arXiv:2603.23143v1 [math.NA] 24 Mar 2026

稳定且高效的矩阵多项式求值系统框架

J.M. Alonso[a]，J. Sastre[b]，J. Ibáñez[c]，E. Defez[c]

[a]分子影像仪器大学研究所，瓦伦西亚理工大学，维拉路 *s/n*，*46022*-瓦伦西亚（西班牙）[b]电信与多媒体应用研究所，瓦伦西亚理工大学，维拉路 *s/n*，*46022*-瓦伦西亚（西班牙）[c]多学科数学大学研究所，瓦伦西亚理工大学，维拉路 *s/n*，*46022*-瓦伦西亚（西班牙）

摘要

近期已有一种矩阵多项式求值方法，其所需的矩阵乘积次数比 Paterson–Stockmeyer（PS）方法少一次（$1M$）。对于大规模矩阵，计算成本渐近上由矩阵乘积的数量决定，因此这一减少会直接影响总执行时间。然而，这些优化公式中的系数是非线性多项式方程组的解，因此可能存在多个潜在解集。若系数选择不当，浮点运算中可能会出现数值不稳定性。

本文提出了一个系统化框架及其 MATLAB 实现 MatrixPolEval1，用于求得并验证次数为 $m \in \{8, 10, 12\}$ 及以上的矩阵多项式的稳定系数组。该框架引入了结构变体，即使原始配置无法得出稳健解，也能维持稳定性。所提供的工具使用可变精度算术（VPA）识别稳定系数组，并提供预期精度的可靠性指标。针对应用中出现的多项式（包括矩阵指数和几何级数）开展的数值实验表明，该框架在保持与 PS 方法相当的数值精度的同时，实现了 $1M$ 节省。

电子邮箱地址：jmalonso@dsic.upv.es (J.M. Alonso), jsastrem@upv.es (J. Sastre), jjibanez@dsic.upv.es (J. Ibáñez), edefez@imm.upv.es (E. Defez)

关键词：矩阵多项式求值，Paterson–Stockmeyer 方法，数值稳定性，舍入误差分析，计算效率，MATLAB 框架*2020*年 *MSC*：15A16, 65F30, 65G50,65Y15

1. 引言

矩阵多项式的求值

$$P_m(A) = \sum_{i=0}^{m} b_i A^i \tag{1}$$

采用 [12], 提出的方案，可以利用矩阵多项式的乘积和线性组合，以较低的计算成本完成求值。具体而言， [12, 命题 1] 表明，对于首项系数非零且次数满足特定条件的一般矩阵多项式，与 Paterson–Stockmeyer（PS）方法相比，计算成本可减少一次矩阵乘法（$1M$）。在此情形下，矩阵乘法的次数被用作主要的成本度量，因为对于大型矩阵而言，其他运算的渐近影响可以忽略不计。近期， [6]进一步刻画了这类求值的理论下界。此外， [6] 中源自猜想 5.1 的 20 次多项式这一具体情形也已在 [16]中得到处理。

然而， [12] 求值公式中的系数由非线性多项式方程的根推导而来，因此可能存在多个解集。如 [12, 第 3.1], 节所示，这些解集的数值稳定性各不相同，因此必须识别出稳定解，才能保证精度。虽然针对某些具体情形已找到了稳定解，例如矩阵指数 [13]和矩阵余弦 [14], 的泰勒近似，但针对任意多项式的一般方法仍有待建立。针对矩阵指数、正弦和余弦在特定近似次数下的相关方案也已在 [2, 17, 3]中提出。

本文提出了一个系统化框架及 MATLAB 实现工具 MatrixPolEval1，用于计算并验证基于 [12, Prop. 1] 的矩阵多项式稳定系数组，适用于次数为 $m = 8$、10、12 及更高的情形。该工具可自动执行筛选流程，从生成的系数组中确定最稳定的解。此外，对于公式

在 [12, Prop. 1]中无法得出稳健结果的情形，本文还提供了其他结构变体。通过应用中的实验（包括与 PS 方法的比较），展示了所提出框架的效率和数值稳定性。

本文中，$\mathbb{N}$ 表示自然数集合 $\{0, 1, 2, \dots\}$。对于 $x \in \mathbb{R}$，$\lceil x \rceil$ 表示大于或等于 x的最小整数，而$\lfloor x \rfloor$ 表示小于或等于 x的最大整数。

本文结构安排如下。第 2 节简要回顾 PS 方法。第 3 节将第 2 节中的方法加以拓展，提出一个通用的矩阵多项式求值框架，该框架比 PS 方法少用一次矩阵乘积。第 4 节介绍算法实现，包括对结构变体和数值稳定性的处理。第 5 节展示数值实验，第 6 节给出总结。

2. Paterson–Stockmeyer 矩阵多项式求值方法

矩阵多项式 (1) 的 PS 求值方案记为 $PS_m(A)$，其定义见 [10, 算法 B]：

$$PS_m(A) = \left(\dots \left(\left(\sum_{j=0}^{s} b_{m-j} A^{s-j} \right) A^s + \sum_{j=1}^{s} b_{m-s-j} A^{s-j} \right) A^s + \dots \right) A^s + \sum_{j=1}^{s} b_{s-j} A^{s-j}, \tag{2}$$

其中，$s \in \mathbb{N} \setminus \{0\}$，且 m 是 s的倍数。如果 m 不是 s的倍数，则公式 (2) 仍然适用：通过取 $m_0 = s\lceil m/s \rceil$并令 $b_k = 0$（$k = m+1$、.....、m_0）来求值 $PS_{m_0}(A)$。

对于给定数量的矩阵乘积，PS 方法可达到的最高次数（记为 m^*）为：

$$m^* = s^2 \quad \text{or} \quad m^* = s(s+1), \tag{3}$$

其中 $s \in \mathbb{N} \setminus \{0\}$。这生成最优次数序列 $\mathcal{M} = \{1, 2, 4, 6, 9, 12, \dots\}$ [11, 第 2 节。1]。此外，[4] 证明了规则 $m^* = (C_{PS} - s + 2)s$ 的最优性，其中 C_{PS} 表示 PS 方法所需的矩阵乘积数量，且 $s = \lfloor C_{PS}/2 \rfloor + 1$。该公式给出了与（3）一致的最大次数 [15, 第 2 节。1]。

