ЖУРНАЛ ВЫЧИСЛИТЕЛЬНОЙ МАТЕМАТИКИ И МАТЕМАТИЧЕСКОЙ ФИЗИКИ, 2021, том 61, № 5, с. 33–53

ОБЩИЕ ЧИСЛЕННЫЕ МЕТОДЫ

УДК 519.61

МЕТОДЫ ЭКСТРАПОЛЯЦИИ ШЭНКСА И ИХ ПРИЛОЖЕНИЯ[1)]

© 2021 г. К. Брезински[1,*], М. Редиво-Дзалья[2,]**

[1] *Université de Lille, CNRS, UMR 8524 – Laboratoire Paul Painlevé, F-59000 Lille, France*

[2] *Università degli Studi di Padova, Dipartimento di Matematica "Tullio Levi-Civita", Via Trieste 63, 35121-Padova, Italy*

**e-mail: Claude.Brezinski@univ-lille.fr*

***e-mail: Michela.RedivoZaglia@unipd.it*

Поступила в редакцию 24.11.2020 г.
Переработанный вариант 24.11.2020 г.
Принята к публикации 14.01.2021 г.

Когда последовательность или серия скаляров, векторов, матриц, тензоров медленно сходится к своему пределу, она может быть преобразована путем преобразования последовательности в новую последовательность или набор новых последовательностей, которые при некоторых предположениях сходятся быстрее к тому же пределу. Такое преобразование можно применять также к расходящимся последовательностям или рядам, обеспечивая тем самым их аналитическое продолжение. Преобразование Шэнкса — хорошо известное преобразование последовательностей для ускорения сходимости в случае скаляров. В этом обзоре мы объясняем его разработку, различные расширения и реализацию. Несколько приложений иллюстрируют его эффективность. Библ. 51. Фиг. 11. Табл. 2.

Ключевые слова: методы ускорения, преобразования последовательностей, преобразование Шэнкса, аппроксимация Паде, тензор.

DOI: 10.31857/S0044466921050069

1. ВВЕДЕНИЕ

В настоящее время экстраполяционные методы подробно исследованы [1]–[7]. Они направлены на ускорение сходимости последовательностей или различных итерационных процессов. Однако области применения методов экстраполяции не ограничены только данной целью. Методы экстраполяции можно использовать в различных аппроксимационных задачах, при решении систем линейных и нелинейных уравнений, при вычислении матричных функций, и это лишь немногие из их потенциальных приложений. Однако эти методы не очень широко известны в сообществе прикладных математиков. В данной обзорной работе мы представляем один из методов экстраполяции, а именно – преобразование Шэнкса [8], так как согласно [9]: "Преобразование Шэнкса, пожалуй, является лучшим универсальным методом ускорения сходимости последовательностей". Мы предложим подробное описание данного метода, его различных расширений и продемонстрируем, что он может быть очень полезен во многих ситуациях.

Когда последовательность чисел, векторов, матриц или тензоров (S_n) медленно сходится к своему пределу S, при стремящемся к бесконечности n, и когда человек не имеет доступа к процессу, генерирующему значения исходной последовательности, этот процесс может быть преобразован путем преобразования последовательности T в новую последовательность (T_n) или набор новых последовательностей $\{(T_k^{(n)})\}$, которые, согласно некоторым предположениям, будут сходиться быстрее к тому же пределу, т.е. $\lim\limits_{n\to\infty} \|T_n - S\| / \|S_n - S\| = 0$.

Существует много таких преобразований последовательности, но общая идея, лежащая в их основе, – это интерполяция с последующей экстраполяцией. Пусть $\mathcal{K}(T)$ – множество последовательностей, члены которых зависят от $k+1$ произвольных неизвестных параметров. Предпо-

[1)]Работа Клода Брезински выполнена при финансовой поддержке Labex CEMPI (ANR-11-LABX-0007-01). Работа Микелы Редиво-Дзальи выполнена при частичной финансовой поддержке проекта университета Падуи, Project 2019 no. DOR1903575/19. Микела Редиво-Дзалья – член группы INdAM Research group GNCS.

ложим, что для каждой последовательности $(t_n) \in \mathcal{K}(T)$, ее предел t может быть вычислен с использованием $k+1$ ее членов. Множество $\mathcal{K}(T)$ назовем *ядром* преобразования T.

Пусть теперь (S_n) – последовательность, подлежащая преобразованию. Предполагается, что она не принадлежит ядру преобразования. Зафиксируем целое число n. Проинтерполируем члены $S_n, \ldots, S_{n+k}$ последовательностью $(t_n) \in \mathcal{K}(T)$ такой, что $S_{n+i} = t_{n+i}$ для $i = 0, \ldots, k$. По построению данные условия для интерполянта позволяют вычислить предел t последовательности (t_n). Такие шаги порождают экстраполяционный процесс, описанный, например, в книге Марчука и Шайдурова, см. [3]. Заметим, что число t – предел интерполирующей последовательности $(t_n) \in \mathcal{K}(T)$, но оно не обязано быть пределом исходной последовательности (S_n), если она не принадлежит множеству $\mathcal{K}(T)$. Таким образом, с изменением n будет изменяться и предел t интерполяционной последовательности ядра, и мы обозначим его через T_n. В результате исходная последовательность (S_n) через ядро преобразования T порождает новую последовательность (T_n).

2. МЕТОД ЭКСТРАПОЛЯЦИИ ШЭНКСА

Среди преобразований для ускорения сходимости скалярных последовательностей преобразование Даниэля Шэнкса, пожалуй, является одним из лучших. Впервые оно было предложено в 1949 г. (см. [10]), но опубликовано лишь в 1955 г. (см. [8]). Его ядро – это множество последовательностей, удовлетворяющих для всех n однородному линейному разностному уравнению порядка k:

$$a_0(t_n - t) + a_1(t_{n+1} - t) + \ldots + a_k(t_{n+k} - t) = 0 \tag{1}$$

при условии $a_0 \times a_k \neq 0$. Без потери общности можно предположить, что выполнено условие нормировки $a_0 + a_1 + \ldots + a_k = 1$, и, таким образом, мы имеем

$$t = a_0 t_n + \cdots + a_k t_{n+k}, \quad n = 0, 1, \ldots . \tag{2}$$

2.1. Преобразование Шэнкса

Запись уравнения (2) для индексов $n, \ldots, n+k$ позволяет определить коэффициенты a_i как решение системы линейных уравнений, а затем вычислить t при помощи (2). Конечно, если последовательность (S_n) не принадлежит ядру, все еще возможно записать и решить систему уравнений, определяющую коэффициенты a_i и далее из уравнений (2) определить так называемое преобразование Шэнкса, обычно обозначаемое $e_k(S_n)$, так как результат будет зависеть и от n, и от k. В результате получим

$$e_k(S_n) = \begin{vmatrix} S_n & S_{n+1} & \cdots & S_{n+k} \\ \Delta S_n & \Delta S_{n+1} & \cdots & \Delta S_{n+k} \\ \vdots & \vdots & & \vdots \\ \Delta S_{n+k-1} & \Delta S_{n+k} & \cdots & \Delta S_{n+2k-1} \end{vmatrix} \Bigg/ \begin{vmatrix} 1 & 1 & \cdots & 1 \\ \Delta S_n & \Delta S_{n+1} & \cdots & \Delta S_{n+k} \\ \vdots & \vdots & & \vdots \\ \Delta S_{n+k-1} & \Delta S_{n+k} & \cdots & \Delta S_{n+2k-1} \end{vmatrix}, \tag{3}$$

где Δ – разностный оператор дифференцирования “справа”, определяемый $\Delta S_i = S_{i+1} - S_i$ для всех i.

Решая разностные уравнения (1) через запись характеристического многочлена и поиск всех его корней, мы получаем следующий результат [11] (см. также [12] с более детальными комментариями).

Теорема 1. *Необходимым и достаточным условием для $e_k(S_n) = S$ при всех n является условие, что последовательность (S_n) удовлетворяет*

$$S_n - S = \sum_{i=1}^{p} A_i(n) r_i^n + \sum_{i=p+1}^{q} [B_i(n)\cos(b_i n) + C_i(n)\sin(b_i n)]e^{\omega_i n} + \sum_{i=0}^{m} c_i \delta_{in},$$

где $r_i \neq 0$ *для* $i = 1, \ldots, p$, A_i, B_i *и* C_i *— многочлены порядка* n *такие, что если* d_i *равняется степени* A_i *плюс* 1 *для всех* $i = 1, \ldots, p$ *и максимуму из степеней* B_i *и* C_i *плюс* 1 *для* $i = p = 1, \ldots, q$, *то*

$$m + \sum_{i=1}^{p} d_i + 2\sum_{i=p+1}^{q} d_i = k - 1,$$

с обязательным условием, что $m = -1$, *если нет члена* δ_{in} (*символ Кронекера*).

Коэффициенты полиномов A_i, B_i и C_i определяются начальными условиями в уравнении (1). Из этого результата видно, что многие итерационные методы, используемые в численном анализе и прикладной математике, порождают последовательности такого вида или близкие к нему, что объясняет важность метода экстраполяции Шэнкса.

2.2. ε*-Алгоритм*

Несколько преобразований последовательности могут быть выражены как отношение двух определителей. Таким образом, для нахождения преобразованных членов исходной последовательности должны быть получены алгоритмы, избегающие численного вычисления этих определителей. Рекурсивная процедура, ε-алгоритм, для реализации преобразования Шэнкса была предложена в 1956 г. Питером Винном [13]. Организация этой процедуры очень проста:

$$\varepsilon_{k+1}^{(n)} = \varepsilon_{k-1}^{(n+1)} + (\varepsilon_k^{(n+1)} - \varepsilon_k^{(n)})^{-1}, \quad k, n = 0, 1, \ldots \tag{4}$$

при $\varepsilon_{-1}^{(n)} = 0$ и $\varepsilon_0^{(n)} = S_n$ для $n = 0, 1, \ldots$, и это выполнено для всех k и n,

$$\varepsilon_{2k}^{(n)} = e_k(S_n) \quad \text{и} \quad \varepsilon_{2k+1}^{(n)} = 1/e_k(\Delta S_n).$$

Таким образом, значения $\varepsilon_{2k+1}^{(n)}$ — это промежуточные результаты и не представляют интереса для наших целей.

