

## Введение

Алгоритм Форда-Сиди был предложен в 1987 году математиками Уильямом Фордом и Авраамом Сиди для решения задачи ускорения сходимости рядов. Данный метод был предложен в [1] как реализация рекурсивного алгоритма генерализации процесса экстраполяции Ричардсона, представленного в [2], позволяющая решить задачу экстраполяции более быстро и эффективно по сравнению с непосредственным решением системы линейных уравнений, определяющих экстраполяцию. Также, стоит отметить, что данный алгоритм схож с  $E$ -алгоритмом и, более того, математически эквивалентен  $E$ -алгоритму, но при этом является более вычислительно эффективным.

## Экстраполяция Ричардсона

Рассмотрим основную идею процесса экстраполяции Ричардсона [6].

Пусть  $\{S_n\}$  – последовательность, а  $(g_j(n)), j = 1, 2, \dots$  – известные вспомогательные последовательности. Предположим, что существуют неизвестные константы  $(c_j), j = 1, \dots, k$ , такие что:

$$S_{n+i} = T_k^{(n)} + \sum_{j=1}^k c_j g_j(n+i), i = 0, \dots, k. \quad (1)$$

Если система линейных уравнений (1) является невырожденной, то по правилу Крамера  $T_k^{(n)}$  можно выразить как отношение двух определителей:

$$T_k^{(n)} = \frac{\begin{vmatrix} S_n & \dots & S_{n+k} \\ g_1(n) & \dots & g_1(n+k) \\ \vdots & \ddots & \vdots \\ g_k(n) & \dots & g_k(n+k) \end{vmatrix}}{\begin{vmatrix} 1 & \dots & 1 \\ g_1(n) & \dots & g_1(n+k) \\ \dots & \ddots & \vdots \\ g_k(n) & \dots & g_k(n+k) \end{vmatrix}}. \quad (2)$$

При этом данная формулировка позволяет наиболее обобщенно определить процесс экстраполяции, что позволяет использовать различные известные преобразования (трансформации) рядов вида  $(S_n) \mapsto (T_k^{(n)})$  для ускорения сходимости изначального ряда  $\{S_n\}$  [3].

$E$ -алгоритм, предложенный независимо Шнайдером [6],

Основная проблема заключается в необходимости вычисления (2), решаемая различными рекурсивными алгоритмами. При этом, большая часть алгоритмов рассчитана на использование конкретного преобразования ряда, что приводит к необходимости использования различных алгоритмов для различных трансформаций рядов. Известны два рекурсивных алгоритма для вычисления  $T_k^{(n)}$ :

- 1) *E*-алгоритм, предложенный независимо К. Брезински [3], Т. Хейви [4] и К. Шнайдером [5]. Шнайдер и Хёви вывели его методом Гауссова исключения, а Брезински – с использованием тождества Сильвестра для определителей.
- 2) Алгоритм Форда-Сиди [1], который требует меньшего числа арифметических операций, чем *E*-алгоритм. Форд и Сиди вывели свой алгоритм, используя тождество Сильвестра.

Форд и Сиди [1] отметили основное различие в рекурсии *E*-алгоритма и алгоритма Форда-Сиди. Брезински и Редиво Дзалья [7] вывели оба алгоритма, используя операторы аннигиляции разностей, и установили связи между ними.

Далее покажем, что *E*-алгоритм и алгоритм Форда-Сиди математически эквивалентны в следующем смысле: оба алгоритма вычисляют одни и те же величины, но разными способами, и рекуррентные соотношения *E*-алгоритма могут быть выведены из соотношений алгоритма Форда-Сиди, и наоборот. Также предложим эффективную реализацию алгоритма Форда-Сиди, которая немного экономичнее оригинальной.

## $E$ -алгоритм

Пусть  $S = (S_n)$  - любая последовательность, которую нужно преобразовать, а  $(g_j(n)), j = 1, 2, \dots$  - любые вспомогательные последовательности. Обозначим 1 как постоянную последовательность с  $1_n = 1$  для  $n = 0, 1, \dots$ . Предположим, что все знаменатели ненулевые.

