 12 ... of the available order with respect to using the Paterson–Stockmeyer method, and increments of $s_{z2s} = 5, 6, 6 \ldots$ with respect to using z_{1ps}.

In [5], real stable solutions were found for the coefficients of (64)–(66) for the exponential Taylor approximation with degrees $6s$ with $s = 4$ and 5, i.e., 24, and 30. The following example deals with the matrix logarithm Taylor approximation.

Example 3. *In this example, we provide real coefficients for calculating the Taylor approximation of the principal matrix logarithm* $\log(B)$ *of order* $m = 6s = 30$, $s = 5$, *in a stable manner based on the results of Proposition 3 and Corollary 1 with the following expressions*

$$y_{05}(A) = A^5(c_1A^5 + c_2A^4 + c_3A^3 + c_4A^2 + c_5A), \tag{91}$$

$$\begin{aligned} y_{15}(A) = {} & (y_{05}(A) + c_6A^5 + c_7A^4 + c_8A^3 + c_9A^2 + c_{10}A) \\ & \times (y_{05}(A) + c_{11}A^5 + c_{12}A^4 + c_{13}A^3 + c_{14}A^2) \\ & + c_{15}y_{05}(A) + c_{16}A^5 + c_{17}A^4 + c_{18}A^3 + c_{19}A^2 + c_{20}A, \end{aligned} \tag{92}$$

$$\begin{aligned} y_{25}(A) = {} & y_{15}(A)(y_{05}(A) + c_{21}A^5 + c_{22}A^4 + c_{23}A^3 + c_{24}A^2 + c_{25}A) \\ & + c_{26}A^5 + c_{27}A^4 + c_{28}A^3 + c_{29}A^2 + c_{30}A, \end{aligned} \tag{93}$$

where the coefficients were numbered correlatively. The coefficients c_i $i = 1, 2, \ldots, 30$, *can be obtained following the procedure from Section 3.3, reducing the whole system of* 30 *equations with* 30 *unknown variables to the system* (89) *of* $s = 5$ *variables with* s *unknowns* e_i, $i = 1, 2 \ldots, 5$, *corresponding in* (93) *to* $e_1 = c_{25}$, $e_2 = c_{24}, \ldots, e_5 = c_{21}$. *Once this was done, we checked that* e_1 *and* e_2 *could be easily solved as functions of* e_3, e_4 *and* e_5, *reducing the system to a system of three equations with three unknown variables. To obtain a real solution of the three coefficients, we used the MATLAB Symbolic Math Toolbox function* `vpasolve` *giving a range* $[-10, 10]$ *for the solutions of the three variables and using 32 decimal digits. The results of the coefficients from* (91)–(93) *rounded to IEEE double precision arithmetic are given in Table 4.*

Note that using the evaluation Formulas (91)–(93), *the Taylor approximation* $y_{25}(A)$ *of order* $m = 30$ *can be computed with a cost of* $7M$. *For the same order the cost of the Paterson–Stockmeyer method is* $9M$, *and using* z_{1ps} *from* (19) *the cost is* $8M$ *(see Table 3). Similarly to [11], we computed the value such that the relative backward error is lower than* u *for the Taylor approximation of* $\log(I - A)$ *of order* $m = 30$ *giving* $\theta_{30} = 0.329365534847136$.

Similarly to Example 2, to check if $y_{25}(A)$ *is competitive, we prepared a new matrix test set with 50* 8×8 *matrices of the Matrix Computation Toolbox [18] reducing their norms so that they are random with a uniform distribution in* $[0.3, \theta_{30}]$, *and the inverse scaling algorithm is not used in either the Padé and Taylor algorithms. Then, we compared the results of using* (91)–(93) *with the results given by function* `logm_iss_full` *from [20] for the previous matrix set, computing the "exact" values of the matrix logarithm in the same way. The error of using the evaluation Formulas* (91)–(93) *was lower than* `logm_iss_full` *in* 97.62% *of the matrices with a* 42.40% *lower relative cost in flops, being competitive in efficiency and accuracy for future implementations for computing the matrix logarithm.*

Table 4. Coefficients of y_{05}, y_{15}, y_{25} from (91)–(93) for computing the Taylor approximation of $\log(B) = \log(I - A) = -y_{25}(A)$ of order $m = 30$.

