标量与向量外推综述

K. JBILOU*

摘要。标量外推和收敛加速方法是数值分析中的核心工具，可提高迭代算法的效率，并加速慢收敛级数的求和。这些方法通过构造变换后的序列，使其更快地收敛到同一极限，同时不改变底层迭代过程，从而降低计算成本并提高数值精度。从历史上看，此类技术可追溯至花拉子米的经典代数方法和牛顿早期的级数加速技术；而系统化方法则于20世纪逐渐形成，包括艾特肯的 Δ^2 过程和理查森外推。后续发展包括 Shanks变换和 Wynn的 ε-算法，这些方法提供了通用框架，能够消除多个占主导地位的误差分量，并与 Pad´é逼近及生成函数的有理逼近密切相关。本文全面综述经典标量外推方法，包括理查森外推、艾特肯的 Δ^2 过程、Shanks变换、Wynn的ε-算法，以及 ρ和 θ算法。我们探讨这些方法的理论基础、渐近误差模型、收敛性质、数值稳定性及实际实现中的注意事项。本文第二部分专门介绍向量外推方法：基于多项式的方法，以及面向向量序列的ε-算法推广。此外，我们还重点介绍其在迭代求解器、Krylov子空间方法和大规模计算模拟中的应用等现代进展。本综述旨在为标量与向量外推技术提供统一的视角，贯通其历史渊源、理论见解和当代计算应用。

arXiv:2602.01301v1 [math.NA] 1 Feb 2026

关键词：标量外推、收敛加速、序列变换、艾特肯的 Δ^2过程、理查森外推、Shanks变换、 ε算法、Pad´é 近似式、向量与矩阵外推。

1 外推方法的历史背景 标量外推方法也称为序列变换或收敛加速技术，是数值分析中的基础工具。其主要目标是加快序列 $\{x_n\}$ 向极限 x^*的收敛，或为缓慢收敛或发散的级数赋予有限值。这些方法广泛应用于微分方程的数值求解、线性与非线性迭代求解器、级数求和以及逼近论 [18, 59]。

迭代近似的概念由来已久。9世纪，花拉子米 [2] 提出了求解代数方程的系统方法。他的算法方法可视为现代迭代方法的先驱，强调通过逐步计算得到近似解。后来，牛顿（1642–1727）和詹姆斯·格雷戈里（1638–1675）等数学家发展了级数展开、插值以及求方程根的迭代公式，这些方法都隐含地依赖于加快收敛以求得解。

这些早期方法主要由实际需求推动：工程师、天文学家和物理学家需要在无法使用现代计算机的情况下，准确近似求根、计算积分和级数求和。挑战在于减少计算工作量，同时达到足够的精度，为序列变换的正式研究奠定基础。

收敛加速的形式化研究始于20世纪：艾特肯的 Δ^2 方法（1926）。艾特肯 [1] 提出了一个简单公式，用于加速线性收敛序列。通过消去误差的首项，他的方法显著减少了达到所需精度所需的迭代次数。该方法尤其适用于误差近似表现为单个指数项的序列。

1911年，理查森 [50] 提出了一种系统方法，适用于依赖参数 h 的序列，其中 h 通常表示数值微分或积分中的步长。通过假设渐近展开式具有 $x(h) = x^* + c_1 h + c_2 h^2 + \dots$ 的形式，理查森的方法构造不同 h 值下序列项的线性组合，以消去低阶误差。这一思想是高精度数值方法的基础，至今仍广泛应用于现代计算数学。艾特肯和理查森都致力于解决减少计算工作量这一实际问题：通过智能预测序列的极限，而不必进行大量迭代。

新的变换和 Wynn 的 ε-算法于 20 世纪中叶出现。20 世纪 50 年代，人们开发出更复杂的技术来处理具有多个误差分量的序列。1955 年，Shanks [57] 将艾特肯方法推广到误差中含有多个指数项的序列。该变换可以用行列式形式表示，理论上十分优雅，但计算成本

*滨海奥帕尔大学，LMPA，F. Buisson 街 50 号，62228 Calais-Cedex，法国。

高昂。Shanks 表明，他的变换能够显著提升各种数值问题中的收敛速度。

1956 年，Wynn [63] 提出了一种称为 ε-算法的递归算法，可高效实现 Shanks变换。 ε-算法避免了直接计算行列式，因此具有数值稳定性，适用于实际应用。该算法可以迭代消除多个误差项，从而加速迭代求解器、级数求和及其他数值计算中相关序列的收敛。更多详情请参阅 [17, 14, 15, 18, 62, 64, 65]。

序列变换理论中最深刻的见解之一，是序列变换与 Pad´é 逼近之间的联系。Shanks变换的行列式形式以及 Wynn 的递归公式，可以理解为对序列 [5, 15, 16] 的生成函数构造有理逼近。这种联系解释了为何这些变换能够有效地对收敛缓慢甚至发散的级数求和，尤其是在级数表示函数在奇点附近的展开时。

标量外推方法的历史演变，既反映了数值分析的实际需求，也体现了其数学精深程度：

- 早期方法侧重于简单的迭代改进和实际计算。
- 20世纪的方法将消除主导误差项的做法形式化，并建立了系统的收敛加速程序。
- 现代发展将这些技术拓展至高维向量序列和矩阵序列，同时兼顾效率与数值稳定性。

了解这一演变，有助于深入认识经典方法如何启发现代算法，以及序列变换为何仍是数值分析中的基础工具。许多数值过程会生成缓慢收敛至目标极限的序列。这类序列常见于不动点迭代、数值求积、微分方程的离散化以及级数求和中。提高收敛速率对于降低计算成本、提升数值效率至关重要。

标量外推方法通过对生成的序列进行后处理，以非侵入式的方式实现收敛加速。其简洁性和有效性使其成为数值分析中经典而经久不衰的工具。

虽然上文讨论的经典方法主要针对标量序列，但许多现代计算问题涉及向量序列或矩阵序列，例如线性和非线性系统的迭代求解器所产生的序列，或离散化偏微分方程所产生的序列。标量外推技术，包括艾特肯的 Δ^2过程、理查森外推和 Wynn 的 ε-算法，为其向量对应方法奠定了理论基础。向量算法和拓扑 ε-算法、最小多项式外推（MPE）以及降秩外推（RRE）等方法，将序列变换的原理扩展到更高维的情形 [18, 42, 59]。这些向量和矩阵外推技术旨在通过消除主要误差分量，以类似于标量方法的方式加速向量序列或矩阵序列的收敛，同时考虑线性相关性和多维相关性。在大规模模拟和迭代求解器中，这些技术尤其有价值，因为每次迭代的计算成本都很高，而快速收敛能够显著降低总体计算成本。因此，理解标量外推至关重要，因为它不仅提供了概念框架，也为这些高维推广提供了分析工具。

本文其余部分安排如下。第 2 节回顾几种著名的标量外推方法，包括理查森外推、艾特肯的 Δ^2 过程、Shanks变换、Wynn 的标量 ε-算法，以及 E-、 ρ- 和 θ-算法。第 3 节介绍经典的向量外推技术。具体而言，我们将介绍降秩外推（RRE）、最小多项式外推（MPE）、修正最小多项式外推（MMPE）、向量和拓扑 ε-算法（VEA 和 TEA），以及 Anderson加速（AA）。此外，我们还将介绍这些向量外推方法在线性问题和非线性问题中的一些应用。最后，第 4 节通过若干数值示例展示这些方法的性能。

2 标量外推方法

2.1 理查森外推

在数值分析中，理查森外推 [50, 51] 是一类用于加速序列或迭代方法收敛的经典技术。收敛加速是指一系列旨在提高序列或迭代近似值接近其精确极限或解的速率的方法。

当迭代方法产生一系列逐次近似时，加速技术会对该序列进行变换，以便用更少的迭代次数得到更精确的估计值，从而降低计算成本。理查森外推是一个典型示例：它结合使用不同步长计算得到的多个近似值，以消除主导阶误差项，从而提高最终估计值的精度。

这种方法在计算科学的许多领域中发挥着核心作用。它广泛应用于数值积分、微分方程的数值求解和物理模拟等领域，在这些领域中，计算效率至关重要。对于给定的精度，理查森外推能够减少所需的迭代次数或步长，也有助于缓解有限精度算术或未来量子计算所引起的数值误差。理查森外推的一个特别重要的应用，是构造高精度数值积分方案。例如，龙贝格积分本质上是通过连续应用理查森外推，对梯形公式进行加速得到的；这一过程系统地消除了主导阶的离散化误差，并使结果快速收敛到精确积分值。

假设渐近展开的形式为

$$x(h) = x^* + c_1 h^p + c_2 h^{p+1} + c_3 h^{p+2} + \cdots, \tag{2.1}$$

其中 $p > 0$ 且 $h \to 0$。其关键思想是利用两个不同步长下的近似值，消去主导误差项 $c_1 h^p$。

设 $x(h)$ 和 $x(h/2)$ 分别为步长 h 和 $h/2$ 下计算得到的近似值。利用 (2.1)，可得

$$\begin{aligned} x(h) &= x^* + c_1 h^p + c_2 h^{p+1} + \cdots, \\ x(h/2) &= x^* + c_1 \left(\frac{h}{2}\right)^p + c_2 \left(\frac{h}{2}\right)^{p+1} + \cdots. \end{aligned}$$

通过消去 c_1 求解 x^*，得到理查森外推值：

$$x^* \approx \frac{2^p x(h/2) - x(h)}{2^p - 1}. \tag{2.2}$$

该公式消去 h^p 项，从而得到一个近似值，其首项误差现为 $\mathcal{O}(h^{p+1})$。理查森外推可递归应用，以获得更高阶收敛。设

$$t_0^{(n)} = x(h/2^n),$$

并对 $k \geq 1$ 递归定义：

$$t_k^{(n)} = t_{k-1}^{(n+1)} + \frac{t_{k-1}^{(n+1)} - t_{k-1}^{(n)}}{2^{p+k-1} - 1}. \tag{2.3}$$

则 $t^{(0)}{}_k = s_n$ 以误差 $\mathcal{O}(h^{p+k})$ 近似 x^*。
更一般地，理查森外推产生一个序列 $(t^{(n)}{}_k)$，定义如下

$$t_k^{(n)} = t_{k-1}^{(n)} - \frac{t_{k-1}^{(n+1)} - t_{k-1}^{(n)}}{x_{n+k} - x_n} x_n, \;\; t_0^{(n)} = s_n \; n \geq 0, \, k \geq 1, \tag{2.4}$$

其中 (x_n)、$n \geq 0$ 为给定点。理查森过程在于在点 $x_n, \ldots, x_{n+k}$ 处对序列项 $s_n, \ldots, s_{n+k}$ 进行插值。

一个经典应用是龙贝格积分。设 $I(h)$ 表示步长为 h 的积分的梯形近似值：

$$I = \int_a^b f(x)\, dx.$$

利用 I(h) 与 I(h/2) 进行理查森外推，可得

$$I \approx \frac{4I(h/2) - I(h)}{3},$$

将收敛速度从 $\mathcal{O}(h^2)$ 提高到 $\mathcal{O}(h^4)$。理查森外推是一种稳健且应用广泛的技术，可用于加速带有代数误差项的序列收敛。其应用涵盖数值积分、微分、微分方程求解以及迭代方法。该方法简单、系统，并且在适当条件下能够达到任意高阶精度。

理查森数列 $(t^{(n)}{}_k)$ 的收敛加速结果广为人知 [45]。主定理指出，对于所有收敛到 s 的序列 (s_n)，$\lim_{n\leftarrow\infty} t^{(n)}{}_k = s$ 的一个必要且充分条件是：存在 α 和 β，满足 $\alpha < 1 < \beta$，使得 $\forall n, x_{n+1}/x_n \notin [\alpha, \beta]$。

2.2 艾特肯的 Δ^2 过程

2.2.1 经典的 Δ^2 算法 艾特肯的 Δ^2 过程是一种经典的标量外推技术，旨在加速线性收敛数列的收敛。设 $\{s_n\}$ 是一个收敛到极限 x^*、线性收敛速率为 ρ 的序列，即

$$\lim_{n\to\infty} \frac{|s_{n+1} - s|}{|s_n - s|} = \rho, \quad 0 < \rho < 1. \tag{2.5}$$

定义前向差分：

$$\Delta s_n = s_{n+1} - s_n, \tag{2.6}$$

$$\Delta^2 s_n = s_{n+2} - 2s_{n+1} + s_n. \tag{2.7}$$

艾特肯变换后的序列为

$$t_n = s_n - \frac{(\Delta s_n)^2}{\Delta^2 s_n}, \quad \Delta^2 s_n \neq 0. \tag{2.8}$$

直观而言，该方法将连续序列元素的线性趋势外推，从而更准确地估计极限。

序列变换的核。设 $\mathcal{T}$ 是一个序列变换，它将标量序列 (s_n) 与变换后的序列 $t_n = \mathcal{T}(s_n)$ 相关联，并假设 (s_n) 收敛到极限 s。

定义（核）。变换 $\mathcal{T}$ 的核记为 $\mathcal{K}(\mathcal{T})$，定义为

$$\mathcal{K}(\mathcal{T}) = \{(s_n) : t_n = s \text{ for all } n \text{ for which the transformation is defined}\}.$$

等价地，引入误差序列 $e_n = s_n - s$，核可表征为

$$\mathcal{K}(\mathcal{T}) = \{(e_n) : (\mathcal{T}(e_n) = 0\},$$

也就是说，该集合由所有被此变换消去的误差序列构成。对于艾特肯–Δ^2 变换，其核为

$$\mathcal{K} = \{(s_n) : s_n = s + c\lambda^n, \qquad c \neq 0,\ \lambda \neq 1\}.$$

应　　用艾特肯的 Δ^2 过程，预设序列 $\{s_n\}$ 收敛到极限 s，并且
误差　具有如下形式的渐近表示

$$e_n = s_n - s = c\lambda^n + o(\lambda^n),$$

其中 $c \neq 0$，且收敛因子满足 $0 < |\lambda| < 1$。在此假设下，原序列是线性收敛的。在这一设定中，Δ^2 变换产生一个新序列 (t_n)，它收敛到同一极限 s，但收敛速度得到改善。特别地，若相继误差之比

$$\frac{e_{n+1}}{e_n}$$

趋于极限 λ ，且 $|\lambda| < 1$，则误差展开的主导项会被该变换有效消去。因此，在不存在高阶项的情况下，艾特肯过程有效消除主导线性误差项，将线性收敛转化为二次收敛；关于艾特肯过程的更多细节见 [17] 。

从几何角度看，艾特肯方法可以理解为沿着由连续点确定的直线投影当前序列，以估计其与极限的交点。这一观点解释了该方法为何对连续迭代呈现近线性行为的序列如此有效。

尽管艾特肯的 Δ^2 过程很简单，但在以下情况下可能会出现数值不稳定性

$$\Delta^2 x_n = x_{n+2} - 2x_{n+1} + x_n \approx 0,$$

从而导致较大的舍入误差。补救措施包括:

- 当{x}n 低于容差时，避免加速,
- 结合平滑或平均技术,
- 对于含有多个指数分量的序列，使用 Shanks变换或 ε算法。

2.2.2 迭代 Δ^2 方法 迭代 Δ^2 方法是艾特肯提出的一种经典序列变换，用于加速序列 $\{x_n\}_{n\geq 0}$的收敛，尤其适用于线性收敛的序列。其主要思想是消去渐近误差的首项，从而生成一个更快收敛到极限的序列。通过反复应用基本的 Δ^2 方法，可以提高其效率——

地进行。将第 k 次迭代所得的序列记为 $t^{(n)}{}_k$，其中

$$t_0^{(n)} = s_n, \qquad t_{k+1}^{(n)} = t_k^{(n)} - \frac{(\Delta t_k^{(n)})^2}{\Delta^2 t_k^{(n)}}, \qquad k \geq 0. \tag{2.9}$$

每次迭代都试图消去变换后序列中剩余误差的首项。实践中，通常只需迭代几次，就能显著加速收敛，尤其是对于近似线性收敛的序列。

迭代 Δ^2 方法对于误差可表示为 $e_n \approx C\lambda^n$且满足 $|\lambda| < 1$ 的序列尤其有效，即线性收敛序列。它是一种标量方法，但可以扩展到向量序列，从而得到一种简单的向量外推形式。该方法广泛用于加速非线性方程迭代方法的收敛、级数求和以及数值分析中的序列外推。

迭代 Δ^2 方法是序列变换最简单的形式，可视为更一般的非线性加速方法的前身，例如Wynn的ε算法、 θ算法以及安德森加速。特别地， ε算法可解释为将迭代 Δ^2过程系统地推广到任意阶外推；这一点将在后续各节中看到。

2.3 Shanks变换 经典Shanks变换 [57] 通过消除误差中的主导指数项来加速标量序列 $\{x_n\}$ 的收敛。一般Shanks变换将这一思想加以推广，以同时去除多个主导误差项，从而能够加速具有多个指数分量的序列:

$$s_n = s^* + \sum_{j=1}^{m} c_j \lambda_j^n, \quad |\lambda_1| > |\lambda_2| > \cdots > |\lambda_m|. \tag{2.10}$$

设 $\{s_n\}$ 为一个标量序列。k阶 Shanks 变换记为$e_n^{(k)}$，旨在去除误差的前k个指数分量。当$k = 1$时，它退化为经典的艾特肯 Δ^2 过程:

$$e_n^{(1)} = s_n - \frac{(\Delta s_n)^2}{\Delta^2 x_n}, \quad \Delta s_n = s_{n+1} - x_n. \tag{2.11}$$

高阶变换可以使用 Wynn 的 ε-算法递归定义，也可以通过行列式表达式定义。

k 阶一般 Shanks 变换可表示为两个行列式之比。令 $\Delta^j x_n$表示第 j 阶前向差分:

$$\Delta^0 s_n = s_n, \quad \Delta^j s_n = \Delta^{j-1} s_{n+1} - \Delta^{j-1} s_n, \quad j \geq 1.$$

于是，k阶 Shanks 变换 $e^{(n)}{}_k$ 表示为

$$e_k^{(n)} = e_k(s_n) = \frac{\begin{vmatrix} s_n & s_{n+1} & \cdots & s_{n+k} \\ \Delta s_n & \Delta s_{n+1} & \cdots & \Delta s_{n+k} \\ \vdots & \vdots & \ddots & \vdots \\ \Delta^{k-1} s_n & \Delta^{k-1} s_{n+1} & \cdots & \Delta^{k-1} s_{n+k} \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \cdots & 1 \\ \Delta s_n & \Delta s_{n+1} & \cdots & \Delta s_{n+k} \\ \vdots & \vdots & \ddots & \vdots \\ \Delta^{k-1} s_n & \Delta^{k-1} s_{n+1} & \cdots & \Delta^{k-1} s_{n+k} \end{vmatrix}}. \tag{2.12}$$