若 $m \notin \mathcal{M}$，则通过选择 $m_0 = \min\{k \in \mathcal{M} : \ k > m\}$，并将额外的首项系数设为零（$b_i = 0$ 对 $i = m+1, \ldots\ldots, m_0$）[12, 第 2 节。1]，使用 $PS_{m_0}(A)$ 方案计算多项式 $P_m(A)$。因此，PS 方法的最低计算成本为：

$$C_{PS} = s + \left\lceil \frac{m}{s} \right\rceil - 2, \tag{4}$$

其中，s 的最优值为$\lfloor \sqrt{m} \rfloor$ 或 $\lceil \sqrt{m} \rceil$。

3. 超越 PS 方法的矩阵多项式求值

[12] 中提出的方法提供了一个框架，可在计算成本低于 PS 方法的情况下对矩阵多项式进行求值。本节详细介绍 $P_8(A)$的求值，其计算成本降低至 $3M$ （比 PS 方法少一次乘法），并将该框架扩展到次数为 $m > 8$的多项式。

3.1. 使用 $3M$对 P_8（A）求值的系数解

使用 $3M$对 $P_8(A)$求值时，采用以下嵌套求值方案 [12, 例 3.1]：

$$\begin{aligned} y_{02}(A) &= A^2(c_4A^2 + c_3A), \\ y_{12}(A) &= (y_{02}(A) + d_2A^2 + d_1A)(y_{02}(A) + e_2A^2) \\ &\quad + e_0y_{02}(A) + f_2A^2 + f_1A + f_0I. \end{aligned} \tag{5}$$

通过令 $y_{12}(A) = P_8(A) = \sum_0^8 b_iA^i$相等，可得到一个关于标量系数 $\{c_i, d_i, e_i, f_i\}$ 的非线性方程组。虽然诸如$f_{0,\ 1,\ 2}$和$c_{3,\ 4}$ 这样的系数可直接由 b_i求得，但其余参数取决于 e_2 [12, 的二次方程的解（见 31]式）：

$$e_2 = \frac{\frac{c_3}{c_4}\underline{de}_2 - d_1 \pm \sqrt{\left(d_1 - \frac{c_3}{c_4}\underline{de}_2\right)^2 + 4\frac{c_3}{c_4}\left(b_3 + \frac{c_3^2}{c_4}d_1 - \frac{c_3}{c_4}b_4\right)}}{2c_3/c_4}, \tag{6}$$

其中 $\underline{de}_2 = d_2 + e_2$。利用 [12,式 27]，可计算该量：$\underline{de}_2 = (b_6 - c_3^2)/c_4$。

根据 $c_4 = \pm\sqrt{b_8}$ [12, Ex. 3.1] 的符号，该表达式最多可得出四组潜在解集。该方案的一个基本要求是 $b_8 \neq 0$。如果 $b_8 < 0$，框架会计算 $y_{12}(A) = -P_m(A)$。这样可以避免 c_4 变为复数，从而在 A 为实矩阵时避免进行复数运算。

3.1.1. 奇异情形的处理 $c_3 = 0$

[12] 中的原始分析未讨论 $c_3 = 0$ 这一情形（发生于 $b_7 = 0$ 时），而这会导致式 (6) 中出现不定式。我们的框架指出，在这种情形下，系统会化简为如下线性形式：

$$\begin{aligned} c_4 &= \pm\sqrt{b_8}, \quad d_1 = b_5/c_4, \quad e_2 = b_3/d_1, \\ d_2 &= b_6/c_4 - e_2, \quad e_0 = (b_4 - d_2 e_2)/c_4. \end{aligned} \tag{7}$$

此特定分支需要附加条件 $b_5 \neq 0$ ，以确保 e_2 定义明确。

3.2. 数值稳定性与系数选择

多个解集的存在 necessitates 基于数值稳定性的选择准则。按照 [12], ，我们通过使用有限精度算术，评估每个重构系数 b_i 的相对误差 er_i ，以此判断稳定性。如果 $\tilde{x}$表示将 x舍入到目标精度后的值，则 b_8 的误差为：

$$er_8 = \frac{|b_8 - \tilde{c}_4^2|}{|b_8|}. \tag{8}$$

如果所有 er_i 都处于目标精度下单位舍入误差u的量级（例如，对于 IEEE 单精度为 $u = 2^{-24} \approx 5.96 \times 10^{-8}$，对于 IEEE 双精度为 $u = 2^{-53} \approx 1.11 \times 10^{-16}$ ），则认为该解集是稳定的。为确保鲁棒性，该框架在$b_i = 0$时进一步使用绝对误差进行检查。这可以避免除以零，并有助于自动选择最准确的求值方案。

3.3. $P_m(A)$ 在 $m \geq 8$ 时的一般求值公式

在 [12, 第 3], 节中，介绍了一种嵌套求值方案，利用 $s + 1$ 次矩阵乘积计算次数为 $m = 4s$ ($s \geq 2$) 的矩阵多项式 $P_m(A)$。其结构定义如下：

$$y_{0s}(A) = A^s \sum_{i=1}^{s} c_{s+i} A^i, \tag{9a}$$

$$\begin{aligned} y_{1s}(A) = &\left(y_{0s}(A) + \sum_{i=1}^{s} d_i A^i \right) \left(y_{0s}(A) + \sum_{i=2}^{s} e_i A^i \right) \\ &+ e_0 y_{0s}(A) + \sum_{i=0}^{s} f_i A^i, \end{aligned} \tag{9b}$$

其中， c_{s+i}、 d_i、 e_i 和 f_i 是标量系数。对于 i = 2、3、.....、 s，幂 A^i 只计算一次并存储起来。计算 $y_{1s}(A)$ 的总成本为 $C_{y_{1s}} = s + 1$ 次矩阵乘积（M）。

令 $y_{1s}(A) = P_m(A)$ 相等，可得一个包含 $4s+1$ 个方程的方程组。若满足以下条件:

$$b_{4s} \neq 0, \tag{10a}$$

$$d_s \neq e_s \quad \text{for } s > 2, \tag{10b}$$

该方程组便可简化为求关于变量e_s的多项式的根。同样，若首项系数 b_{4s} 为负，则框架令$y_{1s}(A) = -P_m(A)$ 相等。这确保系数 $c_{2s} = \pm\sqrt{b_{4s}}$ 保持为实数，从而在 A 为实矩阵时避免复数运算 [12, 第 3] 节。

对于次数为 $m \geq 8$ 的多项式，方案（9b）被整合到类 PS 结构 $z_{1ps}(A)$ [12, 式 52] 中。令 $m = 4s + p$，其中 $p \geq 0$。广义公式 $z_{1ps}(A)$ 定义如下:

$$\begin{aligned} z_{1ps}(A) = \Bigg(\dots \Bigg(\Bigg(y_{1s}(A)A^s + \sum_{i=1}^{s} a_{p-i}A^{s-i} \Bigg) A^s + \sum_{i=1}^{s} a_{p-s-i}A^{s-i} \Bigg) A^s + \dots \\ \times A^s + \sum_{i=1}^{s} a_{p-(t-1)s-i}A^{s-i} \Bigg) A^r + \sum_{j=0}^{r-1} a_j A^j, \end{aligned} \tag{11}$$

