**Постановка математической задачи для Θ – алгоритма**

Дано: медленно сходящаяся последовательность , где – частичные суммы ряда

Условие на сходимость:

1. Последовательность сходится к пределу (т.е. ), но делает это медленно.

Возможные формы :

1. Экспоненциальная: .
2. Рациональная: , d.
3. Смешанные из первых двух.

Цель: обеспечить более быструю сходимость ряда к по сравнению с исходной последовательностью.

**Θ – алгоритм**

Как показано в работе Брезински [1] классический ε-алгоритм может быть представлен в виде:

Нам важно то, что данная формула подчеркивает двухшаговую природу алгоритма, где каждый новый элемент зависит от элементов на двух предыдущих уровнях.

Применяя оператор конечной разности , можно получить соотношение:

В данном контексте оператор действует исключительно на верхние индексы алгоритма, а не на значения последовательности. Для произвольной величины , зависящей от индекса , он определяется как:

Этот оператор анализирует динамику алгоритма, путем отслеживания изменений результатов при переходе от индекса к .

Пример вычислений:

1. Для ε-алгоритма:
2. Для :

Далее рассмотрим условие ускорения сходимости:

Для его выполнения необходимо и достаточно ([1] глава 2.9), чтобы:

Доказательство следует из разложения отношения разностей и анализа предельного поведения компонент. Более подробно об этом пишет Брезински [1].

В случаях, когда условие (5) не выполняется, вводится дополнительный параметр :

Оптимальным образом определить значение можно так:

При таком выборе параметра последовательность будет сходиться быстрее, чем

На практике довольно часто вычисление предела затруднительно, поэтому можно использовать оценку [1, глава 2.9]:

Рассмотрим полную схему Θ-алгоритма. Для удобства будем использовать обозначения вместо .

Инициализация:

Рекуррентные правила:

Формула (11) соответствует нечетному порядку и напрямую следует из определения ε-алгоритма (формула (2)).

Формула (12) соответствует четному порядку . При четных порядках необходимо ввести дополнительный параметр (описан в формуле (9)), который позволит лучше ускорять сходимость путем компенсации главного члена погрешности. Используя этот параметр как коэффициент при , можно добиться ускорения сходимости последовательности.

Иначе формулу четного порядка можно представить следующим образом:

называется второй разностью и вычисляется по формуле:

Численные эксперименты [1] показали, что результаты работы чаще всего почти так же хороши, как и лучшие результаты аналогичных алгоритмов.

**Теорема 1**.

Необходимое и достаточное условие того, что , заключается в том, что имеет одну из следующих форм:

1. Экспоненциальная:

Данная последовательность сходится при условии

Доказательство:

Рассмотрим последовательность из формулы (15), обозначив .

Учитывая формулу (13), получим:

Вычислим :

Вторая разность вычисляется как:

Выполнив подстановку, получим:

1. Рациональная:

Сходимость этой последовательности достигается тогда, когда вещественная часть d строго положительна.

Для удобства обозначим:

Тогда формула (16) примет вид:

Согласно формуле (13) для четного порядка

Выразим

Выразим :

Вторая разность определяется как:

Выразим

Подставим в и после упрощения получим:

После подстановки имеющих значений в формулу (13) получается за счет взаимного уничтожения других членов.

1. Специальные вырожденные случаи при

Сходимость этой последовательности достигается тогда, когда вещественная часть d строго положительна. Доказательство аналогично рациональному случаю и использует рекуррентное свойство произведения. Подробное доказательство теоремы можно найти в книге Брецинского [1] в главе 2.9 (теорема 2.36).

Θ-алгоритм, как показано в теореме 1, эффективно работает с последовательностями экспоненциального и рационального типов, включая их смешанные формы. Численные эксперименты подтверждают его способность ускорять сходимость даже для сложных случаев, таких как логарифмические последовательности. Кроме того, алгоритм демонстрирует устойчивость к колебаниям членов последовательности благодаря введению параметра , который компенсирует главный член погрешности.

**Реализация алгоритма**

|  |
| --- |
| Вход:  n: int - количество членов ряда для частичной суммы  order: int - порядок преобразования (должен быть четным)  series: callable - функция, возвращающая a\_k (k-й член ряда)  Выход:  T - ускоренное значение частичной суммы  Получить n, order, series  if order % 2 != 0 или order < 0:  return "Ошибка: Порядок должен быть четным неотрицательным числом"  if n < 0:  return "Ошибка: n не может быть отрицательным"  if n == 0 или order == 0:  return sum\_{k=0}^n series(k) # Частичная сумма S\_n  Инициализировать:  S\_n = sum\_{k=0}^n series(k)  j = 0  Вычислить Theta(n, order, S\_n, j, series):  Функция Theta(n, order, S\_n, j, series):  if order == 1:  res = 1 / series(n + j + 1)  if not isfinite(res):  return "Ошибка: Деление на ноль"  return res  # Обновление частичной суммы  for k от n+1 до n+j:  S\_n += series(k)  n = n + j  if order == 0:  return S\_n  order1 = order - 1  order2 = order - 2  # Рекурсивные вызовы  t1\_0 = Theta(n, order1, S\_n, 0, series)  t1\_1 = Theta(n, order1, S\_n, 1, series)  t1\_2 = Theta(n, order1, S\_n, 2, series)  t2\_1 = Theta(n, order2, S\_n, 1, series)  if order % 2 != 0: # Нечетный порядок  delta = 1 / (t1\_0 - t1\_1)  if not isfinite(delta):  return "Ошибка: Деление на ноль"  return t2\_1 + delta  else: # Четный порядок  delta2 = 1 / (-2\*t1\_1 + t1\_0 + t1\_2)  if not isfinite(delta2):  return "Ошибка: Деление на ноль"  delta\_n = t2\_1 - Theta(n, order2, S\_n, 2, series)  delta\_n1 = t1\_1 - t1\_2  return t2\_1 + (delta\_n \* delta\_n1 \* delta2) |

**Рисунок 2 –** Псевдокод алгоритма Тета-Брезински

**Список литературы**

1. BrezinskiC. / *Extrapolation Methods: Theory and Practice* / C. Brezinski, M. Redivo Zaglia. — Amsterdam : North-Holland, 1991. — 353 p.