# Введение

***-алгоритм***

-алгоритм (эпсилон – алгоритм) был предложен Питером Винном в 1956 году для вычисления преобразования Шенкса, и до сих пор является одним из самых важных алгоритмов ускорения сходимости, используемых в Численном Анализе, Методах решения уравнений, включая дифференциальные и интегральные, а также во многих других сферах.

Ускорение достигается за счет преобразования (трансформации) последовательности.   
Последовательность , которая расходится или сходится так медленно, что практически не применима, превращается, с помощью функции ,в последовательность, которая сходится быстрее

Считается, что функция ускоряет сходимость, если сходится к быстрее, чем

Трансформация последовательности позволяет улучшить сходимость и/или значительно уменьшить количество необходимых итераций.

Одним из первых методов ускорения сходимости является алгоритм (Дельта 2), открытый Александром Крейгом Эйткеном в 1926 году. ***Метод Эйткена*** не является теоретически обоснованным, но при приближенных значениях параметров позволяет увеличить скорость сходимости [1].

***Метод Эйткена***

Пусть

(1.1)

где и некоторые константы. Тогда:

Следовательно,

Откуда получаем:

Стало быть:

Однако, из – за неточности в качестве следующей итерации мы должны взять значение, близкое к .

Из этого метода и выводится алгоритм :

Обозначим операторы:и

Однако если использовать на для улучшения точности, то это позволит вывести рекурсивную формулу:

(1.2)

Благодаря этой формуле, алгоритм можно имплементировать, используя один одномерный массив.

особенно хорошо подходит для последовательностей с линейной сходимостью (отклонение от их предела ведет себя до бесконечности, как геометрическая последовательность).

К сожалению, это численно нестабильный алгоритм: рекомендуется вычислять последовательность , а также с большим количеством значащих цифр. Некоторые записи алгоритма меньше распространяют ошибки округления, например:

# Эпсилон Алгоритм

Обобщением формулы (1.1) является:

(2.1)

Однако, в обобщенной формуле (1.2) для точно не дает (2.1). Вместо этого используется Преобразование Шенкса, которое определено следующим отношением определителей Ханкеля:

Где обозначает определитель Ханкеля:

Оно полностью соответствует (2.1).

Определители Ханкеля в Преобразовании Шенкса могут быть вычислены нелинейной рекурсией:

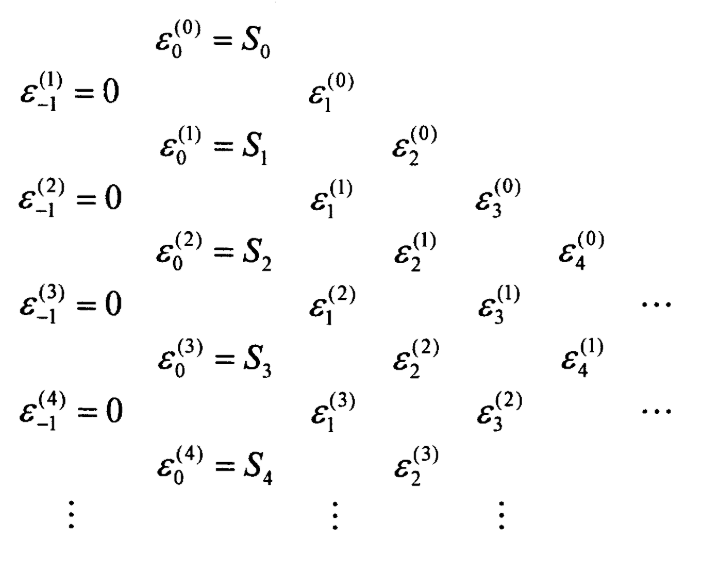
(2.2)

Подробнее о Преобразовании Шенкса можно узнать в файле ***Проект\_ПОПК.pdf,*** а реализация находится в файле ***shanks\_transformation.h.***

Рекурсивная схема (2.2) довольно сложна, и -алгоритм Винна существенно упрощает ее, убирая необходимость в вычислении определение Ханкеля:

(2.3)

-алгоритм является обобщением устаревшего алгоритма и был крайне важным шагом в ускорении сходимости. Результатом работы алгоритма будет -таблица (эпсилон-таблица), которая в теории бесконечна, но при ограниченном даст треугольник.



