

矩阵指数的高效缩放和平方法

Sergio Blanes*

Nikita Kopylov†

Muaz Seydao~

2024 年 4 月 22 日

摘要

本研究提出了一种在给定容差范围内计算矩阵外指数的新算法。结合缩放和平方法，该算法采用了泰勒、分割和经典 Padé 方法，其性能优于最先进软件中使用的近似值。该算法可以计算矩阵与矩阵的乘积，也可以计算矩阵求逆，但可以避免求逆计算，从而使其在某些问题上更加简便。如果矩阵 A 属于一个李代数，那么 e^A 就属于其相关的李群，这是对角线 Padé 近似值预先提供的一个属性，算法还有另一个选择，即只使用这些近似值。数值实验表明，该算法的性能优于最先进的实现方法。

关键词： 矩阵指数、缩放和平方法、Padé approximants、泰勒方法、分数分解、李群

1 引言

我们给出了一种算法，用于计算矩阵指数 e^A [21]，给定公差 tol ，给定大小为 $N \times N$ 的复矩阵 A 。首先，该算法计算矩阵的范数约束 θ ，即 $\|A\|_1 < \theta$ （ $\|\cdot\|_1$ 范数可以用任何其他范数代替），然后根据 θ ，计算矩阵指数 $e_{(A)}$ 。

*Instituto de Matemática Multidisciplinar, Universitat Politècnica de València, E-46022 Valencia, Spain. 电子邮件: serblaza@imm.upv.es

†Instituto de Matemática Multidisciplinar, Universitat Politècnica de València, E-46022 Valencia, Spain. 电子邮件: nikop1@upv.es

~数学系，艺术与科学学院，Muğla Sıhpaçlı 大学，49100, Muğla, 土耳其。电子邮件: m.seydaoglu@alparslan.edu.tr

算法会从一系列选定的方法中选择能提供 e^A 近似值且公差在此范围内的方案。如果找不到匹配的方

案，算法会采用缩放和平方（进一步总结）来找到最便宜的方法匹配 $\|A\|_2 \leq \theta$ 。

应根据手头的具体问题来选择。

在数值方法中，这种适应性是有益的，举例来说，可以考虑用指数积分器求解微分方程。它们已被证明对大量问题非常有用[17]。例如，大多数 Lie-group 方法（参见 [19] 综述），如 Magnus 积分器（参见 [8] 及其中的参考文献）、Crouch-Grossman 方法 [12]、Runge-Kutta-Munthe-Kaas 方法 [22] 等，每一步都需要计算一个或多个矩阵指数。在大多数情况下，对每个矩阵指数的近似只需达到低于舍入精度的给定公差即可，这将使积分的计算成本更高。当必须保留李群结构时，我们仍然可以提供更便宜的指数近似值，但仍然保留结构。

例如，让我们考虑非线性微分方程

$$Y' = A(t, Y) Y, \quad Y(0) = Y_0 \in G, \quad (1)$$

其中， $A \in \mathbb{R}^{(N)}$ ， G 是矩阵李群，它出现在相关物理领域，如刚体力学、Lyapunov 指数的计算以及哈密顿动力学中出现的其他问题。可以证明，在矩阵李群 G 上演化的微分方程可以写成 (1) [19]。在[12]中，作者提出了如下阶数为 3 的 3 级方法

$$\begin{aligned} Y^{(1)} &= Y_n, & K_1 &= A(t_n, Y^{(1)}), \\ Y^{(2)} &= e^{(hA)^{(2)}(Y^{(1)})} K_1, & K_2 &= A(t_n + c_2 h, Y^{(2)}), \\ Y^{(3)} &= e^{(hA)^{(3)}(Y^{(2)})} K_2, & K_3 &= A(t_n + c_3 h, Y^{(3)}), \\ Y^{(n+1)} &= e^{(hA)^{(3)}(Y^{(2)})} e^{(hA)^{(2)}(Y^{(1)})} e^{(hA)^{(1)}(Y^{(0)})} Y^{(n)} \end{aligned} \quad (2)$$

与

$$\begin{aligned} A_{2,1} &= \frac{3}{4}, & A_{3,2} &= \frac{17}{108}, & A_{3,1} &= \frac{119}{216}, & B_1 &= \frac{24}{178}, & B_2 &= \frac{2}{3}, & B_3 &= \frac{13}{51}, \\ C_2 &= A_{2,1}, & C_3 &= A_{3,1} + A_{3,2} \end{aligned}$$

每一步需要计算六个矩阵指数。如果 A 的规范在每一步中变化不大，那么在每个时间步中， $\|K_1\| \sim \|K_2\| \sim \|K_3\|$ ，但系数 a_{ij} 和 b_i 的差值可达

接近一个数量级。对于这些问题，很明显，由于该方法只有三阶：(i) 只要近似值保留了所需的结构，就没有必要将矩阵指数近似到四舍五入的精度；(ii) 每个矩阵指数都可以用不同的方案近似，以达到所需的容差（在不同的步骤中，容差也可以改变），从而在保留精度和质量特性的同时降低计算成本。

第二个例子是随机微分方程 (SDE) 的计算方法，例如 [29] 明确计算矩阵指数。对于这类方程，积分器通常阶数较低，需要重复足够多的模拟次数。因此，与上面的例子类似，使用更便宜的方案来降低整体仿真时间是有益的。

另一方面，文献[20]指出，没有逆的矩阵乘法使得矩阵指数在深度学习中成为一个非常廉价的函数，PyTorch 1.7.0 中添加了泰勒算法，与 Tensorflow 相比，速度有了显著提高。对于这些问题，矩阵指数的多项式近似是有优势的。

缩放和平方也许是 MATLAB 的 `expm`、Mathematica 的 `MatrixExp` 以及 Julia 的 `exp` 和 `ExponentialUtilities.jl` 等流行计算软件包中最常用的方法之一，结合 Padé 近似值 [1, 13, 16, 6, 24]。具体来说，它基于以下关键

$$e^A = e^{A/2^s} \overset{(2)}{\circ} s \in \mathbb{N}. \quad (3)$$

指数 $e^{(A/2)^s}$ 由近似值 $w(A/2^s)$ 代替，通常选择的近似值为指数值的四舍五入误差，然后将其平方 s 次。

这项工作的目标是构建一种算法，当提供矩阵 $A \in \mathbb{C}(N) \times N$ 和容差 tol 时，计算出一个函数 $w_\alpha(A)$ ，其中 α 指的是标识方法的标签，能够近似矩阵指数 e^A ，以便

$$relerr := \frac{\|w_\alpha(A) - e^A\|_1}{\|e^A\|_1} < tol - \|A\|_1. \quad (4)$$

函数 $w_\alpha(A)$ 最常见的选项是

- 泰勒多项式： $w_\alpha(A) = t_\alpha(A)$ ，其中 $t_\alpha(x) = \sum_{n=0}^m \frac{x^n}{n!}$, $x \in \mathbb{C}$ 、
和 $|t_\alpha(x) - e^x| = O(x^{m+1})$.

- 有理 Padé 近似值: $w_{(a)}(A) = r_{(k,m)}(A)$, 其中 $r_{(k,m)}(x) = p_{(k,m)}(x)/q_{(k,m)}(x)$, 其中

$$p_{k,m}(x) = \sum_{j=0}^k \frac{(k+m-j)!k!}{(k+m)!(k-j)!j!} x^j, \quad q_{k,m}(x) = \sum_{j=0}^m \frac{(k+m-j)!m!}{(k+m)!(m-j)!j!} (-x)^j, \quad (5)$$

这是阶 s 的近似值 $= k+m$, 主要误差为项[18, 14], 其公式为

$$e^x - r_{k,m}(x) = (-1)^m \frac{(k!) (m!) (k+m)!}{(k+m)!(k+m+1)!} x^{k+m+1} + O(x^{k+m+2}). \quad (6)$$

请注意, 泰勒多项式是 Padé 近似值的特例, $t_{(s)}(x) = r_{s,0}(x)$, 在 $s = k+m, k, m > 0$ 的情况下, 其前导误差大于 $r_{(k,m)}(x)$ 。不过, 方法的性能在很大程度上取决于计算成本¹, 因此也应加以考虑。

- 有理切比雪夫近似值: $w_a(A) = \tilde{r}_{(k),m}(A)$, 其中 $\tilde{r}_{(k),m}(x) = \tilde{p}_k(x)\tilde{q}_{(m)}(x)$, $\tilde{p}_k(x)$, $\tilde{q}_{(m)}(x)$ 是 k 和 m 的多项式, 再 $k \leq m$, 系数选取如下

$$|\tilde{c}_{k,m}(x) - e| < tol_{k,m}, \quad x \in (-\infty, 0).$$

其中 $tol_{k,m}$ 取决于 k 和 m 的值, 以及多项式系数的选择。

这些方法针对的是矩阵 A 可对角化且具有负实特征值的情况, 其中一些特征值的取值非常大, 因此受到了广泛关注[11, 30]。这个问题不属于我们在本研究中考虑的问题范畴 ($\|A\|$ 必须大小适中), 但本研究中使用的大多数技术都适用, 而且这个问题将在未来得到考虑。