其中， $t = \lfloor p/s \rfloor$ 是次数为 s 的完整分块数量，而 $r = p \bmod s$ 是余数分块的次数，因此 $s > r \geq 0$。

定理 3.1。 设 $P_m(A)$ 为次数为 $m \geq 8$ 的矩阵多项式。对于$s \geq 2$ 和 $m = 4s + p$ $(p \geq 0)$， $P_m(A)$ 可使用(11) 进行计算，计算成本为

$$C_{z_{1ps}}(m) = (s + 1 + \lceil p/s \rceil)M. \tag{12}$$

此外，对于任意 $m \geq 8$ （不包括 $m = 9, 11$），存在某个 s，使得 $C_{z_{1ps}}(m) = C_{PS}(m) - 1$，其中 $C_{PS}(m)$ 是最优 *PS* 方法的计算成本。

证明。 </style $s + 1$ 的代价计入了式（9a）中的 $y_{0s}(A)$、式（9b）中的$y_{1s}(A)$以及初始幂 A^2、 A^3、 $\dots$ A^s的计算代价。项 $\lceil p/s \rceil$ ac-计入了式（11）的嵌套层级中 A^s 的 t 次乘法，以及在$r > 0$时额外的一次最终乘法 A^r。

关于与 PS 方法相比的效率（其成本由 (4) 给出），我们分析可用的次数 m：

- 对于 $s = 2$，公式涵盖 $m = 8 + p$：
  - $m = 8\,(p = 0)$：$C_{z_{1ps}} = 3M$，而 $C_{PS} = 4M$。- $m = 9\,(p = 1)$：$C_{z_{1ps}} = 4M$。最优 PS 成本也是 $4M$（在 (2) 中使用 $s = 3$）；因此，这种方法在此特定情况下并未带来成本降低。- $m = 10\,(p = 2)$：$C_{z_{1ps}} = 4M$，而 $C_{PS} = 5M$。- $m = 11\,(p = 3)$：$C_{z_{1ps}} = 5M$。由于 $m = 11$ 的最优 PS 成本也是 $5M$（例如，使用 $s = 3$ 或 $s = 4$），因此对于此次数无法实现计算节省。

- 对于 $m \geq 12$，通过选择合适的 s 和 p，可以达到任意次数，并始终实现 $1M$ 的成本降低：
  - 对于 $s = 3$，通过改变 $p \in \{0, 1, 2, 3\}$，即可覆盖范围 $m \in [12, 15]$。- 对于 $m = 16$，我们转而采用 $s = 4\,(4s = 16)$。

一般而言，对于任意 $s \geq 3$，次数范围 $[4s, 4s + s]$ 均可完全覆盖。由于对于所有 $s \geq 4$，下一个范围的起点 $4(s + 1)$始终小于或等于当前范围的终点 $5s$（具体来说，$4(3) + 3 = 15$ 之后是 $4(4) = 16$），因此 $m \geq 12$可用的次数之间不存在间隙。在所有这些情况下，与最优 PS 成本相比，成本减少了一次矩阵乘积。 □

公式 (11) 的实际应用取决于数值稳定性。如 [12, Ex. 3.2], 所述，为了保持精度，需要进行系数选择。例如，使用不稳定的系数组，通过 $m = 28$ 和 $s = 7$ 计算 exp(1)，会产生 9.81×10^{-5}的相对误差；而所提出的框架所确定的最稳定系数组得到的结果为 $8.63 \times 10^{-17} < u$，如第 5 节所示。

如果特定数对 (s, p) 找不到稳定解，该框架允许针对相同的 m 探索替代分解。对于 $m = 28$，这包括 $(s = 4, p = 12)$、$(s = 5, p = 8)$ 或 $(s = 6, p = 4)$。对于特定情况 $m = 12$，[6, 第 5.1] 节中给出的替代公式避免了

平方根计算，且只要求首项系数非零。通过将该方案重新表述为以下形式，也可以绕过式（9a）和（9b）中对首项系数平方根的要求：

$$y_{0s}(A) = A^s \left(A^s + \sum_{i=1}^{s-1} c_{s+i} A^i \right), \tag{13a}$$

$$y_{1s}(A) = b_{4s} \left(y_{0s}(A) + \sum_{i=1}^{s} d_i A^i \right) \left(y_{0s}(A) + \sum_{i=2}^{s} e_i A^i \right) + e_0 y_{0s}(A) + \sum_{i=0}^{s} f_i A^i, \tag{13b}$$

其中 $b_{4s} \neq 0$ 是 $P_m(A)$ 的首项系数。将嵌套结构 (9a) 和 (9b) 与 (11) 结合，会生成多个解集。这种冗余使系统能够自动选择最稳定的解，从而提高计算的数值精度。

4. 数值实现与求解器选择

所提出的方法总结于算法 1 中。如 [12, 第 237–240], 页所示，通过令 $y_{1s}(A)$ 和 $P_m(A)$ 的系数相等而得到的 $4s+1$ 非线性方程组，可通过变量代换转化为求解一元多项式的根。因此，实现采用 MATLAB 符号数学工具箱 [18] 中的 vpasolve 函数。

与收敛到单个局部解的标准数值求解器不同，vpasolve 利用可变精度算术（VPA）搜索多组根。这是所提出算法的一项关键特性，因为它能够探索解空间，从而找出数值稳定性最高的一组系数。此外，在求解过程中使用 VPA 并提高有效数字的位数，可以减少处理病态多项式方程组时通常会累积的舍入误差。虽然符号计算的初始成本可能较高，但这项计算是在离线状态下完成的。获得稳定的系数后，即可将其存储起来，用于高效的矩阵多项式求值。

为了提供更大的灵活性，该算法基于基准层级为 $y_{1s}(A)$实现了三种结构变体

$$y_{0s}(A) = A^s \sum_{i=1}^{s} c_{m+1-s+i} A^i, \tag{14}$$

其中，系数 c 按照相关顺序进行索引，以便于数值实现并确保与所提供的 MATLAB 代码保持一致。通过下文详述的 type_pol 参数选择，这些变体扩展了稳定实解的搜索空间，其定义如下：

1. 类型 1（对应原始公式 (9b)）：

$$\begin{aligned} y_{1s}(A) = &\left(y_{0s}(A) + \sum_{i=1}^{s} c_{m+2-2s+i-1} A^i \right) \\ &\times \left(y_{0s}(A) + \sum_{i=2}^{s} c_{m+3-3s+i-2} A^i \right) \\ &+ c_{m+2-3s} y_{0s}(A) + \sum_{i=0}^{s} c_{m+1-4s+i} A^i \end{aligned} \tag{15}$$