Значения $\varepsilon_k^{(n)}$ обычно изображаются в виде двумерного ε-массива, где нижний индекс k сохраняется вдоль столбцов, а верхний индекс n сохраняется вдоль нисходящей диагонали:

$$\begin{array}{ccccccc}
\varepsilon_{-1}^{(0)} = 0 & & & & & & \\
 & \varepsilon_0^{(0)} = S_0 & & & & & \\
\varepsilon_{-1}^{(1)}=0 & & \varepsilon_1^{(0)} & & & & \\
 & \varepsilon_0^{(1)}=S_1 & & \varepsilon_2^{(0)} & & & \\
\varepsilon_{-1}^{(2)}=0 & & \varepsilon_1^{(1)} & & \varepsilon_3^{(0)} & & \\
 & \varepsilon_0^{(2)}=S_2 & & \varepsilon_2^{(1)} & & \ddots & \\
\varepsilon_{-1}^{(3)} = 0 & & \varepsilon_1^{(2)} & & \varepsilon_3^{(1)} & & \\
\vdots & \varepsilon_0^{(3)} = S_3 & & \varepsilon_2^{(2)} & & \ddots & \\
\vdots & \vdots & \varepsilon_1^{(3)} & & \varepsilon_3^{(2)} & & \\
\vdots & \vdots & \vdots & \varepsilon_2^{(3)} & & \ddots & \\
\vdots & \vdots & \vdots & \vdots & \varepsilon_3^{(3)} & & \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots &
\end{array}$$

В процедуре (4) ε-алгоритма обрабатываются значения, расположенные в четырех вершинах следующего ромба:

$$\begin{array}{ccc}
 & \varepsilon_k^{(n)} & \\
\nearrow & & \searrow \\
\varepsilon_{k-1}^{(n+1)} & & \varepsilon_{k+1}^{(n)}. \\
\searrow & & \nearrow \\
 & \varepsilon_k^{(n+1)} &
\end{array}$$

В случае, когда в знаменателе (4) два значения оказались равными, вычислительная процедура должна быть остановлена (данное явление принято называть *обвалом*). Но если значения в знаменателе почти равны, возникает неустойчивость при вычислении очередного члена (называемая *почти–обвалом*). Для разрешения этих недостатков алгоритма Винн, см. [14], также предложил конкретные правила для скалярного ε-алгоритма, которые позволяют повысить устойчивость вычислений.

2.3. Аппроксимации Паде

ε-Алгоритм, описанный в п. 2.2, связан с известными аппроксимациями Паде, используемыми, в частности, для суммирования сходящихся и расходящихся рядов [6], [15].

Пусть $f(z) = \sum_{i=0}^{\infty} c_i z^i$ – формальный степенной ряд. Аппроксимация Паде для f – это рациональная функция, разложение в степенной ряд которой по возрастающим степеням z является близким к ряду f настолько, насколько это возможно, т.е. с точностью до суммы $p+q$ включительно, где p – степень его числителя, а q – степень q его знаменателя.

Обозначим такую аппроксимацию через

$$[p/q]_f(z) = (a_0 + a_1 z + \ldots + a_p z^p) \Big/ (b_0 + b_1 z + \ldots + b_q z^q),$$

где коэффициенты можно получить из следующих выражений:

$$\begin{aligned}
a_0 &= c_0 b_0,\\
a_1 &= c_1 b_0 + c_0 b_1,\\
&\vdots \quad \vdots\\
a_p &= c_p b_0 + c_{p-1} b_1 + \ldots + c_{p-q} b_q,\\
0 &= c_{p+1} b_0 + c_p b_1 + \ldots + c_{p-q+1} b_q,\\
&\vdots \quad \vdots\\
0 &= c_{p+q} b_0 + c_{p+q-1} b_1 + \ldots + c_p b_q
\end{aligned}$$

при условии, что $c_i = 0$ для $i < 0$.

Линейная система, порожденная последними q уравнениями содержит $q+1$ неизвестную $b_0, \ldots, b_q$ и, следовательно, существует ее нетривиальное решение. Положив $b_0 = 1$, мы можем получить значения $b_1, \ldots, b_q$, решая линейную систему, предполагая ее невырожденность. Зная значения b_i, из первых $p+1$ уравнений мы можем вычислить a_i.

Выполнено

$$[n+k/k]_f(z) = \begin{vmatrix} z^k S_n(z) & z^{k-1} S_{n+1}(z) & \cdots & S_{n+k}(z) \\ c_{n+1} & c_{n+2} & \cdots & c_{n+k+1} \\ \vdots & \vdots & & \vdots \\ c_{n+k} & c_{n+k+1} & \cdots & c_{n+2k} \end{vmatrix} \Bigg/ \begin{vmatrix} z^k & z^{k-1} & \cdots & 1 \\ c_{n+1} & c_{n+2} & \cdots & c_{n+k+1} \\ \vdots & \vdots & & \vdots \\ c_{n+k} & c_{n+k+1} & \cdots & c_{n+2k} \end{vmatrix}$$

при $S_n(z) = \sum_{i=0}^{n} c_i z^i$.

Применяя ε-алгоритм к частичным суммам f при $\varepsilon_0^{(n)} = S_n(z)$, можно заметить, сопоставив предыдущую формулу с (3), что

$$e_k(S_n) = [n+k/k]_f(z).$$

Аппроксиманты со степенью знаменателя большей, чем степень числителя, могут быть получены через применение ε-алгоритма к частичным суммам обратного ряда g функции f, формально определяемого через $f(z)g(z) = 1$ и через свойство $[p/q]_f(z)[q/p]_g(z) = 1$. Ряд g существует только тогда, когда $c_0 \neq 0$.

3. НЕСКАЛЯРНЫЕ ПОСЛЕДОВАТЕЛЬНОСТИ

Преобразование Шэнкса и ε-алгоритм можно обобщить на случай нескалярных последовательностей, как мы увидим ниже.

3.1. Случай векторных и матричных последовательностей

В работе [16] Винн предложил расширение своего ε-алгоритма для случая векторных и матричных последовательностей. В данном случае правила (4) остаются прежними, а степень -1 соответствует получению обратной матрицы. Ядро преобразования снова можно записать в форме (1). Позже А. Салам (см. [17]) предложил доказательство, что данный результат все еще верен, если коэффициенты a_i являются квадратными матрицами, и произведение возникает либо слева, либо справа. Алгоритм может быть обобщен и для случая прямоугольных матриц через замену операции обращения матриц вычислением псевдообратных, но результат, касающийся ядра, не остается верным. Подробнее случай ε-алгоритма Винна для матричных последовательностей рассмотрен в [18].

В случае векторных последовательностей Винн определил обращение вектора u через $u^{-1} = u/(u,u)$. Векторы $\varepsilon_{2k}^{(n)}$ тоже могут быть выражены, как и в скалярном случае, через отношение определителей, но размерности $2k+1$ вместо исходной $k+1$, как было доказано П.Р. Грейвс-Моррисом и К. Дженкинсом [9], [19]. Применив алгебру Клиффорда, Дж.Б. МакЛеод доказал, что ядро векторного ε-алгоритма также содержит векторы формы (1) [20], где коэффициенты a_i являются вещественными числами. Позже А. Салам доказал, что ядром векторного ε-алгоритма является непосредственно (1), но с коэффициентами a_i, принадлежащими алгебре Клиффорда, ассоциированной с $\mathbb{R}^p$ [21], [22], где p – размерность векторов последовательности S_n. Аналогичная (3) формула по-прежнему имеет место, но определители необходимо заменить на конструкторы (designants), которые обобщают их в некоммутативной алгебре [17], [23].

3.2. Топологическое преобразование Шэнкса

Столкнувшись с трудностями, возникающими в алгебраических теориях для векторных и матричных ε-алгоритмов, К. Брежинский решил заново заняться этой задачей с начальных этапов [24]. Пусть (S_n) – последовательность элементов топологического векторного пространства E над $\mathbb{C}$, или просто над $\mathbb{R}$ (топология необходима, чтобы уметь рассматривать вопросы сходимости). Задачей было построить преобразование последовательности $(S_n) \mapsto \{(e_k(S_n)\}$ такое, что для всех последовательностей (S_n), удовлетворяющих (1) с $t = S$, при фиксированном k и с коэффициентами $a_i \in \mathbb{C}$, $e_k(S_n) = a_0S_n + a_1S_{n+1} + \ldots + a_kS_{n+k} = S \in E$ для всех n. Записав (1) для индексов n и $n+1$ и вычислив разность, получим

$$a_0\Delta S_n + a_1\Delta S_{n+1} + \ldots + a_k\Delta S_{n+k} = 0 \in E.$$

Следующей задачей стало вычислить коэффициенты a_i. Для этого полученное выражение было преобразовано в выражение над $\mathbb{C}$. Пусть y является элементом алгебраического сопряженного пространства E^* для E, являющегося пространством линейных функционалов над E, и обозначим через $\langle\cdot,\cdot\rangle$ соотношение двойственности между E^* и E. Применение этого соотношения к выражениям, описанным выше, позволяет получить

$$a_0\langle y, \Delta S_n\rangle + a_1\langle y, \Delta S_{n+1}\rangle + \ldots + a_k\langle y, \Delta S_{n+k}\rangle = 0 \in \mathbb{C}. \tag{5}$$

Записав это выражение для индексов от n до $n+k-1$, добавив условие нормировки и решив соответствующую линейную систему для a_i, мы получим (первое) *топологическое преобразование Шэнкса*, определяемое через:

$$\hat{e}_k(S_n) = \frac{\begin{vmatrix} S_n & S_{n+1} & \cdots & S_{n+k} \\ \langle y, \Delta S_n\rangle & \langle y, \Delta S_{n+1}\rangle & \cdots & \langle y, \Delta S_{n+k}\rangle \\ \vdots & \vdots & & \vdots \\ \langle y, \Delta S_{n+k-1}\rangle & \langle y, \Delta S_{n+k}\rangle & \cdots & \langle y, \Delta S_{n+2k-1}\rangle \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \cdots & 1 \\ \langle y, \Delta S_n\rangle & \langle y, \Delta S_{n+1}\rangle & \cdots & \langle y, \Delta S_{n+k}\rangle \\ \vdots & \vdots & & \vdots \\ \langle y, \Delta S_{n+k-1}\rangle & \langle y, \Delta S_{n+k}\rangle & \cdots & \langle y, \Delta S_{n+2k-1}\rangle \end{vmatrix}}. \tag{6}$$

Второе топологическое преобразование Шэнкса, обозначаемое через $\tilde{e}_k(S_n)$, можно определить, сделав замену первой строки числителя на $S_{n+k}, \ldots, S_{n+2k}$. Если $E = \mathbb{C}$, два данных преобразования редуцируются до скалярного преобразования Шэнкса (3).

3.3. Топологические ε*-алгоритмы*

Следующей задачей стало получение рекурсивного алгоритма для реализации полученного преобразования. Этот алгоритм также должен был быть сокращен до (4) в скалярном случае. Ключевым моментом стало определение обращения, которое ограничивается обычным преобразованием для скаляров и векторным в форме, использованной Винном в его векторном ε-алгоритме. Решением оказалось определение обращения пары $(y,u) \in E^* \times E$ через $y^{-1} = u/\langle y,u\rangle \in E$ и $u^{-1} = y/\langle y,u\rangle \in E^*$. Такое обращение удовлетворяет всем свойствам, которым должна удовлетворять данная операция: обратный элемент может быть вычислен, а отношение двойственности элемента со своим обратным равно единице. Это определение обращения привело к тому, что правило этого, так называемого, *топологического* ε-алгоритма пришлось разделить на два правила: одно для членов с еще четным нижним индексом, принадлежащих E, и второе для элементов с нечетным нижним индексом, которые находятся в E^* и являются промежуточными результатами, как в скалярном случае. Эти наблюдения приводят к (первому) *топологическому* ε-алгоритму (TEA1). Второй алгоритм (TEA2) позволяет реализовать рекурсивно второе преобразование. Доказано, что $\hat{\varepsilon}_{2k}^{(n)} = \hat{\varepsilon}_k(S_n)$, как определено уравнением (6), а для второго преобразования и алгоритма, что $\tilde{\varepsilon}_{2k}^{(n)} = \tilde{e}_k(S_n)$ [24].

