Плющенков Б.Д., Турчанинов В.И.

Пошаговая свёртка

Рекомендуемая форма библиографической ссылки: Плющенков Б.Д., Турчанинов В.И. Пошаговая свёртка // Препринты ИПМ им. М.В.Келдыша. 2009. № 24. 24 с. URL: http://library.keldysh.ru/preprint.asp?id=2009-24

РОССИЙСКАЯ АКАДЕМИЯ НАУК

ОРДЕНА ЛЕНИНА
ИНСТИТУТ ПРИКЛАДНОЙ МАТЕМАТИКИ
им. М.В. КЕЛДЫША

Б.Д. Плющенков, В.И. Турчанинов

ПОШАГОВАЯ СВЁРТКА

МОСКВА 2009

Б.Д. Плющенков, В.И. Турчанинов

Пошаговая свёртка

Аннотация. Распространение упругих волн в многофазных средах (в средах "с памятью") описывается системой уравнений в частных производных, содержащих интегральные слагаемые, имеющие вид свёрток найденного решения во все предыдущие данному моменты времени с известными ядрами, зависящими от свойств изучаемой среды. Построение экономичных разностных схем для расчета волновых полей сталкивается с трудностью вычисления свёртки, поскольку само по себе количество арифметических операций, приходящееся на вычисление одного значения свёртки, пропорционально числу шагов по времени. В работе предлагается способ преодоления возникающей трудности. Для этого рассматриваются непрерывные рекуррентные алгоритмы. В частности, линейные алгоритмы приводят к методу аппроксимации ядра суммой экспонент, а для произвольных непрерывных алгоритмов доказывается оценка снизу их точности в зависимости от объема используемой ими памяти. На конкретном примере степенного ядра показано, что линейный алгоритм является одним из лучших среди всех непрерывных.

B.D. Plyushchenkov, V.I. Turchaninov

Step-by-step convolution

Abstract. Propagation of elastic waves in multiphase media "with memory" is described by the system of the partial integrodifferential equations. Integral terms of these equations are convolutions of the computed solution in all time moments previous to time moment under consideration with known kernels depending on properties of the studied medium. Construction of economic finite difference schemes for calculation of wave fields are confronted with difficulty of convolution calculation, since quantity of arithmetic operations for calculation of one value of convolution is proportional to number of steps on time. In present work, the way of overcoming of arising difficulty is offered. Continuous recurrent algorithms are considered for this purpose. In particular, linear algorithms lead to a method of kernel approximation by the sum of exponents, and for any continuous algorithms the lower estimate is proved to their accuracy depending on volume of memory used by them. On a concrete example of a power-like kernel it is shown, that the linear algorithm is one of the best among all continuous algorithms.

Введение

В нестационарных линейных уравнениях в частных производных, описывающих распространение возмущений в дисперсионных средах, значения неизвестных в данный момент времени и в данной точке зависят не только от их значений в бесконечно малой временной и пространственной окрестности (дифференциальная связь), но и от их значений в данной точке во все предшествующие моменты времени. Наиболее общий вид такой линейной зависимости — это свёртка по времени этих значений с заданным ядром, модуль которого должен стремиться к нулю при стремлении аргумента к бесконечности для обеспечения конечности решения. Обычно, задачи для таких процессов формулируют для Фурье или Лаплас изображения по времени этих уравнений. Эти уравнения представляют собой линейные уравнения в частных производных по пространственным переменным с зависящими от частоты коэффициентами.

С точки зрения экономичности вычислений, очевидно, что при численном моделировании таких процессов в неоднородных средах предпочтение следует отдать решению уравнений во временном представлении с помощью конечно-разностных или конечно-элементных схем. До сих пор основная трудность при создании таких схем состояла в отсутствии эффективных (по числу операций и необходимому, для обеспечения заданной точности, объему памяти) алгоритмов для вычисления таких сверток.

Мы столкнулись с этой проблемой при моделировании акустического каротажа скважин на основе решения уточненных уравнений Био [Biot62], [Ply00]. Эти уравнения описывают распространения акустических возмущений в неоднородной пористой флюидонасыщенной упругой среде. Из современной теории динамической проницаемости и извилистости в насыщенных жидкостью пористых средах [Joh87] следует, что сила взаимодействия между каркасом и поровой жидкостью − это свёртка по времени линейной комбинации ускорения и скорости их относительного смещения с ядром $\exp(-x)/\sqrt{x}$, где $x = 2\omega_b t$, ω_b − частота Био и t − время. Построение эффективного алгоритма вычисления такой свёртки для случая, когда характерные частоты волн, распространяющихся в пористой среде, больше, чем частота Био (характеристика дисперсионных свойств породы) - нетривиальная задача. А именно этот случай очень важен для моделирования акустического каротажа формаций с большой проницаемостью и моделирования лабораторных испытаний кернов.