2. 类型 2：

$$\begin{aligned} y_{1s}(A) = &\left(y_{0s}(A) + \sum_{i=0}^{s} c_{m+1-2s+i} A^i \right) \\ &\times \left(y_{0s}(A) + \sum_{i=2}^{s} c_{m+2-3s+i-2} A^i \right) \\ &+ \sum_{i=0}^{s} c_{m+1-4s+i} A^i \end{aligned} \tag{16}$$

3. 类型 3：

$$\begin{aligned} y_{1s}(A) = &\left(y_{0s}(A) + \sum_{i=1}^{s} c_{m+2-2s+i-1} A^i \right) \\ &\times \left(y_{0s}(A) + \sum_{i=1}^{s} c_{m+2-3s+i-1} A^i \right) \\ &+ \sum_{i=0}^{s} c_{m+1-4s+i} A^i \end{aligned} \tag{17}$$

在多个解中选择最优系数组，依据的是稳定性度量 ϵ_{max}。以往的方法 [12, 13, 14]通常依赖相对误差准则，而我们的框架将稳定性检查扩展到系数趋于零的多项式，以确保鲁棒性。具体而言，对于每个重构系数 b_i，其误差 δ_i 在 $b_i \neq 0$ 时按 $|b_i - b_i|/|b_i|$ 计算（参见 [12, 例 3.1]），否则按绝对误差 $|b_i - b_i|$ 计算。这种混合方法可避免验证阶段出现数值奇异性，并使工具能够为更广泛的矩阵多项式识别稳定配置。

算法的主要输入和输出参数详见表 1。为确保正确执行，实现会验证输入，并在首项系数为零（$b_m = 0$）、结构参数 s 不满足所需界限（$s < 2$ 或 $4s > m$），或指定的精度既非“single”也非“double”时发出错误。不过，输出变量 c_vpa 提供了使用 ndigits 位变精度算术计算得到的系数，用户可以手动将其舍入到任意目标精度。此外，输出变量 all_cvpa 包含在同一高精度环境中获得的所有实数和复数解集，便于用户手动选择其他解集。请注意，默认情况下，参数 s 取能够实现相对于 PS 方法节省 $1M$的最小值，从而最大限度地减少存储的中间矩阵幂数量并降低内存开销。

算法 1 的 MATLAB 实现可在 https://github.com/hipersc/MatrixPolynomials/ 获取。函数名称中的后缀“1”表示相较于 PS 方法减少了一次矩阵乘积。为便于使用，该函数采用上述默认参数，因此可以通过 c = MatrixPolEval1(b) 这样的简化调用方式运行。此最简语法会自动返回由向量 b 定义的多项式在 IEEE 双精度下的最优计算系数。有关高级配置和输出的更多详情，请参阅函数内置文档。

5. 数值结果

本节将通过若干数值示例，展示所提出方法的有效性和灵活性。所有数值实验均使用 MATLAB R2024a 在配备 M4 Max 芯片和 48 GB 内存的 Apple MacBook Pro 上完成。

表 1：算法 1 的输入参数和输出参数。

输入参数	
b	多项式系数向量 $\{b_0, \ldots, b_m\}$（默认：指数函数的 8 次泰勒展开）。
精度	目标 IEEE 算术（默认：‘双精度’，选项-：‘单精度’、‘双精度’）。
type_pol	结构变体选择（默认值：1，选项：1, 2, 或 3）。
有效数字位数	VPA 的数字位数（默认值：双精度为 32，单精度为 16）。
s	最大矩阵幂 A^s（默认值：最小的 s 相较于 PS 方法可 $1M$ 节省内存使用，从而优化内存使用）。
输出参数	
c_精度	系数 c舍入至目标 IEEE 精度以最大限度提高稳定性。
首项_系数_符号	b_m (±1) 的符号。如果 -1，则计算系数对于 $-P_m(A)$。
c_vpa	在 VPA（有效数字位数）中计算的稳定系数。
er_min	最大重构误差中的最小相对误差-构误差。
全部_cvpa	求解器找到的所有有效解集的集合
节省量	°与.....相比节省的矩阵乘积数量最优 PS 方法。
s, p	结构参数 s 和的选定值 p.

算法 1 基于重构误差测试选择稳定矩阵多项式求值系数。要求：b、精度、type_pol、s、有效数字位数 确保：c_prec、leading_coeff_sign、c_vpa、er_min、all_cvpa、节省量、s、p 1：如果 $m < 8$ or $m \in \{9, 11\}$ 则 2：推荐使用 PS 方法并退出。3：结束 if 4：如果 $b_m < 0$ 则 5：leading_coeff_sign $\leftarrow -1$6：改变系数的符号：b $\leftarrow -$b。7：否则 8：leading_coeff_sign $\leftarrow 1$9：结束 if 10：如果未提供 s，则根据定理 3.1 选择可节省相较于 PS 方法的最小 s 。11：根据 type_pol 从 (15)–(17) 中选择结构变体。12：以有效数字位数的精度初始化符号环境。13：根据 y_{1s}(A) $= P_m$(A) 构建非线性方程组。14：使用 vpasolve 求解该方程组，得到所有可能的解集 all_cvpa。15：将每个实数解集舍入至目标精度。ˆ 16：对于每个实数解集，重构系数 b_i 并计算稳定性度量 ϵ_{max} =max$_i\{\delta_i\}$，其中：ˆ ˆ$\delta_i = |b_i - b_i|/|b_i|$ 如果 $b_i \neq 0$ [12, 例 3.1], 且 $\delta_i = |b_i - b_i|$ 否则。17：选择使 ϵ_{max} 最小的解集，以定义 c_prec 和 c_vpa，并设 er_min $\leftarrow$ min$\{\epsilon_{max}\}$。18：如果 er_min $> 10u$ 则 19：发出有关潜在不准确性的警告。20：否则 21：发出通知，说明公式是稳定的。22：结束 if 23：计算节省量 $= C_{PS}$(m) $- C_{z1ps}$(m)。24：返回 c_prec、leading_coeff_sign、c_vpa、er_min、all_cvpa、节省量、s、p

示例 5.1. 参数选择灵活性。

在 *[12,* 例 *3.2*，第 *243],* 页中，给出了一个示例，以凸显选择稳定的系数组至关重要。该示例涉及使用 (9a) 和 (9b) 中定义的求值方案，求取次数为 $m = 28$的矩阵指数的泰勒近似，其中 $s = 7$。根据 (12) 和 (4)，该方案的计算成本比 *PS* 方法低 $1M$ 。