Винн доказал, что каждый (четный) ряд -таблицы эквивалентен преобразований Шенкса:

Нечетные же значения нужны лишь для промежуточных вычислений и не несут практической ценности:

Из формулы (2.3) очевидно, что -алгоритм связывает величины, расположенные в четырех вершинах ромба. И самым эффективным решением будет вычисление диагоналей таблицы, постепенно считая новые значения за счет увеличения .

-алгоритм можно представить в виде двух одномерных массивов. Реализация находится в файле ***epsilon\_algorithm.h.*** Однако,в книге за авторством Брезински описан вариант реализации через одномерный массив и несколько дополнительных переменных [2].

# Проблема катастрофического сокращения точности и модификация Винна

**Природа проблемы:**  
Стандарт IEEE754 – широко используемый формат представления чисел с плавающей точкой. Он использует только ограниченное количество битов. Например, представление с двойной точностью использует 64 бита. 1 бит – знак, 11 битов на порядок и 52 бита на мантиссу:

Из – за ограниченного числа битов, для чисел с плавающей точкой имеется фундаментальное ограничение – при операциях с близкими числами возникает катастрофическая потеря значащих разрядов.

Рассмотрим пример: пусть имеются три переменные типа double: x, y, z:

double x = 1.000000000000001; *// Фактическое значение: 1 + 5\*2^{-52}*

double y = 1.000000000000002; *// Фактическое значение: 1 + 9\*2^{-52}*

double z = y - x; *// Теоретически: 1.0\*10^{-15}*

*// Практически: 8.88\*10^{-16} (11% ошибка)*

По логике, должен быть равен , но на практике ответ будет равен , что дает 11% относительной ошибки.

В алгоритме проблема катастрофического сокращения особенно критична при вычислении разности . Когда эти значения очень близки:

1. Результат может стать нулевым, вызывая в конечном итоге деление на ноль,
2. Или слишком маленькое число, которое, при делении на него, приведет к экспоненциальному росту ошибки.

Для решения этой проблемы существуют следующие решения:

**Учетверенная точность**

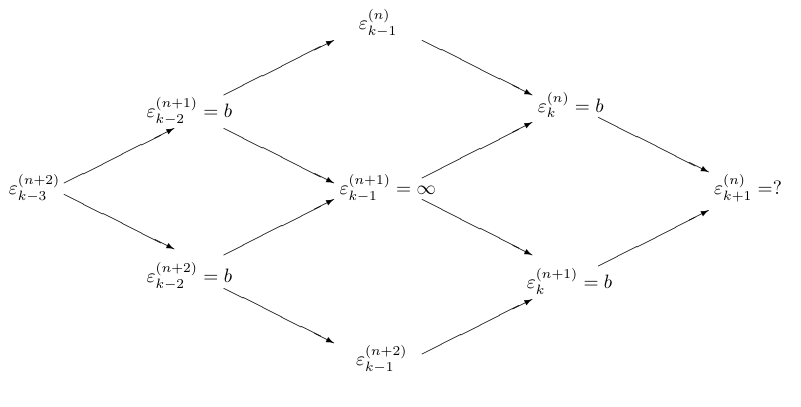
Использование форматов числа четвертой точности 128-битных чисел (float 128/Quad) [3, стр. 19] отодвигает порог возникновения проблемы до ~, но не устраняет её полностью.

**Модификация Винна**

Питер Винн предложил специальное правило для случаев, когда , [3, стр. 20] что вызывает:

* Бесконечность в ,
* Неопределённость в

*Графическое представление проблемы:*

*Модифицированная формула:*

*Ромбовидная схема вычислений   
[3, стр. 20]*

(3.1)   
[3, стр. 21]

Эта формула позволяет нам проигнорировать и , однако требует хранить в памяти целый набор из 4 переменных: и для получения .

Программная реализация этой формулы описана в файле: ***epsilon\_algorithm\_two.h.***   
Для хранения столбцов -таблицы используется четырехмерный массив, кроме того, что бы избежать возможной ошибки в виде деления на 0, которая все равно может возникнуть при очень большом катастрофическом сокращении, применяем резервное правило (если правило 3.1 не позволило избежать ошибки):

Критерий остановки алгоритма:

Можно использовать и другой метод борьбы с катастрофическим сокращением: создание константы , которая остановит алгоритм при достижении необходимой абсолютной погрешности.

* Абсолютная погрешность: (если предел S известен),
* Относительная погрешность: (если предел неизвестен).

1. **Итерационные методы**

Скалярный ε-алгоритм позволяет значительно ускорить сходимость численных методов, таких как метод простой итерации и метод Ньютона.