我们分析了大量泰勒和有理 Padé 方法, 得到了一组公差的误差范围, 并考虑到它们的计算成本, 根据所提供的矩阵 A 和公差 tol , 选择了我们的算法所使用的方法列表。

在所有可能的选择中, 我们应该使用能以最低计算成本提供所需精度的方法, 这就需要进行两类分析:

¹ 在某些情况下, 必须考虑四舍五入的准确性。

后向误差。给定一个特定的方法 w_α ，我们为该方法寻找一个相关的标量函数，例如 $\theta_\alpha(y)$ ，这样，给定一个矩阵 A 和一个正整数 s ，如果 $\|A\|_1 \leq 2^s \theta_\alpha(\text{tol})$ ，当使用 s 平方时，该方法提供的近似值的相对误差低于容差。文献中经常使用正向和反向误差分析，我们将在本文中考虑反向误差分析，类似于 [13, 16] 中的分析，并在 MATLAB 中的函数 (expm) 中实现。

计算成本在分析成本时，我们假设一个稠密矩阵-矩阵乘法的成本为 C ，并将其作为参照成本。如果一种方法的计算成本约为矩阵-矩阵乘积成本的 k 倍，我们就称这种方法的成本为 kC 。那么，给定两个矩阵 A 和 B ，计算 A^{-1} 或 $A^{-1}B$ 的成本将取为 $4C$ 。为了简化符号，我们假设 $C=1$ 。

与误差分析不同，计算成本的研究可以说是一门艺术，而不是一种正确的理论。在过去的几十年里，随着降低算法成本的新技术的出现，解决特定问题的现有最佳方法也在发生变化（未来还将随着新算法和计算机架构的出现而继续变化）。例如，泰勒方法最初被放弃，因为用标准霍纳算法计算时，需要 $m-1$ 个乘积才能计算 t_m 。然而， t_m 的 $m=2, 4, 6, 9, 16, 20$

可分别用 $k=1, 2, 3, 4, 5, 6$ 个乘积计算，大大节省了计算量，使泰勒方法具有竞争力 [23, 27]。此外，[4, 5, 26] 还表明可以进一步减少乘积，这样， $m=2, 4, 8, 12, 18$ 的泰勒多项式就可以是

分别用 $k=1, 2, 3, 4, 5$ 个乘积计算，因此它们是多问题的首选方法。

另一方面，对角 Padé 方法具有 $q_{m,m}(x) = p_{m,m}(-x)$ 的性质，这种对称性允许找到一种程序，在 $m=1, 2, 3, 5, 7, 9, 13, k=0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 13$ 的情况下，同时计算多项式 $p_{(m,m)}(x)$ 和 $q_{(m,m)}(x)$ 的程序。对于 $m=1, 2, 3, 5, 7, 9, 13, k=0, 1, 2, 3, 4, 5, 6$ 的乘积，除了计算 $r_{(m,m)}$ 的一个逆之外，还可以同时计算这两个多项式。文献 [13] 认为，由于计算 $r_{m,m}$ 的成本与计算 $k < m$ 的 $r_{k,m}$ 或 $r_{m,k}$ 的成本相同，且精度更高（见公式 (6)），因此只考虑对角 Padé 方法。然而，[25] 中的研究表明情况并不一定如此。研究表明，在 $k > m$ 的情况下，通过适当的分数分解，可以以与 $r_{m,m}$ 相同的成本计算若干近似值 $r_{(k,m)}$ ，同时提供更高的精度。

此外，在分析方法时还应考虑以下限制：

避免复杂系数。在这项工作中，我们将把一些数学垫‘e 方法分解成更简单的分数，必须注意避免使用复杂系数的方案。这项工作的目标是提出针对实矩阵进行优化的方法，同时也能有效地处理复矩阵，因此我们要寻找具有实系数的方案。请注意，我们算法中使用的方法列表可以通过添加更多针对复矩阵的复系数方法轻松扩展。此时，我们将考虑到，对于 $m=2j$, $r_{k,m}$ 中的分母，例如 $q_{(k,2j)}$, $j=1, 2, \dots$ 在矩阵指数的 Pad‘e approximation 中，有 j 个不同的复根 α_i 和它们的共轭根 $\bar{\alpha}_i$ ，所以

$$q_{(k,2j)}(x) = \delta \prod_{i=1}^j (x - \alpha_i)(x - \bar{\alpha}_i) = (x^2 - 2\operatorname{Re}(\alpha_i)x + |\alpha_i|^2),$$

和 $\delta = (-1)^m k! / (k+m)!$ 。如果 $m=2j+1$, $j=0, 1, 2, \dots$ ，则 $q_{k,2(j)+1}$ 只有一个实数根和 j 个复数根及其共轭根。在将 $r_{k,m}$ 分解为更简单的分数时，我们将考虑这一性质。

李群方法。如果 A 属于一个李代数，那么 e^A 将属于相关的李群。如果要在问题中保留这一性质，所介绍的算法允许只使用对角线 Pad‘e 近似值来计算指数，从而保留这一性质，达到四舍五入的精度。

最终，找到最有效方案的程序如下：

- 为了降低计算成本，一些 Pad‘e 近似值被分解成更简单的分数，避免了系数复杂的方案。
- 对于每个选定的泰勒法和 Pad‘e 法，例如 $w_a(x)$ ，我们用数值方法得到函数 $\theta_a(t)$ ，如果 $\|A\| \leq \theta_a(\text{tol})$ ，那么对于 $\text{tol} = 10^{-k}$, $k=0, 1, \dots, 16$ 。如果不满足这一条件，则采用缩放和平方的程序。

经过分析，我们将在第 3.3 小节中提供泰勒法和 Pad‘e 法的列表，每种方法都是矩阵的公差和规范的至少某些值的最优方法。

2 后向误差分析

在缩放和平方算法的实现过程中，对于给定矩阵 A 和公差 tol ，最佳方法和缩放参数的选择是基于对后退误差的控制 [13]。更具体地说，给定一个指数为 n 的近似值 $w_{(n)}(A)$ 、

即 $w_n(x) = e^x + O(x^{n+1})$ ，可以定义函数 $h_{(n)+1}^{(x)} = \log(e^{-x} w_n'(x))$ 、
则 $w_n(2^{-s}A) = e^{2^{-s}A + (h_{(n)+1}^{(2^{-s}A)})}$ 和

$$w_{(n)}(2^{-s}(A))^{2^s} = e^{A+2^s(h_{(n)+1}^{(2^{-s}A)})} \equiv e^{A+\Delta_A},$$

其中 $\Delta_A := 2^s h_{n+1}(2^{-s}A)$ 是源于 e^A 近似计算的后向误差。此外，如果 $h_{(n)+1}$ 具有幂级数展开

$$h_{(n)+1}(x) = \sum_{k=n+1}^{\infty} c_k x^k,$$

收敛半径不为零，那么显然 $\|h_{(n)+1}(A)\| \leq \tilde{h}_{n+1}(\|A\|)$ ，其中

$$\tilde{h}_{n+1}(x) = \sum_{k=n+1}^{\infty} |c_k| x^k.$$

因此，考虑到 A 的缩放，可以得到

$$\frac{\|\Delta_A\|}{\|A\|} = \frac{\|h_{(n)+1}(2^{-s}A)\|}{\|2^{-s}A\|} \leq \frac{\tilde{h}_{n+1}(\|2^{-s}A\|)}{\|2^{-s}A\|}. \quad (7)$$

在给定公差 tol 的情况下，可以通过数值计算

$$\theta_n(tol) = \max \left\{ \theta : \frac{\tilde{h}_{(n)+1}(\|\theta\|)}{\theta} \leq tol \right\}, \quad (8)$$

对不同的 tol 值进行计算，方法是在 150 个项之后截断相关函数序列 $\tilde{h}_{(m)+1}(\theta)$ 。然后选择 s

$\|2^{-s}A\| \leq \theta_n(tol)$ 和 $(w_n(2^{-s}A))^{2^s}$ 用来逼近 e^A 。请注意，以便

对于 $\|\Delta_A\|$ 的小值，我们可以得出

$$\|\Delta_A\| = \frac{\|w_{(n)}(A) - e^A\|}{\|e^A\|} + O(\|A\|^{n+2}).$$

在 w_n 是对角 Padé 近似值 $r_{m,m}$ 的特殊情况下，则 $n = 2m$ 。 $tol = u$ 时的 $\theta_{2m}(tol)$ 值，其中 u 是单位四舍五入，以单