然而，研究发现存在一个重数为 *10* 的解，使用该特定系数组会导致严重的数值不稳定性。对 $x = 1$ 下的标量情形 $\exp(x)$ 进行求值，清楚地展示了这种不稳定性，其相对误差为 $|\exp(x) - y_{1,s=7}(x)|/\exp(x) = 9.81 \times 10^{-5} \gg u$ （采用 *IEEE* 双精度算术）。此外，最佳可用解的稳定性检查为 $\mathcal{O}(10^{-15}) > 10u$。函数调用证实了这一点

c =MatrixPolEval1(1./factorial(sym(0:28)), *[], [], [], 7*)

该调用会发出警告，指出求值公式可能不准确。

相比之下，使用默认调用

c =MatrixPolEval1(1./factorial(sym(0:28)))

该函数会根据(11)、(9a)和(9b) 以及 $s = 4$ 和 $p = 12$，自动提出求值公式。这一选择会产生*er_min*= 8.63×10^{-17}。使用 *200* 位可变精度算术进行求值时，相对误差为 $|\exp(1) - z_{1,\ p=12,\ s=4}$*<style id='25'>(1)*$|/\exp(1) \approx 5.32 \times 10^{-17} < u$。

如果这些公式的稳定性检查未通过，可以选择类型_多项式 的其他备选方案。如果这些备选方案也不稳定，还可以探索 s 的其他取值，以实现 $1M$ 节省。例如，在以下调用中设置 $s = 5$:

[c, ~, c_vpa, er_min, all_cvpa, 节省量, s, p] =...
MatrixPolEval1(1./factorial(sym(0:28)), [], [], [], 5)

得到 $p = 8$ 和 节省量*=1*。变量 所有*_cvpa* 包含 *8* 个实数解，其中 *er*_最小 $= 8.63 \times 10^{-17}$，同时保持 $\exp(1)$ 的相对误差为$5.32 \times 10^{-17} < u$。

选择 $s = 6$ 和 $p = 4$可得到类似的*er*_最小值，以及相对于$\exp(1)$的相对误差，同时还可节省量*=1*，并得到 *20* 组系数（其中 *4* 组为实数）。本例证实，求值公式在寻找稳定且高效的参数方面具有高度灵活性。

例5.2. 矩阵指数的实数解与稳定性检验。

文献 *[12, Ex. 3.2]*分析了使用(11)与(9a) 和(9b) 求值次数为 $m \in \{8, 12, 16, \ldots, 81\}$ 的矩阵指数泰勒近似的稳定性。在该研究中，求值公式 z_{1ps}、 y_{1s}和y_{0s} 分别由(11)、(9b)和(9a)定义。研究证明，根据其中确立的稳定性准则，对于每个 m 值，都存在实数且稳定的系数组。

以下 *MATLAB* 代码使用函数*MatrixPolEval1* 获取所有泰勒次数 $m \in \{8, 10, 12, 13, \ldots, 81\}$ 对应的系数；与 *PS* 方法相比，可节省一次矩阵乘积（$1M$）：

```
k = 0; for m = [8 10 12:81]k = k + 1; b = 1./
factorial(sym(0:m)); % 获取系数和稳定性检查值[c, ~, ~, er_
min, ~, ~, s, p] = MatrixPolEval1(b); er_minvec(k) =er_
min; end er_min_max =max(er_minvec)
```

该代码的执行结果证实，对于所有考察的次数，求值方案 (9a)、(9b) 和(11) 都存在一组真实且稳定的系数。*er_min* 的最大值为 2.07×10^{-16}，在参数 $s = 6$ 和 $p = 22$ 下于 $m = 46$处取得。

为进一步验证稳定性，考虑一个标量 $x > 0$ ，使得泰勒余项的第一项等于双精度下的单位舍入误差 $u = 2^{-53}$ ；即 $x^{47}/47! = 2^{-53}$，从而得到 $x \approx 8.40$。使用该值对应的求值公式，相对误差为 $|\exp(x) - z_{1,\ p=22,\ s=6}(x)|/\exp(x) = 1.40 \times 10^{-17} < u$。在此示例中，结果证实，即使 x 的幅值增大，所提出的系数仍能保持数值精度。

例5.3。 有限几何级数、控制理论和网络分析中的应用。

考虑有限矩阵几何级数:

$$\Psi(N, A) = I + A + A^2 + \cdots + A^{N-1}. \tag{18}$$

这个多项式是多速率采样数据系统 *[19, 8]*以及大型离散时间控制系统分析中的基础，尤其用于计算可达性和有限时域格拉米矩阵 *[1]*。在网络分析中，它用于评估路径连通性和 *Katz*中心性指数，计算长度最多为 $N-1$ *[7]* 的所有路径的总影响。此外， $\Psi(N, A)$是高阶 *Schulz*型迭代方法的主要计算模块，这些方法用于广义逆的数值计算 *[9]*。

根据 *[19,* 表 *1],* ，计算 $\Psi(17, A)$ （次数为 $m = 16$）的成本为 $6M$。虽然 *[8,* 第 *III* 节*]* 为 $N \leq 36$提出了一种改进算法，但其计算成本仍为 $6M$。通过调用`c = MatrixPolEval1(ones(17,1))`使用所提出的工具，可以获得一组 *IEEE* 双精度系数，用于采用(11) 中的方案并使用参数 $s = 4$ 和 $p = 0$计算 $\Psi(17, A)$ 。这将成本降至 $5M$ （节省 $1M$ ）。稳定性检查结果为 $E_r = 3.69 \times 10^{-16} < 10u$，表明这些公式在数值上稳定，且很可能具有良好的精度。

请注意，调用
`[c,~, c_vpa, er_min, all_cvpa] =MatrixPolEval1(ones(17,1))`可识别出 *4* 组实系数和 *8* 组复系数。该函数默认选择最稳定的实系数组，并通过对`er_min` $\leq 10u$进行稳定性检查来确认其精度。

为便于比较， *[19,* 第 *II* 节*]* 提出的$N \in \{9, 13, 17\}$ 求值方案如下：

$$\Psi(9, A) = I + (I + A^2)(A + A^2)(I + A^4), \tag{19a}$$
$$\Psi(13, A) = I + (I + A^2)(A + A^2)(I + A^4 + A^8), \tag{19b}$$
$$\Psi(17, A) = I + (I + A^2)(A + A^2)(I + A^4)(I + A^8), \tag{19c}$$

其中，矩阵幂只计算一次并重复使用。这些求值的成本分别为 $4M$、 $5M$ 和 $6M$。相比之下，`MatrixPolEval1` 使用 (9a) 、 (9b) 和 (11) 提出的方案在每种情况下都将成本降低了$1M$ 。