$E$ -алгоритм определяется следующим образом. Для  $n = 0, 1, \dots$  величины  $E_k^{(n)}$  и  $g_{k,j}^{(n)}$  определяются как

$$E_0^{(n)} = S_n,$$

$$g_{0,j}^{(n)} = g_j(n),$$

$$j = 1, 2, \dots,$$

$$E_k^{(n)} = \frac{E_{k-1}^{(n)} g_{k-1,k}^{(n+1)} - E_k^{(n+1)} g_{k-1,k}^{(n)}}{g_{k-1,k}^{(n+1)} - g_{k-1,k}^{(n)}}, k = 1, 2, \dots, \quad (3)$$

$$g_{k,j}^{(n)} = \frac{g_{k-1,j}^{(n)} g_{k-1,k}^{(n+1)} - g_{k-1,j}^{(n+1)} g_{k-1,k}^{(n)}}{g_{k-1,k}^{(n+1)} - g_{k-1,k}^{(n)}}, k = 1, 2, \dots, \quad (4)$$

$$j = k + 1, \dots$$

Рекуррентные соотношения (3) и (4) называются основным правилом и вспомогательным правилом  $E$ -алгоритма соответственно. Брезински [3] доказал следующую теорему с использованием тождества детерминантов Сильвестра.

Теорема 1. Для  $n = 0, 1, \dots$  и  $k = 1, 2, \dots$ ,  $E_k^{(n)}$  и  $g_{k,j}^{(n)}$  представляются как

$$E_k^{(n)} = \frac{\begin{vmatrix} S_n & \dots & S_{n+k} \\ g_1(n) & \dots & g_1(n+k) \\ \vdots & \ddots & \vdots \\ g_k(n) & \dots & g_k(n+k) \end{vmatrix}}{\begin{vmatrix} 1 & \dots & 1 \\ g_1(n) & \dots & g_1(n+k) \\ \dots & \ddots & \vdots \\ g_k(n) & \dots & g_k(n+k) \end{vmatrix}},$$

$$g_{k,j}^{(n)} = \frac{\begin{vmatrix} g_j(n) & \dots & g_j(n+k) \\ g_1(n) & \dots & g_1(n+k) \\ \vdots & \ddots & \vdots \\ g_k(n) & \dots & g_k(n+k) \end{vmatrix}}{\begin{vmatrix} 1 & \dots & 1 \\ g_1(n) & \dots & g_1(n+k) \\ \dots & \ddots & \vdots \\ g_k(n) & \dots & g_k(n+k) \end{vmatrix}}, \quad j > k.$$

соответственно.

Если положить  $c_k^{(n)} = g_{k-1,k}^{(n)} / (g_{k-1,k}^{(n+1)} - g_{k-1,k}^{(n)})$ , то (3) и (4) становятся

$$E_k^{(n)} = E_{k-1}^{(n)} - c_k^{(n)} (E_{k-1}^{(n+1)} - E_{k-1}^{(n)}), \quad k > 0, \quad (5)$$

$$g_{k,j}^{(n)} = g_{k-1,j}^{(n)} - c_k^{(n)} (g_{k-1,j}^{(n+1)} - g_{k-1,j}^{(n)}), \quad k > 0, \quad j > k, \quad (6)$$

соответственно.

Для заданных  $S_0, \dots, S_N$  вычисление  $E_k^{(n-k)}$ ,  $0 \leq N, 0 \leq k \leq n$ , требует  $\frac{1}{3}N^3 + O(N^2)$  операций. Количество операций для  $E$ -алгоритма, как упомянуто в [3], составляет  $\frac{5}{3}N^3 + O(N^2)$ , в то время как с использованием (5) и (6) оно становится  $N^3 + O(N^2)$ . Более точно, последнее равно  $N^3 + \frac{5}{2}N^2 + \frac{3}{2}N$ .

Отметим, что Форд и Сиди [1] реализовали  $E$ -алгоритм, переписав его в формах

$$E_k^{(n)} = \frac{E_{k-1}^{(n+1)} - c_k^{(n)} E_{k-1}^{(n)}}{d_k^{(n)}}, \quad k > 0, \quad (7)$$

$$g_{k,j}^{(n)} = \frac{g_{k-1,j}^{(n+1)} - c_k^{(n)} g_{k-1,j}^{(n)}}{d_k^{(n)}}, \quad k > 0, \quad j > k, \quad (8)$$

где  $c_k^{(n)} = g_{k-1,k}^{(n+1)} / g_{k-1,k}^{(n)}$ , а  $d_k^{(n)} = 1 - c_k^{(n)}$ .