表 1: 单精度和双精度的对角线 Pad'e 近似值 r_m 的 $\theta_{(2)(m)}$ 值, 阶次为 $2m$, 最小乘积数为 π 。我们用粗体标出了渐近最优阶, 在这个阶上应用缩放和平方是有利的, 因为每增加一个乘积所增加的 θ 比平方所增加的系数 2 要小。

$\pi :$	0	1	2	3	4	5	6
$2m :$	2	4	6	10	14	18	26
$u \leq 2$	$^{-24}$	8.46e-4	8.09e-2	4.26e-1	1.88	3.93	6.25
$u \leq 2$	$^{-53}$	3.65e-8	5.32e-4	1.50e-2	2.54e-1	9.50e-1	5.37

为方便读者, 表 1 收集了 [1] 中的 expm 和双精度数据。需要注意的是, MATLAB 中的 expm 仅针对 $u = 2$ 的情况实现 $^{-53} \approx 1.1 \cdot 10^{-16}$ 。根据 Higham [13], $m = 13$, 因此当需要缩放时, $r_{(13)(13)}$ 是双精度的最佳选择。当 $\|A\| \leq \theta_{26}(2^{-53}) = 5.37$ 时, [13] 中的算法取前 $m = 3, 5, 7, 9, 13$, 这样 $A \approx \theta_{2m}(u)$ 。我们将分析扩展到其他有理近似值和公差。表 2 只显示了为简化表述而选择的一组公差值的结果。我们只考虑了那些在 $\|A\|$ 和 tol 的某些值下表现最佳的方法。对角线 Pad'e 方法另列在表 3 中。

当然, Pad'e 近似值并不是在这种情况下下的唯一选择。解决问题的另一种方法是使用泰勒多项式

作为指数的基本近似值 t_m , 即 tak-

$$= \sum_{k=0}^m (A^k/k!) / k!$$

计算效率高, 因此具有竞争力
 在许多情况下都是如此。

3 计算成本

3.1 高效计算有理 Pad'e 近似值

对角线 Pad'e 近似值的形式为 $r_{m,m} = [p_m(-A)]^{-1} p_m(A)$, $p_{(m)}(A)$ 和 $p_m(A)$ 的求值都可以尽量减少矩阵乘积的数量。为便于说明, 我们按照 [13, 14] 的方法重现 $r_{(5)(5)}(A)$ 的计算:

$$\begin{aligned} u_5 &= A(b_5 A_4 + b_3 A_2 + b_1 I), v_5 = \\ & b_{(4)} A_4 + b_{(2)} A_2 + b_0 I, (-u_{(5)} + v_{(5)}) \\ r_{(5)(5)}(A) &= u_5 + v_{(5)}, \end{aligned} \quad (9)$$

系数 b_j 来自 (5)，而对于 $r_{(13)(\cdot)(13)}(A)$ 可以得到

$$\begin{aligned} u_{13} &= A(A_{(6)(b13)}A_6 + b_{11}A_4 + b_9A_2) + b_7A_6 + b_5A_4 + b_3A_2 + b_1I, \quad v_{13} = \\ &A_{(6)(b12)}A_6 + b_{10}A_4 + b_8A_{(2)} + b_{(6)}A_6 + b_{(4)}A_4 + b_{(2)}A_2 + b_0I, \quad (-u_{(13)} + v_{(13)})r_{(13)} \\ &_{(13)}(A) = u_{13} + v_{(13)}. \end{aligned} \quad (10)$$

这里 $A_2 = A^2$, $A_4 = A^2$ 和 $A_6 = A_2^2 A_4$ 。以这种形式书写，很明显

因此，只需进行三次和六次矩阵乘法及一次逆运算，就可分别得到 10 阶和 26 阶的指数近似值。对角 Pad'e 近似值 $r_{(m,m)}(A)$ 的 $m = 3, 5, 7, 9$ 和 13 实际上是由 MATLAB 中的函数 expm 使用的。

然而，在许多情况下，可以计算高阶超对角近似值 ($k > m$)，计算成本与对角近似值相同，从而获得更高效的方案[25]。下面的例子可以说明这一点。假设

$$r_{1,1}(x) = \frac{1 + \frac{1}{2}x}{1 - \frac{1}{2}x}$$

可以用一个逆值计算。然而， $r_{(2)(\cdot)(1)}(x)$ 分解如下

$$r_{2,1}(x) = \frac{1 + \frac{2}{3}x + \frac{1}{6}x^2}{1 - \frac{1}{3}x} = \alpha - \frac{1}{2}x + \frac{\beta + \gamma x}{1 - \frac{1}{3}x} =: p_{(0)}(x) + \frac{p_{(1)}(x)}{p_{(2)}(x)}$$

计算成本相同，但阶数更高，对所有相关公差的误差也更小。由于 $r_{2,1}(0) = 1$ ，因此 $\alpha + \beta = 1$ ，有一个自由参数可供选择。一般来说，如果将多项式的自由项放入有理函数中，即 $\alpha = 0$ (或 $p_{(0)}(0) = 0$)，舍入误差就会降低。因此，在所有情况下，我们都将考虑 $p_{(0)}(0) = 0$ 。

我们的目标是计算中、小规范矩阵的指数。因此，使用超对角 Pad'e 近似值是一个有效的选择。同样，不难看出，我们可以将 $r_{(2)(m,m)}(x)$ 写为

$$r_{2m,m}(x) = p_{(0)}(x) + \frac{p_{(1)}(x)}{p_{(2)}(x)},$$

$p_{(i)}(x)$, $i = 0, 1, 2$ 是度数为 m 的多项式，可以用 $m-1$ 积和一个逆来计算。然后，与前面的 $r_{(2)(\cdot)(2)}(x)$ 、 $r_{(3)(\cdot)(3)}(x)$ 和 $r_{(5)(\cdot)(5)}(x)$ 分别为 1、2 和 3 次乘积和一次逆运算) 计算成本相同，就可以计算出 $r_{(4)(\cdot)(2)}(x)$ 、 $r_{(6)(\cdot)(3)}(x)$ 和 $r_{(8)(\cdot)(4)}(x)$ ，它们的精度更高

在所有实际意义上的公差情况下，都比之前成本相同的方法更经济。

这个过程可以扩展到 $r_{(3)(m,)(2)(m)}(x)$ 的情况。

$$r_{3 \text{ 米}, 2 \text{ 米}}(x) = p_0 \frac{(x)}{0} + \frac{(p_{(1)}(x))}{p_{(2)}(x)} + \frac{(p_{(3)}(x))}{p_{(4)}(x)},$$

$p_{(i)}(x)$ 多项式的度数为 m ，那么 $r_{(3)(m,)(2)(m)}(x)$ 可以用 $m-1$ 积和两个反比例来计算。如果 m 是奇数，那么 $p_{(2)}(x)$ 和 $p_{(4)}(x)$ 就是具有复系数的奇数多项式。为了避免这种情况，我们考虑偶数 m ，这样 $p_{(2)}(x)$ 和 $p_{(4)}(x)$ 就包含一对复共轭根，它们可以以不同的方式分布（在舍入精度上也可能不同）。此外，我们还有

$$P_0(0) + \frac{P_{(1)}(0)}{p_{(2)}(0)} + \frac{P_{(3)}(0)}{p_{(4)}(0)} = 1,$$

我们将选择 $p_0(0)=0$ 和 $p_{(1)}(0)=p_{(3)}(0)=(1)$ 。

最后，我们还考虑了 $r_{(4)(m,)(3)(m)}(x)$ ， m 为偶数的情况。

$$r_{4 \text{ 米}, 3 \text{ 米}}(x) = p_0 \frac{(x)}{x} + \frac{(p_{(1)}(x))}{p_{(2)}(x)} + \frac{(p_{(3)}(x))}{p_{(4)}(x)} + \frac{(p_{(5)}(x))}{p_{(6)}(x)},$$

$p_{(i)}(x)$ 多项式的度数为 m ，那么 $r_{(4)(m,)(3)(m)}(x)$ 可以用 $m-1$ 积和三个反向来计算。我们已经分析了 $m=4$ 的情况、

即 $r_{(16)(,)(12)}(x)$ ，比 $r_{(13)(,)(13)}(x)$ 阶数更高，结果更精确，而成本略低。 $p_{(2)(k)}(x)$, $k=1, 2, 3$ 的根的分布有很大的自由度（15 种不同的组合）。此外，我们还有

$$p_0(0) + \frac{p_{(1)}(0)}{p_{(2)}(0)} + \frac{p_{(3)}(0)}{p_{(4)}(0)} + \frac{p_{(5)}(0)}{p_{(6)}(0)} = 1. \text{ 该方案的主要问题是，对于所有的}$$