c_1	$3.218297948685432 \times 10^{-1}$	c_{16}	$2.231079274704953 \times 10^{-1}$
c_2	$1.109757913339804 \times 10^{-1}$	c_{17}	$3.891001336083639 \times 10^{-1}$
c_3	$7.667169819995447 \times 10^{-2}$	c_{18}	$6.539646241763075 \times 10^{-1}$
c_4	$6.192062222365700 \times 10^{-2}$	c_{19}	$8.543283349051067 \times 10^{-1}$
c_5	$5.369406358130299 \times 10^{-2}$	c_{20}	$-1.642222074981266 \times 10^{-2}$
c_6	$2.156719633283115 \times 10^{-1}$	c_{21}	$6.179507508449100 \times 10^{-2}$
c_7	$-2.827270631646985 \times 10^{-2}$	c_{22}	$3.176715034213954 \times 10^{-2}$
c_8	$-1.299375958233227 \times 10^{-1}$	c_{23}	$8.655952402393143 \times 10^{-2}$
c_9	$-3.345609833413695 \times 10^{-1}$	c_{24}	$3.035900161106295 \times 10^{-1}$
c_{10}	$-8.193390302418316 \times 10^{-1}$	c_{25}	$9.404049154527467 \times 10^{-1}$
c_{11}	$-1.318571680058333 \times 10^{-1}$	c_{26}	$-2.182842624594848 \times 10^{-1}$
c_{12}	$1.318536866523954 \times 10^{-1}$	c_{27}	$-5.036471128390267 \times 10^{-1}$
c_{13}	$1.718006767617093 \times 10^{-1}$	c_{28}	$-4.650956099599815 \times 10^{-1}$
c_{14}	$1.548174815648151 \times 10^{-1}$	c_{29}	$5.154435371157740 \times 10^{-1}$
c_{15}	$2.139947460365092 \times 10^{-1}$	c_{30}	1

Note that using the evaluation formulae from Sections 1 and 2 with cost $4M$ and $5M$, one can get an order of approximation 15+ and 21+, respectively, whereas using z_{2ps} from (19) combining (66) with the Paterson–Stockmeyer method, the orders that can be obtained are lower, i.e., 12 and 18, respectively (see Table 3). Note that for the approximation 15+ where $s = 2$ (see Section 3.1), one gets order $15+ = (6s+3)+$ and the total degree of the polynomial obtained is $8s = 16$. For the approximation 21+ where $s = 3$, one gets order $21+ = (6s+3)+$ and the total degree of the polynomial degree is $8s = 24$. The next step in our research is to extend the evaluation formulae from Propositions 1 and 2 to evaluate polynomial approximations of order $(6s+3)+$ of the type

$$y_{0s}(A) = A_s \sum_{i=1}^{s} c_i A_i, \tag{94}$$

$$y_{1s}(A) = \sum_{i=s+1}^{4s} a_i A^i = \left(y_{0s}(A) + \sum_{i=1}^{s} d_i A_i \right) \left(y_{0s}(A) + \sum_{i=2}^{s} e_i A_i \right) + f_0 y_{0s}(A) + \sum_{i=3}^{s} f_i A_i, \tag{95}$$

$$y_{2s}(A) = \left(y_{1s}(A) + \sum_{i=1}^{s} g_i A_i \right) \left(y_{1s}(A) + h_0 y_{0s}(A) + \sum_{i=1}^{s} h_i A_i \right) + j_0 y_{1s}(A) + k_0 y_{0s}(A) + \sum_{i=0}^{s} l_i A_i. \tag{96}$$

Those formulae correspond to a particular case of Formulas (62)–(65) of [1] (Prop. 2) where $k = 2$. It is easy to show that the degree of $y_{2s}(A)$ is $8s$ and the total number of coefficients of y_{2s} is $6s + 4$, i.e., $3s$ coefficients a_i, s coefficients g_i, s coefficients h_i, $s + 1$ coefficients l_i, and coefficients f_0, j_0 and k_0. Using `vpasolve` in a similar way as in Example 2, we could find solutions for the coefficients of (94)–(96) and (19) so that $y_{2s}(A)$ and z_{2ps} allows to evaluate matrix logarithm Taylor-based approximations of orders from 15+ up to 75+. Similarly, we could also find the coefficients for Formulas (94)–(96) to evaluate matrix hyperbolic tangent Taylor approximations of orders higher than 21. Then, our next research step is to show that evaluation Formulas (94)–(96) and its combination with the Paterson–Stockmeyer method from (19) can be used for the general polynomial approximations of matrix functions.

4. Conclusions

In this paper, we extend the family of methods for evaluating matrix polynomials from [1], obtaining general solutions for new cases of the general matrix polynomial evaluation Formulas (62)–(65) from Proposition 2 from [1] (Section 5). These cases allow to compute matrix polynomial approximations of orders 15 and 21 with a cost of $4M$ and $5M$, respectively, whenever a stable solution for the coefficients exist. Moreover, a general method for computing matrix polynomials of order $m = 6s$, for $s = 3, 4...$ more efficiently than the methods provided in [1] was provided. Combining this method with the Paterson–Stockmeyer method, polynomials or degree greater than 30 can be evaluated with two matrix products less than using Paterson–Stockmeyer method as shown in Table 3.

Examples for evaluating Taylor approximations of the matrix cosine and the matrix logarithm were given. The accuracy and efficiency results of the proposed evaluation formulae were compared to state-of-the-art Padé algorithms, being competitive for future implementations for computing both functions.