分子在第一行包含序列值，后续各行包含前向差分，而分母则将第一行替换为全 1。这种行列式形式尤为优雅，因为它清晰地展示了指数误差模型中前 k 项的消除过程。假设该序列具有如下指数模型

$$s_n = s^* + \sum_{j=1}^{k} c_j \lambda_j^n, \tag{2.13}$$

于是，k阶 Shanks变换恰好消除了全部 k 项，得到

$$e_k^{(n)} = s^*.$$

对于包含多于 k 个指数分量的序列，$e_n^{(k)}$会消除其中占主导地位的 k 项，从而改善收敛，使其趋近于下一个最大的 λ_{k+1} 分量。
行列式形式（2.12）也可以用 Pad´é 逼近式来解释。令 x {的生成函数为

$$f(z) = \sum_{n=0}^{\infty} c_n z^n. \tag{2.14}$$

k阶 Shanks 变换 $e_n^{(k)}$对应于在 z = 1处计算 f(z) 的对角 Pad´é 逼近式 [k/k] 。这种有理逼近的观点解释了该方法为何能够有效加速包含多个指数项或接近奇点的序列。

备注 1. 行列式分母可能变得很小，从而导致舍入误差。递归ε算法实现通常可以避免这一问题。当 k 较大时，直接进行行列式计算代价高昂；通过 ε算法递归求解则更有效。对于指数项数量未知的序列，迭代应用 Shanks 变换可实现稳健的收敛加速。

一般 Shanks 变换是收敛加速的有力工具:

- 行列式形式提供了消除多个指数项的**闭式表达式**。
- 递归实现（ε算法）使该方法适用于数值应用。
- 与 Pad´é 逼近的联系解释了为什么该方法往往在线性外推或艾特肯 Δ^2 失效时仍能奏效。
- 适用于来自微分方程数值解、迭代求解器和级数求和的序列。

还要注意，Shanks变换在数学上等价于构造生成函数 S(z) 的对角 Pad´e 近似。更准确地说，第 k 阶 Shanks 变换对应于

类型为 (k, k) 的 Pad´e 近似。
这等价关系解释了为什么 Shanks变换 对误差可表示为如下形式的指数表示的序列特别有效

$$s_n = s^* + \sum_{j=1}^{k} c_j \lambda_j^n, \quad |\lambda_j| < 1. \tag{2.15}$$

在这种情况下，Shanks变换可以精确消除所有 k 个占主导地位的指数误差项。Shanks 引入变换（2.2），是序列变换理论的一项重大突破。它最显著的特点之一，是与 Pad´e 近似之间的密切联系，这一点由 Shanks 本人所证明。

Let

$$f_n(z) = \sum_{\nu=0}^{n} \gamma_\nu z^\nu, \qquad n \in \mathbb{N}_0, \tag{2.16}$$

表示与函数 f 的（形式）幂级数展开相关联的部分和序列。根据一项经典结果（见 [57]），对这些部分和应用 Shanks变换 可得

$$e_k(f_n(z)) = [k+n/k]_f(z), \qquad k,n \in \mathbb{N}_0, \tag{2.17}$$

其中 $[k+n/k]_f(z)$ 表示与 f 的级数相关联的类型为 (k+n,k) 的 Pad´e 近似。

3 Wynn 的 ε-算法 Wynn 的 ε-算法 [63] 是一种强大的递归方法，可用于加速标量序列的收敛，并对收敛缓慢甚至发散的级数求和。它将 Shanks变换推广为一种递归方法，能够逐步消除序列误差中的多个指数分量，并提供一种实用且数值稳定的实现。

考虑一个收敛到极限s^*的标量序列 $\{x_n\}$：

$$s_n = s^* + \sum_{j=1}^{\infty} c_j \lambda_j^n, \quad |\lambda_1| > |\lambda_2| > \cdots. \tag{3.1}$$

ε算法系统地去除主导指数项以加速收敛。令 $\varepsilon^{(n)}{}_k$ 表示ε表的元素。初始化：

$$\varepsilon_{-1}^{(n)} = 0, \tag{3.2}$$
$$\varepsilon_0^{(n)} = s_n. \tag{3.3}$$

$k \geq 0$ 的递推关系为：

$$\varepsilon_{k+1}^{(n)} = \varepsilon_{k-1}^{(n+1)} + \frac{1}{\varepsilon_k^{(n+1)} - \varepsilon_k^{(n)}}, \quad \varepsilon_k^{(n+1)} \neq \varepsilon_k^{(n)}. \tag{3.4}$$

递推沿三角形ε表进行，其中n为序列的指标，k为外推的阶数。偶数指标元素 $\varepsilon_2^{(n)}{}_k$ 被视为序列极限x^*的近似值，而奇数指标元素是辅助量，不直接使用。

Wynn算法等价于迭代应用Shanks变换。具体而言：

$$\varepsilon_2^{(n)} = s_n^{(1)} = s_n - \frac{(\Delta s_n)^2}{\Delta^2 s_n}, \tag{3.5}$$

这正是艾特肯 Δ^2 过程，或一阶Shanks变换。
高阶 $\varepsilon_2^{(n)}{}_k$ 元素对应于k阶 Shanks 变换。该算法为一般 Shanks 变换的行列式公式提供了数值稳定的递归实现。

一般k阶 Shanks 变换可以用行列式形式表示:

$$\varepsilon_{2k}^{(n)} = \frac{\begin{vmatrix} s_n & s_{n+1} & \cdots & s_{n+k} \\ \Delta s_n & \Delta s_{n+1} & \cdots & \Delta s_{n+k} \\ \vdots & \vdots & \ddots & \vdots \\ \Delta^{k-1} s_n & \Delta^{k-1} s_{n+1} & \cdots & \Delta^{k-1} s_{n+k} \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \cdots & 1 \\ \Delta s_n & \Delta s_{n+1} & \cdots & \Delta s_{n+k} \\ \vdots & \vdots & \ddots & \vdots \\ \Delta^{k-1} s_n & \Delta^{k-1} s_{n+1} & \cdots & \Delta^{k-1} s_{n+k} \end{vmatrix}}. \tag{3.6}$$

ε算法递归计算这些行列式形式，避免直接计算行列式。这表明，ε算法是一般 Shanks 变换的实际实现。

请注意， $\varepsilon_2^{(n)}{}_k$ 会消去前 k 个主导指数项。对于长度为 k 的有限指数和序列， $\varepsilon_2^{(n)}{}_k$ 项恰好等于极限 s^*。

Wynn 的 ε算法为序列加速提供了强大且数值稳定的框架。它将 Shanks 变换推广到了更高阶。该算法采用递归实现，无需计算行列式。它可以加速序列、级数部分和以及迭代不动点方法。它还与 Pad´é 近似有着天然联系，这解释了它为何能有效加速具有指数型或近指数型误差分量的序列收敛。有关ε算法收敛的更多详情，请参阅 [14, 63, 64, 65]。

3.1 与 Pad´é 近似的联系 Wynn 的ε算法最深刻的理论诠释之一，是它与 Pad´é 近似的密切联系。这种联系提供了一个统一框架，将序列变换、有理逼近和收敛加速联系起来。Pad´é 近似。考虑与形式幂级数之和相对应的函数

$$f(z) = \sum_{k=0}^{\infty} a_k z^k \tag{3.7}$$

(m,n) Pad´e 逼近式 $[m/n]_f(z)$ 是有理函数

$$[m/n]_f(z) = \frac{P_m(z)}{Q_n(z)}, \qquad Q_n(0) = 1, \tag{3.8}$$

其中 P_m 与 Q_n 分别为次数至多 m 和 n 的多项式，并选取使得

$$f(z) - [m/n]_f(z) = \mathcal{O}(z^{m+n+1}) \quad \text{as } z \to 0. \tag{3.9}$$

将 Pad´e 逼近式 $\{[m/n]\}$ 按 (m,n) 索引排列而成的二维阵列称为 Pad´e 表；有关 Pad´e 近似值的更多细节，请参阅 [5, 15]。设 $(s_n(z))_n$ 为由下式定义的序列

$$s_n(z) = \sum_{k=0}^{n} a_k z^k.$$

将 Wynn 的 ε-算法应用于序列 $\{s_n(z)\}$（将其视为参数 z 的函数），可得基本恒等式

$$\varepsilon_{2k}^{(n)}(z) = [n+k/k]_f(z), \tag{3.10}$$

前提是相应的帕德近似值存在，且算法未发生故障。因此， ε-表中的偶数项与帕德´e表的对角项和近对角项完全一致。特别是，

$$\varepsilon_{2k}^{(0)}(z) = [k/k]_f(z),$$

对应于帕德´e表的主对角线。因此， ε-算法可以被视为帕德´e表生成器，其中索引 k 对应于帕德´e近似值的阶数，而索引 n 则表示沿序列的平移。这种帕德´e解释说明了 ε-算法为何具有非凡的效率。多项式逼近只能再现代数行为，而有理逼近则能够表示含有极点、分支点或其他奇点的函数。因此，对角帕德´e近似值常常能在原始幂级数发散或收敛极其缓慢的区域内收敛。从序列变换的角度看，帕德´e联系意味着， ε-算法会消除数列模型中的主导指数误差项

$$s_n = s + \sum_{j=1}^{m} c_j \lambda_j^n, \tag{3.11}$$

通过构造一个有理逼近式，使其分母捕捉与λ_j相关的主导极点。因此，k 的每次增加都会消除一个额外的指数分量，从而实现大幅加速收敛。

Pad´e 逼近式与 Shanks 变换之间的等价性可通过行列式表示明确体现出来。一般 Shanks 变换产生的近似值与对角 Pad´e 逼近式完全相同，而ε-算法则为这些行列式公式提供了数值稳定的递归实现。从这个意义上说， ε-算法可以被理解为一种高效的计算机制，用于求取 Pad´e 逼近式，而无需显式构造分子和分母多项式。

Pad´e 表的诠释具有若干重要后果：它解释了为什么 ε-算法在求和缓慢收敛或发散级数方面特别有效；为其在生成函数存在奇点时的稳健性提供了理论依据；并确立了收敛加速技术与经典有理逼近理论之间的直接联系。

从 Pad´é 表的角度来看， Wynn 的 ε-算法处于序列变换、有理逼近和数值分析交汇处的核心位置。这种解释不仅阐明了其收敛性质，也说明了它为何能在涉及级数求和和迭代过程的实际计算中长期取得成功。

3.2 连分数与有理逼近 连分数为理解有理逼近提供了一个自然且强大的框架，并在外推和收敛加速方法的理论分析中发挥着核心作用。特别是， Pad´é 近似值可以用连分数表示，许多外推算法也可被解释为隐式生成此类表示的过程。

Pad´é 近似　　值与连分数。给定一个形式幂　　级数

$$f(z) = \sum_{n=0}^{\infty} c_n z^n, \tag{3.12}$$

并考虑连分数

$$C(z) = b_0 + \cfrac{a_1 z}{b_1 + \cfrac{a_2 z}{b_2 + \cfrac{a_3 z}{b_3 + \ddots}}}. \tag{3.13}$$

令 $C_n(z) = A_n(z)\,B_n(z)$ 表示 C(z) 的第 n 个收敛子。对于固定的 z，当且仅当 $(C_n(z))_n$ 在 n 趋于无穷大时收敛到 C(z)，连分数才收敛。f 的 Pad´e 逼近可以表示为连分数的截断，我们有

$$C_{2k} = [k/k]_f(z) \text{ and } C_{2k+1} = [k+1/k]_f(z).$$

另请注意，要获得相应连分式的部分分子和分母，可以使用 H. Rutishauser [52] 提出的著名 qd算法。

通过 ε 算法隐式生成。虽然 Wynn 的 ε 算法不会显式构造连分式，但它通过递归更新隐式生成这种表示的系数。ε 表的每个偶数列都对应于与该序列的生成函数相关联的连分式的一个连续收敛子。从这个意义上说， ε 算法可被视为一种连分式加速器，它将部分和转换为能够捕捉底层函数主要解析结构的有理逼近。

3.3 E 算法 Brezinski 的 E 算法 [19], 是一种通用序列变换，旨在通过消除渐近误差展开中指定的分量来加速收敛。它提供了一个统一框架，涵盖一大类外推方法，包括理查森外推、Shanks变换、Levin 型变换以及 Wynn 的 e 算法。与 Wynn 的 ε 算法等纯递归格式不同，E 算法植根于一种显式代数表述，该表述基于对刻画误差的基函数的消去。

设 $\{s_n\}$ 为收敛于极限 s 的标量序列，并假定截断误差具有如下形式的展开

$$s_n = s + \sum_{j=1}^{\infty} c_j g_j(n), \tag{3.14}$$

其中 $\{g_j(n)\}$ 是已知函数的序列。函数 $g_j(n)$ 描述误差的主导渐近性态，并可表示代数衰减、指数衰减或对数衰减。
对于固定整数 k，E 算法通过施加精确性条件，构造极限 s 的近似值 $E_k^{(n)}$ 。

$$E_k^{(n)} = s \quad \text{whenever} \quad s_n = s + \sum_{j=1}^{k} c_j g_j(n). \tag{3.15}$$

由此得到一个线性方程组，其解给出了变换后的新近似值

$$E_k^{(n)} = \frac{\begin{vmatrix} s_n & s_{n+1} & \cdots & s_{n+k} \\ g_1(n) & g_1(n+1) & \cdots & g_1(n+k) \\ \vdots & \vdots & & \vdots \\ g_k(n) & g_k(n+1) & \cdots & g_k(n+k) \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \cdots & 1 \\ g_1(n) & g_1(n+1) & \cdots & g_1(n+k) \\ \vdots & \vdots & & \vdots \\ g_k(n) & g_k(n+1) & \cdots & g_k(n+k) \end{vmatrix}}. \tag{3.16}$$

这个行列式表达式推广了 Shanks变换，并明确展示了该方法所依据的代数结构。

定理 3.1. $\forall n$, $E^{(n)}$ k =S 当且仅当

$$s_n = s + a_1 g_1(n) + \ldots + a_k g_k(n),\ k, n \geq 0.$$

现在将 $E^{(n)}{}_k$ 写成舒尔补。记向量 g^n 为 $g_n = (g_1(n) \ldots, g_k(n))^T$， $G^{(n)}{}_k$ 为 k ×k

$$[\mathbf{G}_k^{(n)}]_{ij} = \Delta g_i(n + j - 1),\ i, j = 1, \ldots, k.$$

那么， $E^{((n)}{}_k$ 也可以表示为

$$E_k^{((n)} = \begin{vmatrix} s_n & \Delta S_{k,n} \\ \mathbf{g}_n & \mathbf{G}_k^{(n)} \end{vmatrix} / \left| \mathbf{G}_k^{(n)}, \right|$$

其中 $\Delta S_{k,n} = [\Delta s_n, \ldots, \Delta s_{n+k-1}]$. 然后，利用舒尔补公式，并假设矩阵 $G^{(n)}{}_k$ 是非奇异的，我们得到：

$$E_k^{(n)} = s_n - \Delta S_{k,n} \left(\mathbf{G}_k^{(n)} \right)^{-1} \mathbf{g}_n.$$

递归实现。尽管 E 算法是通过行列式定义的，但它也可以采用稳定的递归实现，避免直接进行行列式计算。这些递归显著降低了计算成本并提高了数值稳健性，使该方法适用于实际应用。

Brezinski 提出的 E 算法为加速序列收敛提供了通用框架，它通过显式建模并消去主要误差分量来实现这一目标。虽然其理论定义采用行列式或精确的消去条件，但实际而高效的实现依赖于递归方案，避免计算成本高昂的行列式，并提高数值稳定性。

递归实现的核心在于构造一个双指标表 ($E^{(n)}{}_k$)，定义如下：

$$\begin{cases} \mathbf{E}_0^{(n)} = \mathbf{s}_n, \; g_{0,i}(n) = g_i(n), \\ \mathbf{E}_k^{(n)} = \mathbf{E}_{k-1}^{(n+1)} - \dfrac{\mathbf{E}_{k-1}^{(n+1)} - \mathbf{E}_{k-1}^{(n)}}{\mathbf{g}_{k-1,k}^{(n+1)} - \mathbf{g}_{k-1,k}^{(n)}} \mathbf{g}_{k-1,k}^{(n)} \\ \mathbf{g}_{k,i}^{(n)} = \mathbf{g}_{k-1,i}^{(n+1)} - \dfrac{\mathbf{g}_{k-1,i}^{(n+1)} - \mathbf{g}_{k-1,i}^{(n)}}{\mathbf{g}_{k-1,k}^{(n+1)} - \mathbf{g}_{k-1,k}^{(n)}} \mathbf{g}_{k-1,k}^{(n)}, \quad i = k+1, \ldots, \end{cases}$$

约定 $E_{-1}^{(n)} = \mathbf{0}, \quad \forall n$。
序列 $g^{(n)}{}_{k,i}$ 还有如下行列式表达式

$$g_{k,i}^{(n)} = \frac{\begin{vmatrix} g_i(n) & g_i(n+1) & \cdots & g_i(n+k) \\ g_1(n) & g_1(n+1) & \cdots & g_1(n+k) \\ \vdots & \vdots & & \vdots \\ g_k(n) & g_k(n+1) & \cdots & g_k(n+k) \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \cdots & 1 \\ g_1(n) & g_1(n+1) & \cdots & g_1(n+1) \\ \vdots & \vdots & & \vdots \\ g_k(n) & g_k(n+1) & \cdots & g_k(n+k) \end{vmatrix}}. \tag{3.17}$$

这里，$g^{(n)}{}_{k,i}$ 也可以表示为如下舒尔补

$$g_{k,i}^{(n)} = g_i(n) - [\Delta g_i(n), \ldots, \Delta g_i(n+k-1)] \left(\mathbf{G}_k^{(n)} \right)^{-1} \mathbf{g}_n.$$

对基函数 $g_j(n)$ 的不同选择，可得到众所周知的外推技术，例如 Shanks变换和理查森外推；而选择多项式或有理函数，则可导出广义 Pad´e 型逼近。因此，E 算法为收敛加速提供了统一的代数框架。更多详情请参阅书籍 [19]。E 算法是一种强大而通用的框架，基于显式渐近建模进行序列变换。它将广泛的经典与现代外推方法统一起来，在收敛加速理论中发挥着核心作用，并使人们深入理解外推技术的代数与解析结构。其他著名的标量外推方法见 [19]。其中包括我们将在下文介绍的 ρ 和 θ 算法。