我们已经探讨过，该方案有很大的正反两方面影响。

系数，会带来相对较大的舍入误差，因此，该方案仍在研究之中。

我们还探索了其他方案，发现唯一有效的方案是 $r_{(8)(,)(5)}(x)$ 分解为

$$r_{8,5}(x) = p_0 \frac{(x)}{0} + \frac{(p_{(1)}(x))}{p_{(2)}(x)} + \frac{(q_{(1)}(x))}{q_{(2)}(x)},$$

$q_{(i)}(x)$ 和 $p_{(i)}(x)$ 分别是 2 级和 3 级多项式，然后可以用 2 个乘积和 2 个反函数来计算。

我们采用类似的方法来降低某些对角 Pad'e 近似值的成本。我们将 $r_{m,m}(x)$, $m=4, 8$ 分解为两个分数，而对于 $m=6, 12$ ，我们发现将每个分数分解为三个分数更为经济。我们分析了分数间分母根的不同分布，以减少舍入误差。

3.2 泰勒多项式近似值的高效计算

对于 $n=1, 2, 4$, 使用 $k=0, 1, 2$ 乘积, 我们可以对 $t_n(x)$ 进行微不足道的求值, 而对于 $n=8, 12$ 和 18 , 只需分别使用 $k=3, 4$ 和 5 乘积就能精确计算多项式 [4, 5]。

一般来说, 用 k 个乘积可以得到 2^k 度的多项式, 但对于 $k>3$, 由于没有足够的独立参数, 所以无法计算 $t_{2^k}(x)$ 。但是, 用 $k=4, 5$ 乘积可以得到 16 和 24 度的多项式, 它们与泰勒方程相吻合。

扩展到 15 阶和 21 阶, 我们将用 $t^{(16)}(x)$ 和 $t^{(24)}(x)$ 表示、

15

21

分别。这些方法与 [28] 中的 y_{22} 和 y_{23} 方案相对应

作为对 $k=3$ 个产品的说明, 我们展示了以下计算 t_8 的算法:

$$\begin{aligned} A_2 &= A^2, \\ A_4 &= A_2(x_1 A + x_{(2)} A_{(2)}), \\ A_8 &= (x_3 A_2 + A_4)(x_4 I + x_5 A + x_6 A_2 + x_7 A_{(4)}), \quad t_{(8)}(A) = \\ &= y_0 I + y_1 A + y_{(2)} A_2 + A_8, \end{aligned} \quad (11)$$

其中

$$\begin{aligned} x_1 &= x, & x_2 &= \frac{1 + \sqrt{177}}{352} x_3, & x_4 &= \frac{-271 + 29\sqrt{177}}{315x}, \\ x_{(5)} &= \frac{11(1 + \sqrt{177})}{1260x}, & x_{(6)} &= \frac{11(9 + \sqrt{177})}{5040x}, & x_{(7)} &= \frac{89 - \sqrt{177^3}}{5040x^2}, \\ y_0 &= 1, & y_1 &= 1, & y_2 &= \frac{857 - 58\sqrt{177}}{630}, \\ x_3 &= 2/3. \end{aligned}$$

注意, $t_{(7)}(x)$ 至少需要 4 个乘积, 因此 $t_{(8)}(x)$ 可视为奇异多项式。

3.3 按计算成本排序的选定方法

现在, 我们收集已发现的最佳方法 w_a , 并按其

计算成本 k_a 。计算逆的成本为 1_3

3

1

产品。我们给出了这些方案的结构, 并为

系数 $\alpha_i, \beta_i, \dots, i=1, 2, \dots$ 在每种情况下取值不同。

$k=1$ product $t_2(x) = 1 + x + \frac{1}{2}x^2$.

2

$$k=1 \text{ 产品 } r_{(2)(1)}(x) = p_{(0)}(x) + \frac{p_{(1)}(0)(x)}{p_{(2)}(x)}, p_i(x) = \alpha_i + \beta_i x, p_{(0)}(0) = 0, \alpha_{(2)} = 1。$$

$$k=2 \text{ 产品 } t_{(4)}(x) = 1 + x + x^2 \frac{(1)+(1)x+(1)x^2}{2! \quad 3! \quad 4!}。$$

$$k=2 \text{ 产品 } r_{(4)(2)}(x) = p_0(x) + \frac{p_{(1)}(0)(x)}{p_{(2)}(x)}, p_i(x) = \alpha_i + \beta_i x + \gamma_i x^2, \alpha_0 = 0, \alpha_{(2)} = 1。$$

$$k=3 \text{ 产品 } t_{(8)}(x)。$$

$$k=3 \text{ 产品 } r_{(6)(3)}(x) = p_0(x) + \frac{p_{(1)}(0)(x)}{p_{(2)}(x)}, p_i(x) = \alpha_i + \beta_i x + \gamma_i x^2 + \delta_i x^3, \alpha_0 = 0, \alpha_2 = 1。$$

$$k=3 \text{ 产品 } r_{(6)(4)}(x) = p_{(0)}(x) + \frac{p_{(1)}(0)(x)}{p_{(2)}(x)} + \frac{p_{(3)}(0)(x)}{p_{(4)}(x)}, p_i(x) = \alpha_i + \beta_i x + \gamma_i x^2, \text{ 其中 } \alpha_0 = 0 \text{ 和 } \frac{p_{(1)}(0)-p_{(3)}(0)}{p_{(2)}(0)} = 1, \alpha_2 = \alpha_{(4)} = 1。$$

$$k=4 \text{ 个乘积 } t_{(12)}(x) \text{ 和 } t^{([16])}(x)。$$

$$k=4 \text{ 产品 } r_{(8)(4)}(x) = p_{(0)}(x) + \frac{p_{(1)}(0)(x)}{p_{(2)}(x)}, p_i(x) = \alpha_i + \beta_i x + \gamma_i x^2 + \delta_i x^3 + \sigma_i x^4, \alpha_0 = 0, \alpha_2 = 1。$$

$$k=4 \text{ 产品 } r_{(8)(5)}(x) = p_{(0)}(x) + \frac{p_{(1)}(0)(x)}{p_{(2)}(x)} + \frac{q_{(1)}(0)(x)}{q_{(2)}(x)}, p_i(x) = \alpha_i + \beta_i x + \gamma_i x^2 + \delta_i x^3, q_i(x) = \tilde{\alpha}_i + \tilde{\beta}_i x + \tilde{\gamma}_i x^2 + \tilde{\delta}_i x^3 \text{ with } \alpha_0 = 0 \text{ and } \frac{p_{(1)}(0)-q_{(1)}(0)}{p_{(2)}(0)} = \frac{q_{(2)}(0)-p_{(2)}(0)}{q_{(2)}(0)} = 1。$$

$$k=5 \text{ 个产品 } t_{(18)}(x) \text{ 和 } t^{([24])}(x)。$$

$$k=5 \text{ 产品 } r_{(10)(5)}(x) = p_{(0)}(x) + \frac{p_{(1)}(0)(x)}{p_{(2)}(x)}, p_i(x) = \alpha_i + \beta_i x + \gamma_i x^2 + \delta_i x^3 + \sigma_{(i)x^{(4)+\mu_{(i)}x^{(5)}}}, \alpha_0 = 0, \alpha_2 = 1。 \text{ 不使用这种方法是因为对于所有考虑的公差值，其 } 0 \text{ 都小于 } (r_{(8)(4)}(x))^2 \text{ 的相应值。}$$

$$k=5 \text{ 产品 } r_{(12)(8)}(x) = p_{(0)}(x) + \frac{p_{(1)}(0)(x)}{p_{(2)}(x)} + \frac{p_{(3)}(0)(x)}{p_{(4)}(x)}, p_i(x) = \alpha_i + \beta_i x + \gamma_i x^2 + \delta_i x^3 + \sigma_i x^4, \text{ 其中 } \alpha_0 = 0 \text{ 和 } \frac{p_{(1)}(0)}{p_{(2)}(0)} = \frac{p_{(3)}(0)}{p_{(4)}(0)} = 1, \alpha_2 = \alpha_4 = 1。 \text{ 不稳定因素不过，一般来说，它也会出现舍入误差，尽管小于与 } r_{(16)(12)}(x) \text{ 相比，它的容差更小。不过，对于较低的公差值，它可以与 } r_{(8)(4)}(x) \text{ 和 } r_{(8)(5)}(x) \text{ 相抗衡。}$$

$$k=7 \text{ 产品 } r_{(16)(12)}(x) = p_{(0)}(x) + \frac{p_{(1)}(0)(x)}{p_{(2)}(x)} + \frac{p_{(3)}(0)(x)}{p_{(4)}(x)} + \frac{p_{(5)}(0)(x)}{p_{(6)}(x)}, p_i(x) = \alpha_i + \beta_i x + \gamma_i x^2 + \delta_i x^3 + \sigma_i x^4。 \text{ 当需要非常高的精确度且必须使用缩放时，这种方案将是首选方法。遗憾的是，在我们尝试的所有自由参数选择中，该方法都存在严重的舍入误差，因此目前正在对该方案进行研究，并没有在算法中使用。}$$

$$k=7 \rightarrow \text{产品 } r_{(13)(\cdot)(13)}(x)。$$

在表 2 和表 3 中，我们收集了每种选定方法 $w_{(a)}$ 的容许误差值 tol 和相应的 $\theta_{(a)}(tol)$ ，以及其他一些方法的容许误差值，以供比较。为简化表述，表中仅显示了常用的容差值，而在实验实施中，我们使用了扩展表 $tol = 10^{-k}$, $k = 0, 1, \dots, 16$ 。在给定容差条件下，计算成本低于最优值的方法会以灰色显示，不会在最终算法中使用。在完善查找表（LUT）时，我们选择排除计算成本低于平方便宜法的方法，以及显示较大舍入误差的方案。例如， $r_{(10)(\cdot)(5)}(x)$ 的 θ_s 小于 $r_{(8)(\cdot)(4)}(x)$ ，但有一个额外的缩放。