Реализация с использованием (7) и (8) требует точно такого же количества арифметических операций, как и с использованием (5) и (6). Однако можно избежать потери значимых цифр, используя (5) и (6).

## Алгоритм Форда-Сиди

Алгоритм Форда-Сиди определяется следующим образом. Пусть  $u = (u_n)$  - одна из последовательностей  $S = (S_n)$  или  $g_j = (g_j(n))$ . Величины  $\psi_k^{(n)}(u)$  определяются как

$$\psi_0^{(n)}(u) = \frac{u_n}{g_1(n)}, \quad (9)$$

$$\psi_k^{(n)}(u) = \frac{\psi_{k-1}^{(n+1)}(u) - \psi_{k-1}^{(n)}(u)}{\psi_{k-1}^{(n+1)}(g_{k+1}) - \psi_{k-1}^{(n)}(g_{k+1})}, \quad k > 0. \quad (10)$$

Форд и Сиди [1] доказали следующую теорему с использованием тождества детерминантов Сильвестра следующим образом.

Пусть

$$f_N^j(u) = \begin{vmatrix} g_1(j) & \dots & g_N(j) & u(j) \\ g_1(j+1) & \dots & g_N(j+1) & u(j+1) \\ \vdots & \ddots & \vdots & \vdots \\ g_N(j+N-1) & \dots & g_N(j+N-1) & u(j+N-1) \end{vmatrix}, \quad (11)$$

$$G_p^j = \begin{vmatrix} g_1(j) & \dots & g_p(j) \\ \vdots & \ddots & \vdots \\ g_1(j+p-1) & \dots & g_p(j+p-1) \end{vmatrix}. \quad (12)$$

При этом (2) можно представить в виде:

$$T_k^{(n)} = \frac{f_k^{(n)}(u)}{f_k^{(n)}(g)}. \quad (13)$$

Определим:

$$\psi_p^j(u) = \frac{f_p^j(u)}{G_{p+1}^j}. \quad (14)$$

Следовательно, можно выразить (13) в виде

$$T_k^{(n)} = \frac{\psi_k^{(n)}(S)G_{k+1}^{(n)}}{\psi_k^{(n)}(1)G_{k+1}^{(n)}} = \frac{\psi_k^{(n)}(S)}{\psi_k^{(n)}(1)}. \quad (15)$$

Заметим, что  $f_p^j(u)$  и  $G_{p+1}^j$  с точки зрения определителя отличаются лишь последним столбцом, поэтому будет естественно вывести соотношение между ними. Для этого воспользуемся теоремой Сильвестра.

Теорема 2 (теорема Сильвестра).

Пусть  $C$  – матрица,  $C_{\rho\sigma}$  – матрица, получаемая из  $C$  путем удаления строки  $\rho$  и столбца  $\sigma$ ,  $C_{\rho\rho'\sigma\sigma'}$  – матрица, получаемая из  $C$  путем удаления строк  $\rho$  и  $\rho'$  и столбцов  $\sigma$  и  $\sigma'$ ,  $\rho < \rho'$ ,  $\sigma < \sigma'$

Тогда

$$\det C \det C_{\rho\rho'\sigma\sigma'} = \det C_{\rho\sigma} \det C_{\rho'\sigma'} - \det C_{\rho\sigma'} \det C_{\rho'\sigma}. \quad (16)$$

При этом, если  $C$  – матрица  $2 \times 2$ , то  $\det C_{\rho\rho'\sigma\sigma'} = 1$ .

Если применить данную теорему к  $f_p^j(u)$  размером  $(p+1) \times (p+1)$  при  $\rho = 1$ ,  $\sigma = p$ ,  $\sigma' = \rho' = p+1$  то получим:

$$\det C \det C_{\rho\rho'\sigma\sigma'} = \det C_{\rho\sigma} \det C_{\rho'\sigma'} - \det C_{\rho\sigma'} \det C_{\rho'\sigma}. \quad (17)$$

Тогда можем получить следующее соотношение:

$$f_p^j(u) G_{p-1}^{j+1} = f_{p-1}^{j+1}(u) G_p^j - f_{p-1}^j(u) G_p^{j+1}. \quad (18)$$