3.4 ρ算法 迭代 Δ^2 过程和 Wynn 的 ε-算法是加速线性收敛序列收敛的最有效经典工具之一。此外，它们还能够为一大类交错发散级数赋予有限值。

尽管适用范围广泛，迭代 Δ^2 过程和 ε-算法在应用于表现出对数收敛的序列时都会失效。这一局限性是根本性的，一些作者已经对此进行了形式化证明。为克服这一缺陷，Wynn 在 [63] 引入了一种新的外推法，称为 ρ-算法。

ρ算法递归定义如下：

$$\begin{cases} \rho_{-1}^{(n)} = 0, \qquad \rho_0^{(n)} = s_n, \qquad n \in \mathbb{N}, \\ \rho_{k+1}^{(n)} = \rho_{k-1}^{(n+1)} + \dfrac{x_{n+k+1} - x_n}{\rho_k^{(n+1)} - \rho_k^{(n)}}, \qquad k, n \in \mathbb{N}. \end{cases} ,$$

其中，$x_0, x1$，.....为给定数值。该算法尤其适用于加速渐近性态由对数项主导的序列的收敛。与ε算法类似， ρ算法只有通过原序列的偶数下标元素，才能得到其极限的有意义近似值。也就是说，量 $\rho_2^{(n)}{}_k$ 表示序列 $\{s_n\}$的极限外推近似值，而元素 $\rho_2^{(n)}{}_{k+1}$ 仅作为辅助量。如果变换收敛，这些奇数下标元素通常会发散，不应被视为近似值。

从理论角度看，量 $\rho_2^{(n)}{}_k$ 可以用连分数作自然解释。更确切地说，它们与一个连分数的终止项相同；该连分数在节点集 $\{x_n\}$ 上对序列 $\{s_n\}$ 进行插值，并将这一插值过程外推至无穷远。这种解释立即对插值点 $\{x_n\}$施加了结构条件：它们必须构成一个严格递增且无界的序列：

$$0 < x_0 < x_1 < \cdots < x_m < x_{m+1} < \cdots , \tag{3.18}$$

and

$$\lim_{n\to\infty} x_n = \infty. \tag{3.19}$$

在实际计算中，Wynn 的 ρ-算法通常采用标准形式，即通过选择插值点如下得到：

$$x_n = n + 1, \qquad n \in \mathbb{N}.$$

采用这一选择后，递推关系可简化为

$$\begin{cases} \rho_{-1}^{(n)} = 0, \qquad \rho_0^{(n)} = s_n, \\ \rho_{k+1}^{(n)} = \rho_{k-1}^{(n+1)} + \dfrac{k+1}{\rho_k^{(n+1)} - \rho_k^{(n)}}, \qquad k, n \in \mathbb{N}. \end{cases}$$

这一表述是文献和数值应用中最常见的形式；例如参见 [12]。

3.4.1 θ-算法 θ-算法由 Brezinski 于 [17] 提出，是一种非线性序列变换，旨在克服 Wynn 的 ε-算法和 ρ-算法的某些局限。具体而言， ε-算法对线性收敛序列十分有效，而 ρ-算法则适用于对数收敛序列；但当序列同时呈现线性和对数收敛行为时，这两种方法都无法令人满意地发挥作用。 θ-算法正是为应对这种中间情形而构造的。

设 $\{s_n\}_{n\geq 0}$ 是一个标量序列。 θ-算法定义如下递归方案：

$$\begin{cases} \theta_{-1}^{(n)} = 0, \qquad \theta_0^{(n)} = s_n, \qquad n \in \mathbb{N}, \\ \theta_{2k+1}^{(n)} = \theta_{2k-1}^{(n+1)} + D_{2k}^{(n)} \\ \theta_{2k+2}^{(n)} = \theta_{2k}^{(n+1)} - \dfrac{\Delta\theta_{2k}^{(n+1)}}{\Delta D_{2k+1}^{(n)}} D_{2k+1}^{(n)} \qquad k, n \in \mathbb{N}, \end{cases}$$

前提是分母不为零。
与 ε算法和 ρ算法的情况一样，只有 $\theta_2^{(n)}{}_k$ 下标为偶数的元素才用于逼近原序列的极限。量 $\theta_2^{(n)}{}_{k+1}$ 充当辅助项，并且当变换收敛时通常会发散。

θ算法的定义特征是在分母中显式使用二阶前向差分。这种结构使该变换能够检测并利用序列中的曲率，这是对数收敛的典型特征，同时仍保留加速线性收敛的能力。因此，在 ε算法和 ρ算法均失效或停滞的情况下， θ算法往往仍能取得成功。从理论角度看， θ算法可视为一种混合方法，结合了 Aitken型加速和 Wynn型非线性递推的思想。与 ρ算法不同，它不依赖外部插值点，也不要求预先了解余项的渐近形式。

若序列 $\{s_n\}$ 线性收敛，则θ算法表现出与ε算法相当的收敛性质。对于对数收敛序列，它往往能相对于 ε算法和 ρ算法带来显著改进。这种稳健性使 θ算法成为数值分析中的重要工具，尤其适用于慢收敛级数的求和，以及应用数学和物理学中出现的外推过程。关于 θ算法的进一步理论分析、推广及数值算例，可参见Brezinski及其合作者的著作。

4 向量外推方法 序列变换与外推技术的发展与成功，使得自20世纪70年代以来能够引入多种向量序列外推方法，以处理数值线性代数和不动点迭代中出现的问题。这些向量外推方法中较为著名的有Cabay和Jackson的最小多项式外推（MPE）方法 [24], Eddy [26] 与Mesina [47], 提出的降秩外推（RRE）方法，以及由Pugatchev [49] 提出并经Brezinski [12] 进一步分析的修正最小多项式外推（MMPE）方法，还有Wynn的向量 ε算法 [64]。这些方法的全面理论论述与实际实现可见于 [18, 42, 56, 59, 64]。

多项式外推框架通过与 Pad´e 近似和正交多项式的联系得到了进一步阐明和拓展 [15], ，同时也通过针对线性系统和不动点问题的详细算法研究得到了进一步发展。这方面的重要贡献见于 [20, 22, 40, 41, 43, 56, 59]。相关进展还包括 [58, 60] 中的研究，以及 Hansen 对秩亏和病态问题的讨论 [33, 34]。
近来，这些思想已从向量推广到分块与张量情形。 [44], 研究了分块外推方法，而 [7, 8]则发展了张量外推技术及其应用。这些拓展表明，外推方法仍是加速高维、结构化数值问题收敛的灵活而强大的工具。向量外推方法可分为两类：一类是多项式方法，包括 MPE、RRE 和 MMPE；另一类是基于 ε的方法，包括向量 ε算法和拓扑 ε算法。

由于 后文将广泛使用舒尔补公式，因此我们在此回顾其定义，以
求完整性。
Let

$$M = \begin{pmatrix} A & B \\ C & D \end{pmatrix},$$

其中 $A \in \mathbb{R}^{m\times m}$ 和 $D \in \mathbb{R}^{n\times n}$ 是方阵。若 A 可逆，则 M 中 A 的舒尔补定义为

$$(M/A) = D - CA^{-1}B.$$

类似地，如果 D 可逆，则 M 中 D 的舒尔补为

$$(M/D) = A - BD^{-1}C.$$

例如，请注意 $\det[(M/A)] = \det(M)/\det(A)$。

4.1 基于多项式的方法

4.1.1 向量多项式外推（VPE）方法 本小节介绍几种应用最广泛的向量多项式外推（VPE）方法，即最小多项式外推（MPE）方法 [24], 降秩外推（RRE）方法 [26, 47], 以及修正最小多项式外推（MMPE）方法 [12, 49]。这些方法旨在通过对序列项作适当的线性组合，加速向量序列 $(s_n) \subset \mathbb{R}^N$ 向极限（或不动点）的收敛。

设 (s_n) 为给定的向量序列，并考虑如下定义的 k 阶一般变换 T_k

$$T_k : \mathbb{R}^N \longrightarrow \mathbb{R}^N, \quad s_n \mapsto t_k^{(n)},$$

with

$$t_k^{(n)} = s_n + \sum_{i=1}^{k} a_i^{(n)} \phi_i(n), \quad n \geq 0, \tag{4.1}$$

其中，$(\phi_i(n))_n$ 是根据具体外推法选取的辅助向量序列。其思路是基于该序列的增量或差分的线性组合，构造 $t^{(n)}{}_k$，作为对 s_n 的修正。

我们还定义移位变换 $\widetilde{T}_k$ 如下:

$$\tilde{t}_k^{(n)} = s_{n+1} + \sum_{i=1}^{k} a_i^{(n)} \phi_i(n+1), \quad n \geq 0, \tag{4.2}$$

其中，系数 $a^{(n)}{}_i$ 与 T_k 中使用的系数相同。利用这个移位序列，我们可以定义 $t^{(n)}{}_k$ 的广义残差:

$$\begin{aligned} \tilde{r}(t_k^{(n)}) &= \tilde{t}_k^{(n)} - t_k^{(n)} \\ &= \Delta s_n + \sum_{i=1}^{k} a_i^{(n)} \Delta \phi_i(n), \end{aligned} \tag{4.3}$$

其中，前向差分算子 Δ 作用于序列指标:

$$\Delta \phi_i(n) = \phi_i(n+1) - \phi_i(n), \quad i = 1, \ldots, k.$$

对于多项式向量外推方法，辅助序列通常选为

$$\phi_i(n) = \Delta s_{n+i-1}, \quad i = 1, \ldots, k, \tag{4.4}$$

这意味着修正项基于序列的连续差分。随后通过对广义残差施加正交条件来确定系数 $a^{(n)}{}_i$:

$$\tilde{r}(t_k^{(n)}) \perp \operatorname{span}\{y_1^{(n)}, \ldots, y_k^{(n)}\}, \tag{4.5}$$

其中，$y^{(n)}{}_i$的选取（$i = 1,\ldots, k$）决定了所采用的方法:

$$y_i^{(n)} = \begin{cases} \Delta s_{n+i-1}, \text{ for} & \text{MPE} \\ \Delta^2 s_{n+i-1}, \text{ for} & \text{RRE} \\ y_i \text{ fixed vectors, for} & \text{MMPE.} \end{cases}$$

这里，$\Delta^2 s_n = \Delta(\Delta s_n)$ 表示二阶前向差分，而 MMPE 中的 y_i 通常按以下方式选取最小多项式构造。

定义子空间

$$\widetilde{W}_{k,n} = \operatorname{span}\{\Delta^2 s_n, \ldots, \Delta^2 s_{n+k-1}\}, \quad \widetilde{L}_{k,n} = \operatorname{span}\{y_1^{(n)}, \ldots, y_k^{(n)}\}.$$

那么，广义残差满足

$$\tilde{r}(t_k^{(n)}) - \Delta s_n \in \widetilde{W}_{k,n}, \tag{4.6}$$

$$\tilde{r}(t_k^{(n)}) \perp \widetilde{L}_{k,n}. \tag{4.7}$$

用矩阵形式表示，令 $L_{k,n}$,$\Delta S_{k,n}$，以及$\Delta^2 S_{k,n}$为 N ×k 矩阵，其列分别为 $y_1^{(n)}$,... , $y^{(n)}{}_k$、 Δs_n,... ,Δs_{n+k-1} 和 $\Delta^2 s_n, \ldots, \Delta^2 s_{n+k-1}$。那么

$$\tilde{r}(t_k^{(n)}) = \Delta s_n - \Delta^2 S_{k,n}\left(L_{k,n}^T \Delta^2 S_{k,n}\right)^{-1} L_{k,n}^T \Delta s_n, \tag{4.8}$$

近似值 $t^{(n)}{}_k$ 由下式给出

$$t_k^{(n)} = s_n - \Delta S_{k,n}\left(L_{k,n}^T \Delta^2 S_{k,n}\right)^{-1} L_{k,n}^T \Delta s_n, \tag{4.9}$$

这也可以表示为舒尔补。事实上，令 $\mathscr{T}_{(n)\,k}$ 为由下式定义的矩阵

$$\mathscr{T}_k^{(n)} = \begin{pmatrix} s_n & \Delta S_{k,n} \\ L_{k,n}^T \Delta s_n & L_{k,n}^T \Delta^2 S_{k,n} \end{pmatrix}. \tag{4.10}$$

因此，近似值 $t^{(n)}{}_k$ 可表示为以下舒尔补

$$t_k^{(n)} = \left(\mathscr{T}_k^{(n)} / L_{k,n}^T \Delta^2 S_{k,n}\right). \tag{4.11}$$

注意，近似值 $t^{(n)}{}_k$ 也可以写成两个行列式之比

$$t_k^{(n)} = \begin{vmatrix} s_n & \Delta S_{k,n} \\ L_{k,n}^T \Delta s_n & L_{k,n}^T \Delta^2 S_{k,n} \end{vmatrix} / \left| L_{k,n}^T \Delta^2 S_{k,n}, \right| \tag{4.12}$$

其中，分子中的向量行列式是按其第一行展开后得到的向量。

Eq等价地，$t^{(n)}{}_k$ 也可以表示为序列元素的凸线性组合（见 [42]） :

$$t_k^{(n)} = \sum_{j=0}^{k} \beta_j^{(n)} s_{n+j}, \quad \sum_{j=0}^{k} \beta_j^{(n)} = 1, \tag{4.13}$$

其中系数满足以下方程组

$$\sum_{j=0}^{k} \alpha_{i,j}^{(n)} \beta_j^{(n)} = 0, \quad i = 0, \ldots, k-1, \tag{4.14}$$

with

$$\alpha_{i,j}^{(n)} = \begin{cases} (\Delta s_{n+i}, \Delta s_{n+j}), & \text{MPE}, \\ (\Delta^2 s_{n+i}, \Delta s_{n+j}), & \text{RRE}, \\ (y_{i+1}, \Delta s_{n+j}), & \text{MMPE}. \end{cases}$$

最后，$t^{(n)}{}_k$ 还具有以下行列式表示:

$$t_k^{(n)} = \frac{\begin{vmatrix} s_n & s_{n+1} & \cdots & s_{n+k} \\ \alpha_{0,0}^{(n)} & \alpha_{0,1}^{(n)} & \cdots & \alpha_{0,k}^{(n)} \\ \vdots & \vdots & & \vdots \\ \alpha_{k-1,0}^{(n)} & \alpha_{k-1,1}^{(n)} & \cdots & \alpha_{k-1,k}^{(n)} \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \cdots & 1 \\ \alpha_{0,0}^{(n)} & \alpha_{0,1}^{(n)} & \cdots & \alpha_{0,k}^{(n)} \\ \vdots & \vdots & & \vdots \\ \alpha_{k-1,0}^{(n)} & \alpha_{k-1,1}^{(n)} & \cdots & \alpha_{k-1,k}^{(n)} \end{vmatrix}}. \tag{4.15}$$

这种表示突显了向量外推方法与经典行列式之间的联系，明确揭示了序列差分之间的线性相关，而这种相关性正是加速收敛的关键。

显然，RRE是一种正交投影，而MPE和MMPE是斜投影方法。降秩外推通过最小化前向差分线性组合的范数来构造外推向量。具体而言，RRE求解约束最小二乘问题

$$\min_{\boldsymbol{\alpha}} \left\| \sum_{j=0}^{k} \alpha_j \Delta s_{n+j} \right\|, \quad \text{subject to} \quad \sum_{j=0}^{k} \alpha_j = 1. \tag{4.16}$$

外推向量由此给出

$$\mathbf{x}_n^{\mathrm{RRE}} = \sum_{j=0}^{k} \alpha_j s_{n+j}. \tag{4.17}$$

RRE具有良好的数值稳健性，在用于线性不动点迭代时等价于GMRES方法。其最小二乘表述可确保稳定性，因此特别适用于大规模问题。

最小线性组合

多项式外推通过强制精确消去一个 $\{\alpha_j\}$ 来确定系数前向差分的组合：

$$\sum_{j=0}^{k} \alpha_j \Delta s_{n+j} = \mathbf{0}, \quad \sum_{j=0}^{k} \alpha_j = 1. \tag{4.18}$$

由此得到的外推向量为

$$\mathbf{x}_n^{\mathrm{MPE}} = \sum_{j=0}^{k} \alpha_j s_{n+j}. \tag{4.19}$$

当误差可表示为至多包含 k 个分量的指数展开时，MPE 方法可得到精确结果。然而，由于其实现需要求解一个可能病态的线性方程组，在有限精度算术中进行计算时，MPE 的数值稳健性可能不如 RRE。应用于非线性问题时，MPE 的表现与 RRE 和 Anderson加速有明显差异。当非线性迭代在解附近近似呈线性时，MPE 仍可能有效，但其性能通常对偏离线性的情况更为敏感

MMPE 在由选定测试向量 $\{\mathrm{y}_i\}_{i=1}^{k}$ 张成的约化子空间上施加正交条件：

$$\sum_{j=0}^{k} \alpha_j y_i^T \Delta s_{n+j} = 0, \quad i = 1, \ldots, k. \tag{4.20}$$

这使得系数 $\{\alpha_j\}$ 对应的线性方程组规模更小。外推向量仍由下式给出

$$\mathbf{x}_n^{\mathrm{MMPE}} = \sum_{j=0}^{k} \alpha_j s_{n+j}. \tag{4.21}$$

在实际计算中，RRE、MPE 和 MMPE 之间的选择取决于数值稳定性与精确性之间的平衡。在大规模问题中，RRE 通常因其稳健性而受到青睐；而当问题条件良好且追求高精度时，MPE 可能更为有效。MMPE 则介于这两种方法之间。总的来说，多项式向量外推方法构成了一个灵活而强大的加速收敛框架。借助多项式误差表示和 Krylov 子空间性质，RRE、MPE 和 MMPE 等方法已被证明在广泛的科学与工程应用中有效 [22, 25, 27]。

4.1.2 Sβ-算法 设 $\{s_n\}_{n\geq 0}$ 是 $\mathbb{R}^N$ 中的一个向量序列。设 k 为满足 $k < p$ 的正整数，且 $n \in \mathbb{N}$。我们考虑通过 MMPE 方法对给定的线性无关向量 $y1, \ldots, y_k$进行变换 (s_k^n)，并令 Y_k 为这些向量作为列组成的矩阵：

$$s_k^{(n)} = s_n - \Delta S_{n,k}\left(Y_k^T \Delta^2 S_{n,k}\right)^{-1} Y_k^T \Delta s_n. \tag{4.22}$$