Аналогичная проблема возникает при численном моделировании распространении электромагнитных волн в дисперсионных средах и в т.п. задачах.

В данной работе для вычисления свёртки рассматривается семейство ре-

куррентных алгоритмов, которые классифицируются по объему используемой ими памяти. В частности показывается, что линейные с постоянными коэффициентами рекуррентные алгоритмы приводят к методу аппроксимации ядра суммой экспонент (экспоненциальный алгоритм). Для непрерывных рекуррентных алгоритмов дается оценка снизу их точности в зависимости от объема используемой памяти. Используя эту оценку удается показать, что экспоненциальный алгоритм вычисления рекуррентной свёртки со степенным ядром оптимален по порядку величины. Для нескольких конкретных ядер приводятся их аппроксимации суммой экспонент и соответствующие им расчетные соотношения.

Основные результаты данной работы впервые были опубликованы в докладах 2-ой конференции памяти М.А. Био по поромеханике, Гренобль, Франция, 26 − 28 августа 2002 г. [Ply02], а результаты применения экспоненциального алгоритма в явной конечно-разностной схеме для решения модифицированных уравнений Био опубликованы там же [Ply02fd]. В 2003 г. независимо от нас в [Sofr03] был предложен аналогичный алгоритм для вычисления свёртки.

1. Пошаговая свёртка

Определим дискретную свёртку (Лапласа) $u = K * v$ произвольной последовательности $v = \{v_n\}$ с заданной последовательностью (ядром) $K = \{K_n\}$:

$$u_n = (K * v)_n = \sum_{k=0}^{n} K_{n-k}\, v_k, \quad n = 0, 1, \ldots . \tag{1}$$

Задача о пошаговой свёртке состоит в том, чтобы при $n = 0, 1, \ldots$ шаг за шагом вычислять с достаточной точностью значения u_n по мере поступления извне заранее неизвестных значений v_n.

Целью работы является изучение рекуррентных экономичных алгоритмов приближённого вычисления свёртки (1). Свойства рекуррентности и экономичности обусловлены той прикладной задачей, в которую алгоритмы встроены. Обсудим их более подробно.

Работа рекуррентных алгоритмов осуществляется последовательно по шагам $n = 0, 1, 2, \ldots$, начиная с некоторого исходного состояния. На n-ом шаге на вход алгоритма подается v_n, а на выходе получается значение $\tilde{u}_n$, которое мы будем интерпретировать как приближённое значение нашей свёртки: $u_n \approx \tilde{u}_n$.

Требование экономичности алгоритма состоит в том, что его память − пространство состояний S состоит из небольшого количества m вещественных чисел − компонент вектора $s = (s_1, s_2, \ldots, s_m)$. Именно в памяти ал-

горитма накапливается вся информация о предыдущем поведении входной последовательности $\{v_n\}$.

Таким образом, наш алгоритм полностью определяется некоторыми отображениями F_n:

$$\begin{pmatrix} s^{(n)} \\ \tilde{u}_n \end{pmatrix} = F_n \begin{pmatrix} s^{(n-1)} \\ v_n \end{pmatrix}, \quad s^{(-1)} = 0, \quad n = 0, 1, \dots , \tag{2}$$

где $s^{(n)}$ – состояние памяти алгоритма, получившееся сразу после n-ого шага. Начальное состояние $s^{(-1)}$ можно принять за нуль, что, очевидно, не приводит к потере общности. Выходной вектор $\tilde{u}$ будем обозначать также через $\tilde{K}(v)$: $\tilde{K}(v) = \tilde{u} = (\tilde{u}_0, \tilde{u}_1, \tilde{u}_2, ...)$.

Чтобы глубже понять суть рассматриваемой проблемы, приведём два примера.

Хорошо известно, что свёртки можно считать с помощью быстрого преобразования Фурье. Но для быстрого преобразования Фурье требуется знание всей входной последовательности, в отличие от того, что на тот момент, когда нужно вычислить очередное значение свёртки, входная последовательность известна лишь частично.

Непосредственная реализация свёртки по исходной формуле (1) неэкономична, потому что, во-первых, нужно хранить все предыдущие входные значения, во-вторых, число операций, приходящееся на вычисление одного значения свёртки, имеет порядок $O(n)$. На фоне решения всей дифференциальной задачи это приводит к многократному удорожанию всего расчёта. Наша цель состоит в том, чтобы дополнительная память, предназначенная для вычисления свёртки, была величиной порядка $O(1)$. Как следствие этого, время вычисления одного значения свёртки окажется величиной порядка $O(1)$.