**Метод простой итерации (метод неподвижной точки)**

Цель задач с фиксированной точкой состоит в том, чтобы найти действительное число такое, что для

Метод решения задачи с фиксированной точкой заключается в построении последовательности, генерируемой даёт

,

пока последовательность не сойдется. Критерием остановки является с небольшим положительным вещественным числом. Построение итераций с фиксированной точкой имеет то преимущество, что оно простое в реализации. Однако сходимость сгенерированной последовательности к решению не гарантирована (она зависит от некоторых свойств функции f) и часто является линейной с низкой скоростью сходимости. Метод простой, но может расходиться или сходиться медленно (обычно линейная скорость).

**Метод Ньютона**

Мы используем метод Ньютона, когда хотим найти действительное число

такое, что для

Метод Ньютона — это итерационный метод.

S₀ — задано

, ,

пока последовательность (Sn)n не сойдется. Критерием остановки является |f(Sn)|≤ δ, где δ – малое положительное действительное число.

Метод Ньютона обладает квадратичной сходимостью (порядка 2) при определенных допущениях, и это его главное преимущество. Однако для этого требуется производная от функции f.

ε-алгоритм в этом случае не ускоряет сходимость, поскольку метод Ньютона уже сходится быстро. Более того, иногда ε-алгоритм даже ухудшает точность из-за численных ошибок.

Вывод: ε-алгоритм эффективен при применении к медленно сходящимся методам (линейная сходимость), но не обязательно улучшает результат при применении к быстро сходящимся (высокого порядка) методам.

1. **Векторный ε-алгоритм**

Векторная форма ε-алгоритма также была изучена П. Винном. Формулировка этого алгоритма основана на выражении (2.3) скалярного ε-алгоритма и адаптирована для работы с векторами. Векторная версия ε-алгоритма позволяет ускорять сходимость последовательностей векторов, в отличие от скалярного случая. Однако, поскольку операции деления векторов не определены, необходимо изменить формулу. Векторный ε-алгоритм, предложенный П. Винном, использует скалярные произведения для замены делений.

Пусть (Sn)n - векторная последовательность с Sn ∈ ℝ d, для n ∈ N.

где для фиксированных значений n и k, εk(n) ∈ ℝd

Чтобы применить этот алгоритм, мы должны определить значение, обратное вектору.   
εk(n+1) - εk(n) — это вектор в ℝd. Для этого П. Винн определил обратную величину вектора **v** = (ui)1≤i≤d ∈ ℝd.

с точечным произведением (v; v), определяемым

Поскольку этот алгоритм основан на формулировке скалярного ε-алгоритма, векторный ε-алгоритм представлен таким же образом. Действительно, мы можем построить такую же таблицу с двойной записью, где элементы последовательности (Sn)n расположены во втором столбце. Однако величины εk(n) будут векторами. Мы продвигаемся слева направо и сверху вниз по таблице ε. Более того, для каждого нового элемента исходной последовательности (Sn)n мы можем построить восходящую диагональ в таблице ε.

Все еще основываясь на результатах скалярного ε-алгоритма, только величины с еще более низким индексом могут быть интегрированы в новую последовательность, созданную ε-алгоритмом. Мы также можем найти правило пересечения для векторного ε-алгоритма, которое является:

# Модификации

Итерированный и -алгоритм прекрасно подходят для ускорения линейно сходящихся последовательностей, а так же многих расходящихся. Однако, и те и другие не эффективны в случае логарифмической сходимости. Для решения этой проблемы Винн создал – алгоритм (Rho – алгоритм):

(6.1)

Алгоритм эффективен для ускорения в случае логарифмической сходимости, но абсолютно не подходит для линейной и, тем более, для расходящихся рядов.

Попыткой получить преимущества обоих версий алгоритмов был -алгоритм (тета – алгоритм), разработанный в 1971 году Брезински [2].

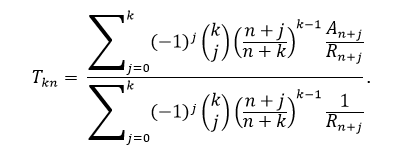
Как и в случае с -алгоритмом и – алгоритмом, -алгоритм дает практически применимые значения только в случаях четных , нечетные значения являются лишь вспомогательными данными.

Тета алгоритм оказался удачным экспериментом, и он позволяет получить стабильно хорошие результаты для большого количества различных рядов. Возможны реализации через один трехмерный массив или один двумерный [5].