我们使用以下 *MATLAB* 代码，对 $\Psi(17, A)$ 的两种方法进行了比较，使用了 *100* 个随机 100×100矩阵：

```
rng(0); % 确保结果可复现 c =
MatrixPolEval1(ones(17,1)); for i = 1:
100
```

```
    A = rand(100);
    I = eye(size(A)); A2 = A^2; A4 = A2^2; A8 = A4^2;
    % Westreich evaluation
    W = I + (I + A2) * (A + A2) * (I + A4) * (I + A8);
    % Proposed evaluation (s=4, p=0)
    A3 = A2 * A;
    y0 = A4 * (c(1)*A4 + c(2)*A3 + c(3)*A2 + c(4)*A);
    y1 = (y0 + c(5)*A4 + c(6)*A3 + c(7)*A2 + c(8)*A) ...
         * (y0 + c(9)*A4 + c(10)*A3 + c(11)*A2) ...
         + c(12)*y0 + c(13)*A4 + c(14)*A3 ...
         + c(15)*A2 + c(16)*A + c(17)*I;
    er_rel(i) = norm(y1 - W, 1) / norm(W, 1);
end
```

最大_误差 =最大(er_相对)

结果显示，最大相对差为 $4.00\times10^{-16} \approx 3.61u$。对于 10000×10000 个矩阵，差值为 $8.77 \times 10^{-17} < u$。这些比较结果汇总于表 *2*，表明精度较高，且与 *[12, Ex. 3.1]* 中的稳定性检查结果一致。

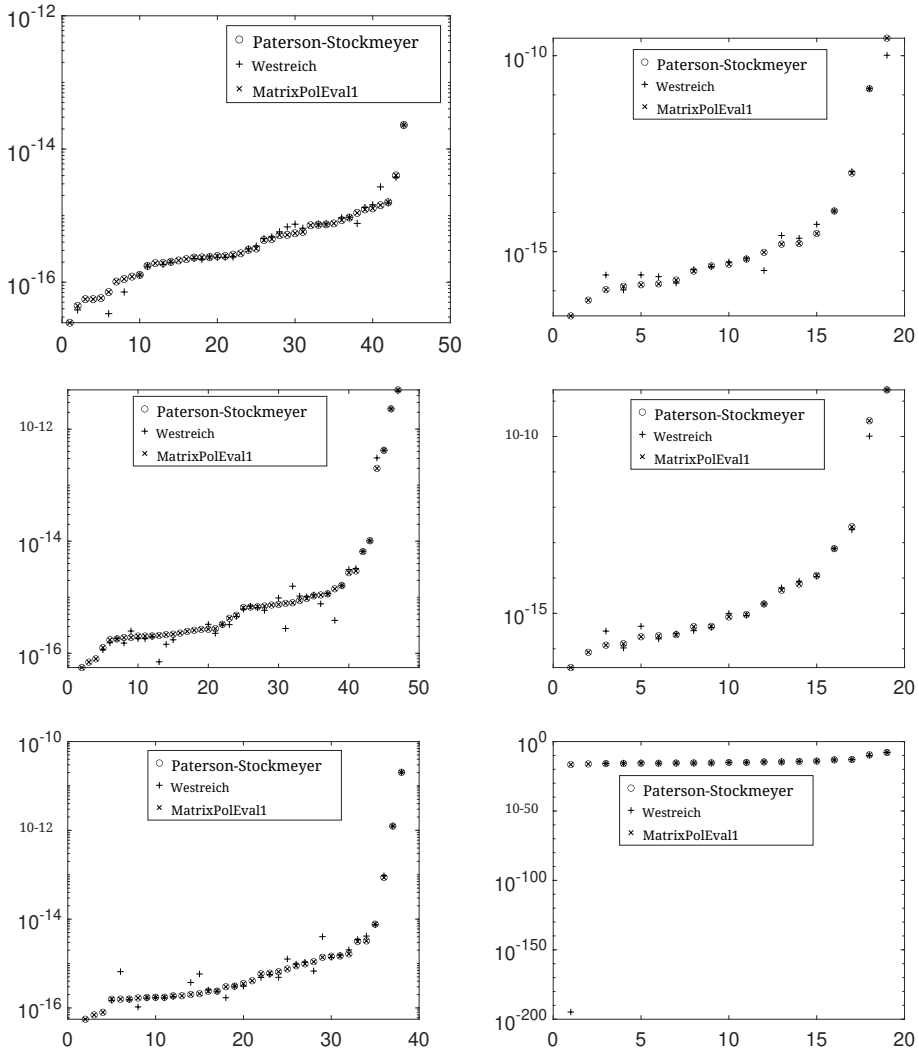
矩阵大小 (n)	相对差异（ u的倍数）		
	所提方法与 PS 对比	所提方法与 W 对比	W 与 PS 对比
100	$3.28u$	$3.61u$	$3.30u$
1000	$2.03u$	$2.01u$	$2.06u$
10000	$1.92u$	$0.76u$	$1.91u$

表 2：对 100 个大小为 $n = 100$、1000 和 10000 的随机矩阵求值 $\Psi(17, A)$ 时的相对差。结果比较了使用 MatrixPolEval1 中的系数进行矩阵多项式求值、采用 $(m, s) = (16, 4)$ 的 PS 方法 (2)，以及 Westreich (W) 公式 (19c)。数值以 IEEE 双精度算术中的单位舍入误差的倍数表示 $u = 2^{-53} \approx 1.11 \times 10^{-16}$。

最后，我们测试了矩阵计算工具箱 *(MCT) [5]* 和 *Eigtool MATLAB* 软件包 *(EMP) [20]* 中的矩阵，尺寸为 $n = 100$，500 和 1000。对于 $\Psi(N,\ A)$，当 $N = 9$，13 和 17 时的参考“精确”值，是通过 *MATLAB* 符号数学工具箱使用可变精度算术 *(VPA)* 计算得到的。具体而言，将每个矩阵 A 转换为符号表示，并使用 *PS* 公式 (2) 计算求和 $\sum_{i=0}^{N-1} A^i$；当次数为 $m = 8$，12 和 16 时，采用了 $s = 4$，精度为 *100* 位小数