С помощью (2) нами выделено достаточно общее семейство рекуррентных алгоритмов и дана их классификация по параметру m – размерности их активного рабочего пространства.

Рассматривая линейные F_n не зависящие от n мы, с помощью несложной линейной алгебры, немедленно приходим к практичному экспоненциальному методу решения нашей задачи.

Рассматривая непрерывные F_n мы даём оценки снизу погрешности алгоритма (2). Эти оценки, как показывает опыт, не являются чересчур заниженными. В частности, с их помощью удаётся показать для некоторых конкретных ядер, что экспоненциальный метод является одним из лучших.

2. Линейные автономные алгоритмы

Изучим линейные автономные алгоритмы (2), то есть алгоритмы, для которых отображения F_n линейны и не зависят от n.

Линейный автономный алгоритм (2) можно записать в виде

$$\begin{array}{rcl} s^{(n)} & = & As^{(n-1)} + v_n B, \quad s^{(-1)} = 0, \\ \tilde{u}_n & = & Cs^{(n-1)} + v_n d, \end{array} \tag{3}$$

где A – $m * m$ матрица с постоянными коэффициентами, B – вектор-столбец, C – вектор-строка, d — число. Наиболее просто алгоритм (3) выглядит в базисе собственных векторов матрицы [1] A

$$q_i^{(n)} = \lambda_i q_i^{(n-1)} + b_i v_n, \; q_i^{(-1)} = 0, \;\; i = 1, ..., m \tag{4}$$

$$\tilde{u}_n = \sum_{i=1}^{m} c_i q_i^{(n-1)} + c_0 v_n, \tag{5}$$

где λ_i – собственные значения A, $q_i^{(n)}$ – коэффициенты разложения $s^{(n)}$ по собственным векторам матрицы A, c_i и b_i – некоторые числа.

Некоторые замечания. Назовем алгоритмы эквивалентными, если они выдают одинаковые значения, коль скоро на вход им подают равные величины. Строго говоря, алгоритмы (3) и (4)–(5) не тождественны друг другу, а лишь являются эквивалентными. Ясно, что с точки зрения результата, пользователю совершенно неважно знать внутреннее строение алгоритма, то есть какой именно из эквивалентных алгоритмов реализован. С точки зрения конкретной реализации естественно выбрать тот алгоритм, который проще и экономичнее. Таким образом, переход от алгоритма (3) к алгоритму (4)–(5) есть не что иное, как переход к эквивалентному алгоритму с более простым внутренним устройством.

Отметим, что если в равенствах (4),(5) заменить $q_i^{(n)}$ на $b_i q_i^{(n)}$, то после сокращения равенства (4) на b_i и подходящего переопределения коэффициентов c_i в (5), равенства (4),(5) сохранят свой вид, а множитель b_i станет равным 1.

Алгоритм (4)–(5) при $b_i = 1$ назовем нормальной формой алгоритма (3).

Теорема 1.

1. Линейный автономный алгоритм с памятью, состоящей из m чисел, имеет вид (3).

2. Нормальная форма алгоритма (3) в случае, когда матрица A имеет простой спектр, дается формулами

$$q_i^{(n)} = \lambda_i q_i^{(n-1)} + v_n, \;\; q_i^{(-1)} = 0, \;\; i = 1, ..., m, \tag{6}$$

$$\tilde{u}_n = \sum_{i=1}^{m} c_i q_i^{(n-1)} + c_0 v_n, \tag{7}$$

[1] Предполагается, что A имеет диагональную Жорданову форму.

где $s^{(n)} = (q_i^{(n)})_{i=1,...,m}$ — вектор состояния памяти алгоритма, λ_i и c_i — параметры алгоритма.

3. Алгоритм (6)–(7) вычисляет свёртку

$$\tilde{u} = \tilde{K} * v, \tag{8}$$

последовательности v с ядром $\tilde{K} = \{\tilde{K}_n\}$, определённым по формуле:

$$\tilde{K}_n = \begin{cases} \sum\limits_{i=1}^{m} \alpha_i \lambda_i^n, & n > 0, \quad (\alpha_i = c_i/\lambda_i), \\ c_0, & n = 0. \end{cases} \tag{9}$$

4. Для устойчивости алгоритма (6)–(7) необходимо, чтобы

$$|\lambda_i| \leq 1. \tag{10}$$

Доказательство.

Пункт 1 теоремы есть следствие предположений.

Пункт 2 доказан выше.

Докажем третье утверждение теоремы.