Пусть:

$$D_p^j = \frac{G_{p+1}^j G_{p-1}^{j+1}}{G_p^j G_p^{j+1}}. \quad (19)$$

Тогда из (18) и (19) можно выразить

$$\psi_p^j(u) = \frac{[\psi_{p-1}^{j+1}(u) - \psi_{p-1}^j(u)]}{D_p^j}. \quad (20)$$

Из (16) и (20) можем видеть, что  $\psi_k^{(n)}(u)$  и  $T_k^{(n)}$  могут быть вычислены рекурсивно. Следовательно, задача сводится к эффективному вычислению  $D_p^{(j)}$ . При этом учитывая отсутствия конкретных данных о  $g_i(n)$ , учитывая, что  $G_{p+1}^{(j)} = f_p^j(g_{p+1})$  согласно (4) и (5) можем сделать вывод, что:

$$\psi_p^j(g_{p+1}) = 1. \quad (21)$$



Из этого можем выразить:

$$D_p^{(j)} = \psi_{p-1}^{(j+1)}(g_{p+1}) - \psi_{p-1}^{(j)}(g_{p+1}). \quad (22)$$

Из чего следует, что

$$\psi_p^{(n)}(u) = \frac{\psi_{p-1}^{(n+1)}(u) - \psi_{p-1}^{(n)}(u)}{\psi_{p-1}^{(n+1)}(g_{k+1}) - \psi_{p-1}^{(n)}(g_{k+1})}. \quad (23)$$

То есть (15) и (23) позволяет рекурсивно решить поставленную задачу экстраполяции.

Рассмотрим непосредственно сам алгоритм:

1) Для  $j = 0, 1 \dots$  вычислим:

$$\psi_0^{(j)}(s) = \frac{s^{(j)}}{g_1^{(j)}}, \psi_0^{(j)}(1) = \frac{1}{g_1^{(j)}}, \psi_0^{(j)}(g_k) = g_k, k = 2, 3 \dots$$

2) Для  $j = 0, 1, \dots$  и  $p = 1, 2, \dots$  вычислим

$$D_p^{(j)} = \psi_{p-1}^{(j+1)}(g_{p+1}) - \psi_{p-1}^{(j)}(g_{p+1}).$$

А также вычислим

$$\psi_p^{(j)}(u) \text{ при } u = s, u = 1, u = g_k, k = p + 2, p + 3 \dots$$

$$\psi_p^{(j)}(u) = \frac{\psi_{p-1}^{(j+1)}(u) - \psi_{p-1}^{(j)}(u)}{D_p^{(j)}}.$$

И, соответственно, вычислим:

$$T_k^{(n)} = \frac{\psi_k^{(n)}(S)}{\psi_k^{(n)}(1)}.$$

Процесс вычисления  $\psi_p^{(j)}(u)$  можно представить следующим образом:

$\psi_0^{(0)}(u)$			
$\psi_0^{(1)}(u)$	$\psi_1^{(0)}(u)$		
$\vdots$	$\vdots$	$\ddots$	
$\psi_0^{(L)}(u)$	$\psi_1^{(L-1)}(u)$	$\dots$	$\psi_L^{(0)}(u)$

Таблица 1. Представление массива  $\psi_p^{(j)}(u)$

Замечание. Несмотря на кажущуюся необходимость в вычислении  $g_{L+1}$ , для вычисления каждого  $T_p^{(j)}$ ,  $0 \leq j + p \leq L$ , на практике, это нужно только для  $D_L^{(0)}$ , чтобы затем вычислить  $\psi_L^{(0)}(S)$  и  $\psi_L^{(0)}(1)$ , и, соответственно,

$T_L^{(0)}$ . Однако, как было отмечено в [6], на практике этих вычислений можно избежать, вычислив  $T_n^{(0)}$ ,  $n = 1, 2, \dots$  по формуле:

$$T_k^{(0)} = \frac{\psi_{k-1}^{(1)}(S) - \psi_{k-1}^{(0)}(S)}{\psi_{k-1}^{(1)}(1) - \psi_{k-1}^{(0)}(1)}. \quad (24)$$