我们假设求逆过程中涉及的所有矩阵都是非奇异的，这保证了变换后序列的存在。

在特殊情况下，当 $k = N$ 且向量 y_i选为 $\mathbb{R}^N$的标准基向量 $e_i, i = 1,..., N$ 时，该变换便简化为 Henrici 变换 [35, p。 116], , 该变换随后在 [13]中得到了进一步研究。量 $s^{(n)}{}_k$ 也可以表示为两个行列式的比值。具体而言，

$$s_k^{(n)} = \frac{\begin{vmatrix} s_n & s_{n+1} & \cdots & s_{n+k} \\ (y_1, \Delta s_n) & (y_1, \Delta s_{n+1}) & \cdots & (y_1, \Delta s_{n+k}) \\ \vdots & \vdots & & \vdots \\ (y_k, \Delta s_n) & (y_k, \Delta s_{n+1}) & \cdots & (y_k, \Delta s_{n+k}) \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \cdots & 1 \\ (y_1, \Delta s_n) & (y_1, \Delta s_{n+1}) & \cdots & (y_1, \Delta s_{n+k}) \\ \vdots & \vdots & & \vdots \\ (y_k, \Delta s_n) & (y_k, \Delta s_{n+1}) & \cdots & (y_k, \Delta s_{n+k}) \end{vmatrix}}, \tag{4.23}$$

其中，$(\cdot,\cdot)$ 表示 $\mathbb{R}^N$ 中的欧几里得内积，而分子中的行列式是沿第一行展开该行列式所得到的向量。我们还引入序列 $\beta^{(n)}{}_k$，它由 $s^{(n)}{}_k$ 的行列式表达式得到，方法是将 (4.23)) 分子中的第一行替换为：$\Delta s_n, \ldots, \Delta s_{n+k}$。以下称为 Sβ-算法的算法可递归计算两个序列 $s^{(n)}{}_k$ 和 $\beta^{(n)}{}_k$ 的各项；更多详情请参见 [38, 39] 。

$$\begin{cases} s_0^{(n)} = s_n,\ \beta_0^{(n)} = \Delta s_n, & n = 0, 1, \ldots, \\ \beta_k^{(n)} = \dfrac{\beta_{k-1}^{(n)} - a_k^{(n)} \beta_{k-1}^{(n+1)}}{1 - a_k^{(n)}}, & k \geq 1, \\ s_k^{(n)} = \dfrac{s_{k-1}^{(n)} - a_k^{(n)} s_{k-1}^{(n+1)}}{1 - a_k^{(n)}},\ a_k^{(n)} = \dfrac{\left(y_k, \beta_{k-1}^{(n)}\right)}{\left(y_k, \beta_{k-1}^{(n+1)}\right)}. \end{cases}$$

4.1.3 向量H算法 设 (s_n) 是一个向量序列，且 $(g_i(n)$ 是给定的某些序列。定义新的变换 $\mathscr{H}_k$ by $H^{(n)}{}_k = \mathscr{H}_k(s_n)$，其定义如下列两个行列式之比

$$H_k^{(n)} = \frac{\begin{vmatrix} s_n & s_{n+1} & \cdots & s_{n+k} \\ g_1(n) & g_1(n+1) & \cdots & g_1(n+k) \\ \vdots & \vdots & & \vdots \\ g_k(n) & g_k(n+1) & \cdots & g_k(n+k) \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \cdots & 1 \\ g_1(n) & g_1(n+1) & \cdots & g_1(n+k) \\ \vdots & \vdots & & \vdots \\ g_k(n) & g_k(n+1) & \cdots & g_k(n+k) \end{vmatrix}}.$$

可以看出，由 Sβ算法得到的 $s^{(n)}{}_k$ 是 $H^{(n)}{}_k$ 的一个特例。向量 $H^{n)}{}_k$ 可通过以下向量H算法递归计算

$$\begin{cases} H_0^{(n)} = s_n,\ g_{0,i}^{(n)} = g_i(n), & n = 0,1,\ldots, \\ H_k^{(n)} = H_{k-1}^{(n)} - \dfrac{g_{k-1,k}^{(n)}}{g_{k-1,k}^{(n+1)} - g_{k-1,k}^{(n)}} (H_{k-1}^{(n+1)} - H_{k-1}^{(n)}), & k \geq 1, \\ g_{k,i}^{(n)} = g_{k-1,i}^{(n)} - \dfrac{g_{k-1,k}^{(n)}}{g_{k-1,k}^{(n+1)} - g_{k-1,k}^{(n)}} (g_{k-1,i}^{(n+1)} - g_{k-1,i}^{(n)}), \end{cases}$$

向量H算法最早由 Brezinski 在 [16] 中提出，并用于实现 Henrici 的变换 [35, 55]。还要注意，H 算法等同于对向量 s_n 的每个分量同时应用标量 E 算法，并使用相同的辅助标量序列 $(g_i(n))$。对于 H 算法，也可以得到一种跳过失效和近失效的特定规则，见 [18]。

现在，设 ΔS_{n+k-1} 为列向量是 Δs_n、…、$\Delta s_{n+k-1}]$ 的 $N \times k$ 矩阵，g_n 为元素是 $g_1(n)\ldots$、$g_k(n)$ 的 k 维向量，G_n 为列向量是 Δg_n、Δg_{n+1}、…、$\Delta\Delta g_{n+k-1}$ 的 $k \times k$ 矩阵。那么，假设矩阵 G_n 非奇异，可得

$$H_k^{(n)} = s_n - \Delta S_{n+k-1} (G_n)^{-1} \mathbf{g}_n,$$

这也可以表示为以下舒尔补

$$H_k^{(n)} = \left(\mathscr{H}_k^{(n)} / G_n \right),$$

其中 $\mathscr{H}_{(n)\,k}$ 是由下式给出的矩阵

$$\mathscr{H}_k^{(n)} = \begin{pmatrix} s_n & \Delta S_{n+k-1} \\ \mathbf{g}_n & G_n \end{pmatrix}.$$

4.1.4 VEP 在线性方程组中的应用 考虑线性方程组

$$Cx = f, \tag{4.24}$$

其中 $C \in \mathbb{R}^{N\times N}$ 且 $f \in \mathbb{R}^N$。设 M 为一个非奇异预条件子。预处理后的线性系统为 s

$$M^{-1}Cx = M^{-1}f. \tag{4.25}$$

从 s_0，开始，生成

$$s_{j+1} = Bs_j + b, \quad j = 0,1,\ldots, \tag{4.26}$$

其中 $B = I - A$，$A = M^{-1}C$，且 $b = M^{-1}f$。于是，广义残差变为真实残差:

$$\tilde{r}(t_k^{(n)}) = r(t_k^{(n)}) = b - At_k^{(n)}. \tag{4.27}$$

此外，$\Delta^2 s_n = -A\Delta s_n$ 意味着 $\Delta^2 S_{k,n} = -A\Delta S_{k,n}$。接下来，为简化起见，令 $n = 0$，并记 $t_k = t^{(0)}{}_k$。设 d 为 B 关于 $s_0 - x^*$的最小多项式的次数。那么当 $k \leq d$ 时，ΔS_k 和 $\Delta^2 S_k$ 均为满秩，且 t_d 存在，并等于方程 (4.24) 的解。RRE、MPE 和 MMPE 生成的残差满足

$$\begin{cases} r_k \in \widetilde{W}_k = A\widetilde{V}_k, \\ r_k \perp \widetilde{L}_k, \end{cases}$$

~其中 V_k = 张成{$\Delta s0$、.....、Δs_{k-1}} 并且

$$\widetilde{L}_k = \begin{cases} \widetilde{W}_k, & \text{RRE}, \\ \widetilde{V}_k, & \text{MPE}, \\ \widetilde{Y}_k = \text{span}\{y_1,\ldots,y_k\}, & \text{MMPE}. \end{cases}$$

~设 P_k 为到 W_k上的正交投影算子。则 RRE 残差为

$$r_k^{\mathrm{RRE}} = r_0 - P_k r_0. \tag{4.28}$$

令 Q_k 和 R_k 分别表示 MPE 和 MMPE 的斜投影算子，可得

$$r_k^{\mathrm{MPE}} = r_0 - Q_k r_0, \text{ and}$$
$$r_k^{\mathrm{MMPE}} = r_0 - R_k r_0.$$

定义 r_0 与正交投影 $P_k r_0$ 之间的锐角 θ_k

$$\cos\theta_k = \max_{z\in\widetilde{W}_k\setminus\{0\}} \frac{|(r_0,z)|}{\|r_0\|\,\|z\|}. \tag{4.29}$$

我们还定义向量 r_0,$Q_k r_0$ 和 r0,$\mathbb{R}_k r_0$，分别与其间的两个锐角 ϕ_k 和 ψ_k。下面给出三种外推方法的残差范数所满足的一些关系。

定理 4.1. *[42]* 设 θ_k 为 r_0 与正交投影 $P_k r_0$，之间的锐角；设 ϕ_k 为 r_0 与 $Q_k r_0$ 之间的锐角；设 ψ_k 表示 r_0 与 $R_k r_0$ 之间的锐角。则有以下关系

1. $\| r_k^{rre} \|^2 = (1-\cos^2\theta_k)\ \| r_0 \|^2$。
2. $\| r^{mpe} k \|^2 = (\tan^2\phi_k)\ \| r_0 \|^2$。
3. $\| r^{mpe} k \| \le (\cos\ \phi_k)\ \| r_k^{rre} \|$。

此外，若 y_j 为 MMPE y_j r_0，或对于某个 $j = 1, \dots, k$，则我们还有

- $\| r_k^{mmpe} \|^2 = (tan^2\psi_k)\ \| r_0 \|^2$.
- $\| r_k^{mmpe} \| \le (\cos\psi_k)\ \| r_k^{rre} \|$.

用于求解线性方程组的一般向量外推算法总结于算法 1 中。

算法 1 线性系统的向量多项式外推（VPE）

初始化 $k = 0$，选择 x_0, ，并设置整数 p 和 m。

1. 基本迭代：令 $t_0 = x_0, z_0 = t_0$, ，然后对于 $j = 0, \dots, p-1$，计算

$$z_{j+1} = Bz_j + b.$$

2. 外推：令 $s_0 = z_p$，然后对于 $j = 0,\dots, m$，计算

$$s_{j+1} = Bs_j + b,$$

并使用 RRE、MPE 或 MMPE 计算近似值 t_m。

3. 更新 $s_0 = t_m$，将 $k = k+1$ 增加 1，然后返回步骤 1，直到收敛。

这些方法的稳定实现使用 QR分解 [58] 处理 RRE 和 MPE，或使用带主元策略 [41] 的 LU分解处理 MMPE，以减少存储需求并提高数值稳定性。向量多项式外推方法的一个关键性质是：当应用于线性生成的向量序列时，它们在数学上等价于一些广为人知的Krylov子空间方法，例如 GMRES [53] 和完全正交化方法（FOM） [54]；更多详情见 [6, 38, 42]。这在下面的定理中表述。

T定理 4.2. 当应用于线性生成的序列时，向量外推方法 RRE 和 MPE 在数学上分别等价于 GMRES 和 Arnoldi方法，而 MMPE 则等价于一种斜投影Krylov子空间方法。

根据该定理的结果，我们可以指出，所有多项式向量外推（PVE）方法都可以用多项式逼近理论来解释。系数 $\{\alpha_j\}$ 确定了一个多项式 $P(\lambda)$，其在迭代矩阵的主导特征值 λ_j 处的取值很小，即 $P(\lambda_j) \approx 0$。

因此，所得的外推迭代点能够有效抑制误差的主成分，从而加快收敛速度。

还应注意，由于 VPE方法在用于求解线性方程组时与Krylov子空间方法在数学上等价，因此它们至多经过 N 次迭代即可收敛，其中 N 为空间的维数。然而，与Krylov子空间方法一样，外推方法通常以重启模式使用，这样可以为大型问题节省存储空间。

4.1.5 在线性离散不适定问题中的应用 本小节仅关注将降秩外推（RRE）应用于截断奇异值分解（TSVD）生成的向量序列。TSVD迭代点被视为一个有序的近似值序列，其中截断指标起到迭代参数的作用。在此框架下，纳入更多奇异分量会逐步丰富近似值，但在不适定问题中，也可能因噪声放大而引入不稳定性。

应用RRE的目的是加速TSVD序列的收敛，同时保留其正则化性质。通过对连续的TSVD迭代点进行适当的线性组合，RRE利用问题内在的低秩结构，并增强最相关奇异模态的贡献。因此，RRE可作为一种有效的TSVD正则化后处理技术。

此外，RRE 方法能够隐式且自适应地控制近似的有效秩。由此得到的外推解通常能在精度和稳定性之间取得良好折衷，因此 TSVD–RRE 组合策略对于大规模病态问题尤其具有吸引力。详情及说明请参阅论文 [43]。

我们考虑对以下线性方程组的解进行近似的问题

$$Ax = b, \tag{4.30}$$

其中 $A \in \mathbb{R}^{m\times n}$ 是一个秩确定不良的矩阵；也就是说，A 有许多大小相差悬殊的奇异值，其中一些接近于零。特别地，A 严重病态，且可能是奇异矩阵。这里针对 $m \geq n$ 的情形介绍该方法，但它也适用于 $m < n$ 的情形。
Let $\hat{b} \in \mathbb{R}^m$ 表示与 b 对应的未知精确右端向量，因此

$$b = \hat{b} + e, \tag{4.31}$$

其中 e 表示测量误差或计算误差。我们的目标是计算精确相容方程组的最小范数解 $\hat{x}$ 的一个高精度近似

$$Ax = \hat{b}.$$

形式上，$\hat{x} = A^{\dagger}\hat{b}$，其中 $A^{\dagger}$ 表示 A 的摩尔－彭若斯伪逆。由于误差 e 的存在以及 A 的病态性，将 $A^{\dagger}$ 直接应用于 b，

$$A^{\dagger}b = A^{\dagger}(\hat{b} + e) = \hat{x} + A^{\dagger}e,$$

通常并不能给出 $\hat{x}$ 的可靠近似值。近似 $\hat{x}$ 的一种标准方法是采用截断奇异值分解（TSVD），它用低秩近似值替换 $A^{\dagger}$；参见例如 Golub 与 Van Loan [28] 或 Hansen [33]。设 A 的奇异值分解由下式给出

$$A = \sum_{j=1}^{n} \sigma_j u_j v_j^T, \tag{4.32}$$

其中奇异值满足

$$\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_\ell > \sigma_{\ell+1} = \cdots = \sigma_n = 0. \tag{4.33}$$

这里，向量 $u_j \in \mathbb{R}^m$ 和 $v_j \in \mathbb{R}^n$ 分别构成正交归一矩阵 $U = [u_1, \ldots, u_n] \in \mathbb{R}^{m\times n}$ 和 $V = [v_1, \ldots, v_n] \in \mathbb{R}^{n\times n}$ 的列。对于所讨论的问题，许多最小的非零奇异

值都非常接近零。
对于任意 $1 \leq \mathrm{k} \leq \ell$，A 的秩-k近似定义为

$$A_k = \sum_{j=1}^{k} \sigma_j \mathrm{u}_j \mathrm{v}_j^T, \tag{4.34}$$

其对应的伪逆为

$$A_k^{\dagger} = \sum_{j=1}^{k} \sigma_j^{-1} v_j u_j^T. \tag{4.35}$$

最小二乘问题的最小范数解

$$\min_{x \in \mathbb{R}^n} \|A_k x - b\|$$

则为

$$x_k := A_k^{\dagger} b = \sum_{j=1}^{k} \frac{u_j^T b}{\sigma_j} v_j. \tag{4.36}$$

一个关键问题是如何选择索引 k，使得 x_k 能够准确近似 ˆx。为对此进行形式化，令 $\mathrm{k}_{\mathrm{opt}} \geq 1$ 表示满足以下条件的最小整数

$$\|x_{k_{\mathrm{opt}}} - \hat{x}\| = \min_{k \geq 1} \|x_k - \hat{x}\|. \tag{4.37}$$

在下文中，我们令

$$y_i = \Delta^2 s_{i-1}; \text{ for } 1 \leq i \leq m-2,$$

并考虑由 TSVD 生成的序列 $\{\mathrm{s}_k\}_{k \geq 0}$。因此，

$$s_0 := 0, \qquad s_k := x_k = A_k^{\dagger} b = \sum_{j=1}^{k} \frac{u_j^T b}{\sigma_j} v_j = \sum_{j=1}^{k} \delta_j v_j, \tag{4.38}$$

其中

$$\delta_j := \frac{u_j^T b}{\sigma_j}, \qquad 1 \leq j \leq k. \tag{4.39}$$

因此，我们有

$$\Delta s_{k-1} = s_k - s_{k-1} = \delta_k v_k, \tag{4.40}$$

我们可以假设 $\delta_k \neq 0$，因为如果情况并非如此，那么就从序列 (4.38) 中删除相应项，并保持相同的指标记法来计算下一项。矩阵 $\Delta \mathrm{S}_{k-1} = [\Delta \mathrm{s0}, \ldots, \Delta \mathrm{s}_{k-1}]$ 可以分解为

$$\Delta S_{k-1} = [\delta_1 v_1, \ldots, \delta_k v_k] = V_k \, \mathrm{diag}[\delta_1, \ldots, \delta_k], \tag{4.41}$$

其中 $\mathrm{V}_k = [\mathrm{v1}, \ldots, \mathrm{v}_k]$. 此外，由于 $\Delta^2 \mathrm{s}_{k-1} = \delta_{k+1} \mathrm{v}_{k+1} - \delta_k \mathrm{v}_k$，我们推得

$$\Delta^2 S_{k-1} = V_{k+1} \begin{bmatrix} -\delta_1 & & & \\ \delta_2 & -\delta_2 & & \\ & \ddots & \ddots & \\ & & \delta_k & -\delta_k \\ & & & \delta_{k+1} \end{bmatrix}. \tag{4.42}$$

然后利用 (4.41)，得到

$$\Delta^2 S_{k-1}^T \Delta S_{k-1} = \begin{bmatrix} -\delta_1 & & & \\ \delta_2 & -\delta_2 & & \\ & \ddots & \ddots & \\ & & \delta_k & -\delta_k \\ & & & \delta_{k+1} \end{bmatrix}^T V_{k+1}^T \left[\delta_1 v_1, \ldots, \delta_k v_k\right].$$

另一方面，由于

$$V_{k+1}^T \left[\delta_1 v_1, \ldots, \delta_k v_k\right] = \begin{bmatrix} \delta_1 & 0 & \ldots & 0 \\ 0 & \delta_2 & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \ldots & 0 & \delta_k \\ 0 & 0 & \ldots & 0 \end{bmatrix}, \tag{4.43}$$