需要注意的是，根据不同的应用，方法搜索可以有多种实现方式：

- 如果问题要求避免反演，LUT 可以只包含 Tailor 方法。
- 如果只考虑高公差，则可以禁用高阶方法，如 $r_{(13)(\cdot)(13)}(x)$ ，从而优先使用廉价的低阶方法和额外的缩放-平方。
- 另一方面，如果由于容差要求较低而希望避免平方法，则可以对平方法进行额外的惩罚，例如，人为地增加其成本或将其 θ 乘以小于 1 的系数。
- 采用了其他标准，例如 $(\frac{\theta}{k_{\alpha}})$ 。

自适应方案选择算法 考虑到上述因素，我们的实验实施采用以下算法，在提供矩阵、公差值和 LUT（按公差和方法成本升序排列）的情况下选择方法：

1. 计算 $m = \log_{10}(tol)l$ ，其中表示四舍五入操作。
2. 选择与 $10^m < tol$ 条件相对应的 θ_{α} 栏。
3. 对于每个 θ_{α} 计算相应的缩放 $s_{\alpha} = \max(0, \log \frac{\|A\|_2}{\theta_{\alpha}(10^m)})$ 其中 $\lceil \cdot \rceil$ 表示四舍五入操作。

4. 对于成本为 $k(a_j)$ 的每种方法 $w(a_j)$ (在上面的列表中提供) , 计算总成本 $k_a + 1.1 \cdot s_a$ 。为了打破平局, 需要缩放的方法将受到 s_a 乘以系数 1.1 的惩罚。
5. 选择总成本最低的方法。

4 李群中的矩阵指数

给定 $J \in GL(N, \mathbb{R})$, $GL(N)$ 是所有 $N \times N$ 个非奇异矩阵的群, 我们可以将 J 型交群或二次李群定义为集合

$$o_J(n) = \{x \in gl(n) : X^T J X = J\},$$

(X^T 代表共轭转置) , 二次李群的李代数为

$$o_J(N) = \{A \in gl(N) : A^T J + J A = 0_N\},$$

其中, $gl(N)$ 是所有 $N \times N$ 矩阵的代数, 0_N 是维数为 N 的空马利克斯。

众所周知, 如果 $A \in o_J(N)$, 那么 $e^A \in O_J(N)$ 。如果 J 是同位矩阵, 当 A 是偏赫米特 (偏斜对称) 矩阵且指数矩阵是单元 (正交) 矩阵时, 这一点得到满足; 而如果 J 是交映体矩阵, 则 e^A 属于交映体群, 等等。

大多数李群积分器的阶数相对较低, 通常能提供中等精度 (例如 (2)) , 但对于许多问题, 在计算指数时必须保留李群结构[9, 10, 19]。在这种情况下, 在本文所考虑的方案中, 我们应该使用对角 Padé 近似值, 这是唯一能完全保留这种结构的方案。注意, 由于 $A^T J = -JA$, 那么

$$A^{T^k} J = (-1)^{(k)} J A^{(k)},$$

因此, 如果 k 为奇数, 则 A^k 属于代数, 其指数属于李群。任何奇函数 $f(A)$ 也有类似的性质, 即如果 $f(-A) = -f(A)$, 其中 $f(x)$ 是原点的解析函数, 那么 $f(A)$ 属于李代数, $e^{f(A)}$ 属于李群。对角线 Padé 近似值是唯一对称的 Padé 方法, 因此

$$r_{m,m}(-x) = (r_{m,m}(x))^{-1},$$

表 2：超对角线方法的后向误差比较。星号（*）代表没有成本估算的方法。为完整起见，行和列均为灰色。

费用	方法	宽容						
		2^{-11}	10^{-4}	2^{-24}	10^{-8}	10^{-12}	2^{-53}	10^{-16}
1	$t_{2,2}$	5.31e 2-	2.43e 2-	5.98e 4-	2.45e 4-	2.45e 6-	2.58e 8-	2.45e 8-
$1\frac{1}{3}$	1	3.18e 1-	1.90e 1-	1.62e 2-	8.96e 3-	4.16e 4-	2.00e 5-	1.93e 5-
2	t_4	4.48e 1-	3.10e 1-	5.12e 2-	3.29e 2-	3.31e 3-	3.40e 4-	3.31e 4-
$2\frac{1}{3}$	$r_{4,2}$	1.66	1.30	3.98e 1-	2.97e 1-	6.48e 2-	1.42e 2-	1.40e 2-
3	t_8	1.59	1.35	5.80e 1-	4.70e 1-	1.54e 1-	4.99e 2-	4.93e 2-
$3\frac{1}{3}$	$R_{6,3}$	3.28	2.81	1.31	1.09	4.01e 1-	1.47e 1-	1.45e 1-
$3\frac{2}{3}$	$R_{6,4}$	4.10	3.57	1.79	1.51	6.12e 1-	2.48e 1-	2.46e 1-
4	t_{12}	2.79	2.50	1.46	1.28	6.24e 1-	3.00e 1-	2.97e 1-
*	t_{15}	3.68	3.38	2.22	2.00	1.14	6.41e 1-	6.37e 1-
4	$(t)^\#$	3.91	3.59	2.35	2.11	1.20	4.92e 1-	4.63e 1-
$4\frac{1}{3}$	\int^{16}	4.95	4.43	2.55	2.22	1.07	5.07e 1-	5.03e 1-
$4\frac{2}{3}$	$R_{8,4}$ $R_{8,5}$	5.83	5.25	3.14	2.76	1.40	7.05e 1-	6.99e 1-
5	t_{18}	4.57	4.26	3.01	2.76	1.75	1.09	1.08
*	$t_{21}^{(t)}$	5.45	5.13	3.82	3.56	2.42	1.62	1.62
5	\int^{24}	5.62	5.29	3.95	3.67	2.50	4.54e 1-	4.21e 1-
$5\frac{1}{3}$	$r_{10,5}$	6.64	6.08	3.95	3.54	2.00	1.11	1.10
$5\frac{2}{3}$	$r_{12,8}$	1.02e1	9.54	6.91	6.37	4.16	2.69	2.68
$7\frac{1}{3}$	$r_{16,12}$	1.55e1	1.48e1	1.18e1	1.12e1	8.27	6.09	6.07
$7\frac{1}{2}$	$r_{13,13}$	1.53e1	1.45e1	1.12e1	1.06e1	7.55	5.37	5.35

或者换句话说

$$r_{(m,m)}(x) = e^{f(0)(x)} \quad \text{与} \quad f(-x) = -f(x),$$

则它们保留了李群结构。

我们的算法有一个选项，可以使用对角 Padé 近似值来近似矩阵指数。在这种情况下，我们将使用 MATLAB 中已经包含的方案，但在前面提到的分数分解之后，我们还将增加一些方案。表 3 列出了这些方案。请注意，我们不会使用 $m = 10, 11$ 的方法，因为 $r_{(12)(12)}(x)$ 的阶数更高，计算成本相同。遗憾的是，分解后的 $r_{(12)(12)}(x)$ 与 $r_{(12)(8)}(x)$ 和 $r_{(16)(12)}(x)$ 存在相同的舍入误差，因此在算法中没有使用。

5 数值性能

在数值实验中，我们展示了所提算法的行为以及组成算法的各个方案。在示例中，算法计算 $\|A\|_1$ ，并使用它和规定的公差来从表 2 或表 3 3.3 小节。中选择最合适的方案，如所述

数值示例 1 考虑一个随机对角占优矩阵 $A = D + R$, $A \in \mathbb{R}^{101 \times 101}$, 这样 $D = \text{diag}(-50, -49, \dots, 50)$ 且 R 的元素从 $[-1; 1]$ 中均匀采样，矩阵被归一化（即 $A := A/A_1$ ）。