Правила алгоритмов TEA были довольно сложными (два разных правила, они требуют хранения элементов E и E^*, отношение двойственности было рекурсивно использовано в правилах). Таким образом, было трудно реализовать их для линейных функционалов общего вида, и в прошлом эти алгоритмы использовались только в простом случае, когда E совпадает с $\mathbb{R}^p$ (или $\mathbb{C}^p$), y был вектором, и отношение двойственности сводится к внутреннему произведению между действительными или комплексными векторами. В 2014 г. авторы этой работы осознали, спустя 40 лет после публикации [24], что соотношение, записанное ранее, позволяет существенно упростить правила двух топологических ε-алгоритмов [25], что привело к формулировке *упрощенных топологических* ε-алгоритмов (STEA). Каждый из двух этих упрощенных алгоритмов может быть записан в 4 эквивалентных формах. Приведем только правило второго (STEA2), которое еще более эффективно и включает работу только с величинами с четными нижними индексами:

$$\tilde{\varepsilon}_{2k+2}^{(n)} = \tilde{\varepsilon}_{2k}^{(n+1)} + \frac{\varepsilon_{2k+2}^{(n)} - \varepsilon_{2k}^{(n+1)}}{\varepsilon_{2k}^{(n+2)} - \varepsilon_{2k}^{(n+1)}}(\tilde{\varepsilon}_{2k}^{(n+2)} - \tilde{\varepsilon}_{2k}^{(n+1)})$$

при $\tilde{\varepsilon}_0^{(n)} = S_n$, и где значения $\varepsilon_k^{(n)}$ вычисляются с использованием скалярного ε-алгоритма Винна, примененного к последовательности $(\varepsilon_0^{(n)} = \langle y, S_n\rangle)$.

По сравнению с оригинальными топологическими ε-алгоритмами упрощенные используют только одно треугольное правило вместо двух более сложных, также они позволяют использовать только элементы E с нижними четными индексами, отсутствует необходимость вычислять или хранить элементы E^*, соотношение двойственности используется только на этапе инициализации скалярного ε-алгоритма, требования по количеству используемых ячеек памяти были существенно снижены (что важно в случае работы с векторными и матричными последовательностями), а устойчивость алгоритмов была повышена с использованием конкретных правил Винна для скалярного ε-алгоритма [14]. Больше деталей и информацию о программной реализации можно найти в работе [26].

3.4. Другие преобразования Шэнкса

Как мы видим теперь, существуют различные способы использования уравнения (5) для вычисления коэффициентов a_i в преобразовании. Всегда предполагается, что сумма коэффициентов a_i равна 1.

Вместо использования одного элемента $y \in E^*$ при меняющемся n, мы можем зафиксировать n и выбрать k линейно-независимых функционалов $y_i \in E^*$. Эта процедура приводит к *методу минимальной полиномиальной экстраполяции* (ММРЕ) [24], который записывается через:

$$e_k(S_n) = \frac{\begin{vmatrix} S_n & S_{n+1} & \cdots & S_{n+k} \\ \langle y_1, \Delta S_n \rangle & \langle y_1, \Delta S_{n+1} \rangle & \cdots & \langle y_1, \Delta S_{n+k} \rangle \\ \vdots & \vdots & & \vdots \\ \langle y_K, \Delta S_n \rangle & \langle y_k, \Delta S_{n+1} \rangle & \cdots & \langle y_k, \Delta S_{n+k} \rangle \end{vmatrix}}{\begin{vmatrix} 1 & 1 & \cdots & 1 \\ \langle y_1, \Delta S_n \rangle & \langle y_1, \Delta S_{n+1} \rangle & \cdots & \langle y_1, \Delta S_{n+k} \rangle \\ \vdots & \vdots & & \vdots \\ \langle y_k, \Delta S_n \rangle & \langle y_k, \Delta S_{n+1} \rangle & \cdots & \langle y_k, \Delta S_{n+k} \rangle \end{vmatrix}}.$$

Для этого преобразования мы используем те же обозначения $e_k(S_n)$, что и для топологического преобразования Шэнкса, даже если оно отличается от него. Это преобразование может быть реализовано рекурсивно через $S\beta$-алгоритм согласно Л. Джейбилоу [27] (см. также [28]).

Метод полиномиальной экстраполяции (MPE) [29] и метод *экстраполяции с редуцированным рангом* (RRE) [30], [31] можно напрямую получить из выражения ММРЕ, подставив $y_i = \Delta S_{n+i-1}$ для первого и $y_i = \Delta^2 S_{n+i-1}$ для второго. Однако оба этих метода применимы лишь в случае векторных последовательностей.

Существует и несколько других обобщений, упомянем, например, [18], [32] и книги [1], [4], [5].

4. СТРАТЕГИИ И РЕАЛИЗАЦИИ

Обсудим два способа использования преобразований Шэнкса: методы *ускорения* и *рестарта*, непосредственно посвященные решению задач о неподвижной точке. Реализация обсуждаемых стратегий будет детально обсуждаться ниже.

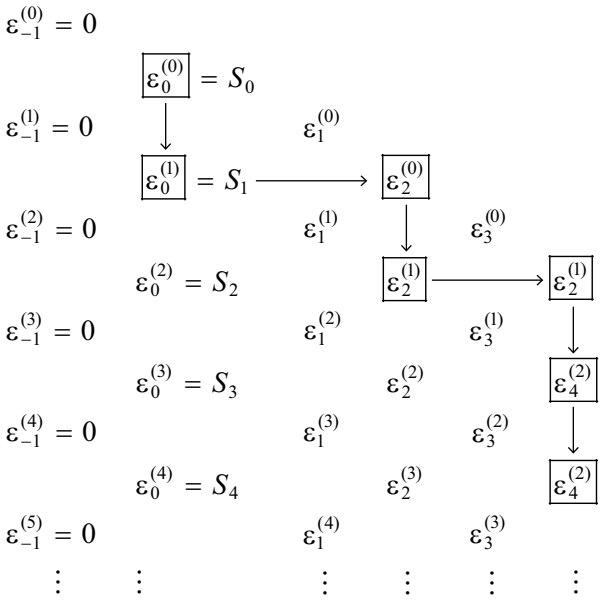
4.1. Метод ускорения (АМ)

Пусть (S_n) – последовательность скаляров, векторов, матриц или тензоров, сходимость которой необходимо ускорить с помощью одного из ε-алгоритмов. Предположим, что для фиксированного значения k мы хотим вычислить последовательность $(\varepsilon_{2k}^{(n)})$. Члены последовательности (S_n) определяются один за другим в соответствующей программной реализации, и четные столбцы ε-массива вычислены настолько далеко, насколько это возможно. Начиная с S_0 и S_1, ни один член не может быть получен. Тогда, вводя S_2, можно вычислить $\varepsilon_2^{(0)}$. С использованием дополнительно еще S_3 мы можем получить $\varepsilon_2^{(1)}$. Далее с S_4, можно получить $\varepsilon_4^{(0)}$. Таким образом, нисходящая "лестница" вдоль главной диагонали ε-массива получается до тех пор, пока, введя член S_{2k}, мы не сможем вычислить $\varepsilon_{2k}^{(0)}$. Далее каждый новый член $S_{2k+1}, S_{2k+2}, \ldots$ позволяет вычислить следующий член в столбце $2k$, т.е. $\varepsilon_{2k}^{(1)}, \varepsilon_{2k}^{(2)}, \ldots$. Если вместо вычисления при фиксированном значении k последовательности $(\varepsilon_{2k}^{(n)})$ требуется вычислить элементы главной нисходящей диагонали или "лестницу" вдоль нее, достаточно дать большое значение k и прекратить вводить новые члены последовательности для преобразования. Данная стратегия называется *методом ускорения* (АМ). Фиг. 1 демонстрирует сопутствующий алгоритм, а фиг. 2 приводит пример данной стратегии.

Поясним далее, как построить таблицу, см. фиг. 2 по последовательности (S_n). Простейший способ построения ε-массива – это вычислить столбцы один за другим, начиная с первого столбца, используя далее для всех столбцов (кроме первого) значения из двух предыдущих. С точки зрения требований по количеству используемых ячеек памяти и количества действий данная процедура может быть слишком затратной в случае, если членами последовательности являются векторы или матрицы. Поэтому обычно при реализации алгоритмов используется

Выбрать $2k$ и S_0
For $n = 1, 2, 3, \ldots,$
Вычислить S_n
Использовать метод и вычислить на каждом цикле новый экстраполированный член последовательности
$\varepsilon_0^{(0)} = S_0, \varepsilon_0^{(1)}, \varepsilon_2^{(0)}, \varepsilon_2^{(1)}, \ldots, \varepsilon_{2k}^{(0)}, \varepsilon_{2k}^{(1)}, \varepsilon_{2k}^{(2)}, \ldots$
end for n

Фиг. 1. Метод ускорения (AM).



Фиг. 2. Пример ε-массива при $2k = 4$.

прием *бегущего окна*, который известен благодаря Винну [14], использовавшему данную технику в своем скалярном ε-алгоритме.

Конечно, эта техника может использоваться и в других алгоритмах, в частности, в топологическом ε-алгоритме или в его упрощенных версиях. Идея состоит в том, чтобы вести работу от возрастающих диагоналей, т.е. мы вычисляем каждую новую возрастающую диагональ массива, используя предыдущую диагональ (или две предшествующие в случае первого топологического ε-алгоритма). Большее число деталей сообщено в работе [26] и в пользовательской инструкции соответствующего программного обеспечения [33].

4.2. Метод рестарта (RM)

В частном случае вычисления скалярной, векторной, матричной или тензорной неподвижной точки основная проблема состоит в том, чтобы найти x, удовлетворяющий уравнению $x = F(x)$. Очевидно, можно организовать итерационный процесс в форме

$$S_{n+1} = F(S_n), \quad n = 0, 1, \ldots,$$

и затем ускорить его сходимость с помощью AM. Однако можно использовать и другую стратегию стеффенсенского типа, известную под названием *метода рестарта* (RM). Начиная с заданного x_0, мы вводим $u_0 = x_0$ и проводим некоторое количество итераций Пикара, *базовых итераций*, $u_{i+1} = F(u_i)$ для $i = 0, \ldots, 2k-1$ для фиксированного значения k. Далее применение ε-алгоритма к этим $2k+1$ элементам генерирует $x_1 = \varepsilon_{2k}^{(0)}$. Базовые итерации после этого можно начать

Выбрать $2k$ и x_0.
For $n = 0, 1, \ldots$ (внешние итерации)
  Присвоить $u_0 = x_n$
  Вычислить $u_{i+1} = F(u_i)$, для $i = 0, \ldots, 2k-1$ (внутренние итерации)
  Применить метод экстраполяции для $u_0, \ldots, u_{2k}$ и вычислить $\varepsilon_{2k}^{(0)}$
  Присвоить $x_{n+1} = \varepsilon_{2k}^{(0)}$
end for n

Фиг. 3. Метод рестарта (RM).