由此可得

$$\Delta^2 S_{k-1}^T \Delta S_{k-1} = \begin{bmatrix} -\delta_1^2 & \delta_2^2 & & \\ & -\delta_2^2 & \delta_3^2 & \\ & & & \\ & & -\delta_{k-1}^2 & \delta_k^2 \\ & & & -\delta_k^2 \end{bmatrix}.$$

外推向量 $t_k = t^{(0)}{}_k$ 为

$$t_k = \sum_{j=0}^{k} \gamma_j^{(k)} s_j, \tag{4.44}$$

其中，分量为 $\gamma_0^{(k)}, \ldots, \gamma^{(k)}{}_k$ 的向量 $\boldsymbol{\gamma}$ 满足以下线性方程组

$$\sum_{j=0}^{k} \gamma_j^{(k)} = 1, \quad \sum_{j=0}^{k} \gamma_j^{(k)} \eta_{ij} = 0, \tag{4.45}$$

其中 $\eta_{ij} = (\Delta^2 s_i, \Delta s_j)$.
令 $\beta_i = \gamma^{(k)}{}_i \quad / \gamma^{(k)}{}_k$，则可通过归一化变量 β_i 求得每个系数 $\gamma^{(k)}{}_i$

$$\gamma_i^{(k)} = \frac{\beta_i^{(k)}}{\sum_{j=0}^{k} \beta_j^{(k)}}.$$

因此，该向量

$$\boldsymbol{\beta}_k = (\beta_0^{(k)}, \ldots, \beta_{k-1}^{(k)})^T$$

满足以下线性方程组：

$$\left(\Delta^2 S_{k-1}^T \Delta S_{k-1}\right) \boldsymbol{\beta}_k = -\Delta^2 S_{k-1}^T \Delta s_k. \tag{4.46}$$

由于有限差分矩阵的结构，右端可简化为

$$\Delta^2 S_{k-1}^T \Delta s_k = [0, \ldots, 0, \delta_{k+1}^2]^T,$$

由此立即得到显式解

$$\beta_i^{(k)} = \frac{\delta_{k+1}^2}{\delta_{i+1}^2}, \qquad 0 \le i < k.$$

将这些系数相加，我们得到

$$\sum_{i=0}^{k} \beta_i^{(k)} = 1 + \sum_{i=0}^{k-1} \frac{\delta_{k+1}^2}{\delta_{i+1}^2} = \delta_{k+1}^2 \sum_{i=0}^{k} \frac{1}{\delta_{i+1}^2}.$$

因此，系数 $\gamma^{(k)}{}_j$ 可表示为闭式表达式

$$\gamma_j^{(k)} = \frac{\dfrac{1}{\delta_{j+1}^2}}{\displaystyle\sum_{i=0}^{k} \frac{1}{\delta_{i+1}^2}}, \qquad 0 \le j < k. \tag{4.47}$$

现在假设系数 $\gamma_0^{(k)}$、$\gamma_1^{(k)}$、...、$\gamma^{(k)}{}_k$ 已经计算得到。我们引入辅助变量

$$\alpha_0^{(k)} = 1 - \gamma_0^{(k)}, \qquad \alpha_j^{(k)} = \alpha_{j-1}^{(k)} - \gamma_j^{(k)}, \quad 1 \le j < k, \qquad \alpha_{k-1}^{(k)} = \gamma_k^{(k)}, \tag{4.48}$$

借助这些变量，我们可以将外推向量 t_k 改写为望远镜形式：

$$t_k = s_0 + \sum_{j=0}^{k-1} \alpha_j^{(k)} \Delta s_j = s_0 + \Delta S_{k-1} \boldsymbol{\alpha}_k, \tag{4.49}$$

其中 $\boldsymbol{\alpha}_k = [\alpha_0^{(k)}, \ldots, \alpha^{(k)}{}_{k-1}]^T$。利用（4.47），系数 α_i 可显式表示为

$$\alpha_i^{(k)} = \sum_{j=i+1}^{k} \gamma_j^{(k)} = \frac{\displaystyle\sum_{j=i+1}^{k} \frac{1}{\delta_{j+1}^2}}{\displaystyle\sum_{l=0}^{k} \frac{1}{\delta_{l+1}^2}}, \qquad 0 \le i < k. \tag{4.50}$$

最后，外推向量 t_k 具有如下谱表示

$$t_k = \sum_{j=1}^{k} \alpha_{j-1}^{(k)} \frac{u_j^T b}{\sigma_j} v_j. \tag{4.51}$$

这一表示揭示了，应用 RRE 方法相当于对截断奇异值分解(TSVD)实施一种滤波机制，其滤波因子为

$$f_j^{(k)} = \alpha_{j-1}^{(k)}, \qquad 1 \le j \le k.$$

我们需要评估 $\|t_{k+1} - t_k\|$，以确定我们的停止准则。由(4.51)可得

$$t_{k+1} - t_k = \sum_{j=1}^{k} \frac{(\alpha_{j-1}^{(k+1)} - \alpha_{j-1}^{(k)})}{\sqrt{\delta_j}} v_j + \frac{\alpha_k^{(k+1)}}{\sqrt{\delta_{k+1}}} v_{k+1}.$$

由于向量 $v_j, 1 \le j \le k+1$ 正交归一，因此可得

$$\|t_{k+1} - t_k\| = \sqrt{\sum_{j=1}^{k} \frac{|\alpha_{j-1}^{(k+1)} - \alpha_{j-1}^{(k)}|^2}{\delta_j} + \frac{|\alpha_k^{(k+1)}|^2}{\delta_{k+1}}}.$$

广义残差可写为

$$\tilde{r}(t_k) = \Delta S_k \boldsymbol{\gamma}_k = V_{k+1}\ \mathrm{diag}\left[\frac{u_1^T b}{\sigma_1}, \ldots, \frac{u_{k+1}^T b}{\sigma_{k+1}}\right] \boldsymbol{\gamma}_k, \tag{4.52}$$

其中 $\boldsymbol{\gamma}_k = [\gamma_0^{(k)}, \gamma_1^{(k)}, \ldots, \gamma^{(k)}{}_k\ \ ]^T$ 由(4.47)给出。由(4.39)可得

$$\|\tilde{r}(t_k)\|^2 = \sum_{i=0}^{k} \delta_{i+1}^2 \left(\gamma_i^{(k)}\right)^2,$$

再利用(4.47)，我们得到简单表达式

$$\|\tilde{r}(t_k)\| = \frac{1}{\sqrt{\sum_{j=0}^{k} \frac{1}{\delta_{j+1}^2}}}.$$

此外，由关系式(4.47)，我们还可得到广义残差范数的表达式 l:

$$\|\tilde{r}(t_k)\| = \sqrt{\delta_k^2 \gamma_{k-1}^{(k)}}.$$

对于适定问题，范数 $\|\tilde{}\mathrm{r}(\mathrm{t}_k)\|$ 随k的增大而单调递减并趋于零。相比之下，对于受右端扰动影响的不适定问题，只要k保持足够小，量 $\|\tilde{}\mathrm{r}(\mathrm{t}_k)\|$ 起初会随k的增大而减小。然而，超过某一阈值后，残差范数便开始随k的增大而增大。使 $\|\tilde{}\mathrm{r}(\mathrm{t}_k)\|$ 不再减小的指标k的取值，往往能给出接近期望解ˆx的近似值x_k。RRE-TSVD算法总结如下:

算法2 用于不适定问题的RRE-TSVD算法

- Compute the SVD of the matrix A: $[U, \Sigma, V] = svd(A)$.
  Set $s_0 = 0$, $s_1 = \frac{u_1^T b}{\sigma_1} v_1$, and $t_1 = s_1$, with
  $u_i = U(:,i)$ and $v_i = V(:,i)$ for $i = 1, \ldots, n$.
- For $k = 2, \ldots, n$
  1. Compute s_k from (4.38).
  2. Compute the $\gamma_i^{(k)}$ and $\alpha_i^{(k)}$ for $i = 0, \ldots, k-1$, using (4.47) and (4.50).
  3. Form the approximation t_k by (4.51).
  4. If $\|t_k - t_{k-1}\| / \|t_{k-1}\| < tol$, stop.
- End

我们通过考虑积分方程来说明序列 $\{\mathrm{t}_k\}_{k\geq 1}$ 和 $\{\|\tilde{}\mathrm{r}(\mathrm{t}_k)\|\}_{k\geq 1}$ 的收敛行为

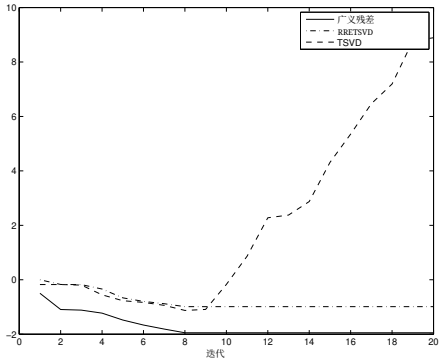
$$\int_0^{\pi/2} \kappa(s,t) x(t) dt = g(s), \qquad 0 \leq s \leq \pi, \tag{4.53}$$

其中

$$\kappa(s,t) = \exp(s\cos(t)) \ \text{ and } \ g(s) = 2\sin(s)/s.$$

解为 x(t) =sin(t)。该积分方程在 [4] 中有所讨论。我们使用 [34] 中的 MATLAB 代码 baart，通过以 600 个正交归一箱函数作为检验函数和试探函数的 Galerkin方法，对 (4.53) 进行离散化。这得到非对称矩阵 $\mathrm{A} \in \mathbb{R}^{600\times 600}$。其条件数 κ(A) :‖ ‖ $^{-1}\| = 4 \cdot 10^{18}$

A A。因此，A 在数值上是奇异的。向量 ˆb 和 b 按照上述方法确定，噪声水平为 $\nu = 1 \cdot 10^{-2}$。



FIG。4.1. 相对误差

图4.1展示了 TSVD 和 RRE-TSVD 的收敛情况。从图中可以看到，当 k 增大且足够小时，相对误差范数

$$\|s_k - \hat{x}\|/\|\hat{x}\| \text{ and } \|t_k - \hat{x}\|/\|\hat{x}\|$$

序列（s_k）由TSVD方法生成，而序列（t_k）则通过对TSVD迭代点应用RRE得到。对于较小的k值，两个序列中的近似误差均稳步减小。当截断指标超过最优值$k_{opt} = 9$时，与s_k 相关的误差由于噪声放大而开始增大，而t_k 的误差范数基本保持不变，并在$k = k_{opt}$处达到最小值。在广义残差范数中也可以观察到类似的停滞行为。

这一观察表明，广义残差范数不再减小的点，为确定最优截断指标提供了可靠且计算成本低廉的判据。因此，监测这种停滞现象为在TSVD和RRE–TSVD方法中选择k提供了一种实用策略（关于k_{opt}的定义，请参见方程（4.37））。

4.1.6 VEP 在非线性问题上的应用 考虑非线性方程组

$$G(x) = x, \quad G : \mathbb{R}^N \to \mathbb{R}^N, \tag{4.54}$$

残差为 $r(x) = G(x) - x$。从 s_0, 开始生成序列

$$s_{j+1} = G(s_j), \quad j = 0, 1, \ldots, \tag{4.55}$$

使得 $r(s_j) = \Delta s_j$。外推算法总结如下:

算法 3 非线性系统向量外推法

1. 令 $k = 0$，选择初始猜测 x_0、整数 p 和 m，以及容差 ε。2
. **Basic iteration:**
(a) 令 $t_0 = x_0, w_0 = t_0$。(b) 对于 $j = 0, \ldots, p - 1$,
计算 $w_{j+1} = G(w_j)$。
3. **外推 Extrapolation phase:**
(a) 令 $s_0 = w_p$。(b) 如果 $\|s_1 - s_0\| < \varepsilon$，则停止。
(c) 否则，对于 $j = 0, \ldots, m$，计算 $s_{j+1} = G(s_j)$。
(d) 使用 RRE、MPE 或 MMPE 计算 t_m。
4. 令 $s_0 = t_m$、$k = k + 1$、$x_0 = s_0$,，然后返回步骤 2。

记

$$G'(x^*) = J,$$

其中 x^* 是一个解，并假设函数 G 满足以下条件。

˜注意，这些向量方法不需要 G 的雅可比矩阵，并且以其完整形式，可在下一定理所述的某些条件下达到二次收敛。

矩阵 J –I 是非奇异的。我们设

$$M = (J - I)^{-1}. \tag{4.56}$$

此外，G 的 Fr´echet 导数 G′ 满足利普希茨条件，即

$$\|G'(x) - G'(y)\| \le L\|x - y\|, \quad \text{for all } x, y \in D, \tag{4.57}$$

其中 L 为正的常数，D 表示关注域。

定理 4.3. *[39]* 在 *[39],* 中所定义的关于 G 的更多条件下，存在 x^* 的一个邻域 U，使得对所有 $x_0 \in$ U,

$$\|x_{k+1} - x^*\| = o\left(\|x_k - x^*\|^2\right),$$

对 RRE 和 MPE 两种方法均成立。

在实践中，这些外推技术以黑箱方式应用于非线性过程生成的迭代点序列 (s_k),

$$s_{i+1} = G(s_i),\ i \ge 0,$$

无需显式访问雅可比矩阵 $G'(x^*)$。当非线性迭代局部收敛且映射 G 足够光滑时，相应的误差序列通常可渐近表示为有限个线性模态之和。在这些情况下，外推方法能够消去主要误差分量，从而加速收敛。

在这三种方法中，RRE 通常被认为在非线性情形下最为稳健。将其表述为最小二乘问题，可提高其对扰动、舍入误差以及偏离线性模型情形的稳定性。因此，在大规模非线性应用中，RRE 常被优先采用，包括非线性偏微分方程、积分方程和反问题的数值求解。

MPE 方法在多项式意义上对线性问题具有最优性，但用于非线性迭代时，其表现可能较难预测。具体而言，MPE 所依据的线性误差模型在远离不动点时可能并不准确，从而导致停滞甚至发散。尽管如此，当迭代点足够接近解时，MPE 仍可显著加速收敛，在某些情况下甚至优于 RRE。

MMPE方法可视为介于RRE与MPE之间的一种中间方法。通过借助辅助向量或权重引入额外的灵活性，MMPE可能提高某些非线性问题中的稳定性。然而，其有效性更强烈地依赖于参数选择和问题结构，这限制了它在一般非线性应用中的广泛使用。RRE、MPE和MMPE在非线性问题中的性能依赖于迭代在解附近局部线性化的有效性。当这一近似值充分时，外推方法可以显著增强收敛。否则，往往需要采用重启、截断或自适应窗口策略等保护技术来保持稳定性。

4.2 ε算法 在本部分中，我们考虑两种著名的向量 ε算法：Wynn的向量 ε以及Brezinski的拓扑 ε算法，并给出这些算法的一些变体。

4.2.1 向量ε算法 Wynn [64], 提出的标量 ε算法是一种广泛用于加速标量序列收敛的方法。一个自然的问题是如何将此算法推广到向量序列$(s_n) \subset \mathbb{R}^N$ 或 $\mathbb{C}^N$。这一推广的主要困难在于，向量不像标量那样具有自然的乘法逆元。

Wynn 提出的一种方法 [64], 是将向量 $z \in \mathbb{R}^N$ 的逆定义为

$$z^{-1} = \frac{z}{\|z\|^2}, \tag{4.58}$$

其中 $\|\cdot\|$ 表示 $\mathbb{R}^N$中的欧几里得范数。此定义保留了将标量算法推广到向量所需的某些代数性质。

利用这个向量逆，可以递归地定义向量 ε-算法（VEA）：

$$\begin{cases} \varepsilon_{-1}^{(n)} = 0, \quad \varepsilon_{0}^{(n)} = s_n, \quad n = 0, 1, \ldots, \\ \varepsilon_{k+1}^{(n)} = \varepsilon_{k-1}^{(n+1)} + \left[\varepsilon_{k}^{(n+1)} - \varepsilon_{k}^{(n)}\right]^{-1}, \quad k, n = 0, 1, \ldots \end{cases} \tag{4.59}$$

人们已对该算法的收敛性质进行了广泛研究。文献 [30, 46] 证明，若存在有限个标量 $a_0, \ldots, a_k$ 使得

$$\sum_{i=0}^{k} a_i (s_{n+i} - s) = 0, \quad a_k \neq 0, \quad a_0 + \cdots + a_k \neq 0, \tag{4.60}$$

对所有 $n \geq N_0$, 均成立，则向量ε-算法满足

$$\varepsilon_{2k}^{(n)} = s, \quad \forall n \geq N_0. \tag{4.61}$$

向量 ε-算法也可应用于由线性方程组的迭代求解所产生的序列。具体而言，若 (s_n) 由以下形式的迭代格式生成：

$$s_{n+1} = Bs_n + b, \tag{4.62}$$

那么，经过有限步后，标量和向量ε-算法都能得到该线性方程组的精确解 s^*：

$$\varepsilon_{2N}^{(n)} = s^*, \quad \forall n \geq 0, \tag{4.63}$$

中间量 $\varepsilon_2^{(n)}{}_k$（其中 $k < N$）能给出越来越精确的 x^*近似值。

向量 ε 算法也已成功应用于非线性序列，从而加速非线性问题中的不动点迭代。例如，它可应用于由非线性算子 G 生成的序列：

$$s_{n+1} = G(s_n), \tag{4.64}$$

以加速收敛至不动点 $s^* = G(s^*)$。

尽管功能强大，向量 ε 算法的计算量仍高于向量多项式外推方法，例如 RRE、MPE 和 MMPE。具体而言：

- 《计算》 $\varepsilon_{(n)}2_k$需要 $2k+1$序列向量 $s_n \cdots$、s_{n+2k}；而多项式方法只需要 $k+2$向量 $s_n \cdots$、s_{n+k+1}。
- 每一步都涉及向量求逆，还需要额外存储中间的 ε 量；当 n 较大或 k 较高时，这可能会占用大量资源。

- 尽管存储和计算需求较高，该算法仍能提供精度很高的近似值，尤其适用于序列收敛缓慢或呈现振荡行为的情况。

总之，向量 ε-算法是用于线性和非线性序列的强大收敛加速工具，可与向量多项式方法互为补充。它的主要优势在于良好的收敛性质，而主要局限则是所需的计算工作量较大。有关向量 ε-算法的更多性质和进展，可参见 [18, 31, 56]。