那么，对于 $h = 10^m$, $m = -3, -2, \dots, 2$ 和 $\text{tol} = 10^{-k}$, $k = 0, 1, \dots$ 对于每个规范 h, τ 我们计算归一化的相对误差。

$$\frac{\|w_a(hA) - e^{hA}\|_1}{\|A\|_1 \|e^{hA}\|_1} \quad (12)$$

其中，参考值 e^{hA} 是以任意算术方式数值计算得出的误差成本对。然后，我们在图 1 中将这些误差成本对与表 2 中的理论估计值绘制在一起图 2。显示了对表 3 所列对角线方法的同样数值检验

从图 1 中可以看出，表 2 中超过对角线方法的平均误差低于反向误差分析预测的误差范围

(8). 唯一的例外是 $r_{(12)(8)}(x)$ ，对于低公差，应首选 $r_{(13)(13)}(x)$ 。

表 3: 对角线 Pad'e 近似值的后向误差比较。为完整起见, 表中的行和列均为灰色。

费用	方法	公差 10^{-8}						
		2^{-11}	10^{-4}	2^{-24}	10^{-12}	2^{-53}	10^{-16}	
$2^{\frac{1}{2}}$	R2,2	7.63e 1-	5.16e 1-	8.09e 2-	5.18e 2-	5.18e 3-	5.32e 4-	5.18e 4-
$3^{\frac{1}{2}}$	R3,3	1.87	1.45	4.26e 1-	3.16e 1-	6.82e 2-	1.50e 2-	1.47e 2-
$3^{\frac{2}{3}}$	R4,4	3.14	2.60	1.05	8.40e 1-	2.66e 1-	8.54e 2-	8.43e 2-
$4^{\frac{1}{2}}$	R5,5	4.46	3.85	1.88	1.58	6.31e 1-	2.54e 1-	2.51e 1-
5_3	r6,6	5.81	5.15	2.85	2.47	1.15	5.41e 1-	5.37e 1-
$5^{\frac{1}{3}}$	R7,7	7.16	6.47	3.93	3.47	1.82	9.50e 1-	9.43e 1-
$5^{\frac{2}{3}}$	R8,8	8.53	7.80	5.06	4.55	2.59	1.47	1.46
$6^{\frac{1}{2}}$	r9,9	9.89	9.15	6.25	5.69	3.46	2.10	2.09
7	r10,10	1.13e1	1.05e1	7.47	6.86	4.40	2.81	2.80
7	r11,11	1.26e1	1.18e1	8.71	8.07	5.41	3.60	3.59
7_3	r12,12	1.40e1	1.32e1	9.97	9.31	6.46	4.46	4.44
$7^{\frac{1}{2}}$	r13,13	1.53e1	1.45e1	1.12e1	1.06e1	7.55	5.37	5.35
$8^{\frac{1}{2}}$	r14,14	1.67e1	1.59e1	1.25e1	1.18	8.67	6.33	6.31
$8^{\frac{2}{3}}$	r15,15	1.80e1	1.72e1	1.38e1	1.31e1	9.83	7.34	7.31
$8^{\frac{1}{4}}$	r16,16	1.94e1	1.86e1	1.51e1	1.44e1	1.10e1	8.37	8.35
9	r17,17	2.08e1	1.99e1	1.64e1	1.57e1	1.22e1	9.44	9.41
9	r18,18	2.21e1	2.13e1	1.77e1	1.70e1	1.34e1	1.05e1	1.05e1

为便于比较，以垂直虚线的形式列出了使用当前数值软件的方案所得到的结果，这些结果与不变的舍入误差近似值相对应。误差线上方标注了算法使用的具体方案。更具体地说，如果我们取 $m = -1$ ，即 $\|A\|_1 \leq 0.1$ ，则将使用图 1 所示六种方法中的一种，而在 MATLAB 或 Julia 中将使用成本更高的 $r_{5) (15)}$ 。

数值示例 2：交映矩阵 假设 J 是交映矩阵， B 是对称矩阵，则哈密顿矩阵 $A = JB$ 属于交映代数，且

$$e^{A^T} J e^A = J, \quad J = \begin{pmatrix} 0 & I \\ -I & 0 \end{pmatrix}.$$

对角 Padé 近似值保留了这一特性，我们可以分析表 3 中方法的精度与计算成本，并检查交映特性是否得到保留。

为了说明交映特性的保留，我们设定了三个公差值 (10^{-4} , 10^{-8} , 10^{-16}) 来计算 $w_{(a)}(A)$ 和两类矩阵。作为一个简单的例子，我们使用以下带有对角线块的矩阵：

$$A = \begin{pmatrix} 0 & D \\ -D & 0 \end{pmatrix}, \quad D = \text{diag}(-26, -25, \dots, 26) \in \mathbb{R}^{(53) \times 53}. \quad (13)$$

我们以哈密顿矩阵 A 为例，其元素取样如下

来自 $[-1; 1]$ ：

$$A = \begin{pmatrix} F & H \\ G & -F^T \end{pmatrix}, \quad f, g, h \in \mathbb{R}^{53 \times 53}, \quad (14)$$

其中 G 和 H 是对称的。与之前一样，首先对 A 进行归一化处理，然后使用建议的算法计算与 J

$$\left\| \frac{w_{(a)}(A)^T J w_{(a)}(A) - J C_{(1)}}{\|J\|_1} \right\| = \left\| \begin{pmatrix} A \\ J \end{pmatrix}^T \begin{pmatrix} A \\ J \end{pmatrix} - J \right\|_1, \quad (15)$$

为一组准则。然后，对于每个公差，我们绘制出误差 (15) 与矩阵规范的对比图，表明成本和相应的方法。可以看出，对于每个规定的公差，算法都能保持与参考值 $r_{(13) (13)}$ 相同的误差，同时所需的计算量较少。从图 2 和图 3 可以看出，对于所有考虑的公差值，计算量都有所节省，同时交映特性与 $r_{(13) (13)}$ 相同。