Для этого вычислим сначала $q_i^{(n)}$, рекурсивно раскрывая выражение (6):

$$\begin{aligned} q_i^{(n)} &= v_n + \lambda_i(v_{n-1} + \lambda_i(v_{n-2} + ... + \lambda_i(v_1 + \lambda_i v_0)...)) \\ &= \sum_{k=0}^{n} \lambda_i^{n-k} v_k, \quad (i = 1, ..., m). \end{aligned} \tag{11}$$

Подставляя (11) в (7) получим

$$\tilde{u}_n = \sum_{k=0}^{n-1} v_k \sum_{i=1}^{m} \alpha_i \lambda_i^{n-k} + c_0 v_n, \quad \alpha_i = c_i/\lambda_i. \tag{12}$$

Формула (12) есть в точности развёрнутое выражение соотношения (8).

Пункт 4 очевиден и включен для того, чтобы не забывать про устойчивость.

Теорема доказана.

Теорема 1 является, конечно, пересказом теории рекуррентных последовательностей. Однако, она является руководством к действию.

Во-первых, и это главное, в ней содержится рецепт для эффективного рекуррентного вычисления свёртки, а именно: следует аппроксимировать ядро K суммой вида (9) (это сумма экспонент) [2] и считать свёртку по формулам (6), (7).

[2] Некоторые методы аппроксимации функции суммой экспонент рассмотрены в [Sofr03] и [Ham68]. Мы использовали метод наименьших квадратов.

Во-вторых, мы ответили на вопрос, что же вообще можно получить с помощью линейных автономных методов.

Отметим также, что нормальная форма – не единственное экономное для вычислений представление линейного автономного алгоритма. Однако, по нашему опыту, она устойчива для вычислений, в то время как другие формы, например, форма Фробениуса, могут быть численно неустойчивыми.

3. Определение погрешности

Прежде всего надо определить класс функций v, на котором следует добиваться хорошей аппроксимации. Имея ввиду наиболее широкую область приложений предлагаемого метода будем предполагать, что все функции лежат в классе l_2. Никаких других ограничений на класс функций мы налагать не будем. На практике расчёты ведутся до некоторого конечного шага $n = N$, поэтому достаточно рассматривать свёртку (1) на отрезке $n \in [0, N]$.

В соответствии со сказанным введём евклидовы нормы как для входных последовательностей v так и для выходных последовательностей u:

$$\| w \| = \sqrt{w_0^2 + w_1^2 + ... + w_N^2}\,, \text{ где } w = v \text{ либо } w = u. \tag{13}$$

Погрешность алгоритма (2) будем вычислять по формуле

$$\varepsilon = \max_{||v|| \le 1} ||\tilde{K}(v) - K * v||. \tag{14}$$

Альтернативное определение погрешности:

$$\varepsilon_\delta = \sup_{0 < ||v|| \le \delta} \frac{||\tilde{K}(v) - K * v||}{||v||}, \tag{15}$$

где δ – положительный параметр, и его предельный вариант:

$$\varepsilon_0 = \lim_{\delta \to +0} \varepsilon_\delta \tag{16}$$

дают то же самое ε, как в определении (14), только в том случае, когда оператор $\tilde{K}(v)$ — линейный. В нелинейном случае эти определения различны, однако отметим, что последующие результаты сохраняют свою силу также тогда, когда вместо определения погрешности (14) используется какое-нибудь из упомянутых альтернативных определений.

В общем случае величина ε_δ возрастает с увеличением δ вследствие вложенности шаров $||v|| \le \delta$ друг в друга. Это обосновывает существование предела в (16).

4. Непрерывные алгоритмы

Рассмотрим непрерывные алгоритмы (2), то есть алгоритмы, для которых отображения F_n непрерывны.

Для того чтобы оценить качество любого конкретного алгоритма (2), мы докажем теорему 2, которая дает оценку снизу для погрешности вычисления свёртки любым непрерывным нелинейным алгоритмом с m ячейками памяти. Такая оценка вычисляется для заданного ядра и при заданных N и m. Если окажется, что исследуемый алгоритм имеет погрешность, близкую к этой оценке, то это говорит о его высокой эффективности.

Ниже мы приведём пример исследования "на оптимальность" одного линейного алгоритма вычисления свёртки со степенным ядром.