后来，一种更为复杂的发展建立在非交换代数结构的应用之上。在这一框架下，经典的行列式概念被设计式取代，后者为非交换域提供了恰当的推广。这种方法后来在实克利福德代数的框架内得到进一步发展 [56]。设计式的左式和右式版本均已在

早期的奠基性研究中得到定义和分析 [37]。
为阐明这一构造，考虑与矩阵 $\Delta_r^{(n)}$相关联的右设计式

$$\Delta_r^{(n)} = \begin{vmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{vmatrix}_r ,$$

其中，系数 a_{ij} 取值于一个非交换域。该量并非通过置换展开来定义，而是通过递归过程引入的。在最简单的非平凡情形，即对于一个 2×2 矩阵，可得

$$\Delta_r^{(2)} = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}_r = a_{22} - a_{12}a_{11}^{-1}a_{21}.$$

一般而言，令 A_{pq}^r 表示由 $\Delta_r^{(n)}$选取第 1 至 q 行、第 1 至 p 列以及索引为 r 的列所构成的 $p+1$ 阶右设计式。于是有如下递推关系：

$$\Delta_r^{(n)} = A_{n;n}^n - A_{n-1;n}^n \left(A_{n-1;n-1}^{n-1}\right)^{-1} A_{n;n-1}^n.$$

这一恒等式与舒尔补公式十分相似，凸显了设计式与块矩阵分解之间的结构相似性。

设计式自然地出现在非交换域 [48], 上线性系统的理论中，这也解释了它们在当前背景下的重要性。在这一代数框架中，文献 [56] 证明，通过对序列 (s_n) 应用向量 ε-算法（VEA）得到的向量$\varepsilon_{2\ k}^{(n)}$ 可表示为

$$\varepsilon_{2k}^{(n)} = \begin{vmatrix} s_n & \cdots & s_{n+k-1} & S_n \\ \vdots & & \vdots & \vdots \\ s_{n+k} & \cdots & s_{n+2k-1} & S_{n+k} \end{vmatrix}_r \left(\begin{vmatrix} s_n & \cdots & s_{n+k-1} & 1 \\ \vdots & & \vdots & \vdots \\ s_{n+k} & \cdots & s_{n+2k-1} & 1 \end{vmatrix}_r \right)^{-1} .$$

4.2.2 拓扑 ε-算法 在 [12], Brezinski 提出了另一种针对向量序列的标量ε-算法推广，它与向量 ε-算法有很大不同，被称为拓扑 ε-算法（TEA）。

这种方法包括计算序列 (s_n) 的极限或反极限的近似值 $e_k(s_n) = t^{(n)}{}_k$，使得

$$t_k^{(n)} = s_n + \sum_{i=1}^{k} a_i^{(n)} \Delta s_{n+i-1},\ n \geq 0. \tag{4.65}$$

我们考虑由下式定义的新变换 $\tilde{t}_{k,j}$，$j = 1, \ldots, k$:

$$\tilde{t}_{k,j}^{(n)} = s_{n+j} + \sum_{i=1}^{k} a_i^{(n)} \Delta s_{n+i+j-1},\ j = 1, \ldots, k.$$

令 $\tilde{t}^{(n)}{}_{k\,0} = t^{(n)}{}_k$，并将第 j 个广义残差定义如下

$$\begin{aligned} \tilde{r}_j(t_k^{(n)}) &= \tilde{t}_{k,j}^{(n)} - \tilde{t}_{k,j-1}^{(n)} \\ &= \Delta s_{n+j-1} + \sum_{i=1}^{k} a_i^{(n)} \Delta^2 s_{n+i+j-2}, j = 1, \ldots, k. \end{aligned}$$

因此，表达式 (4.65) 中涉及的 $t^{(n)}{}_k$ 系数的计算方式是，使每个第 j 个广义残差都与某个选定的向量 $y \in \mathbb{R}^N$正交，即

$$(y, \tilde{r}_j(t_k^{(n)})) = 0;\ \ j = 1, \ldots, k. \tag{4.66}$$

Henc向量 $a_n=(a_1^{(n)},\ldots,a^{(n)}{}_k)^T$是 k ×k 线性方程组 (4.66) 的解，该方程组写作 s

$$T_{k,n}\,a_n=\Delta S_{k,n}^T y \tag{4.67}$$

其中，$T_{k,n}$是一个矩阵，其列为 $\Delta^2 S_{k,n}^T y\ldots,\Delta^2 S_{k,n+k-1}^T y$（假设其为非奇异矩阵），而$\Delta_j S_{k,n}$、j = 1、2 是 N ×k 矩阵，其列为 $\Delta_j s_n,\ldots,\Delta_j s_{n+k-1}$、j = 1、2。注意，k ×k 矩阵 $T_{k,n}$ 还有如下公式

$$T_{k,n}=\mathscr{S}_{k,n}\,(I_N\otimes y)$$

其中，$\mathscr{S}_{k,n}$是一个 k ×Nk 矩阵，其块列为 $\Delta^2 S_{k,n}^T,\ldots,\Delta^2 S_{k,n+k-1}^T$。利用（4.65）和（4.67），$t^{(n)}{}_k$ 可用矩阵形式表示为

$$t_k^{(n)}=s_n-\Delta S_{k,n}\,T_{k,n}^{-1}\,\Delta S_{k,n}^T y. \tag{4.68}$$

利用 Schur 公式，$t^{(n)}{}_k$ 可表示为两个行列式之比

$$t_k^{(n)}=\begin{vmatrix} s_n & \Delta S_{k,n} \\ \Delta S_{k,n}^T y & T_{k,n}\end{vmatrix}\Big/\,det(T_{k,n}).$$

对于拓扑ε-算法的核，容易看出，如果
∀n ∃a0, . . . , a_k，其中 $a_k\neq 0$ 且 $a_0+\ldots a_k\neq 0$，满足 $_k\sum a_i(s_{n+i}-s)=0$，那么 ∀ n, $t^{(n)}{}_k=s._{i=0}$
向量 $e_k(s_n)=t^{(n)}{}_k$ 可以表示为以下行列式之比

$$\mathbf{e}_k(s_n)=\frac{\begin{vmatrix} s_n & \ldots & s_{n+k} \\ (y,\Delta s_n) & \ldots & (y,\Delta s_{n+k}) \\ \ddots & \ldots & \ddots \\ (y,\Delta s_{n+k-1}) & \ldots & (y,\Delta s_{n+2k-1})\end{vmatrix}}{\begin{vmatrix} 1 & \ldots & 1 \\ (y,\Delta s_n) & \ldots & (y,\Delta s_{n+k}) \\ \ddots & \ldots & \ddots \\ (y,\Delta s_{n+k-1}) & \ldots & (y,\Delta s_{n+2k-1})\end{vmatrix}} \tag{4.69}$$

并可通过 Brezinski [12] 发现的拓扑ε-算法（TEA）递归计算，其计算方案如下

$$\begin{cases} \widehat{\boldsymbol{\varepsilon}}_{-1}^{(n)}=0;\ \widehat{\boldsymbol{\varepsilon}}_{0}^{(n)}=s_n, \\ \widehat{\boldsymbol{\varepsilon}}_{2k+1}^{(n)}=\widehat{\boldsymbol{\varepsilon}}_{2k-1}^{(n+1)}+\dfrac{y}{(y,\Delta\widehat{\boldsymbol{\varepsilon}}_{2k}^{(n)})} \\ \widehat{\boldsymbol{\varepsilon}}_{2k+2}^{(n)}=\widehat{\boldsymbol{\varepsilon}}_{2k}^{(n+1)}+\dfrac{\Delta\widehat{\boldsymbol{\varepsilon}}_{2k}^{(n)}}{(\Delta\widehat{\boldsymbol{\varepsilon}}_{2k+1}^{(n)},\Delta\widehat{\boldsymbol{\varepsilon}}_{2k}^{(n)})},\quad n,k=0,1,\ldots. \end{cases}$$

前向差分算子 Δ 作用于上标 n，因此我们有

$$\widehat{\boldsymbol{\varepsilon}}_{2k}^{(n)}=\mathbf{e}_k(s_n);\ \ \widehat{\boldsymbol{\varepsilon}}_{2k+1}^{(n)}=\frac{y}{(y,\mathbf{e}_k(\Delta s_n))},\ n,k=0,1,\ldots.$$

对于向量序列 (s_n)，第一拓扑Shanks变换 (4.69) 可通过一种称为第一简化拓扑 ε-算法 (STEA1) [20]的递归方案计算。该算法定义如下:

$$\widehat{\boldsymbol{\varepsilon}}_{2k+2}^{(n)}=\widehat{\boldsymbol{\varepsilon}}_{2k}^{(n+1)}+\frac{\boldsymbol{\varepsilon}_{2k+2}^{(n)}-\boldsymbol{\varepsilon}_{2k}^{(n+1)}}{\boldsymbol{\varepsilon}_{2k}^{(n+1)}-\boldsymbol{\varepsilon}_{2k}^{(n)}}\left(\widehat{\boldsymbol{\varepsilon}}_{2k}^{(n+1)}-\widehat{\boldsymbol{\varepsilon}}_{2k}^{(n)}\right),\qquad k,n=0,1,\ldots, \tag{4.70}$$

$$\widehat{\boldsymbol{\varepsilon}}_{0}^{(n)}=s_n,\qquad n=0,1,\ldots. \tag{4.71}$$

标量 $\varepsilon^{(n)}{}_k$ 是将标量 ε-算法应用于标量序列 z_n =(y, s_n) 后得到的。

现在令 ~$e_k(s_n)$ 为由（4.69）得到的向量，其中将第一行替换为向量：s_{n+k},... , s_{n+2k}。第二拓扑Shanks变换也可以通过第二种简化拓扑ε-算法（STEA2） [20], 计算，该算法采用相同的初始化（4.71），并定义如下：

$$\widetilde{\boldsymbol{\varepsilon}}_{2k+2}^{(n)} = \widetilde{\boldsymbol{\varepsilon}}_{2k}^{(n+1)} + \frac{\varepsilon_{2k+2}^{(n)} - \varepsilon_{2k}^{(n+1)}}{\varepsilon_{2k}^{(n+2)} - \varepsilon_{2k}^{(n+1)}} \left(\widetilde{\boldsymbol{\varepsilon}}_{2k}^{(n+2)} - \widetilde{\boldsymbol{\varepsilon}}_{2k}^{(n+1)} \right), \qquad k, n = 0, 1, \ldots. \tag{4.72}$$

正如 [20], 所指出的，作者已在已发表的论文中确立了拓扑 ε-算法的若干收敛与加速性质。然而，这些结果所需的假设在实践中往往难以验证。简化算法的出现，使得我们能够在更适用于实际情形的假设下确立新的收敛性结果。因此，简化的 ε-算法不仅是拓扑Shanks变换数值实现的工具，也是构建和分析其底层理论框架的关键要素。

4.3 TEA在线性和非线性系统中的应用 再次考虑方程组 Ax = b，并设 (s_n) 为由线性过程 $s_{n+1} = Bs_n + b$ 生成的向量序列，其中 B = I −A。

利用 $\Delta^2 s_{n+i} = B\Delta^2 s_{n+i-1}$ 这一事实，矩阵 $T_{k,n}$ 现在可表示为

$$T_{k,n} = -L_k^T A \Delta S_{k,n} \tag{4.73}$$

其中，L_k是一个 N × k 矩阵，其各列为 y、B^Ty.....、$B^T{}_{k-1}$y。由于 n 是固定整数，为简便起见，令 n = 0，并将 $T_{k,}0$ 记为 T_k，将 $\Delta S_{k,}0$ 记为 ΔS_k。另一方面，不难看出

$$\Delta S_k^T y = L_k^T r_0. \tag{4.74}$$

因此，将 (4.73) 和 (4.74) 代入 (4.68)，可得 TEA 产生的第 k 个残差为

$$r_k^{tea} = r_0 - A\Delta S_k (L_k^T A \Delta S_k)^{-1} L_k^T r_0. \tag{4.75}$$

设 E_k 表示到 Krylov子空间 K_k(A,Ar_0) 上、且与 Krylov子空间 $K_k(B^T, y) = K_k(A^T, y)$ 正交的斜投影算子。由 (4.75) 可知，TEA 生成的残差可写为

$$r_k^{tea} = r_0 - E_k r_0. \tag{3.9}$$

这表明，topological ε-算法在数学上等价于 Lanczos方法。注意，当且仅当 k × k 矩阵 $L_k^T A\Delta S_k$ 非奇异时，TEA 定义的第 k 个近似值才存在。下面的结果给出了 $y = r_0$ 时残差范数满足的一些关系。

T定理4.4. 设 φ_k 为 r_0 与 $E_k r_0$ 之间的锐角，并设 y = r_0。则有以下关系：

1. $\| r_k^{tea} \| = (\tan\varphi_k) \| r_0 \|$; k > 1。
2. $\| r_k^{tea} \| \leq (\cos \varphi_k) \| r_k^{rre} \|$。

注2。

- 定理4.4的第一个结果表明，当且仅当 cosφk ≠ 0 时，TEA 残差才有定义。
- 我们还注意到，如果 RRE 中出现停滞（$\| rrrek \| = \| r_0 \|$ 对某个 k 成立），则 cosθk = 0，这意味着 cosφk = 0，从而表明 TEA 近似没有定义。

拓扑 ε-算法也可用于求解非线性方程组。在此情形下，该算法应用于由底层非线性迭代过程生成的序列 (s_n)。TEA 的一个重要优点是，无需显式了解非线性映射 G 的雅可比矩阵~，同时在适当假设下仍能实现二次收敛。将 VEA 和 TEA 用于求解线性和非线性问题时，所需的计算量和存储空间会随迭代步数增加。因此，在实际应用中，对于大型问题，必须重新启动这些算法。在外推阶段之前先进行若干次基本迭代也很有用

。将 VEA 或 TEA 应用于线性和非线性系统 $x = G(x)$，可得到以下算法。

算法 4：用于非线性系统的向量外推算法（VEA/TEA）

---

1. 初始化：令 $k = 0$，选择初始猜测 x_0, 以及整数 p 和 m。
2. 基本迭代：(a) 令 $t_0 = x_0, w_0 = t_0$。(b) 对于 $j = 0, \ldots, p-1$，计算 $w_{j+1} = G(w_j)$ 3. 外推阶段：

   (a) 令 $s_0 = w_p$。(b) 如果 $\|s_1 - s_0\| < \varepsilon$，则停止。(c) 否则，对于 $j = 0, \ldots, 2m-1$，计算 $s_{j+1} = G(s_j)$。(d) 计算近似值

$$t_m = \varepsilon_{2m}^{(0)}$$

   使用 VEA 或 TEA。
4. 更新：令 $s_0 = t_m$，$k = k+1$，并返回步骤 3。

4.4 Anderson 加速方法 Anderson加速方法由 D. G. Anderson 于 1965 [3] 提出，作为一种提高不动点迭代收敛性的技术。最初的动机是加速用于求解非线性积分方程的简单迭代方案。此后，该方法得到推广，并广泛用于计算科学、优化和数值线性代数领域中的非线性方程组求解。

Anderson加速是一种强大的迭代技术，用于求解以不动点形式表示的非线性系统

$$x = G(x), \tag{4.76}$$

或者，等价地，

$$F(x) := G(x) - x = 0. \tag{4.77}$$

该方法最初由 Anderson 提出，之后在现代数值优化和科学计算背景下得到重新表述和分析；例如参见 [3, 22, 36, 61]。

设 $\{x_k\}_{k\geq 0} \subset \mathbb{R}^p$ 表示一个迭代点序列，并定义相应的残差向量

$$F_k = F(x_k) = G(x_k) - x_k.$$

经典的不动点迭代形式如下

$$x_{k+1} = x_k + \beta_k F_k,$$

其中 $\beta_k > 0$ 是一个松弛（或阻尼）参数。安德森加速通过结合多个先前迭代点及其残差信息，构造更准确的更新方向，从而增强这一基本迭代。

选择一个初始向量 x_0, 、整数深度参数 $m \geq 1$ 和初始阻尼参数 $\beta_0 > 0$。第一个迭代点按下式计算：

$$x_1 = x_0 + \beta_0 F_0. \tag{4.78}$$

对于每个迭代索引 $k \geq 1$，定义

$$m_k = \min(m, k),$$

并引入差分向量

$$\Delta x_i = x_{i+1} - x_i, \qquad \Delta F_i = F_{i+1} - F_i.$$

利用这些量，构造矩阵

$$\Delta X_k = \left[\Delta x_{k-m_k}, \ldots, \Delta x_{k-1}\right], \qquad \Delta \mathscr{F}_k = \left[\Delta F_{k-m_k}, \ldots, \Delta F_{k-1}\right].$$

通过求解最小二乘问题确定 Anderson组合的系数

$$\theta_k = \arg\min_{\theta \in \mathbb{R}^{m_k}} \|F_k - \Delta \mathscr{F}_k \theta\|_2, \tag{4.79}$$

该问题旨在寻找过去残差差分的线性组合，使其能够最佳地逼近当前残差。当 $\Delta \mathscr{F}_k$ 的列线性无关时，解可显式表示为

$$\theta_k = (\Delta \mathscr{F}_k^T \Delta \mathscr{F}_k)^{-1} \Delta \mathscr{F}_k^T F_k. \tag{4.80}$$

加速更新：计算出 θ_k 后，定义加速中间量

$$y_k = x_k - \Delta X_k \theta_k, \text{ and } \bar{F}_k = F_k - \Delta \mathscr{F}_k \theta_k.$$

然后通过以下方式得到下一迭代

$$x_{k+1} = y_k + \beta_k \bar{F}_k, \tag{4.81}$$

或者，等价地，

$$x_{k+1} = x_k + \beta_k F_k - (\Delta X_k + \beta_k \Delta \mathscr{F}_k)\theta_k. \tag{4.82}$$

这里，$\beta_k > 0$ 是一个可能依赖于迭代的阻尼参数，其中 $\beta_k = 1$ 对应于无阻尼安德森方法。

向　量$\mathbf{y}_k$可视为通过最优组合过去迭代而得到的外推近似，
以便　减小当前残差。事实上，利用(4.80)，我们可以写出

$$y_k = x_k - \Delta X_k (\Delta \mathscr{F}_k^T \Delta \mathscr{F}_k)^{-1} \Delta \mathscr{F}_k^T F_k, \tag{4.83}$$