уже с $u_0 = x_1$ и продолжать далее, что порождает последовательность (x_n), которая при некоторых предположениях сходится к неподвижной точке x. Фиг. 3 демонстрирует соответствующий алгоритм. В векторном случае $F : \mathbb{R}^p \mapsto \mathbb{R}^p$, Х. Ле Ферран [34] доказал, что при выборе $k = p$ использование любого из топологических ε-алгоритмов в RM генерирует последовательность, которая при некоторых предположениях сходится квадратично к неподвижной точке $x \in \mathbb{R}^p$ для отображения F. Данная процедура называется *обобщенным методом Стеффенсена* (GSM), так как при $p = 1$ процедура оказывается непосредственно методом Стеффенсена [35]. Для метода RRE подробный анализ представлен в работе Сиди [36].

4.3. Программные реализации

Несмотря на то, что во многих публикациях в литературе утверждается, что преобразования последовательностей и соответствующие алгоритмы могут быть полезными методами для получения или улучшения численных решений широкого круга задач численного анализа и прикладной математики, потенциальных пользователей часто отговаривают использовать их из-за сложности их реализации простым и оптимизированным способом. По этой причине в 1991 г. в дополнение к [1] на языке FORTRAN 77 была опубликована библиотека (процедуры и демонстрационные программы). Недавно библиотека EPSfun [33] на языке MATLAB была внедрена в библиотеку с открытым программным кодом Netlib. Она содержит необходимые функции по реализации ε-алгоритмов Винна, оригинальных топологических ε-алгоритмов и их упрощенных версий. Напомним, что оригинальные и упрощенные топологические ε-алгоритмы требуют определения элемента $y \in E^*$. В оригинальных алгоритмах y непосредственно используется в одном из правил, в то время как в упрощенных данный элемент играет роль только на этапе инициализации скалярного ε-алгоритма, который применяется к последовательности $(\langle y, S_n \rangle)$. Демонстрационные программы, шаблоны проектов (как для методов AM, так и для RM), а также пользовательские инструкции являются частью данного программного обеспечения.

5. ПРИМЕРЫ ПРИЛОЖЕНИЙ

Все эти алгоритмы получили множество применений: в ускорении сходимости скалярных, векторных и матричных последовательностей (в частности, для рекурсивного решения систем линейных и нелинейных уравнений), при работе с осцилляциями Гиббса для рядов Фурье, при вычислении матричных функций, при поиске решений интегральных уравнений, в задачах, связанных с тензорами и многих других. В данном разделе мы постараемся дать обзор некоторых из таких задач и проиллюстрировать конкретные улучшения, достигаемые в результате использования методов экстраполяции.

5.1. Скалярный ε-алгоритм

Начнем с демонстрации двух случаев применения скалярного ε-алгоритма. В первом случае рассматривается задача об ускорении сходимости конкретной последовательности. Во втором случае алгоритм применяется к рядам Фурье.

Таблица 1. Паде аппроксимации для логарифма

k	$f_{2k}(1)$	$[k/k]_f(1)$	k	$f_{2k}(2)$	$[k/k]_f(2)$
1	0.830	0.7	1	0.260×10^{1}	1.14
2	0.783	0.6933	2	0.506×10^{1}	1.101
3	0.759	0.693152	3	0.126×10^{2}	1.0988
4	0.745	0.69314733	4	0.375×10^{2}	1.098625
5	0.736	0.6931471849	5	0.121×10^{3}	1.0986132
6	0.730	0.69314718068	6	0.410×10^{3}	1.09861235
7	0.725	0.693147180563	7	0.142×10^{4}	1.098612293
8	0.721	0.69314718056000	8	0.504×10^{4}	1.0986122890
9	0.718	0.6931471805599485	9	0.181×10^{5}	1.098612288692
10	0.716	0.6931471805599454	10	0.655×10^{5}	1.0986122886698

5.1.1. Скалярные последовательности. Применим скалярный ε-алгоритм к частичным суммам f_{2k} ряда

$$f(z) = \ln(1+z) = z - z^2/2 + z^3/3 - z^4/4 + \ldots,$$

который сходится для $|z| \le 1$, кроме $z = -1$.

Для $z = 1$ имеем $\ln 2 = 0.6931471805599453\ldots$. Для $z = 2$ ряд расходится, и мы имеем $\ln 3 = 1.098612288668110\ldots$. Аппроксимации Паде приводят к результатам, содержащимся в табл. 1 ниже (напомним, что $\varepsilon_{2k}^{(0)} = [k/k]_f(z)$).

Мы можем видеть, что ε-алгоритм позволяет получить суммы для расходящихся последовательностей (рядов, в данном конкретном случае). Это следует из факта, что значения $\varepsilon_{2k}^{(n)}$ связаны с аппроксимациями Паде и что эти аппроксимации позволяют получить аналитическое продолжение для некоторых рядов вне области сходимости, что и демонстрирует приведенный пример.

5.1.2. Эффект Гиббса. Далее мы приведем численные примеры, показывающие, что ε-алгоритм позволяет локализовать разрывы в рядах Фурье, ускорить их сходимость и уменьшить выбросы, возникающие из-за эффекта Гиббса.

Рассмотрим ряд Фурье общего вида (без ограничений общности, предположим, что ряд является периодическим на отрезке $[0, 2\pi]$ и обладает нулевым постоянным членом):

$$S(t) = \sum_{k=1}^{\infty} a_k \cos kt + \sum_{k=1}^{\infty} b_k \sin kt.$$

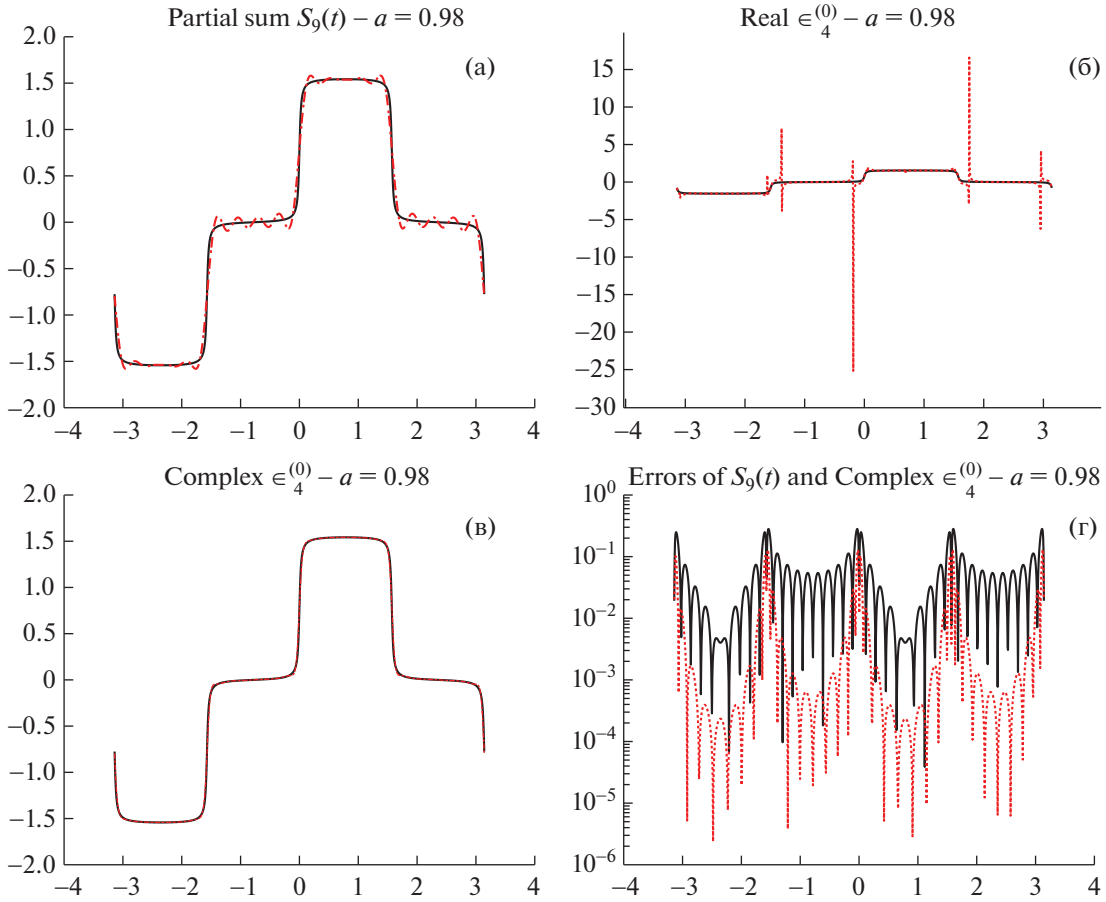
Складывая этот ряд с сопряженным:

$$\widetilde{S}(t) = \sum_{k=1}^{\infty} a_k \sin kt - \sum_{k=1}^{\infty} b_k \cos kt$$

как с мнимой частью, мы получаем комплексный ряд Фурье:

$$F(t) = S(t) + i\widetilde{S}(t) = \sum_{k=1}^{\infty} (a_k - ib_k)e^{ikt}.$$

Применим теперь ε-алгоритм к частичным суммам $F(t)$ и далее возьмем вещественную часть значений $\varepsilon_{2k}^{(n)}$. Данная процедура известна под названием *комплексного* ε-алгоритма, а получаемые значения обозначаются ${}_C\varepsilon_{2k}^{(n)}$. В случае, если скалярный ε-алгоритм применяется к частичным суммам $S(t)$ (как в п. 2.3), мы считаем, что речь идет о *вещественном* ε-алгоритме и обозначаем получаемые значения ${}_R\varepsilon_{2k}^{(n)}$.



Фиг. 4. Эффект Гиббса и скалярный ε-алгоритм.