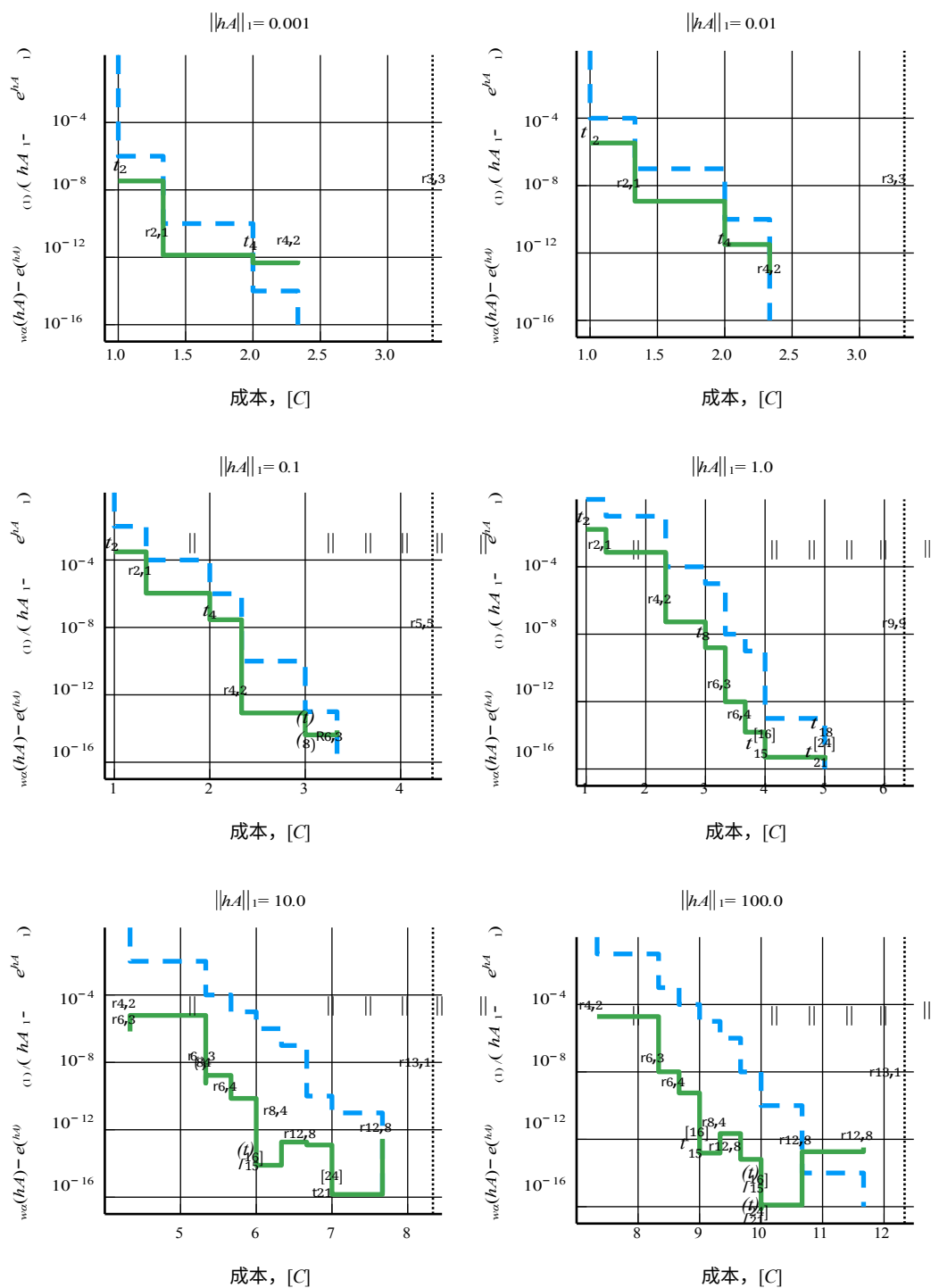


图 1: 公差 (虚线) 和实际平均成本与超对角线方法总计算成本的对比图。细的垂直虚线表示当前软件实现 [14] 时的成本。请注意垂直虚线与实线之间的距离节约了成本。

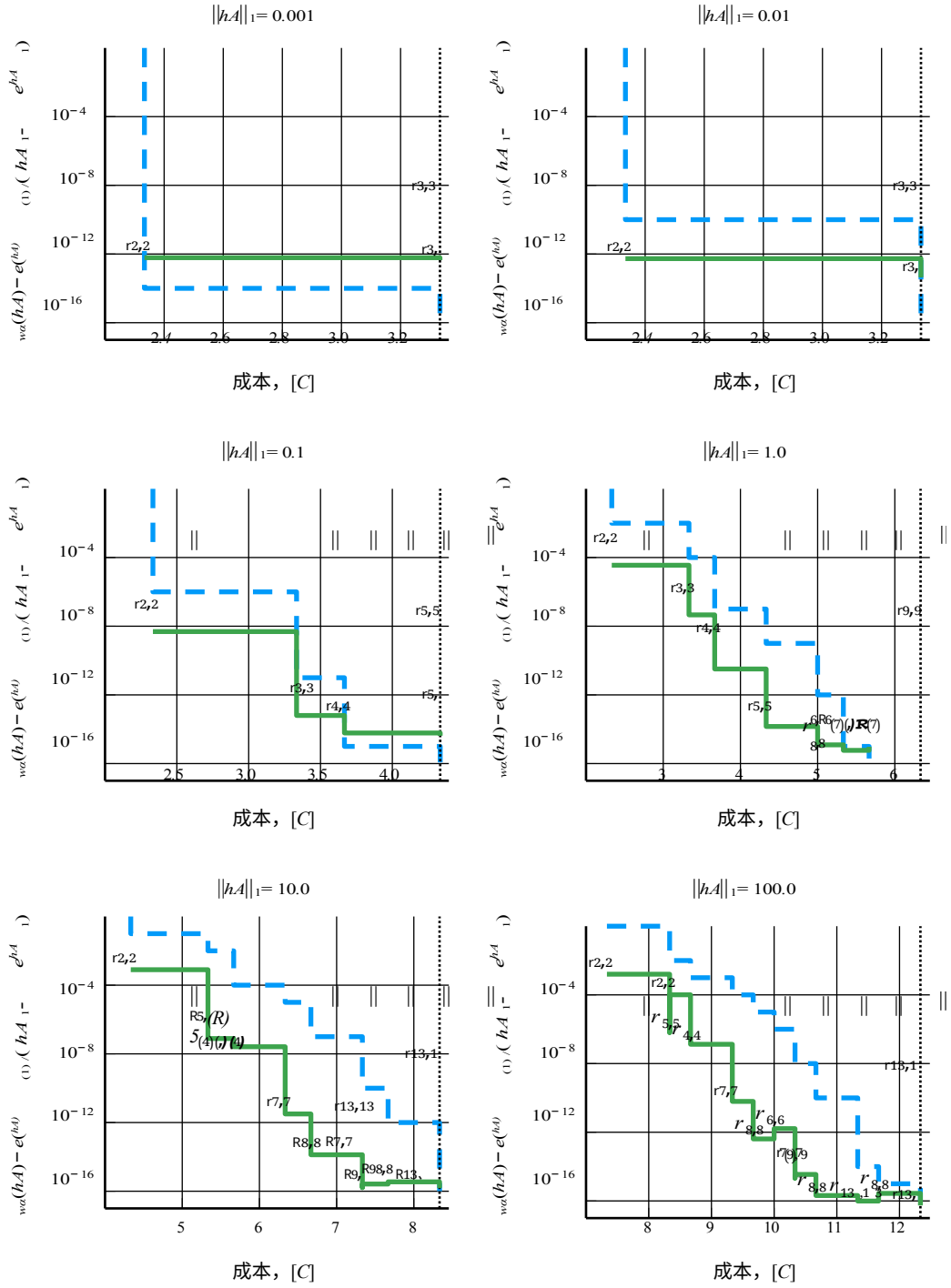


图 2：公差（虚线）和实际平均成本与对角线方法总计算成本的对比图。细的垂直虚线表示当前软件实现 [14] 时的成本。请注意，垂直虚线与实线之间的距离表示节约成本。

我们将较大规范的差异归因于我们的实验代码和 Julia 的 LinearAlgebra（线性代数）代码中更简单的实现 $r_{(13)}(x)$ 。例如，使用矩阵再平衡和直接调用某些 BLAS 函数。高度优化的实现可以在必要时切换到内置方法。

数值示例 3：单元式 与折中情况类似，我们考虑单元式情况。为此，我们使用两个矩阵：前一个 (13) 以及 $A = iB + C$, $A \in \mathbb{C}^{101 \times 101}$ ，其中 B 是对称矩阵， C 是倾斜对称矩阵。我们测量相对误差

$$\frac{\|w_{(a)}(A)^T w_{(a)}(A) - I\|_1}{\|I\|_1} = \|r_{m,m}^T(A) r_{m,m}(A) - I\|_1, \quad (16)$$

结果如图 4 所示。与前面的例子一样，所关注的属性得到了保留，计算成本也有所降低。

对于这个问题，如果求解的精度也要求四舍五入，那么计算成本就会降低。如果不这样做，量身定制的多项式近似方法可能是非常高效的方案 [2]。

6 结束语

我们提出了一种新算法，它能以比标准方法更低的计算成本逼近不同公差的矩阵指数函数，适用于各种矩阵。该算法可以很容易地调整为只使用泰勒方法（即矩阵-矩阵乘积，不含反演），以解决乘积比反演便宜得多的问题，或者只使用对角 Padé 近似值来保留李群结构。

在 $\|A(k)\|^{1/k} \ll \|A\|$, $k > 1$ 的某些情况下，可以将误差扩展到

通过分析，可以使用更便宜的方案或减少缩放次数，从而降低成本。由于我们的算法计算的是 A 的几个幂，这一点很容易检查，如果用户对这一类问题感兴趣，也可以很容易地按照 [14, 4, 5] 进行调整。我们建议只将其作为一个选项，否则会增加所有问题的额外成本，而在很多情况下这是不必要的。

本文的分析也可用于其他矩阵函数的近似 [15]，一般来说，对角 Padé 近似值不再对称，即 $q_{(m,m)}(x) = p_{(m,m)}(x)$ 。显然，如果矩阵具有某些特殊结构，如矩阵的扰动，其指数计算既简单又便宜，则应使用其他技术 [3]。