Теорема 3. Для  $n = 0, 1, \dots$ , и  $k = 1, 2, \dots$ ,  $\psi_k^{(n)}(u)$  представляются как

$$\psi_k^{(n)}(u) = \frac{\begin{vmatrix} u_n & \dots & u_{n+k} \\ g_1(n) & \dots & g_1(n+k) \\ \vdots & \ddots & \vdots \\ g_k(n) & \dots & g_k(n+k) \end{vmatrix}}{\begin{vmatrix} g_{k+1}(n) & \dots & g_{k+1}(n+k) \\ g_1(n) & \dots & g_1(n+k) \\ \dots & \ddots & \vdots \\ g_k(n) & \dots & g_k(n+k) \end{vmatrix}}.$$

Следовательно, по теоремам 1 и 3,  $T_k^{(n)}$  можно вычислить как

$$T_k^{(n)} = \frac{\psi_k^{(n)}(S)}{\psi_k^{(n)}(1)}. \quad (25)$$

Следуя реализации Форда и Сиди [1], вычисление  $T_k^{(n-k)}$ ,  $0 \leq n \leq N$ ,  $0 \leq k \leq n$ , требует  $\frac{1}{3}N^3 + \frac{3}{2}N^2 + \frac{7}{6}N$  вычитаний и  $\frac{1}{3}N^3 + \frac{5}{2}N^2 + \frac{25}{6}N + 2$  делений, всего  $\frac{2}{3}N^3 + 4N^2 + \frac{16}{3}N + 2$  арифметических операций.

Замечание. (1) Алгоритм Форда-Сиди требует  $g_{k+1}$  для вычисления  $\psi_k^{(n)}$ . Однако для вычисления  $T_k^{(n)}$  алгоритм Форда-Сиди не требует  $g_{k+1}$ . Причина кроется в следующем: пусть  $a_1, \dots, a_{k+1}$  - любая последовательность, такая что

$$\begin{vmatrix} a_1 & \dots & a_{k+1} \\ g_1(n) & \dots & g_1(n+k) \\ \vdots & \ddots & \vdots \\ g_k(n) & \dots & g_k(n+k) \end{vmatrix} \neq 0.$$

Пусть

$$\bar{\psi}_k^{(n)}(u) = \frac{\begin{vmatrix} u_1 & \dots & u_{k+1} \\ g_1(n) & \dots & g_1(n+k) \\ \vdots & \ddots & \vdots \\ g_k(n) & \dots & g_k(n+k) \end{vmatrix}}{\begin{vmatrix} a_1 & \dots & a_{k+1} \\ g_1(n) & \dots & g_1(n+k) \\ \vdots & \ddots & \vdots \\ g_k(n) & \dots & g_k(n+k) \end{vmatrix}}.$$

Тогда из (25) следует

$$T_k^{(n)} = \frac{\bar{\psi}_k^{(n)}(S)}{\bar{\psi}_k^{(n)}(1)}.$$

Используя этот прием, когда заданы  $S_0, \dots, S_N, g_1, \dots, g_N$ , можно определить все  $T_k^{(n)}, 0 \leq n+k \leq N$ , с помощью алгоритма Форда-Сиди.

(2) Более того, ни значение  $g_{k+1}$ , ни последовательность  $a_1, \dots, a_{k+1}$  не требуются для вычисления  $T_k^{(n)}$ , когда используется

$$T_k^{(n)} = \frac{\psi_{k-1}^{(n+1)}(S) - \psi_{k-1}^{(n)}(S)}{\psi_{k-1}^{(n+1)}(1) - \psi_{k-1}^{(n)}(1)},$$

что выводится из (10) и (25). Реализация этого факта и есть эффективный алгоритм Форда-Сиди, которая будет описана позже.

## Вывод алгоритма Форда-Сиди из $E$ -алгоритма

Пусть  $u = (u_n)$  – любая последовательность. Предположим, что преобразованная последовательность  $E_k$ : ли  $u = (u_n) \mapsto E_k(u) = (E_k^{(n)}(u))$  определены как

$$E_0^{(n)} = u_n, \quad (26)$$

$$E_k^{(n)}(u) = \frac{E_{k-1}^{(n)}(u)E_{k-1}^{(n+1)}(g_k) - E_{k-1}^{(n+1)}(u)E_{k-1}^{(n)}(g_k)}{E_{k-1}^{(n+1)}(g_k) - E_{k-1}^{(n)}(g_k)}, \quad (27)$$

$$k > 0.$$

Предположим, что преобразования последовательностей  $\psi_k: u = (u_n) \mapsto \psi_k(u) = (\psi_k^{(n)}(u))$  определены как

$$\psi_k^{(n)}(u) = \frac{E_k^{(n)}(u)}{E_k^{(n)}(g_{k+1})}, \quad k = 0, 1, \dots \quad (28)$$

Теорема 4. Величины  $\psi_k^{(n)}(u)$ , определенные (28), удовлетворяют (9) и (10).