В качестве примера рассмотрим ряд Фурье на отрезке $[0, \pi]$:

$$S(t) = \frac{1}{2}\left(\arctan\frac{2a\cos t}{1-a^2} + \arctan\frac{2a\sin t}{1-a^2}\right) =$$
$$= \sum_{k=1}^{\infty}(-1)^{k+1}\frac{a^{2k-1}}{2k-1}\cos(2k-1)t + \sum_{k=1}^{\infty}\frac{a^{2k-1}}{2k-1}\sin(2k-1)t.$$

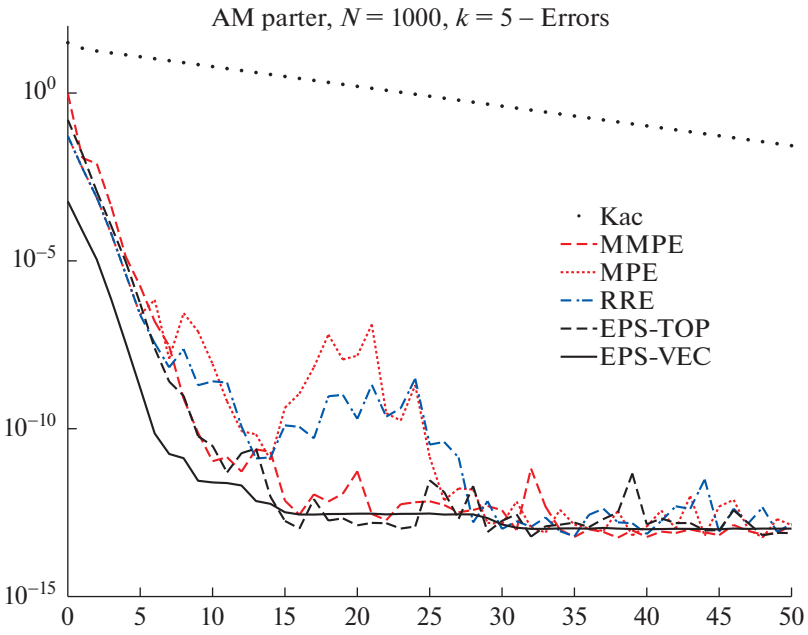
На фиг. 4а–в сплошная линия соответствует точному результату, пунктирная представляет результаты использования ε-алгоритма. На фиг. 4г мы демонстрируем уровень ошибки $S_9(t)$ (что соответствует частичной сумме из первых 9 членов $S(t)$) относительно точных значений при значении $a = 0.98$. График ${}_R\varepsilon_4^{(0)}$ демонстрирует, что алгоритм позволяет выявить почти все разрывы.

График значений ошибки демонстрирует улучшения, полученные в результате использования комплексного ε-алгоритма. Эта идея была предложена Винном [37], затем приведена в работе [38], где ε-алгоритм использовался для локализации разрывов, и окончательно оформлена в [39].

5.2. Работа с системами уравнений

Применим теперь ранее описанные алгоритмы к задачам о решении систем линейных и нелинейных уравнений.

5.2.1. Метод Качмарца. Метод Качмарца – итерационный метод решения систем линейных уравнений [40]. Хотя история этого алгоритма началась в 1937 г., в последнее время к нему возрождается интерес из-за его хорошей структуры для параллельной реализации. Другим достоин-



Фиг. 5. Ускорение сходимости метода Качмарца для `parter` матрицы, $N = 1000$, $k = 5$.

ством этого подхода является его сходимость для всех случаев, как было доказано [41]. Тем не менее сходимость обычно оказывается медленной, и несколько процедур для ее ускорения можно найти в литературе.

Одна итерация *циклического* метода Качмарца для решения линейной системы $Ax = b$ размерности N состоит в следующих действиях:

$$p_0 = x_n,$$
$$p_i = p_{i-1} + \frac{(b - Ap_{i-1}, e_i)}{(A^{\mathrm{T}}e_i, A^{\mathrm{T}}e_i)} A^{\mathrm{T}}e_i, \quad i = 1, \ldots, N,$$
$$x_{n+1} = p_N.$$

Обозначим через $a_i = A^{\mathrm{T}}e_i$ вектор-столбец, сформированный из i-го столбца A, а через b_i – компоненту с номером i вектора правой части b. Тогда вычисление каждого из векторов p_i в шаге итерации не требует вычисления умножения матрицы на вектор, что приводит к:

$$p_i = p_{i-1} + \frac{b_i - (p_{i-1}, a_i)}{\|a_i\|^2} a_i.$$

Данное замечание позволяет легко получить параллельную реализацию алгоритма.

Обозначим далее $P_i = I - A\alpha_i e_i^{\mathrm{T}}$ и $Q_i = A^{-1}P_iA$ при $\alpha_i = A^{\mathrm{T}}e_i \Big/ \left\|A^{\mathrm{T}}e_i\right\|^2$. Выполнено $x_{n+1} - x = Q(x_n - x)$ при $Q = Q_N \cdots Q_1$, что есть $x_n - x = Q^n(x_0 - x)$. Данное выражение демонстрирует, что итерационная последовательность (x_n) принадлежит к ядру топологического преобразования Шэнкса. Так, в теории его применение может привести к получению точного решения системы. Однако, так как N обычно (очень) велико, преобразование нельзя использовать на практике, хотя его можно использовать в целях ускорения сходимости, как это делается в работе [42].

Мы рассматриваем `parter` матрицу A, $N = 1000$, $\varkappa(A) \simeq 4.2306$, – теплицеву матрицу с сингулярными числами, близкими к π. Фигура 5 демонстрирует сравнение метода Качмарца с MMPE, MPE, RRE, топологическим ε-алгоритмом и с векторным ε-алгоритмом при $k = 5$.

Мы видим, что все методы достигают хорошей точности с преимуществом векторного ε-алгоритма. Более того, его сходимость оказывается более гладкой. В приведенном примере стар-

шее собственное значение матрицы A равно 0.8732178, а следующее за ним – 0.3170877. Таким образом, согласно теоретическим результатам, общим для всех этих методов, хорошее ускорение может наблюдаться даже при $k = 1$.

5.2.2. Интегральные уравнения. Рассмотрим следующее нелинейное *интегральное уравнение Фредгольма* II *рода* с ядром K:

$$u(t) = \int_a^b K(t, x, u(x))dx + f(t), \quad t \in [a, b].$$

Будем считать, что выполнены все обычные условия, гарантирующие существование единственного решения.

Стандартным методом решения этого уравнения является приближение интегрального оператора с помощью квадратурных формул:

$$\int_a^b K(t, x, u(x))dx \simeq \sum_{j=0}^{p} w_j^{(p)} K(t, x_j^{(p)}, u(x_j^{(p)})),$$

где $x_0^{(p)}, \ldots, x_p^{(p)}$ – это $p+1$ точка на отрезке $[a, b]$, а верхний индекс p соответствует зависимости от номера выбранной точки. Напомним, что веса $w_j^{(p)}$ строго положительны, а их сумма равна $b - a$. Таким образом, уравнение аппроксимируется через

$$u_p(t) = \sum_{j=0}^{p} w_j^{(p)} K(t, x_j^{(p)}, u_p(x_j^{(p)})) + f(t).$$

Далее мы приближаем решение u_p методом коллокаций для точек $t_i^{(p)} = x_i^{(p)}$ при $i = 0, \ldots, p$. При фиксированном значении p обозначим для удобства $t_i = x_i = t_i^{(p)} = x_i^{(p)}$, $f_i = f(t_i)$, $w_i^{(p)} = w_i$ и определим далее аппроксимации u_i для $u_p(t_i)$, $i = 0, \ldots, p$, как решение системы из $p+1$ нелинейных уравнений:

$$u_i = \sum_{j=0}^{p} w_j K(t_i, t_j, u_j) + f_i, \quad i = 0, \ldots, p.$$

Для решения этой системы для начала мы используем итерационный метод Пикара:

$$u_i^{(n+1)} = \sum_{j=0}^{p} w_j K(t_i, t_j, u_j^{(n)}) + f_i, \quad i = 0, \ldots, p,$$

или, в более общем случае схему релаксации:

$$u_i^{(n+1)} = u_i^{(n)} - \alpha \left\{ u_i^{(n)} - \sum_{j=0}^{p} w_j K(t_i, t_j, u_j^{(n)}) - f_i \right\}, \quad i = 0, \ldots, p,$$

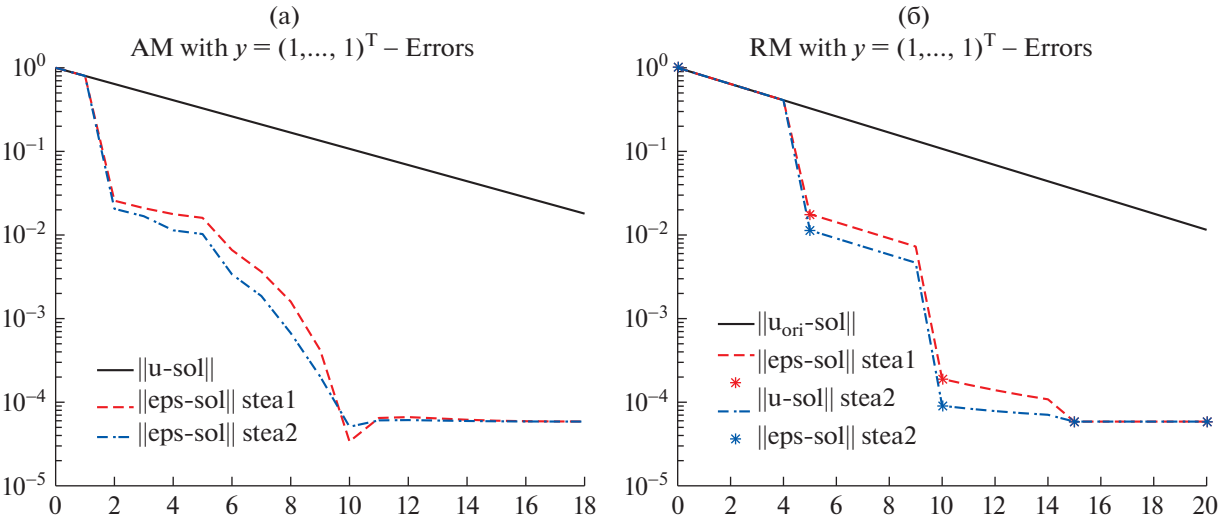
где α – параметр, который необходимо найти, а $u_i^{(0)}$, $i = 0, \ldots, p$ – начальное приближение решения в точках t_i.

Затем эти итерации вводятся в один из методов, полученных из преобразования Шэнкса, таких как MPE, MMPE, RRE, векторный или упрощенный топологический ε-алгоритмы. Полный математический анализ этой методологии и алгоритм описаны в работе [43]. Программы на языке MATLAB с несколькими примерами находятся в свободном доступе на сайте Matlab File Exchange.

В качестве примера мы рассмотрим интегральное уравнение

$$u(t) = t^2 \int_0^1 \frac{x^2}{1 + u^2(x)} dx + (1/2 - \ln 2)t^2 + \sqrt{t},$$

чьим решением является $u(x) = \sqrt{x}$. Из преобразований Шэнкса здесь используются первый и второй упрощенные топологические ε-алгоритмы, каждый с использованием AM и RM стратегий.



Фиг. 6. Стратегии AM (а) и RM (б) при $y = (1, \ldots, 1)^T$.

Результаты $\alpha = 0.2$, $p = 31$, $y = (1, \ldots, 1)^T$ использования стратегии AM при параметрах и $2k = 10$ представлены на фиг. 3а, результаты при $2k = 4$ для стратегии RM представлены на фиг. 3б.

Нашей целью было не конкурировать со сложными методами, которые можно найти в литературе, а показать, что простая процедура, сопровождаемая методом ускорения, может быть весьма эффективной.

5.2.3. Нелинейные алгебраические уравнения. Для ускорения сходимости итерационного метода $x_{n+1} = x_n + \alpha f(x_n)$ при решении системы линейных или нелинейных уравнений $f(x) = 0$ мы можем воспользоваться *методом ускорения* (AM) или *методом рестарта* (RM).