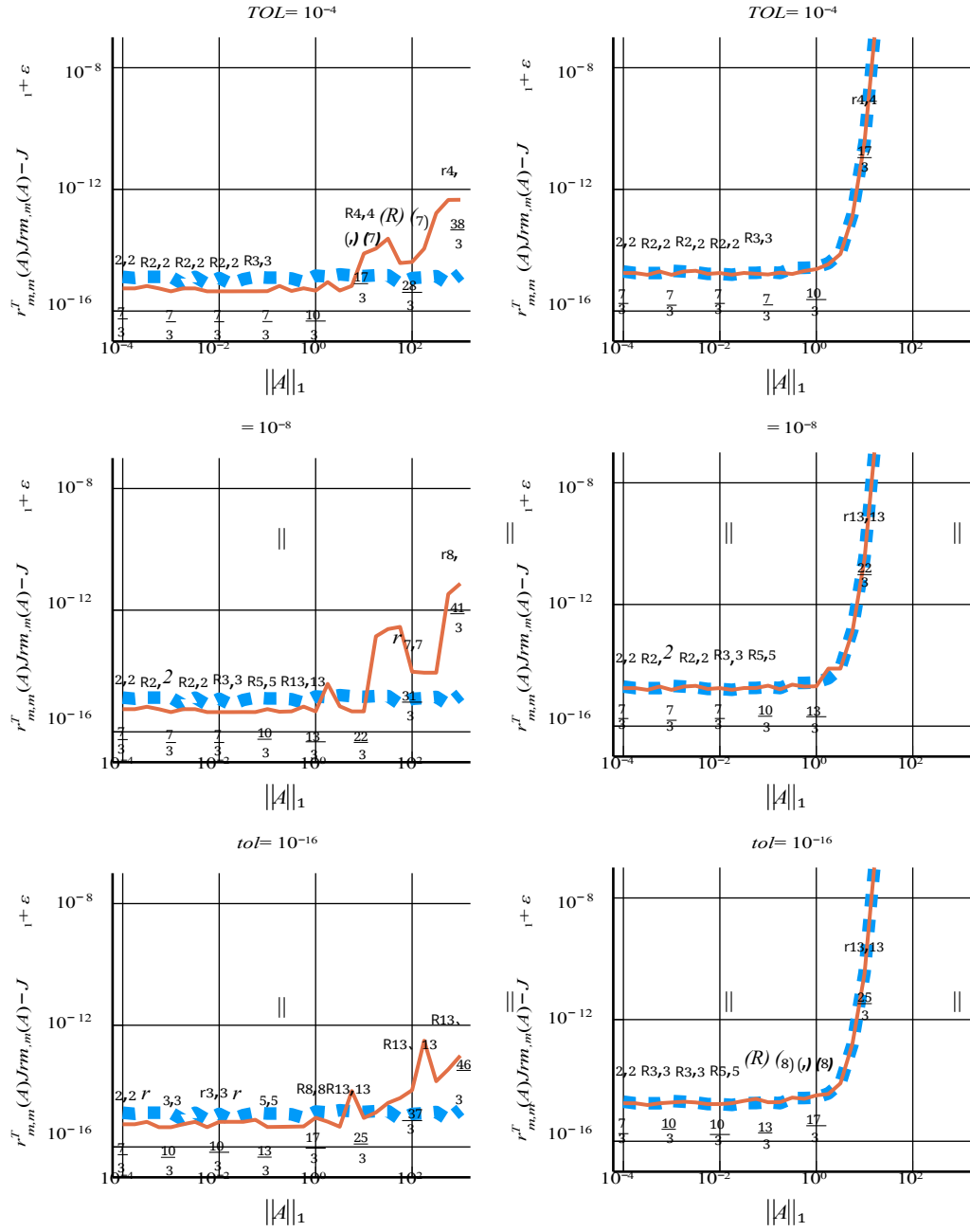


图 3：只使用对角线 Padé 方法计算矩阵 (13) (左) 和 (14) (右) 的交换特性保持情况比较。参考（虚线）是 Julia 1.10.2 中的 LinearAlgebra.exp 函数。为避免奇异性，添加了机器 ε_0 。

我们已经把几个 Padé 近似分解成可以独立计算的低度分数，使它们适合并行化。我们将在 [7]。之后对矩阵指数的并行计算进行深入研究

致谢

这项工作得到了西班牙科技创新部的资助。

通过项目 PID2022-136585NB-C21, MCIN/AEI/10.13039/501100011033/FEDER、欧盟以及巴伦西亚大区（西班牙）通过 CIAICO/2021/180 项目提供了支持。

参考资料

- [1] A.H. Al-Mohy and N.J. Higham, A new scaling and squaring algorithm for the matrix exponential, SIAM J. Matrix Anal.应用, **31**, (2009 年), 第 970-989 页。
- [2] P.Bader, S. Blanes, F. Casas, M. Seydaoglu, An efficient algorithm to compute the exponential of skew-Hermitian matrices for the time integration of the Schrödinger equation, Math. Computing.Sim.Sim., **194**, (2022), pp.383-400.
- [3] P.Bader, S. Blanes, and M. Şulu, The scaling, splitting and squaring method for the exponential of perturbed matrices, SIAM J. Matrix Anal.应用, **36**, (2015 年), 第 594-614 页。
- [4] P.Bader, S. Blanes, and F. Casas, An improved algorithm to compute the exponential of a matrix, arXiv:1710.10989 [math.NA], (2017), preprint.
- [5] P.P.Bader, S.Blanes 和 F.Casas, 用优化的泰勒多项式近似计算矩阵指数, 《数学》, **7** (2019), 1174。
- [6] J.Bezanson, A. Edelman, S. Karpinski, & V. Shah, Julia: A fresh approach to numerical computing, SIAM Review.**59**, 65-98 (2017).
- [7] S.Blanes, 矩阵函数的并行计算及其对向量的作用 arXiv:2210.03714 [math.NA], (2022), preprint.
- [8] S.S. Blanes, F. Casas, J.A. Oteo, and J. Ros, The Magnus expansion and some of its applications.物理报告, **470** (2009), 第 151-238 页。

- [9] E.Celledoni and A. Iserles, Approximating the exponential from a Lie algebra to a Lie group, *Math.Comput.*, **69** (2000), pp.
- [10] E.Celledoni and A. Iserles, Methods for the approximation of the matrix exponential in a Lie-algebraic setting, *IMA J. Numer.Anal.*, **21** (2001), pp.463-488.
- [11] W.J. Cody, G. Meinardus, and R. S. Varga, Chebyshev rational approximation to e^{-x} in $[0, +\infty)$ and applications to heat-conduction problems, *J.Approximation Theory*, **2** (1969), pp.
- [12] P.E. Crouch and R. Grossman, Numerical integration of ordinary differential equations on manifolds, *J. Nonlinear Sci.* **3** (1993), 1-33.
- [13] N.J. Higham, The scaling and squaring method for the matrix exponential revisited, *SIAM J. Matrix Anal.Appl.*, **26**, (2005 年), 第 1179-1193 页。
- [14] N.J. Higham, The scaling and squaring method for the matrix exponential revisited, *SIAM Review* **51** (2009), pp.
- [15] N.J. Higham, *Functions of Matrices: Theory and Computation*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2008).
- [16] N.J. Higham 和 A.H. Al-Mohy, 计算矩阵函数, *Acta Numerica*, **51**, (2010 年), 第 159-208 页。
- [17] M.Hochbruck 和 A. Ostermann, 《指数积分器》, *Acta Numerica*, **19**, (2010 年), 第 209-286 页。
- [18] A.Iserles, Composite exponential approximations. *Mathematics Of Computation*.**38**, 99-112 (1982), <http://dx.doi.org/10.1090/S0025-5718-1982-0637289-7>
- [19] A.Iserles, H. Z. Munthe-Kaas, S.P. Nørsett and A. Zanna, Lie-group methods, *Acta Numerica*, **9**, (2000), pp.
- [20] M.Lezcano-Casado, 流形上的自适应方法和动量方法, *arXiv:2010.04617*, (2020).
- [21] C.B. Moler and C.F. Van Loan, Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later, *SIAM Review*, **45** (2003), pp.3-49.
- [22] H.Munthe-Kaas, Runge-Kutta methods on Lie groups, *BIT* **38** (1998), 92-111.

- [23] M.S. Paterson 和 L.J. Stockmeyer, 论求解多项式所需的非标量多项式次数, SIAM J. Comput., **2** (1973), pp.60-66.
- [24] C.Rackauckas, Q. Nie, DifferentialEquations.jl - A Performant and Feature-Rich Ecosystem for Solving Differential Equations in Julia.The Journal Of Open Research Software.**5** (2017), <https://doi.org/10.5334/jors.151>
- [25] J.Sastre, Efficient mixed rational and polynomial approximations of matrix functions, Appl.计算》, **218** (2012), 第 11938-11946 页。
- [26] J.Sastre, Efficient evaluation of matrix polynomials, Linear Algebra Appl., **539**, (2018), pp.
- [27] J.Sastre, J. Ib̃ez, P. Ruiz, and E. Defez, New scaling-squaring Taylor algorithms for computing the matrix exponential, SIAM J. Sci. Comp., **37** (2015), pp.
- [28] J.Sastre, J. Ib̃ez, and E. Defez, Boosting the computation of the matrix exponential, Appl. Math.Comput., **340** (2019), pp.
- [29] C.Ta, D. Wang, Q. Nie, An integration factor method for stochastic and stiff reaction-diffusion systems.计算物理学报**295**
pp.505-522 (2015,8), <http://dx.doi.org/10.1016/j.jcp.2015.04.028>
- [30] L.N. Trefethen, J.A.C. Weideman, and T. Schmelzer, Talbot quadratures and rational approximations, BIT, **46** (2006), pp.