Future work will deal with the generalization of more efficient evaluation formulae based on the evaluation Formulas (62)–(65) from Proposition 2 from [1] (Section 5), its combinations with Paterson–Stockmeyer method (19), and in general, evaluation formulae based on products of matrix polynomials.

Author Contributions: Conceptualization, J.S.; methodology, J.S. and J.I.; software, J.S.; validation, J.S. and J.I.; formal analysis, J.S. and J.I.; investigation, J.S. and J.I.; resources, J.S. and J.I.; writing—original J.S. and J.I.; writing—review and editing, J.S. and J.I. Both authors have read and agreed to the published version of the manuscript.

Funding: This research was partially funded by the European Regional Development Fund (ERDF) and the Spanish Ministerio de Economía y Competitividad grant TIN2017-89314-P, and by the Programa de Apoyo a la Investigación y Desarrollo 2018 of the Universitat Politècnica de València grant PAID-06-18-SP20180016.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MDPI	Multidisciplinary Digital Publishing Institute
DOAJ	Directory of open access journals
TLA	Three letter acronym
LD	Linear dichroism

References

1. Sastre, J. Efficient evaluation of matrix polynomials. *Linear Algebra Appl.* **2018**, *539*, 229–250. [CrossRef]
2. Paterson, M.S.; Stockmeyer, L.J. On the number of nonscalar multiplications necessary to evaluate polynomials. *SIAM J. Comput.* **1973**, *2*, 60–66. [CrossRef]
3. Higham, N.J. *Functions of Matrices: Theory and Computation*; Society for Industrial and Applied Mathematics: Philadelphia, PA, USA, 2008.
4. Sastre, J.; Ibáñez, J.; Alonso, P.; Peinado, J.; Defez, E. Fast taylor polynomial evaluation for the computation of the matrix cosine. *J. Comput. Appl. Math.* **2019**, *354*, 641–650. [CrossRef]
5. Sastre, J.; Ibáñez, J.E. Defez, Boosting the computation of the matrix exponential. *Appl. Math. Comput.* **2019**, *340*, 206–220.
6. Al-Mohy, A.H.; Higham, N.J. A new scaling and squaring algorithm for the matrix exponential. *SIAM J. Matrix Anal. Appl.* **2009**, *31*, 970–989. [CrossRef]
7. Al-Mohy, A.H.; Higham, N.J.; Relton, S. New Algorithms for Computing the Matrix Sine and Cosine Separately or Simultaneously. *SIAM J. Sci. Comput.* **2015**, *37*, A456–A487. [CrossRef]
8. Bader, P.; Blanes, S.; Casas, F. Computing the Matrix Exponential with an Optimized Taylor Polynomial Approximation. *Mathematics* **2019**, *7*, 1174. [CrossRef]
9. Bader, P.; Blanes, S.; Casas, F. An improved algorithm to compute the exponential of a matrix. *arXiv* **2017**, arXiv:1710.10989.
10. Sastre, J. On the Polynomial Approximation of Matrix Functions. Available online: http://personales.upv.es/~jorsasma/AMC-S-16-00951.pdf (accessed on 20 April 2020).
11. Al-Mohy, A.H.; Higham, N.J. Improved inverse scaling and squaring algorithms for the matrix logarithm. *SIAM J. Sci. Comput.* **2012**, *34*, C153–C169. [CrossRef]
12. Moler, C.B.; Loan, C.V. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Rev.* **2003**, *45*, 3–49. [CrossRef]
13. Blackford, S.; Dongarra, J. Installation Guide for LAPACK, LAPACK Working Note 41. Available online: http://www.netlib.org/lapack/lawnspdf/lawn41.pdf (accessed on 20 April 2020).
14. Sastre, J. Efficient mixed rational and polynomial approximation of matrix functions. *Appl. Math. Comput.* **2012**, *218*, 11938–11946. [CrossRef]
15. Sastre, J.; Ibáñez, J.; Alonso, P.; Peinado, J.; Defez, E. Two algorithms for computing the matrix cosine function. *Appl. Math. Comput.* **2017**, *312*, 66–77. [CrossRef]
16. Fasi, M. Optimality of the Paterson–Stockmeyer method for evaluating matrix polynomials and rational matrix functions. *Linear Algebra Appl.* **2019**, *574*, 182–200. [CrossRef]
17. Ibáñez, J.; Alonso, J.M.; Sastre, J.; Defez, E.; Alonso-Jordá, P. Advances in the Approximation of the Matrix Hyperbolic Tangent. *Mathematics* **2021**, *9*, 1219. [CrossRef]
18. Higham, N.J. The Matrix Computation Toolbox. Available online: http://www.ma.man.ac.uk/~higham/mctoolbox (accessed on 18 April 2020).

19. Davies, E.B. Approximate diagonalization. *SIAM J. Matrix Anal. Appl.* **2007**, *29*, 1051–1064. [CrossRef]
20. Higham, N. Matrix Logarithm. 2020. Available online: https://www.mathworks.com/matlabcentral/fileexchange/33393-matrix-logarithm (accessed on 18 April 2020).