Рассмотрим далее нелинейную систему:

$$x_1 = x_1 x_2^3/2 - 1/2 + \sin x_3,$$
$$x_2 = (\exp(1 + x_1 x_2) + 1)/2,$$
$$x_3 = 1 - \cos x_3 + x_1^4 - x_2$$

с решением $(-1, 1, 0)^T$.

Начиная с точки $x_0 = 0 \in \mathbb{R}^3$, мы наблюдаем результаты, представленные на фиг. 4а для AM при $\alpha = 0.2$ и $\varepsilon_4^{(n)}$. При $\alpha = 0.1$ наблюдаются аналогичные результаты. Для GSM ($2k = 6$) при $\alpha = 0.1$ использование STEA2 позволяет получить результаты лучше, чем STEA1, что показано на фиг. 7б. Ступенчатая структура графика на фиг. 4б наглядно демонстрирует квадратичную сходимость.

Квадратичная сходимость GSM также наблюдается для MMPE, MPE и RRE, что доказано в работе [44].

5.2.4. Матричные уравнения. Рассмотрим симметричное матричное уравнение Стейна, известное также как уравнение Ляпунова с дискретным временем:

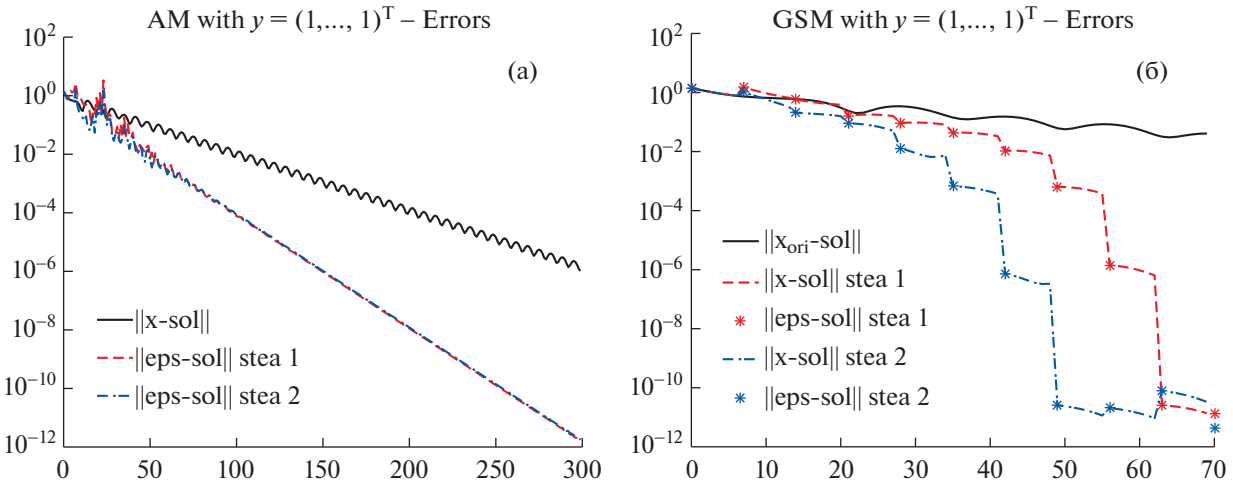
$$S - ASA^T = FF^T,$$

$F \in \mathbb{R}^{m\times s}$, $s \ll m$, при собственных значениях A, находящихся внутри единичного круга.

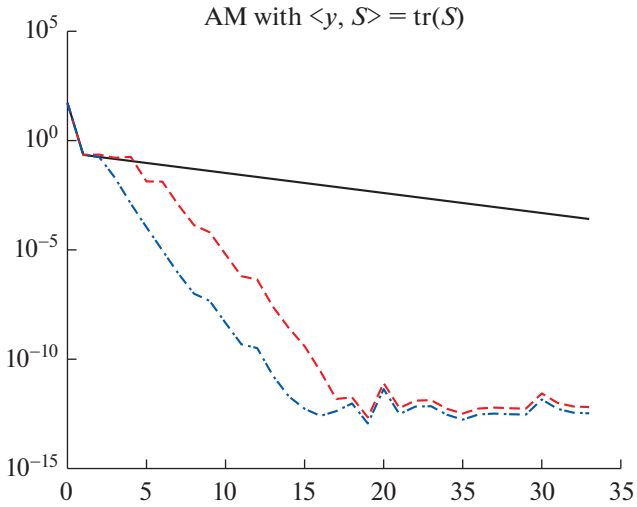
Известен итерационный метод, позволяющий получить численное решение (см. [25]):

$$S_{n+1} = FF^T + AS_nA^T,$$

$n = 0, 1, \ldots$, при $S_0 = 0$.



Фиг. 7. Нелинейная система: а – для AM, б – для GSM.



Фиг. 8. Уравнение Стейна.

Результаты на фиг. 8 соответствуют матрице `moler` размерности 500 для матрицы A, нормированной на скалярное значение такое, что ее спектральный радиус стал равен 0.9. Матрица `moler` – симметричная положительно определенная матрица с одним малым собственным значением. Ее элементы определяются как $a_{ij} = \min(i, j) - 2$ и $a_{ii} = i$. Матрица F – `parter` матрица размерности 500×30. Данные две матрицы принадлежат набору примеров MATLAB© `gallery`. При $2k = 6$ для упрощенных топологических ε-алгоритмов, где дуальное произведение с функционалом y соответствует следу матрицы, мы применяем стратегию AM.

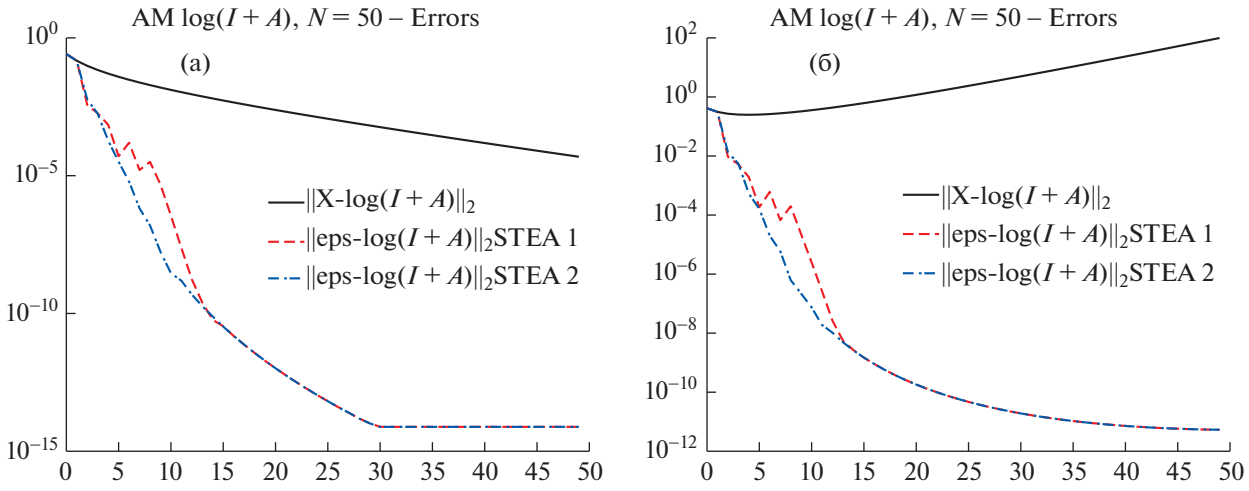
Сплошная линия на фиг. 8 соответствует итерационному методу, штриховая – алгоритму STEA1, а штрихпунктирная – методу STEA2.

5.3. Матричные функции

Рассмотрим задачу вычисления:

$$\log(I + A) = A - \frac{A^2}{2} + \frac{A^3}{3} - \frac{A^4}{4} + \ldots + (-1)^{n+1}\frac{A^n}{n} + \ldots,$$

где A – вещественная матрица.



Фиг. 9. Вычисление $\log(I + A)$: а – для $\rho(A) = 0.9$, б – для $\rho(A) = 1.2$.

Данный ряд сходится, если спектральный радиус матрицы A строго меньше единицы.

В данном примере мы покажем, что упрощенные топологические ε-алгоритмы позволяют ускорить сходимость частичных сумм данного ряда, а также получить сходимость к приближенному значению в случае расходимости исходного ряда.

Выберем квадратную случайную матрицу B размерности 50, значение r и определим

$$A = r \times B/\rho(B).$$

Таким образом, спектральный радиус A равен r.

Используя упрощенные топологические ε-алгоритмы при $\langle y, \cdot \rangle = \mathrm{tr}(\cdot)$, при $\varepsilon_8^{(n)}$ мы получаем результаты для частичных сумм ряда, представленные на фиг. 6, соответствующие $r = 0.9$ (случай сходимости, на фиг. 9а) и $r = 1.2$ (случай расходимости, на фиг. 9б). На графиках изображены значения евклидовой нормы погрешности с использованием функции `logm` для языка программирования MATLAB (которая позволяет вычислить основной матричный логарифм).

6. ТЕНЗОРЫ

Далее продемонстрируем два примера применения топологического ε-алгоритма и его упрощенной формы к задачам, связанным с тензорами (известных также как гиперматрицы). В нашей работе под тензором понимается многомерный массив.

Пусть $\mathbf{T} = (t_{i_1,\ldots,i_d}) \in \mathbb{R}^{n\times\cdots\times n}$ – вещественный кубический тензор с d измерениями размерности n. Его можно кратко обозначить через $\mathbf{T} \in \mathbb{R}^{[d,n]}$. Конечно, в случае $d = 1$ тензор является вектором, а при $d = 2$ – матрицей.

Для любого кубического тензора $\mathbf{T}$ и для любого вектора $\mathbf{x}$ размерности n можно записать вектор $\mathbf{y}$, вводя понятие умножение тензора $\mathbf{T}$ на вектор $\mathbf{x}$. Сделать это можно с помощью следующего отображения:

$$T : \mathbb{R}^n \to \mathbb{R}^n, \quad \mathbf{x} \mapsto T(\mathbf{x})_{i_1} = \sum_{i_2,\ldots,i_d} t_{i_1,i_2,\ldots,i_d} x_{i_2} \cdots x_{i_d},$$

для $i_1 = 1, \ldots, n$.

6.1. Тензорные ℓ^p-собственные пары

Так как T переводит $\mathbb{R}^n$ в себя, мы можем ввести концепцию собственных пар для $\mathbf{T}$ через отображение T. Вещественное число $\lambda \in \mathbb{R}$ будем считать ℓ^p-собственным значением $\mathbf{T}$ [45] (при $p > 1$), соответствующим ℓ^p-собственному вектору $\mathbf{x} \in \mathbb{R}^n$, если

$$T(\mathbf{x}) = \lambda \Phi_p(\mathbf{x}), \quad \|\mathbf{x}\|_p = 1,$$

где $\|\mathbf{x}\|_p$ соответствует обычно ℓ^p-норме:

$$\|\mathbf{x}\|_p = \left(|x_1|^p + \ldots + |x_n|^p\right)^{1/p}$$

и отображение $\Phi_p : \mathbb{R}^n \to \mathbb{R}^n$ определяется поэлементно $\Phi_p(\mathbf{x})_i = |x_i|^{p-2} x_i = \operatorname{sign}(x_i)|x_i|^{p-1}$ при $i = 1, \ldots, n$. Если $p \neq d$, то ℓ^p-собственные векторы нельзя определить с точностью до скалярного множителя. По этой причине мы добавляем условие нормировки $\|\mathbf{x}\|_p = 1$.