Доказательство. Из (26) и (28) следует, что  $\psi_0^{(n)}(u) = u_n/g_1(n)$ .

Используя математическую индукцию по  $k$ , легко доказать, что  $E_k^{(n)}(1) = 1$ .

Таким образом, имеем

$$\psi_k^{(n)}(1) = \frac{1}{E_k^{(n)}(g_{k+1})}, \quad k = 0, 1, \dots \quad (29)$$

Из (28) следует

$$\psi_k^{(n)}(g_j) = \frac{E_k^{(n)}(g_j)}{E_k^{(n)}(g_{k+1})}, \quad k = 0, 1, \dots; \quad j = k + 1, k + 2, \dots \quad (30)$$

Из (27) и (30) получаем

$$\begin{aligned}
\psi_{k-1}^{(n+1)}(g_{k+1}) - \psi_{k-1}^{(n)}(g_{k+1}) &= \frac{E_{k-1}^{(n+1)}(g_{k+1})}{E_{k-1}^{(n+1)}(g_k)} - \frac{E_{k-1}^{(n)}(g_{k+1})}{E_{k-1}^{(n)}(g_k)} = \\
&= \frac{E_{k-1}^{(n)}(g_{k+1})E_{k-1}^{(n+1)}(g_k) - E_{k-1}^{(n+1)}(g_{k+1})E_{k-1}^{(n)}(g_k)}{E_{k-1}^{(n+1)}(g_k)E_{k-1}^{(n)}(g_k)} = \\
&= E_k^{(n)}(g_{k+1}) \left( \frac{1}{E_{k-1}^{(n+1)}(g_k)} - \frac{1}{E_{k-1}^{(n)}(g_k)} \right). \tag{31}
\end{aligned}$$

Разделив обе части (13) на  $E_k^{(n)}(g_{k+1})$ , получаем

$$\frac{E_k^{(n)}(u)}{E_k^{(n)}(g_{k+1})} = \frac{\frac{E_{k-1}^{(n)}(u)}{E_{k-1}^{(n)}(g_k)} - \frac{E_{k-1}^{(n+1)}(u)}{E_{k-1}^{(n+1)}(g_k)}}{E_k^{(n)}(g_{k+1}) \left( \frac{1}{E_{k-1}^{(n)}(g_k)} - \frac{1}{E_{k-1}^{(n+1)}(g_k)} \right)},$$

следовательно, из (14) и (17) получаем

$$\psi_k^{(n)}(u) = \frac{\psi_{k-1}^{(n+1)}(u) - \psi_{k-1}^{(n)}(u)}{\psi_{k-1}^{(n+1)}(g_{k+1}) - \psi_{k-1}^{(n)}(g_{k+1})}.$$

Отметим, что Брезински и Редиво Залья [7] вывели соотношения (28)-(30) из своих определений  $E_k^{(n)}(u)$  и  $\psi_k^{(n)}(u)$ .

## Вывод E-алгоритма из алгоритма Форда-Сиди

Предположим, что преобразования последовательностей  $\psi_k$  удовлетворяют (9) и (10) для любой последовательности  $u = (u_n)$ . Пусть преобразования последовательностей  $E_k$  определены как

$$E_k^{(n)}(u) = \frac{\psi_k^{(n)}(u)}{\psi_k^{(n)}(1)}. \quad (32)$$

Теорема 4.  $E_k^{(n)}(u)$ , определенные (32), удовлетворяют (26) и (27).