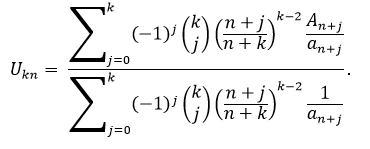
Алгоритм Левина - основан на нелинейных преобразованиях взвешенных частичных сумм.

Имеет три модификации:

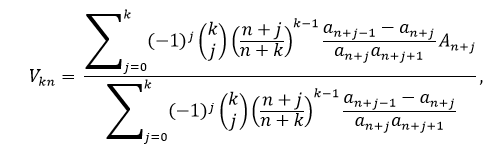
Tkn - преобразование — нужен для знакопеременных рядов.



Ukn - преобразование — для монотонных медленно сходящихся рядов.



Vkn - преобразование — универсальный вариант.



Алгоритм Левина не требует четкого разделения на четные/нечетные шаги в отличие от Ѳ-алгоритма, из-за чего усложняется реализация. К тому же алгоритм Левина благодаря своим модификациям позволяет выбирать тип преобразования под конкретный ряд.

Алгоритм Левина ускоряет сходимость линейно сходящихся рядов с помощью линейных комбинаций частичных сумм, а Алгоритм Левина-Сиди более универсален. Он ускоряет сходимость как линейно сходящихся, так и медленно сходящихся или расходящихся рядов.

Анализируя модификации -алгоритма, а так же алгоритм Левина **(см. алгоритм Левина)**, Левина - Сиди, Ченг создал эффективный алгоритм, схожий по параметрам даже иногда превосходящий с -алгоритмом [6].

Данный алгоритм можно представить в виде одного двумерного массива и одного одномерного. Реализация находится в файле ***chang\_whynn\_algorithm.h.***

# Заключение

В качестве вывода можно отметить, что ε-алгоритм представляет собой эффективный способ ускорения сходимости числовых последовательностей, особенно в тех случаях, когда используется метод простой итерации и наблюдается линейная скорость сходимости. Его применение позволяет существенно сократить количество итераций и общее время вычислений при минимальных затратах.

Однако данный метод оказывается малоэффективным для последовательностей с уже высокой скоростью сходимости, например, квадратичной, как в методе Ньютона. Кроме того, логарифмически сходящиеся последовательности также слабо поддаются ускорению с помощью ε-алгоритма.

Таким образом, ε-алгоритм стоит рассматривать как практичный и доступный инструмент для определённого класса задач, в частности — для линейно сходящихся итеративных методов. В более сложных случаях целесообразно исследовать альтернативные подходы, такие как ρ-алгоритм или θ-алгоритм, обладающие иными свойствами и потенциально более высокой эффективностью при работе с «трудными» последовательностями.

# Литература

[1] Ионкин Н.: [Лекции по курсу «Численные методы»](http://cmcstuff.esyr.org/vmkbotva-r15/3%20%D0%BA%D1%83%D1%80%D1%81/6%20%D0%A1%D0%B5%D0%BC%D0%B5%D1%81%D1%82%D1%80/%D0%A7%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%BD%D1%8B%D0%B5%20%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D1%8B/%D0%BB%D0%B5%D0%BA%D1%86%D0%B8%D0%B8.pdf) (2019) 55-56 стр.

[2] Brezinski, C.: [Algorithmes d’Accel´ eration de la Convergence— ´ Etude Num ´ erique. ´ Editions Technip, ´ Paris](https://books.google.ru/books?id=TxDghaunVjkC&printsec=frontcover&hl=ru#v=onepage&q&f=false) (1978) Chapter 4.3.2

[3]Clément V.: [Acceleration of convergence for numerical sequences](https://hal.science/hal-04207550/document) (2023) 18-27 стр.

[4] Weniger, E.: [Nonlinear sequence transformations for the acceleration of convergence and the summation of divergent series](https://arxiv.org/abs/math-ph/0306302) (1989) pp. 279–281

[5] Xiang-Ke C.: [Construction of new generalizations of Wynn’s epsilon and rho algorithm by solving finite difference equations in the transformation order](https://link.springer.com/article/10.1007/s11075-019-00695-w) (2019) 25 стр.

[] Steele J.: [SOME RESULTS CONCERNING THE FUNDAMENTAL NATURE OF WYNN'S VECTOR EPSILON ALGORITHM](https://www.researchgate.net/publication/344177013_SOME_RESULTS_CONCERNING_THE_FUNDAMENTAL_NATURE_OF_WYNN'S_VECTOR_EPSILON_ALGORITHM) (2002) 21-23 стр.