Тензор $\mathbf{T} = (t_{i_1,\ldots,i_d})$ будем называть неотрицательным и писать $\mathbf{T} \geq 0$, если $t_{i_1,\ldots,i_d} 0$ для всех $i_j = 1, \ldots, n$ и $j = 1, \ldots, d$. Далее можно ввести несколько обобщений понятия матричного спектрального радиуса для тензоров (см., например, [46]). В этой работе мы используем следующее определение:

$$r_p(\mathbf{T}) = \sup\{|\lambda| \lambda\text{: является } \ell^p \text{ собственным значением } \mathbf{T}\}.$$

Доказано, что для кубических неотрицательных тензоров $\mathbf{T}$ таких, что $T(1)$ поэлементно положителен, где 1 – вектор из единиц, при $p > d$ выполнено $r_p(\mathbf{T}) > 0$, и существует единственный поэлементно положительный $\mathbf{u} \in \mathbb{R}^n$ такой, что $\|\mathbf{u}\|_p = 1$ и $T(\mathbf{u}) = r_p(\mathbf{T})\Phi_p(\mathbf{u})$.

Определим дополнительно:

$$\mathbf{x} \mapsto f_p(\mathbf{x}) = \frac{\mathbf{x}^{\mathrm{T}} T(\mathbf{x})}{\|\mathbf{x}\|_p^d}, \quad \mathbf{x}^{\mathrm{T}} T(\mathbf{x}) = \sum_{i_1,\ldots,i_d} t_{i_1,i_2,\ldots,i_d} x_{i_1} \cdots x_{i_d}.$$

Для вычисления максимальной ℓ^p-собственной пары тензора $\mathbf{T}$ степенной метод, возможно, лучший из известных подходов. Начнем с $\mathbf{x}_0 \in C_+(\mathbf{T})$, где $C_+(\mathbf{T})$ – конус неотрицательных векторов, имеющих такой же шаблон нулей, как $T(1)$. Выберем $p > d$, точность $\varepsilon > 0$ и $q = p/(p-1)$ (сопряженный показатель). Степенной метод, записанный ниже, позволяет получить последовательность (λ_k), сходящуюся к требуемому собственному значению, а последовательность $(\mathbf{x}_k)$ сходится к соответствующему собственному вектору при любом выборе начального условия $\mathbf{x}_0 \in C_+(\mathbf{T})$:

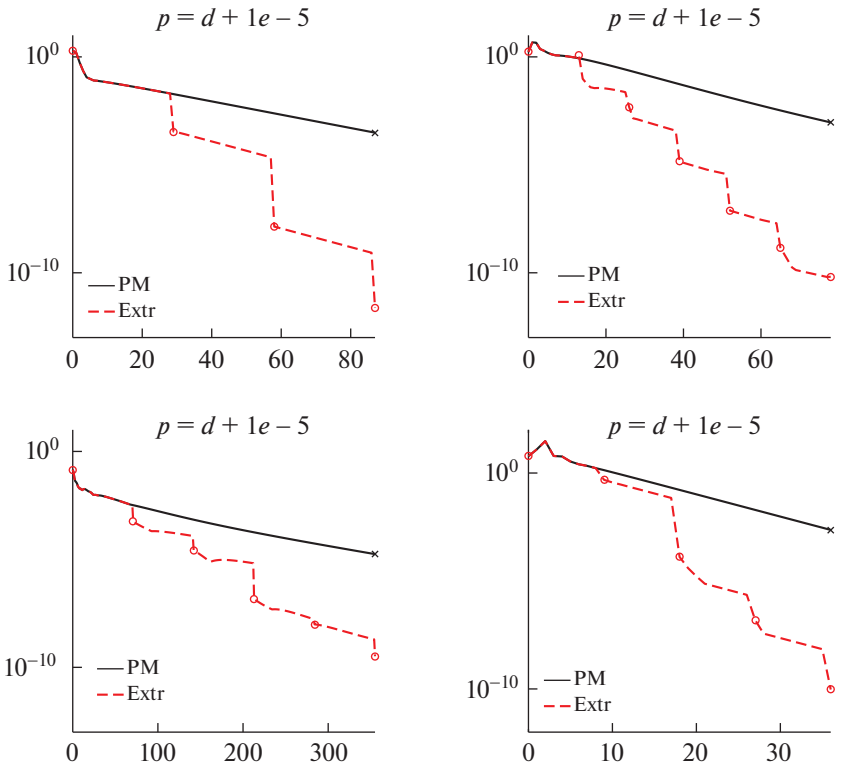
For $k = 0, 1, 2, 3, \ldots$ **repeat**
$\mathbf{y}_{k+1} = \Phi_q(T(\mathbf{x}_k))$
$\mathbf{x}_{k+1} = \mathbf{y}_{k+1} / \|\mathbf{y}_{k+1}\|_p$
$\lambda_{k+1} = f_p(\mathbf{x}_{k+1})$
until $\|\mathbf{x}_{k+1} - \mathbf{x}_k\|_p < \varepsilon$

Данный алгоритм может работать достаточно медленно, что делает его неприменимым для решения реальных прикладных задач, например, для работы с тензорами, соответствующими сетевым данным. Таким образом, для ускорения сходимости последовательности $(\mathbf{x}_k)$ мы используем алгоритм STEA2 с техникой рестарта.

На графиках ниже (фиг. 10) мы отображаем результаты вплоть до 30-й итерации для поэлементной невязки собственного вектора

$$\|T(\mathbf{x}_k) - \lambda_k \Phi_p(\mathbf{x}_k)\|_\infty,$$

как для исходной последовательности, так и для экстраполированной, и мы “подчеркиваем” крупными точками каждый рестарт внешнего цикла, например, невязку, сгенерированную ите-



Фиг. 10. Тензор ℓ^p-собственных векторов. Первая строка: `dolphins` (*nb*) и `yeast` ($n = 2361$). Вторая строка: `gre1107` ($n = 1107$) и `wb - cs - stanford` ($n = 9914$).

рациями экстраполяции. Мы фиксируем $p = d + 10^{-5}$ и устанавливаем $\mathbf{x}_0$ случайным вектором. С выбранным p мы можем вычислить приближение H-собственных пар (соответствующих $p = d$), так как в данном случае сходимость не гарантируется.

Линейный функционал $\mathbf{y}$ обновляется в конце каждого внешнего цикла как $\mathbf{y} = \tilde{\mathbf{e}}_h(\mathbf{x}_0)$ (для первого шага экстраполяции мы выбираем $\mathbf{y} = \mathbf{x}_0$).

В табл. 2 показаны четыре примера реальных сложных сетей, описываемых с помощью неотрицательных тензоров $\mathbf{T}$ третьего порядка, порожденных данными из работы [47].

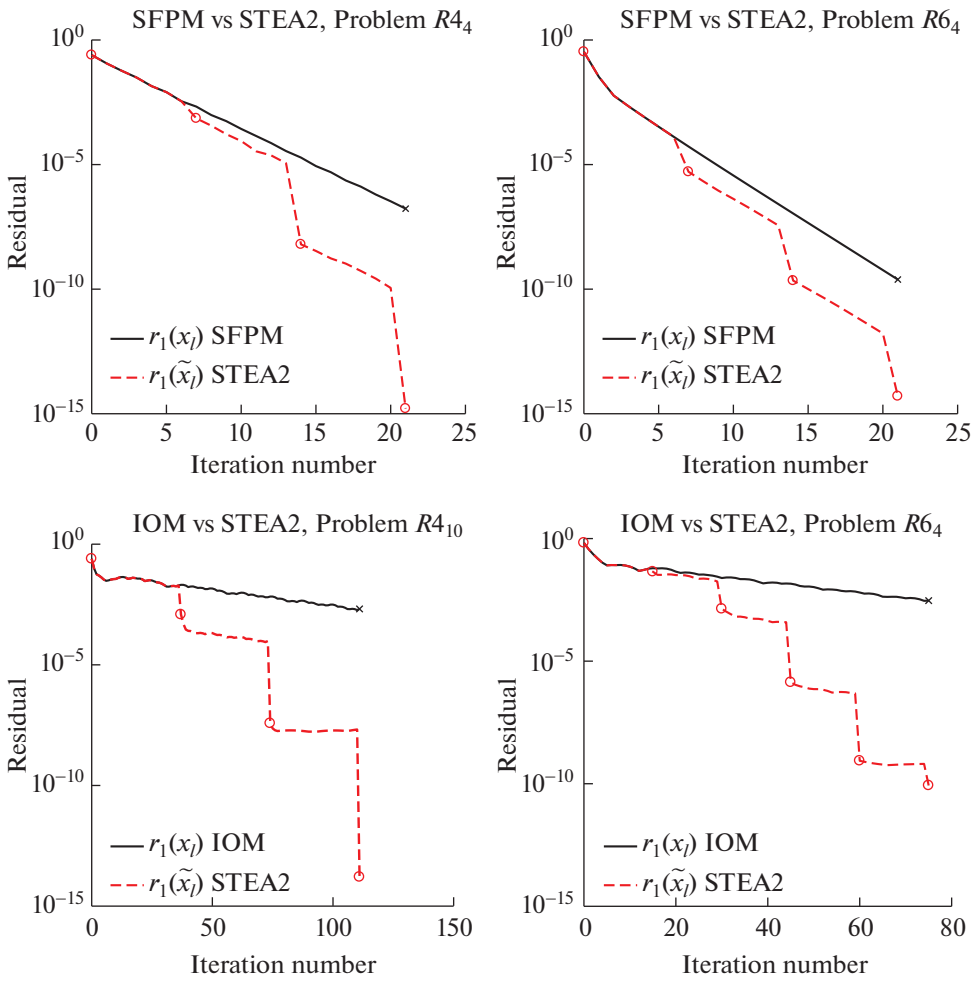
Больше подробностей, информацию о других алгоритмах и примерах для данного типа задач можно найти в [48].

6.2. Вычисление полилинейного PageRank

Недавно идея Pagerank была обобщена для случая многомерных марковских цепей [49]. Исходное многомерное распределение Pagerank аппроксимируется так называемым полилиней-

Таблица 2. Характеристики сетей, описываемых с помощью неотрицательных тензоров $\mathbf{T}$ третьего порядка, порожденных данными из работы [47]

Название задачи	Размер
`dolphins` (undirected)	62
`yeast` (undirected)	2361
`gre1107` (directed)	1107
`wb - cs - stanford` (directed)	9914



Фиг. 11. Полилинейный PageRank. Первая строка – *метод простой итерации со сдвигами* SFPM, $\alpha = 0.499$, $2k = 6$. Вторая строка – *метод внутренних и внешних итераций* ~~iom~~, $\alpha = 0.99$, $2k = 36$ (в) и $2k = 14$ (г).