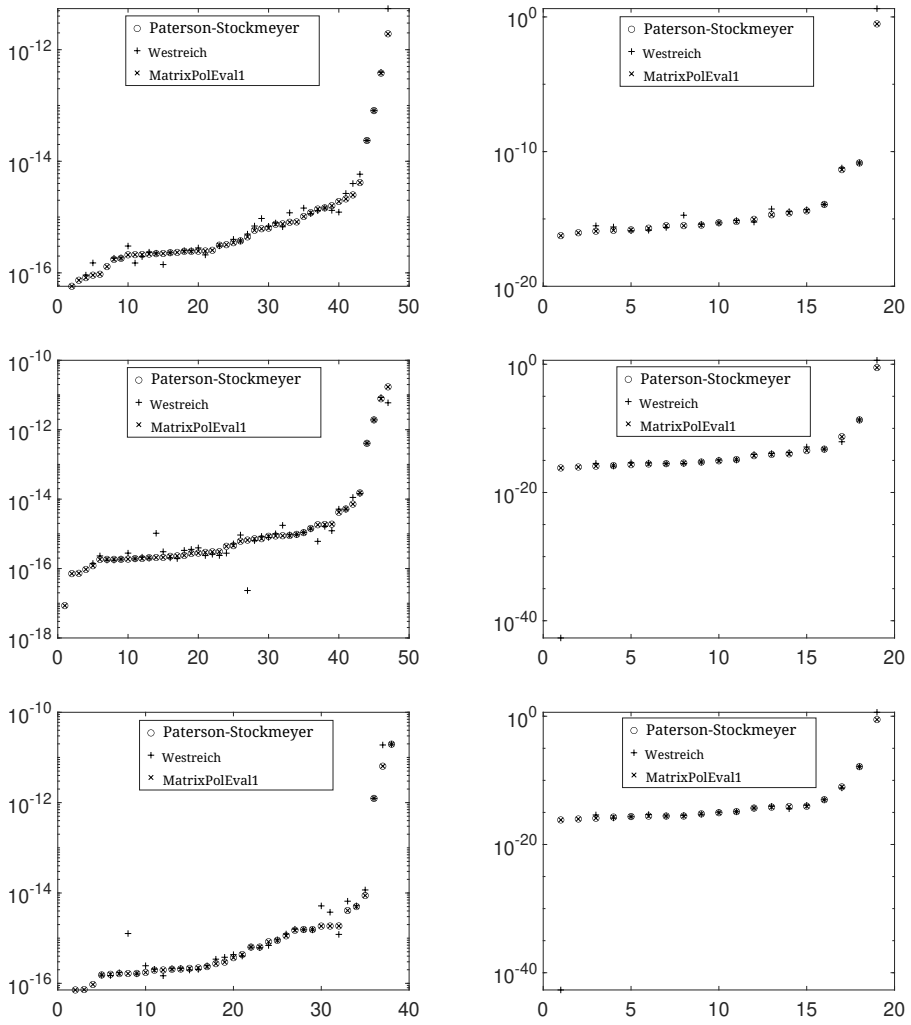
位。按照该方法计算时，*MCT* 中的矩阵 *16*、*17*、*21*、*42* 和 *44*，以及 *EMP* 中的矩阵 *4* 得到了无穷大或 *NaN* 结果，因此从测试中剔除。

图 *1*、图 *2* 和图 *3* 展示了排序后的 *1*-范数相对误差。虽然 *Westreich* 的公式偶尔会呈现略有差异的误差范数，但 *PS* 方法和所提出的 *MatrixPolEval1* 方案相对于高精度 *VPA* 基准的精度几乎完全相同。



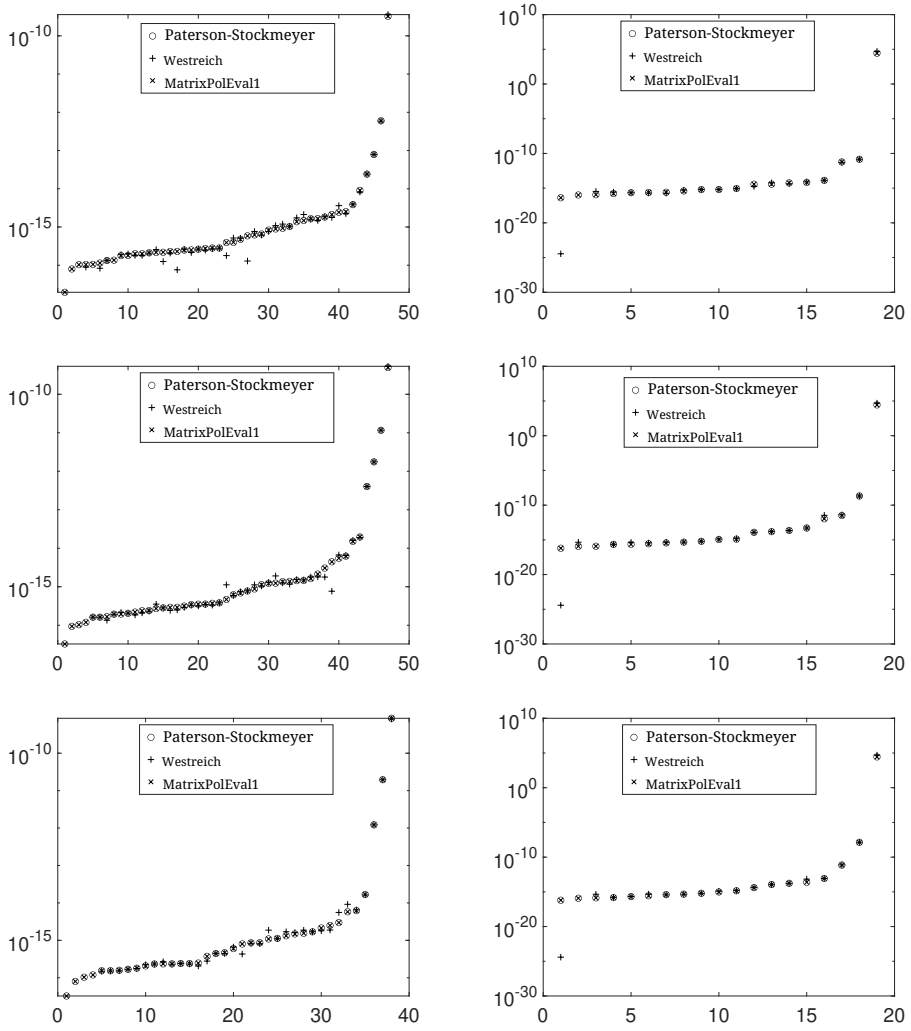
(a) MCT。 (b) EMP。

图 1： $\Psi(9, A)$ 求值的相对误差比较。图中展示了矩阵维数为 $n = 100$ （上）、 $n = 500$ （中）和 $n = 1000$ （下）时，IEEE 双精度下按升序排列的 1-范数相对误差。比较的方法包括 Paterson–Stockmeyer (PS) 公式 (2)、Westreich 方法 (19a) [19], 以及通过 MatrixPolEval1 得到的优化求值公式。左侧图对应 MCT 测试矩阵，右侧图对应 EMP 矩阵。



(a) MCT。 (b) EMP。

图 2： $\Psi(13, A)$ 求值的相对误差对比。图中展示了在 IEEE 双精度下，针对矩阵维数为 $n = 100$ （上）、$n = 500$ （中）和 $n = 1000$ （下）的测试结果，按 1-范数相对误差排序。比较的方法包括 PS 公式 (2)、Westreich 方法 (19b) [19], 以及通过 MatrixPolEval1 得到的优化求值公式。左侧图对应 MCT 测试矩阵，右侧图对应 EMP 矩阵。