这表明$\mathbf{y}_k$ 依赖于$\mathbf{F}_k$ 在由先前残差差分所张成的子空间上的投影。

此外，由于 $\mathbf{y}_k$ 是一个舒尔补，因此可以将其表示为两个行列式的比值，其中分子中的行列式是一个向量，可通过沿第一行展开该行列式得到：

$$y_k = \begin{vmatrix} x_k & \Delta x_{k-m_k} & \cdots & \Delta x_{k-1} \\ \Delta \mathscr{F}_k^T F_k & & \Delta \mathscr{F}_k^T \Delta \mathscr{F}_k & \end{vmatrix} \Big/ \left|\Delta \mathscr{F}_k^T \Delta \mathscr{F}_k\right|. \tag{4.84}$$

直观地说，Anderson加速通过对过去迭代进行线性组合，以最优方式减小残差，从而显著提高收敛速度。实践中观察到，Anderson加速通常比标准不动点迭代收敛得更快，有时还能稳定原本会发散的序列。

该方法已成功用于求解计算流体力学、电子结构计算和优化中的非线性系统。该方法还用于加速量子化学中的自洽场（SCF）迭代，以及增强应用于线性系统的Krylov子空间方法。

请注意，在物理学和化学文献中，该方法也称为向量安德森加速或安德森混合。现代实现通常会采用阻尼、预条件或重启，以提高其在大规模问题上的稳健性。有关该方法收敛性的更多详情，请参阅论文 [3, 22, 23, 36, 61]。

备注 3. 参数 m 决定了计算新迭代 x_{k+1} 时使用多少先前迭代的信息。如果 m 过小，可利用的历史信息有限，可能导致慢收敛。反之，如果 m 过大，较早迭代中的信息可能会在过多的步骤中保留，从而降低更新效果，再次导致收敛变慢。此外，m 的取值会直接影响相关优化问题的维数。m 取值较大可能导致病态最小二乘问题，并增加求解这些问题的计算成本。

在上述算法框架下，参数 m_k 可以在不同迭代之间变化。一种简单策略是在第 k 次迭代时将 m_k =设为 k，这种选择通常称为无截断Anderson加速。在这种情况下，新迭代点 x_{k+1} 的计算会纳入此前生成的所有迭代点的信息。另一种更精细的策略是动态调整 m_k 的值，以控制相关最小二乘问题的条件数。

用于求解非线性方程组 G(x) = x 的 Anderson 加速（AA）方法的各个步骤总结于算法 5 中。

算法 5 Anderson 加速方法

输入：不动点映射 G、初始猜测 x_0, 、深度 m ≥ 1、阻尼参数 $\{\beta_k\}$、容差 ε。
输出：满足 $\|F(x_k)\| \leq \varepsilon$ 的近似解 x_k。
步骤 1。计算初始残差 $F_0 = F(x_0)$，并设置 $x_1 = x_0 + \beta_0 F_0$.
步骤 2. 对于 k = 1, 2, ... :

- 令 $m_k = \min(m, k)$，并令 $F_k = F(x_k)$。
- 构造差分向量 $\Delta x_i = x_{i+1} - x_i$ 和 $\Delta F_i = F_{i+1} - F_i$，其中 i = k $-m_k \cdots$，k -1。
- 构造矩阵 $\Delta X_k = \left[\Delta x_{k-m_k} \cdots, \Delta x_{k-1}\right]$, $\Delta\mathscr{F}_k = \left[\Delta F_{k-m_k} \cdots, \Delta F_{k-1}\right]$
- 将系数向量 $\theta_k \in \mathbb{R}_{mk}$ 计算为以下最小二乘问题的解

$$\theta_k = \arg\min_{\theta \in \mathbb{R}^{m_k}} \|F_k - \Delta\mathscr{F}_k \theta\|_2 .$$

- 定义加速量：$y_k = x_k - \Delta X_k \theta_k$　　$\bar{F}_k = F_k - \Delta\mathscr{F}_k \theta_k$.
- 更新迭代值

$$x_{k+1} = y_k + \beta_k \bar{F}_k .$$

- 如果 $\|F_k\| \leq \varepsilon$，则停止。

End

4.5 分块外推法

4.5.1 多项式块外推方法 设 (S_n) 是一个 $\mathbb{R}^{N\times s}$ 的矩阵序列，并考虑从 $\mathbb{R}^{N\times s}$ 到 $\mathbb{R}^{N\times s}$ 的变换 T_k，其定义如下：

$$T_k : S_n \to T_k^{(n)} = S_n + \sum_{i=1}^{k} G_i(n) A_i^{(n)}, \; n \geq 0 \tag{4.85}$$

其中 辅助的 N × 个 s 阶矩阵序列 $(G_i(n))_n$；i = 1,... , k 已给定，且 s × 个系数 $A^{(}$ $_i^{n)}$ 将会 ˜确定。令 T_k 表示由 T_k 按如下方式得到的新变换

$$\widetilde{T}_k^{(n)} = S_{n+1} + \sum_{i=1}^{k} G_i(n+1) A_i^{(n)}, \; n \geq 0. \tag{4.86}$$

我们将 T (n) k 的广义残差定义为 ˜R(T (n) k) = ˜T (n) k −T (n) k，也可表示为 ˜R(T (n) k) = ΔS_n + k $\sum \Delta G_i(n) A^{(n)}$ i . i=1前向差分算子 Δ 作用于指标 n，即 $\Delta G_i(n) = G_i(n+1) - G_i(n)$，i = 1,..., k。我们

稍后将看到，在求解线性方程组时，序列 $(S_n)_n$ 由线性过程生成，此时广义残差与经典残差一致。矩阵系数 $A^{(n)}{}_i$由正交关系得到

$$\widetilde{R}(T_k^{(n)}) \in (\mathrm{range}(Y_1^{(n)},\ldots,Y_k^{(n)}))^{\perp} \tag{4.87}$$

其中，$Y_{(n)}1,\ldots,Y_{(n)k}$是给定的 N ×s 矩阵。若 $\widetilde{\mathscr{W}_k}_{,n}$ 和 $\widetilde{\mathscr{L}_k}_{,n}$ 分别表示块子空间 $\widetilde{\mathscr{W}_k}_{,n}=\mathrm{rang}(\Delta G_1(n)\ldots,\Delta G_k(n))$（由 $\Delta G_1(n)\ldots,\Delta G_k(n)$ 的列生成的子空间）和 $\widetilde{\mathscr{L}_k}_{,n}=\mathrm{range}(Y_1^{(n)},\ldots,Y_{(n)k})$（由 $Y_{(n)}1,\ldots,Y_{(n)k}$的列生成）则由 (4.86) 和 (4.87) 可得，广义残差满足以下关系

$$\widetilde{R}(T_k^{(n)}) - \Delta S_n \in \widetilde{\mathscr{W}_{k,n}}, \tag{4.88}$$

and

$$\widetilde{R}(T_k^{(n)}) \in \widetilde{\mathscr{L}_{k,n}^{\perp}}. \tag{4.89}$$

关系 (4.88) 和 (4.89) 表明，广义残差 $\widetilde{R}(T^{(n)}k)$ 是通过将矩阵 ΔS_n 的列正交投影到块子空间 $\widetilde{\mathscr{W}_k}_{,n}$ 上得到的。在矩阵形式下，$\widetilde{R}(T^{(n)}k)$ 可写为 $\widetilde{R}(T^{(n)}k) = \Delta S_n - \Delta\mathscr{G}_{k,n}(L_{k,n}^T\Delta\mathscr{G}_{k,n})^{-1}L_{k,n}^T\Delta S_n$，其中 $\Delta\mathscr{G}_{k,n}$ 和 $L_{k,n}$ 是 N ×k 阶矩阵，分别由以下矩阵的列生成：$\Delta G_1(n)\ldots,\Delta G_k(n)$ 和 $Y(n)1,\ldots,Y(n)k$。近似值 $T^{(n)}k$ 定义为

$$T_k^{(n)} = S_n - \mathscr{G}_{k,n}(L_{k,n}^T\Delta\mathscr{G}_{k,n})^{-1}L_{k,n}^T\Delta S_n, \tag{4.90}$$

其中 $\mathscr{G}_{k,n}$是由 $\mathscr{G}_{k,n}=[G_1(n)\ldots,G_k(n)]$定义的 N ×ks 块矩阵。注意，只有当 ks ×ks 矩阵 $L_{k,n}^T\Delta\mathscr{G}_{k,n}$ 非奇异时，$T_{(n)k}$才有定义，本文中均假设这一条件成立。令 $\mathscr{T}_{k,n}$为如下矩阵：

$$\mathscr{T}_{k,n} = \begin{pmatrix} S_n & \mathscr{G}_{k,n} \\ L_{k,n}^T\Delta S_n & L_{k,n}^T\Delta\mathscr{G}_{k,n} \end{pmatrix}. \tag{4.91}$$

于是，近似值 $T^{(n)}{}_k$ 可表示为以下 Schur 补

$$T_k^{(n)} = (\mathscr{T}_{k,n} / L_{k,n}^T\Delta\mathscr{G}_{k,n}). \tag{4.92}$$

若令 $G_i(n) = \Delta S_{n+i-1}$; $i = 1,\ldots,k$ 且 $Y_i(n) = \Delta G_i(n)$; $i = 1,\ldots,k$，则得到块降秩外推（Bl-RRE）方法。在这种情况下，近似值 $T^{(n)}{}_k$ 为

$$T_{k,Bl-RRE}^{(n)} = (\mathscr{T}_{k,n} / \Delta\mathscr{G}_{k,n}^T\Delta\mathscr{G}_{k,n}). \tag{4.93}$$

若令 $G_i(n) = \Delta S_{n+i-1}$; $i = 1,\ldots,k$ 且 $Y_i(n) = G_i(n)$; $i = 1,\ldots,k$，则得到块最小多项式外推（Bl-MPE）方法。

最后，若 $G_i(n) = \Delta S_{n+i-1}$；$i = 1,\ldots,k$，且 $Y_i(n) = Y_i$；$i = 1,\ldots,k$（任意 N ×s 矩阵），则得到块修正最小多项式外推（Bl-MMPE）方法。

注意，若$N \times ks$ 矩阵$\Delta\mathscr{G}_{k,n}$ 为满秩，则由 Bl-RRE 方法产生的近似值是良定义的，而由 Bl-MPE 和 Bl-MMPE 方法生成的近似值可能不存在。由于 Bl-RRE 是一种正交投影方法，相应的广义残差满足如下最小化性质 $\|\widetilde{R}(T_{(n)k,Bl-RRE})\|_F = \min_{Z\in\widetilde{W}_{k,n}} \|\Delta S_n - Z\|_F$；其中 $\|X\|$ 表示 X 的 Frobenius 范数。

4.5.2 块拓扑 ε-算法（Bl-TET）在 [12], Brezinski 提出了标量 ε-算法 [63] 针对向量序列的一种推广，称为拓扑 ε-算法（TEA）。TEA 的主要思想是通过构造一列改进近似 $\mathrm{T}^{(n)}{}_k$，加速向量序列（S_n）的收敛，使其更快地收敛到（S^n）的极限或反极限。

在本小节中，我们聚焦于该算法的一个块版本，称为块拓扑 ε-变换（Bl-TET）。块版本特别适用于大型问题或具有多个右端项的系统所产生的向量序列，因为它能够同时处理多个分量。

设 (S_n) 是 $\mathbb{R}^N$ 中的一个向量序列。我们将其极限（或反极限）的近似值 $\mathrm{T}^{(n)}{}_k$ 构造为以下形式

$$T_k^{(n)} = S_n + \sum_{i=1}^{k} \Delta S_{n+i-1} A_i^{(n)}, \quad n \geq 0, \tag{4.94}$$

其中，$\Delta \mathrm{S}_n = \mathrm{S}_{n+1} - \mathrm{S}_n$ 是前向差分，而 $\mathrm{A}^{(n)}{}_i$ 是待确定的 $\mathrm{N} \times$ 个系数矩阵。项 k 表示变换的深度，它控制使用序列中多少个先前差分来加速收敛。

为定义广义残差，我们引入移位变换

$$\widetilde{T}_{k,j}^{(n)} = S_{n+j} + \sum_{i=1}^{k} \Delta S_{n+i+j-1} A_i^{(n)}, \quad j = 1, \ldots, k, \tag{4.95}$$

其中 $\tilde{}\mathrm{T}^{(n)}{}_{k\,0} = \mathrm{T}^{(n)}{}_k$。广义残差递归定义为

$$\widetilde{R}_j(T_k^{(n)}) = \widetilde{T}_{k,j}^{(n)} - \widetilde{T}_{k,j-1}^{(n)}, \quad j = 1, \ldots, k. \tag{4.96}$$

这些残差度量了连续变换后序列之间的差异，并在定义最优系数 $\mathrm{A}^{(n)}{}_i$ 时起着核心作用。

矩阵 $\mathrm{A}^{(n)}{}_i$ 的选取应满足：每个广义残差都与给定的 $\mathrm{N} \times \mathrm{s}$ 矩阵 Y 正交，而该矩阵定义了一个合适的约束子空间：

$$Y^T \widetilde{R}_j(T_k^{(n)}) = 0, \quad j = 1, \ldots, k. \tag{4.97}$$

这一条件确保 $\mathrm{T}^{(n)}{}_k$ 能够捕捉序列差分中最显著的分量，并消除与慢收敛相关的方向。

我们可以用紧凑的分块矩阵形式写出这些正交条件。定义

$$\mathscr{D}_{k,n} = \begin{bmatrix} Y^T \Delta^2 S_n & Y^T \Delta^2 S_{n+1} & \cdots & Y^T \Delta^2 S_{n+k-1} \\ Y^T \Delta^2 S_{n+1} & Y^T \Delta^2 S_{n+2} & \cdots & Y^T \Delta^2 S_{n+k} \\ \vdots & \vdots & \ddots & \vdots \\ Y^T \Delta^2 S_{n+k-1} & Y^T \Delta^2 S_{n+k} & \cdots & Y^T \Delta^2 S_{n+2k-2} \end{bmatrix}, \tag{4.98}$$

and

$$A_n = \begin{bmatrix} (A_1^{(n)})^T \\ \vdots \\ (A_k^{(n)})^T \end{bmatrix}, \quad Z_n = \begin{bmatrix} (Y^T \Delta S_n)^T \\ \vdots \\ (Y^T \Delta S_{n+k-1})^T \end{bmatrix}.$$

那么，(4.97) 等价于线性方程组

$$\mathscr{D}_{k,n} A_n = -Z_n. \tag{4.99}$$

如果块矩阵 $\mathscr{D}_{k,\,n}$是非奇异的，那么系数 $\mathrm{A}_1^{(n)}$、. . .、$\mathrm{A}^{(n)}{}_k$便可唯一确定，从而保证 $\mathrm{T}^{(n)}{}_k$ 存在且唯一。

块变换序列的最终表达式可写为

$$T_k^{(n)} = S_n - \Delta\mathscr{S}_{k,n}\,\mathscr{D}_{k,n}^{-1} Z_n, \tag{4.100}$$

其中 $\Delta\mathscr{S}_{k,\,n} = [\Delta \mathrm{S}n,\ \ldots,\ \Delta \mathrm{S}_{n+k-1}]$。等价地，$\mathrm{T}_{(n)\,k}$可表示为一个舒尔补：

$$T_k^{(n)} = (\mathscr{M}_{k,n}/\mathscr{D}_{k,n}), \quad \mathscr{M}_{k,n} = \begin{bmatrix} S_n & \Delta\mathscr{S}_{k,n} \\ Z_n & \mathscr{D}_{k,n} \end{bmatrix}. \tag{4.101}$$

注 4。

- 块拓扑 ε 算法推广了标量和拓扑 ε-算法，能够同时加速多个向量序列。
- 正交条件 (4.97) 确保在由 Y 定义的子空间中使残差最小化，这对于源自离散化微分方程或线性系统的序列至关重要。
- 块拓扑 ε 算法的递归实现可显著降低计算成本，因为它们避免了在每一步重复计算差分和矩阵求逆。
- 对于大规模问题、分块迭代方法，或具有自然分块结构的序列（例如来自多参数问题或张量方程的序列），分块形式尤其有利。

我们注意到，本文介绍的向量外推方法也已推广到张量序列 [7, 8, 9]，并广泛应用于彩色图像、视频和高光谱图像的补全与复原。张量外推方法也已用于多重 Google PageRank，参见 [10, 11]。

5 数值实验

5.1 一个偏微分方程示例 为评估向量外推方法 RRE、MPE、MMPE、拓扑ε-算法（TEA）以及安德森方法的性能，我们考虑一个源自反应–扩散模型的非线性椭圆边值问题。

令 $\Omega = (0,1) \times (0,1)$，并考虑以下非线性偏微分方程

$$-\Delta u(x,y) + u(x,y)^3 = f(x,y), \qquad (x,y) \in \Omega, \tag{5.1}$$

满足齐次狄利克雷边界条件

$$u(x,y) = 0, \qquad (x,y) \in \partial\Omega. \tag{5.2}$$

右端项 f 的选取使得精确解为

$$u^*(x,y) = \sin(\pi x)\sin(\pi y). \tag{5.3}$$

问题 (5.1)–(5.2) 采用标准二阶有限差分格式在均匀的 $\mathrm{p} \times \mathrm{p}$ 网格上离散。令 $\mathrm{h} = 1/(\mathrm{p}+1)$ 表示网格尺寸，$\mathrm{N} = \mathrm{p}^2$ 为内部网格点的数量。离散拉普拉斯算子由对应于五点差分模板的稀疏矩阵 $\mathrm{A} \in \mathbb{R}^{N\times N}$ 表示。离散后得到的非线性方程组可写为

$$Au + u^{\circ 3} = b, \tag{5.4}$$

其中，$\mathrm{u} \in \mathbb{R}^p$ 是节点值向量，$\mathrm{u}^{\circ 3}$ 表示 u 的三次哈达玛幂，即逐分量立方；$\mathrm{b} \in \mathbb{R}^p$ 是包含强迫项和边界条件的离散右端项。

为了应用向量外推方法，将非线性方程组（5.4）改写为一个不动点问题 m

$$u = G(u). \tag{5.5}$$

从初始猜测 $\mathrm{u}_0 = 0$ 开始，基本的皮卡迭代

$$u_{k+1} = G(u_k), \qquad k = 0, 1, \ldots, \tag{5.6}$$

生成序列 $\{\mathrm{u}_k\}$，其线性收敛于精确解 u^*。

通过向量外推进行加速。利用降秩外推（RRE）、最小多项式外推（MPE）、修正最小多项式外推（MMPE）、拓扑 ε-算法（TEA）和安德森方法，对皮卡迭代产生的序列进行加速。对于每种方法，均对连续迭代点 $\{\mathrm{u}_k\}$ 的块应用外推，以生成加速近似。