ным PageRank, а задача поиска истинного многомерного стационарного распределения заменяется задачей, связанной с вычислением симметричного тензора ранга один.

В рамках предложенной модели требуется решить следующую задачу о неподвижной точке:

$$\alpha \mathbf{T}\mathbf{x}^{d-1} + (1-\alpha)\mathbf{v} = \mathbf{x}, \quad \|\mathbf{x}\|_1 = 1,$$

где $\mathbf{T}$ – стохастический тензор и $\mathbf{v}$ – поэлементно положительный стохастический вектор, и где по определению из предыдущего раздела $\mathbf{T}\mathbf{x}^{d-1} = T(\mathbf{x})$.

Для решения данной задачи в работе [50] рассмотрены два метода решения задачи о неподвижной точке: *метод простой итерации со сдвигами* (кратко SFPM) [49] в формулировке, адаптированной для конкретного случая вычисления полилинейного Pagerank, а также *многомерный степенной метод со сдвигами* [51] и дополнительно *метод внутренних и внешних итераций* из работы [49]. Скорость сходимости может быть очень низкой. Таким образом, мы применили топологический ε-алгоритм с техникой рестарта для последовательностей $\mathbf{x}_\ell$, порожденных этими методами. Многочисленные численные эксперименты на синтетических и реальных данных демонстрируют улучшение работы перечисленных методов из-за применения методов экстраполяции. Мы демонстрируем результаты для нескольких задач из работы [49] при $d = 3$, $n = 4$ или $n = 6$, и где $y \in \mathbb{R}^n$ в STEA2 – последний экстраполированный член с предыдущего цикла. На фиг. 11 демонстрируются графики изменения относительной первой нормы невязки.

7. ЗАКЛЮЧЕНИЕ

Целью данной работы было показать, что методы экстраполяции могут быть весьма полезными инструментами при решении различных задач численного анализа и прикладной математики. Мы сосредоточились на изучении преобразований Шэнкса и ε-алгоритмах для их реализации. Дополнительно мы демонстрируем ряд конкретных примеров, иллюстрирующих выгоду использования методов экстраполяции. Мы надеемся, что эта статья будет стимулировать исследователей использовать эти алгоритмы в будущем. Соответствующие программные реализации алгоритмов на языке программирования MATLAB доступны в открытом доступе.

СПИСОК ЛИТЕРАТУРЫ

1. *Brezinski C., Redivo-Zaglia M.* Extrapolation Methods: Theory and Practice. Amsterdam: North–Holland, 1991.
2. *Delahaye J.P.* Sequence Transformations. Berlin: Springer–Verlag, 1988.
3. *Marchuk G.I., Shaidurov V.V.* Difference Methods and Their Extrapolations. New York: Springer–Verlag, 1983.
4. *Sidi A.* Practical Extrapolation Methods: Theory and Applications. Cambridge: Cambridge University Press, 2003.
5. *Sidi A.* Vector Extrapolation Methods with Applications. Philadelphia: Society for Industrial and Applied Mathematics, 2017.
6. *Weniger E.J.* Nonlinear sequence transformations for the acceleration of convergence and the summation of divergent series // Comput. Phys. Rep. 1989. V. 10. № 5. P. 189–371.
7. *Wimp J.* Sequence Transformations and Their Applications. New York: Academic Press, 1981.
8. *Shanks D.* Nonlinear Transformations of Divergent and Slowly Convergent Sequences // J. of Math. and Phys. 1955. V. 34. P. 1–42.
9. *Graves-Morris P.R., Jenkins C.D.* Vector-valued, rational interpolants III // Constructive Approximation. 1986. V. 2. P. 263–289.
10. *Shanks D.* An analogy between transient and mathematical sequences and some nonlinear sequence-to-sequence transforms suggested by it. Part I. White Oak: Naval Ordnance Laboratory, 1949.
11. *Brezinski C., Crouzeix M.* Remarques sur le procédé Δ^2 d'Aitken // C. R. Acad. Sci. Paris. 1970. P. 896–898.
12. *Brezinski C., Redivo-Zaglia M.* The genesis and early developments of Aitken's process, Shanks' transformation, the ε-algorithm, and related fixed point methods // Numerical Algorithms. 2019. V. 80. P. 11–133.
13. *Wynn P.* On a Device for Computing the $e_m(S_n)$ Transformation // Math. Tables and Other Aids to Comput. 1956. V. 10. № 54. P. 91–96.
14. *Wynn P.* Singular rules for certain non-linear algorithms // BIT Numerical Math. 1963. V. 3. P. 175–195.
15. *Baker G.A., Jr., Graves-Morris P.R.* Padé Approximants, 2nd Edition. Cambridge: Cambridge University Press, 1996.
16. *Wynn P.* Acceleration Techniques for Iterated Vector and Matrix Problems // Math. of Comput. 1962. V. 16. № 79. P. 301–322.
17. *Salam A.* Non-commutative extrapolation algorithms // Numerical Algorithms. 1994. V. 7. P. 225–251.
18. *Brezinski C., Redivo-Zaglia M.* Matrix Shanks Transformations // Electronic Journal of Linear Algebra. 2019. V. 35. P. 248–265.
19. *Graves-Morris P.R., Jenkins C.D.* Generalised inverse vector valued rational interpolation // Padé Approximat. and its Appl. Bad Honnef 1983. Lecture Notes in Math. 1984. P. 144–156.
20. *McLeod J.B.* A note on the ε-algorithm // Comput. 1971. V. 7. P. 17–24.
21. *Artin E.* Geometric Algebra. New York: Interscience, 1957.
22. *Porteous I.R* Topological Geometry, Second Edition. New York: Cambridge University Press, 1981.
23. *Salam A.* An algebraic approach to the vector ε-algorithm // Numerical Algorithms. 1996. V. 11. P. 327–337.
24. *Brezinski C.* Généralisation de la transformation de Shanks, de la table de Padé et de l'ε-algorithme // Calcolo. 1975. V. 12. P. 317–360.
25. *Brezinski C., Redivo-Zaglia M.* The simplified topological ε-algorithms for accelerating sequences in a vector space // SIAM Journal on Sci. Comput. 2014. V. 36. P. A2227–A2247.
26. *Brezinski C., Redivo-Zaglia M.* The simplified topological ε-algorithms: software and applications // Numerical Algorithms. 2017. V. 74. P. 1237–1260.
27. *Jbilou K.* Méthodes d'Extrapolation et de Projection. Applications aux Suites de Vecteurs. Thèse de 3ème cycle. Université des Sciences et Techniques de Lille, 1988.
28. *Jbilou K., Sadok H.* Some results about vector extrapolation methods and related fixed point iteration // J. Comp. Appl. Math. 1991. V. 36. P. 385–398.

29. *Cabay S., Jackson L.W.* A polynomial extrapolation method for finding limits and antilimits of vector sequences // SIAM J. Numer. Anal. 1976. V. 13. P. 734–752.
30. *Kaniel S., Stein J.* Least-square acceleration of iterative methods for linear equations // J. Optim. Theory Appl. 1974. V. 14. P. 431–437.
31. *Mešina M.* Convergence acceleration for the iterative solution of $x = Ax + f$ // Comput. Meth. in Appl. Mech. and Eng. 1977. V. 10. P. 165–173.
32. *Brezinski C., Redivo-Zaglia M., Saad Y.* Shanks Sequence Transformations and Anderson Acceleration // SIAM Review. 2018. V. 60. P. 646–669.
33. *Brezinski C., Redivo-Zaglia M.* EPSfun: a Matlab toolbox for the simplified topological ε-algorithm. Netlib (http://www.netlib.org/numeralgo/), 2017, na44 package.
34. *Le Ferrand H.* The quadratic convergence of the topological epsilon algorithm for systems of nonlinear equations // Numerical Algorithms. 1992. V. 3. P. 273–283.
35. *Steffensen J.F.* Remarks on iteration // Scandinavian Actuarial Journal. 1933. V. 1933. P. 64–72.
36. *Sidi A.* A convergence study for reduced rank extrapolation on nonlinear systems // Numerical Algorithms. 2020. V. 84. P. 957–982.
37. *Wynn P.* Transformations to accelerate the convergence of Fourier series // Gertrude Blanch Anniversary Volume. 1967. P. 339–379.
38. *Guilpin C., Gacougnolle J., Simon Y.* The ε-algorithm allows to detect Dirac delta functions // Appl. Numerical Math. 2004. V. 48. P. 27–40.
39. *Brezinski C.* Extrapolation algorithms for filtering series of functions, and treating the Gibbs phenomenon // Numerical Algorithms. 2004. V. 36. P. 309–329.
40. *Kaczmarz S.* Angenäherte Auflösung von Systemen linearer Gleichungen // Bull. Acad. Polon. Sci. 1937. V. 35. P. 355–357.
41. *Gastinel N.* Linear Numerical Analysis. New York: Acad. Press, 1970.
42. *Brezinski C., Redivo-Zaglia M.* Convergence acceleration of Kaczmarz's method // J. of Eng. Math. 2013. V. 93. P. 3–19.
43. *Brezinski C., Redivo-Zaglia M.* Extrapolation methods for the numerical solution of nonlinear Fredholm integral equations // J. Integral Equat. Appl. 2019. V. 31. № 1. P. 29–57.
44. *Jbilou K., Sadok H.* Vector extrapolation methods. Applications and numerical comparison // J. of Comput. and Appl. Math. 2000. V. 122. P. 149–165.
45. *Lim L.-H.* Singular values and eigenvalues of tensors: a variational approach // 1st IEEE Internat. Workshop on Comput. Advances in Multi-Sensor Adaptive Proc. 2005. P. 129–132.
46. *Gautier A., Tudisco F., Hein M.* The Perron-Frobenius theorem for multi-homogeneous mappings // SIAM J. Matrix Anal. Appl. 2019. V. 40. P. 1179–1205.
47. *Davis T.A., Yifan H.* The University of Florida Sparse Matrix Collection // ACM Trans. Math. Softw. 2011. V. 38. № 1. P. 1–25.
48. *Cipolla S., Redivo-Zaglia M., Tudisco F.* Shifted and extrapolated power methods for tensor ℓ^p-eigenpairs // Electron. Trans. Numer. Anal. 2020. V. 53. P. 1–27.
49. *Gleich D.F., Lim L.H., Yu Y.* Multilinear pagerank // SIAM J. Matrix Anal. Appl. 2015. V. 36. № 4. P. 1507–1541.
50. *Cipolla S., Redivo–Zaglia M., Tudisco F.* Extrapolation methods for fixed-point multilinear PageRank computations // Numer. Linear. Algebra. Appl. 2020. V. 27. P. e2280.
51. *Kolda T.G., Mayo J.R.* Shifted power method for computing tensor eigenpairs // SIAM J. Matrix Anal. Appl. 2011. V. 32. № 4. P. 1095–1124.