(a) MCT。 (b) EMP。

图3：求值 $\Psi(17, A)$ 的相对误差比较。图中展示了矩阵维数为 $n = 100$ （上）、$n = 500$ （中）和 $n = 1000$ （下）时，以 IEEE 双精度计算得到的排序后 1-范数相对误差。比较的方法包括 PS 公式 (2)、Westreich 方法 (19c) [19], 以及通过 MatrixPolEval1 得到的优化求值公式。左侧图对应 MCT 测试矩阵，右侧图对应 EMP 矩阵。

6. 结论

本文提出了一个框架及其相应的 MATLAB 工具 MatrixPolEval1，旨在优化矩阵多项式求值。与广泛使用的 PS 方法相比，所提出的方案通过节省一次矩阵乘积（$1M$），始终能够降低计算成本。该方法具有通用性，可用于求值次数为 8、10 和 ≥ 12 且首项系数非零的矩阵多项式，为各种数学和工程应用提供比传统技术更高效的替代方案。

所提出方法的一项显著优势，是能够从所有潜在解集中识别出稳定的系数组。如果未找到稳定的解集，该工具会通过内置机制向用户发出警告；此外，其参数具有灵活性，因此在其他配置可能产生不稳定结果的情况下，也能够发现稳定解。

对于有限几何级数等特定应用 $\Psi(N, A)$，所提出的方法在保持所需数值精度的同时，为经典专用算法（例如 Westreich 的方法）提供了一种具有竞争力的替代方案。使用矩阵计算工具箱（MCT）和 Eigtool MATLAB 软件包（EMP）进行的大量测试证实，稳定性检查参数 er_min 是实际数值性能的可靠预测指标。这一行为与先前的结果一致，在先前的研究中，同一稳定性判据已成功应用于矩阵指数、矩阵余弦及其他相关矩阵函数的求值[13, 14]。除这些具体情形外，该工具还旨在作为一种通用实用程序，以较低的计算成本稳定地求值任意矩阵多项式。未来的工作将致力于提供新的矩阵多项式求值工具，以实现更大的计算节省。

CRediT 作者贡献声明

Jose Miguel Alonso： 软件、研究、验证、数据整理、撰写——审阅与编辑。

Jorge Sastre： 概念化、方法、软件、形式分析、研究、资源、撰写——初稿、撰写——审阅与编辑、监督、项目管理、经费获取。

Javier Ibáñez： 研究、验证、形式分析、撰写——审阅与编辑。

Emilio Defez： 形式分析、验证、撰写——审阅与编辑。

写作过程中使用生成式人工智能和人工智能辅助技术的声明

在准备本作品期间，作者使用 Gemini 改进稿件的语言表达、行文流畅度和技术清晰度。使用该工具后，作者根据需要审阅并编辑了内容，并对技术准确性和出版物的最终内容承担全部责任。

利益冲突声明

作者声明，他们不存在任何已知的、可能影响本文所报告工作的竞争性经济利益或个人关系。

致谢

本研究得到了瓦伦西亚自治区政府 CIAICO/2023/270 项目的资助。

参考文献

[1]A.C. Antoulas，《大规模动力系统的逼近》，工业与应用数学学会，费城，2005年。[2] P. Bader、S. Blanes、F. Casas，使用优化的泰勒多项式逼近计算矩阵指数，《Mathematics》，7（12）（2019年）1174。[3] P. Bader、S. Blanes、F. Casas、M. Seydaoğlu，一种高效算法：计算斜厄米矩阵的指数以用于薛定谔方程的时间积分，《数学与计算机模拟》，194（2022年）383–400。[4] M. Fasi，Paterson–Stockmeyer 方法在求值矩阵多项式和有理矩阵函数方面的最优性，《线性代数及其应用》574（2019年）182–200。[5] N.J. Higham，《矩阵计算工具箱》。2002年。在线地址：https://www.mathworks.com/matlabcentral/fileexchange/2360-the-matrix-computation-toolbox（访问于2026年3月24日）。

[6] E. Jarlebring、G. Lorentzon，与固定次数矩阵-矩阵乘法相关的多项式集合，arXiv:2504.01500，2025年。[7] L. Katz，基于社会计量分析得出的新地位指数，《Psychometrika》18（1）（1953年）39–43。[8] L. Lei、T. Nakamura，一种快速求值矩阵多项式 $I + A + \ldots + A^{N-1}$ 的算法，《IEEE电路与系统汇刊I：基础理论与应用》，39:4（1992年）299–300。[9] H. Li、Y. Li，用于计算广义逆的 Schulz型迭代，《Appl. Math. Comput.》217（22）（2011年）9173–9181。[10] M.S. Paterson、L.J. Stockmeyer，求值多项式所需的非标量乘法次数，《SIAM J. Comput.》2（1）（1973年）60–66。[11] J. Sastre，有效的矩阵函数混合有理与多项式逼近，《Appl. Math. Comput.》218（24）（2012年）11938–11946。[12] J. Sastre，高效求值矩阵多项式，《线性代数及其应用》539（2018年）229–250。[13] J. Sastre、J. Ibáñez、E. Defez，加速矩阵指数的计算，《Appl. Math. Comput.》340（2019年）206–220。[14] J. Sastre、J. Ibañez、P. Alonso、J. Peinado 和 E. Defez，快速泰勒多项式求值以计算矩阵余弦，《J. Comput. Appl. Math.》354（2019年）641–650。[15]J. Sastre、J. Ibáñez，超越 Paterson–Stockmeyer 方法的矩阵多项式求值，《Mathematics》9（14）（2021年）1600。[16]J. Sastre、J. Ibáñez、J.M. Alonso 和 E. Defez，超越 Paterson–Stockmeyer：推进矩阵多项式计算，《WSEAS Trans. Math.》24（2025年）684–693。发表于 AMCSE，法国巴黎，2025年4月14日至16日。[17] M. Seydaoğlu、P. Bader、S. Blanes、F. Casas，以减少乘积数量的方式同时计算矩阵正弦和余弦，《Appl. Numer. Math》163（2021年）96–107。

[18]The MathWorks Inc., 《符号数学工具箱》，版本 24.1 (R2024a), 美国马萨诸塞州纳蒂克，2024年。 [在线]。网址：https://www.mathworks.com/products/symbolic.html（访问于2026年3月24日）。[19] D. Westreich, 《计算矩阵多项式 $I + A + \ldots + A^{N-1}$》, IEEE 电路与系统汇刊，36:1 (1989年) 162–164。[20]T.G. Wright, Eigtool，版本 2.1。2009年。在线网址：http://www.comlab.ox.ac.uk/pseudospectra/eigtool（访问于2026年3月24日）。