当相对非线性残差满足以下条件时，所有方法均停止

$$\frac{\|G(u_k) - u_k\|}{\|G(u_0) - u_0\|} \leq 10^{-6}. \tag{5.7}$$

该示例具有代表性，反映了偏微分方程离散化所产生的大规模非线性问题，并为比较向量外推技术的效率、稳健性和计算成本提供了有意义的基准。该判据独立应用于每种向量外推法。

表 5.1 报告了收敛所需的迭代次数、最终相对残差范数以及相应的 CPU时间。问题为通过二维偏微分方程的有限差分离散化得到的非线性椭圆问题，定义在 80×80 网格上（$\mathrm{N} = 6400$ 个未知量）。所有方法均采用重启模式计算，其中 $\mathrm{m} = 5$ 为每个周期的大小。

TABLE 5.1 向量外推方法与安德森加速在 80×80 偏微分方程网格上的比较。

方法	迭代	最终相对残差	CPU时间（秒）
RRE	16	9.8×10^{-7}	3.85
MPE	17	9.9×10^{-7}	4.10
MMPE	15	9.7×10^{-7}	4.85
TEA	18	9.6×10^{-7}	4.30
安德森	14	9.8×10^{-7}	4.05

正如预期，与基本的皮卡迭代相比，所有向量外推方法都显著减少了迭代次数。在多项式向量外推方法中，RRE 和 MPE 在收敛速度与计算成本之间实现了最佳平衡。向量安德森方法也取得了良好结果，优于 MPE、MMPE 和 TEA。总体而言，监测相对非线性残差可为非线性向量外推方法提供可靠且实用的停止准则。RRE 方法在求解某些偏微分方程中的其他应用见 [25]。

5.2 Google PageRank 在本例中，我们考虑 amazon-0302 数据集，它表示亚马逊产品的共同购买网络。由于其规模贴近实际且结构稀疏，该数据集被广泛用作 PageRank 等大规模图算法的基准。该数据集被建模为有向图 $\mathrm{G} = (\mathrm{V},\mathrm{E})$，其中每个节点对应一种产品，从节点 i 指向节点 j 的有向边表示产品 j 经常与产品 i 被共同购买。整个网络构成一个连通分量，其中包含一个由 241.761 个节点和 1.131.217 条边组成的强连通核心。目标是计算 PageRank向量 $\mathrm{x} \in \mathbb{R}^N$，其中 **N=262,111**，使得

$$x = \alpha P x + (1 - \alpha) v,$$

其中，P 是网络的列随机转移矩阵，$\alpha \in (0, 1)$ 是阻尼因子（通常为 $\alpha = 0.85$），v 是个性化向量（通常为均匀分布），N 是网络中的节点数。

PageRank向量表示网络上带有随机传送的随机游走的平稳分布，并根据产品的相对重要性对其进行排名。

我们采用两种方法计算网页排名向量：经典的幂方法，通过迭代应用 $\mathbf{x}_{k+1} = \alpha \mathbf{P}\mathbf{x}_k + (1-\alpha)\mathbf{v}$ 直至收敛；以及艾特肯的 Δ^2 过程，将其应用于幂方法生成的序列，以加速收敛；关于艾特肯的 Δ^2 在 Google PageRank 中的应用，详见 [29] 。两种方法的停止准则均基于相对残差范数

$$\frac{\|x_{k+1} - x_k\|}{\|x_{k+1}\|} \le 10^{-6}.$$

表 5.2 采用和不采用艾特肯加速时网页排名计算的性能比较。

方法	迭代	最终相对残差	CPU时间（秒）
幂方法（经典）	245	9.2×10^{-7}	18.5
艾特肯的 Δ^2	62	8.7×10^{-7}	7.8

如表 5.2 所示，幂方法需要迭代 245 次才能满足停止准则，而艾特肯过程仅迭代 62 次便收敛，并达到相同的相对残差容差。CPU 时间从 18.5 秒缩短至 7.8 秒，证明了加速的有效性。两种方法得到的最终 PageRank向量均与网络的真实平稳分布一致。两种方法的最终相对残差范数均低于容差 10^{-6}，表明算法成功收敛。这些结果符合外推方法在大型稀疏图上的预期表现。

5.3 Fredholm 积分的 ε算法 考虑如下第二类 Fredholm 积分方程

$$u(x) = f(x) + \lambda \int_0^1 K(x,t)\,u(t)\,dt, \tag{5.8}$$

其中，K(x,t) 是给定的核，f 是已知函数，而 $\lambda \in \mathbb{R}$ （或 $\mathbb{C}$）是一个参数。引入积分算子

$$(\mathcal{K}u)(x) = \int_0^1 K(x,t)\,u(t)\,dt,$$

方程 (5.8) 可写成如下算子形式

$$u = f + \lambda \mathcal{K} u. \tag{5.9}$$

Star从初始猜测 $\mathbf{u}^{(0)}$(x) 出发，诺伊曼迭代（也称为皮卡迭代）定义为 by

$$u^{(k+1)}(x) = f(x) + \lambda \int_0^1 K(x,t)\,u^{(k)}(t)\,dt, \qquad k = 0,1,2,\dots \tag{5.10}$$

或者，等价地，

$$u^{(k+1)} = f + \lambda \mathcal{K} u^{(k)}. \tag{5.11}$$

在适当的假设下，例如当 $|\lambda|\,\|\mathcal{K}\| < 1$ 时，序列 $\{\mathbf{u}^{(k)}\}_{k\ge 0}$ 收敛到方程 (5.8) 的唯一解 u。在这种情况下，极限可表示为诺伊曼级数

$$u = \sum_{j=0}^{\infty} (\lambda \mathcal{K})^j f. \tag{5.12}$$

对积分算子进行离散化（例如，采用求积规则）后，可得到如下形式的线性方程组

$$(I - \lambda K_h)u = f, \tag{5.13}$$

其中 K_h 表示离散化算子。此时，诺伊曼（或皮卡德）迭代变为不动点迭代

$$u^{(k+1)} = f + \lambda K_h u^{(k)}, \qquad k = 0, 1, 2, \ldots \tag{5.14}$$

它生成诺伊曼级数的部分和，并作为自然起点，用于收敛加速技术，例如标量 ε-算法；本实验将采用该算法。这里我们设

$$K(x,t) = e^{-|x-t|}, \qquad f(x) = \sin(\pi x), \qquad \lambda = 0.5.$$

该问题通过梯形公式在 $[0,1]$ 上的均匀网格中离散为 $N = 500$ 个点。由此得到一个稠密线性方程组，其解通过诺伊曼不动点迭代近似求得。为加速收敛，将 Wynn 的 ε-算法逐分量应用于迭代点序列。两种方法的停止准则均基于连续迭代值的相对差，

$$\frac{\|u^{(k+1)} - u^{(k)}\|_2}{\|u^{(k)}\|_2} \le 10^{-6}. \tag{5.15}$$

表 5.3 报告了满足停止准则所需的迭代次数，以及最终的相对残差范数。

表 5.3 $N = 500$时的收敛性结果

方法	迭代	最终相对残差	CPU时间（秒）
诺伊曼迭代	61	9.3×10^{-7}	28.4
ε-算法	15	3.7×10^{-8}	11.6

The r 结果表明，Wynn 的 ε-算法显著提高了收敛速率。所得 CPU - 时间 也表明，使用 ε-算法加速收敛具有优势。
尽管 ε-算法的递归外推过程会带来额外开销，但迭代次数的减少足以弥补这一成本。因此，总 CPU 时间得以缩短。

随着离散化参数 N 增大，由于离散化积分算子所产生的稠密矩阵-向量乘积，每次迭代的计算开销也随之增加。因此，对于大规模问题，加速技术的优势会更加明显。本次实验中 $N = 500$ 的结果证实，Wynn 的 ε-算法在加速第二类Fredholm积分方程的不动点迭代收敛方面非常有效，显著减少了迭代次数和计算时间。关于标量 ε-算法在 Fredholm 积分中的更多示例和应用，请参阅优秀论文 [21]。

6 结论 标量外推方法（例如理查森外推、艾特肯的 Δ^2 过程、Shanks变换、Wynn 的 ε-算法以及 ρ和 θ 算法）构成了一类用于加速数值序列收敛的基础方法。每种方法都基于系统性的方法，旨在减少或消除收敛或缓慢收敛序列中的主导阶误差项，从而以更少的迭代次数获得更精确的序列极限近似值。

理查森外推利用采用不同离散步长计算出的近似值来消去主导误差项，因此在数值微分、积分以及微分方程求解中尤为有效。艾特肯的 Δ^2 过程为加速线性收敛序列提供了一种简单且计算效率高的方法，而 Shanks变换则通过使用行列式推广了这一思想，从而提高更复杂级数的收敛速度。Wynn 的 ε-算法进一步拓展了

这些概念，提供了一种能够处理具有振荡或非线性收敛行为的序列的递归方案。标量外推方法通过提升迭代计算的效率和精度，在数值分析中发挥着至关重要的作用。向量序列变换，包括 MPE、RRE 和 MMPE 等多项式外推方法，以及向量和块 ε-算法，为加速迭代数值方法中产生的序列收敛提供了强有力的工具。在处理求解线性或非线性方程组所产生的向量序列时，这些技术尤其有价值，因为直接方法的计算成本可能很高，甚至不可行。对于这类向量序列，安德森方法也是一种适用的数值方法。我们还注意到，向量外推方法已推广到矩阵和张量情形。对向量序列变换技术的研究与实现，是现代数值分析的重要组成部分。这些技术为改善收敛、优化计算资源以及在广泛的科学与工程应用中实现更高精度提供了严谨的框架。

参考文献

[1] A. C. Aitken，《伯努利代数方程数值解法》，《爱丁堡皇家学会会刊》，46: 289–305，1926。[2] 花拉子米，《代数》，巴格达，825。（有拉丁文译本。）[3] D. G. Anderson，《非线性积分方程的迭代过程》，ACM 会刊（JACM），12(4):547–560，1965。[4] M. L. Baart，利用自相关在含噪病态最小二乘问题中确定伪秩，《IMA 数值分析期刊》，2:41–247，1882。[5] G. A. Baker Jr. 和 P. R. Graves-Morris，《Pad´e 逼近》，Addison-Wesley，雷丁，1981。[6] J. Beuneu，《用于线性问题的投影-最小化方法》，RAIRO 数值分析，17:221–248，1983。[7] F.P.A. Beik、A. ElIchi、K. Jbilou 和 R. Sadaka，张量外推方法及其应用，《数值算法》，87:1421–1444，2021。[8] A. Bentbib、K Jbilou、R Tahiri，N 模最小张量外推方法，《数值算法》95(2):665-691，2024。[9] A.H. Bentbib、K Jbilou 和 R. Tahiri，Hosvd-tmpe：一种用于多维序列的外推方法，Calcolo，61(2)，27，2024。[10] M. Boubekraoui、A. Bentbib 和 K. Jbilou，多线性 PageRank 的外推方法，《数值算法》，2024。[11] M Boubekraoui、A. Bentbib、K Jbilou，用于多线性 PageRank 计算的向量 Aitken 外推法，《应用数学与计算期刊》69 (1)，1145-1172，2023。[12] C. Brezinski，Shanks 变换、Padé 表和 ε 算法的推广，Calcolo，：–。[13] C. Brezinski，《外推算法：理论与实践》，North-Holland，1977。[14] C. Brezinski，关于 εand ρ-算法的研究。《数值数学》，17:153-162，1971。[15] C. Brezinski，《Pad´e 型近似与一般正交多项式》，Birkh¨auser，1980。[16] C. Brezinski，关于 Henrici 非线性方程方法的讨论。数值分析与计算复分析研讨会，苏黎世，未发表，1983年8月。

[17] 273 727 730, 1971 C. Brezinski, 1991

[18] 对数收敛序列的加速，巴黎科学院报告，A: - C. Brezinski，《收敛加速：技术与应用》，North-Holland，。[19] C. Brezinski 和 M. Redivo Zaglia，《外推方法：理论与实践》，North-Holland，阿姆斯特丹，1991。[20] C. Brezinski 和 M. Redivo Zaglia，用于加速向量空间中序列的简化拓扑 ε-算法，《SIAM 科学计算期刊》第 36 卷(5): A2227–A2247，2014。[21] C. Brezinski 和 M. Redivo Zaglia，数值求解非线性 Fredholm 积分方程的外推方法（与 M. Redivo Zaglia 合著）。积分方程应用期刊，31:29-57，2019。[22] C. Brezinski、S. Cipolla、M. Redivo Zaglia 和 Y. Saad，非线性方程组的 Shanks 型与 Anderson 型加速技术，《IMA 数值分析期刊》，42(4):3058–3093，2022。[23] C. Brezinski、M. Redivo Zaglia 和 Y. Saad，Shanks 序列变换与 Anderson 加速，《SIAM评论》，60(3): 646–669，2018。[24] S. Cabay 和 L. W. Jackson，一种用于求向量序列极限与反极限的多项式外推方法，《SIAM 数值分析期刊》，13: 734–752，1976。[25] S. Duminil、H. Sadok 和 D. B. Szyld，结合降秩外推的非线性 Schwarz 迭代，《应用数值数学》，94:209–221，2015。[26] J. A. Eddy，向量序列的加速收敛，《计算与应用数学杂志》，4 138，1979 131: –。[27] W. Gander、G. H. Golub 和 D. Gruntz，通过外推求解线性方程，载于《超级计算》，J. Kowalik 编，NATO 高级科学研究所系列 F：计算机系统科学 62，Springer，柏林，1989，第279–293页。[28] G. H. Golub 和 C. F. Van Loan，《矩阵计算》，第3版，约翰斯·霍普金斯大学出版社，巴尔的摩，1996。[29] S. D. Kamvar、T. H. Haveliwala、C. D. Manning 和 G. H. Golub，用于加速 PageRank 计算的外推方法，载于《第十二届国际万维网大会论文集》（WWW 2003），ACM 出版社，2003，第261–270页。[30] P. R. Graves-Morris，向量序列的外推方法，《数值数学》，61:475–487，1992。[31] P.R. Graves-Morris、D.E. Roberts 和 A. Salam，《ε 算法及相关主题》，《计算与应用数学杂志》，122(1–2):51–80，2000。

[32] P. R. Graves-Morris 和 C. D. Jenkins，向量值有理插值函数，III，《构造性逼近》，2:263–289，1986。[33] P. C. Hansen，正则化工具：用于分析和求解离散不适定问题的 MATLAB 软件包，《数值算法》，6:1–35，1994。[34] P. C. Hansen，《秩亏与离散不适定问题》，工业与应用数学学会，费城，1998。[35] P. Henrici，《数值分析基础》，威立，纽约，1964。[36] N. J. Higham 和 N. Strabic，用 Anderson 加速交替投影法以计算最近相关矩阵，《数值算法》，72:1021–1042，2016。[37] A. Heyting，《非交换乘法数域中的线性方程理论》，《数学年刊》，98，465–490，1927。[38] K. Jbilou，《外推与投影方法：向量序列的应用》，博士论文，里尔科学与技术大学，1988。[39] K. J bilou 和 H. Sadok，关于向量外推方法及相关不动点迭代的一些结果，《计算与应用数学杂志》，36:385–398，1991。[40] K. Jbilou 和 H. Sadok，线性系统若干向量外推方法的分析，《数值数学》，70 89，1995 73: –。[41] K. Jbilou 和 H. Sadok，修正最小多项式外推法的 LU实现，《IMA 数值分析期刊》，19:549–561，1999。[42] K. Jbilou 和 H. Sadok，向量外推方法：应用与数值比较，《计算与应用数学杂志》，122:149–165，2000。[43] K. Jbilou、L. Reichel、H. Sadok，用于线性离散不适定问题的 TSVD 增强型向量外推，《数值算法》，51(2):195-208，2009。[44] K. Jbilou 和 A. Messaoudi，分块外推法及其应用，《应用数值数学》，106 164，2016 154: -。[45] P.J. Laurent，《数值分析中的外推过程研究》，国家数学科学博士论文，格勒诺布尔大学，1964。[46] J. B. McLeod，关于 ε-算法的注记，《计算》，7:17-24，1971。[47] M. Me`sina，用于迭代求解 x = Ax + f 的收敛加速，《计算机在应用力学与工程中的方法》，10:165–173，1977。[48] O. Ore，非交换域中的线性方程，《数学年刊》，32:463–477，1931。[49] B. P. Pugatchev，迭代过程的收敛加速及一种求解非线性方程组的方法，《苏联计算数学与数学物理》，17:199–207，1978。[50] L. F. Richardson，《通过有限差分近似求解涉及微分方程的物理问题》，《英国皇家学会哲学汇刊 A 辑》，210:307–357，1911。[51] L. F. Richardson，《延迟趋近极限法》，《英国皇家学会哲学汇刊A辑》，226:299–361，1927。[52] H. Rutishauser，《商差分算法》，Birkh¨auser 出版社，巴塞尔，1957。[53] Y. Saad 和 M. H. Schultz，GMRES：求解非对称线性系统的广义最小残差算法，《SIAM科学与统计计算杂志》，7:856–869，1986。
[54] 2 2003, Y. Saad，《稀疏线性系统的
[55] 迭代方法》，第 nd 版，工业与应用数学学会，费城，PA，H. Sadok，关于 Henrici 变换加速向量序列的研究，《计算与应用数学杂志》，29:101-110，1990。[56] A. Salam，向量 ε-算法的代数方法，《数值算法》，11:327-337，1996。[57] D. Shanks，《发散与慢收敛序列的非线性变换》，《数学与物理杂志》，34:1–42，1955。[58] A. Sidi，最小多项式外推法和降秩外推法的高效实现，《计算与应用数学杂志》，36:305–337，1991。[59] A. Sidi，《实用外推法》，剑桥大学出版社，2003。[60] D. A. Smith、W. F. Ford 和 A. Sidi，向量序列的外推方法，《SIAM评论》，29 (1987)，199–233；更正，《SIAM评论》，30:623–624，1988。[61] H. F. Walker 和 P. Ni，不动点迭代的 Anderson 加速，《SIAM 数值分析期刊》，49 1735，2011 1715: –。
[62] 1981 J. Wimp，《序列变换及其
[63] 应用》，学术出版社，纽约，P. Wynn，关于计算 $e_n^{(m)}(S)$ 变换的一种方法，《数学表格及其他计算辅助工具》，10:91–96，1956。[64] P. Wynn，迭代向量与矩阵问题的加速技术，《数学计算》，16:301-322，1962。[65] P. Wynn，ε 算法的收敛性与稳定性，《SIAM 数值分析期刊》，3:91–122，1966。