Доказательство. Из (10), имеем

$$\psi_k^{(n)}(g_{k+1}) = 1.$$

Следовательно, по определению (32), получаем

$$E_k^{(n)}(g_{k+1}) = \frac{1}{\psi_k^{(n)}(1)}. \quad (33)$$

Из (32) и (10) следует

$$\begin{aligned} E_k^{(n)}(u) &= \frac{\psi_k^{(n)}(u)}{\psi_k^{(n)}(1)} = \frac{\psi_{k-1}^{(n+1)}(u) - \psi_{k-1}^{(n)}(u)}{\psi_{k-1}^{(n+1)}(1) - \psi_{k-1}^{(n)}(1)} = \\ &= \frac{\frac{E_{k-1}^{(n+1)}(u)}{E_{k-1}^{(n+1)}(g_k)} - \frac{E_{k-1}^{(n)}(u)}{E_{k-1}^{(n)}(g_k)}}{\frac{1}{E_{k-1}^{(n+1)}(g_k)} - \frac{1}{E_{k-1}^{(n)}(g_k)}} = \\ &= \frac{E_{k-1}^{(n+1)}(g_k)E_{k-1}^{(n)}(u) - E_{k-1}^{(n)}(g_k)E_{k-1}^{(n+1)}(u)}{E_{k-1}^{(n+1)}(g_k) - E_{k-1}^{(n)}(g_k)}. \end{aligned}$$

Следовательно, по теоремам 4 и 5, мы считаем, что E-алгоритм и алгоритм Форда-Сиди математически эквивалентны.

## Эффективная реализация для алгоритма Форда-Сиди

Пусть  $\psi_k^{(n)}(u)$  определены (9) и (10). Из (25) и (10) следует, что  $T_k^{(n)}$  в уравнении (2) представляется как

$$T_k^{(n)} = \frac{\psi_k^{(n)}(S) - \psi_{k-1}^{(n)}(S)}{\psi_k^{(n)}(1) - \psi_{k-1}^{(n)}(1)}, \quad k > 0. \quad (34)$$

Вычисление  $T_k^{(n-k)}, 0 \leq n \leq N, 0 \leq k \leq n$  с использованием эффективной реализации для алгоритма Форда-Сиди требует  $\frac{1}{3}N^3 + N^2 + \frac{2}{3}N$  вычитаний и  $\frac{1}{3}N^3 + 2N^2 + \frac{8}{3}N$  делений, всего  $\frac{2}{3}N^3 + 3N^2 + \frac{10}{3}N$  арифметических операций. Хотя количество операций предложенного метода и алгоритма Форда-Сиди асимптотически равны, предложенный метод немного более экономичен, чем алгоритм Форда-Сиди.

Предположим, что мы ускоряем сходимость обычной последовательности с помощью подходящего метода, такого как  $u$ -преобразование Левина в двойной точности, и что  $T_N^{(0)}$  - оптимальное экстраполированное значение. Тогда обычно  $10 \leq N \leq 20$ . (см., например, [8, 9].) Таким образом, предложенный метод на практике на 6–11% более экономичен, чем алгоритм Форда-Сиди.

## Заключение

В данной работе были рассмотрены  $E$ -алгоритм и алгоритм Форда-Сиди, выведена его рекуррентная формула, а также была доказана математическая эквивалентность алгоритма Форда-Сиди и  $E$ -алгоритма. Также была предложена эффективная реализация алгоритма Форда-Сиди, которая немного экономичнее оригинальной.



## Список литературы

1. An Algorithm for a Generalization of the Richardson Extrapolation Process // SIAM J. Numer. Anal. 24 (5) // W. F. Ford, A. Sidi – 1987. – P. 1212–1232.
2. Some Properties of a Generalization of the Richardson Extrapolation Process // Journal of Computational and Applied Mathematics // A. Sidi. – 1979. P. 327-346.
3. A general extrapolation algorithm // Numer. Math. 35 // C. Brezinski. – 1980. – P. 175–187.
4. Generalized neville type extrapolation schemes // BIT 19 // T. Håvie. – 1979. – P. 204–213.
5. Vereinfachte Rekursionen zur Richardson-Extrapolation in Spezialfällen // Numer. Math. 24 // C. Schneider. – 1975. – P. 177– 184.
6. The E-algorithm and the Ford–Sidi algorithm // Journal of computational and applied mathematics 122 (1) // N. Osada. – 2000. P. 223-230.
7. A general extrapolation procedure revisited // Adv. Comput. Math. 2 // C. Brezinski, M. Redivo Zaglia. – 1994. – P. 461-477.
8. Acceleration of linear and logarithmic sequence // SIAM J. Numer. Anal, 16 (2) // D.A. Smith, W.F. Ford. – 1979. - P. 223–240.
9. Extrapolation Methods, Theory and Practice // // C. Brezinski, M. Redivo Zaglia. - 1991.