Теорема 2. Пусть N и p некоторые целые, удовлетворяющие условию $N - m > p > m$. Погрешность ε_0 в диапазоне $n \in [0, N]$ любого алгоритма (2) с непрерывными F_n и памятью из m чисел удовлетворяет оценке

$$\varepsilon_0 \geq \sigma_{m+1}, \tag{17}$$

где $\sigma_1 \geq \cdots \geq \sigma_m \geq \sigma_{m+1} \geq \ldots$ — сингулярные числа матрицы

$$G = \begin{pmatrix} K_p & K_{p-1} & \ldots & K_1 \\ K_{p+1} & K_p & \ldots & K_2 \\ \ldots & \ldots & \ldots & \ldots \\ K_N & K_{N-1} & \ldots & K_{N-p+1} \end{pmatrix}. \tag{18}$$

Доказательство. Для каждого $\delta > 0$ требуется найти такую нормированную входную последовательность v, что $\| \tilde{K}(v) - K * v \| \geq \delta\sigma_{m+1}$. Будем искать её среди последовательностей, обращающихся в нуль при $n \geq p$. Поэтому на протяжении доказательства будем считать, что начиная с шага $n = p$ на вход подаются нули, то есть

$$v_p = v_{p+1} = ... = v_N = 0. \tag{19}$$

В этом случае можно отождествлять векторы $v = (v_0, v_1, ..., v_{p-1})$ и последовательности $(v_0, v_1, ..., v_{p-1}, 0_p, ..., 0_N)$, что мы и сделаем. Отметим, что при этом вектор Gv совпадает с вектором, составленным из последних $N - p + 1$ координат свёртки $u = K * v$, то есть

$$Gv = (u_p, u_{p+1}, ..., u_N). \tag{20}$$

В последнем легко убедиться, если записать оператор свёртки (1) в матрич-

ном виде:

$$\begin{pmatrix} K_0 & 0 & \dots & 0 & 0 & \dots & 0 \\ K_1 & K_0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ K_{p-1} & K_p & \dots & K_0 & 0 & \dots & 0 \\ K_p & K_{p-1} & \dots & K_1 & K_0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ K_N & K_{N-1} & \dots & K_{N-p+1} & K_{N-p} & \dots & K_0 \end{pmatrix} \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{p-1} \\ 0_p \\ \vdots \\ 0_N \end{pmatrix} = \begin{pmatrix} X & Y \\ G & Z \end{pmatrix} \begin{pmatrix} v_0 \\ v_1 \\ \vdots \\ v_{p-1} \\ 0_p \\ \vdots \\ 0_N \end{pmatrix}.$$

Матрица G находится в левом нижнем углу матрицы оператора свёртки (1).

Алгоритм (2) устроен так, что входные данные по пути на выход проходят через узкое место — память нашего алгоритма. Именно это используется в доказательстве.

Выясним, какие функциональные зависимости возникают между входными, выходными и промежуточными данными в процессе работы алгоритма. Изобразим их на следующей схеме:

$$\begin{array}{ccccccccccccc} & & \downarrow & & \downarrow & & & & \downarrow & & \downarrow & & & & \downarrow \\ 0 & \to & s^{(0)} & \to & s^{(1)} & \to & \dots & \to & s^{(p-1)} & \to & s^{(p)} & \to & \cdots & \to & s^{(N)} \\ & & \searrow & & \searrow & & & & \searrow & & \searrow & & & & \searrow \\ & & \tilde{u}_0 & & \tilde{u}_1 & & & & \tilde{u}_{p-1} & & \tilde{u}_p & & & & \tilde{u}_N \end{array} \qquad (21)$$

В первой, средней и последней строках находятся, соответственно, входная последовательность $v_0, v_1, ...,$ цепочка состояний $s^{(0)}, s^{(1)}, ...$ и значения $\tilde{u}_0, \tilde{u}_1, ...$ на выходе. То обстоятельство, что на n-ом шаге, в соответствии с определением нашего алгоритма (1), переход в состояние $s^{(n)}$ однозначно определяется по v_n и $s^{(n-1)}$, изображается стрелками, ведущими в $s^{(n)}$. Выходное значение $\tilde{u}_n$ тоже является функцией от v_n и $s^{(n-1)}$. Эта зависимость изображается наклонной стрелкой.

Прослеживая по схеме (21) историю перехода в состояние $s^{(p-1)}$, убеждаемся, что $s^{(p-1)}$ является непрерывной функцией от $v_0, v_1, ..., v_{p-1}$:

$$s^{(p-1)} = \Phi(v_0, v_1, ..., v_{p-1}). \qquad (22)$$

Из схемы (21) (разумеется с учётом условия (19)) видно также, что выходные значения $(\tilde{u}_p, \tilde{u}_{p+1}, ..., \tilde{u}_N)$ однозначно определяются по состоянию $s^{(p-1)}$. Запишем это в виде функциональной зависимости

$$(\tilde{u}_p, \tilde{u}_{p+1}, ..., \tilde{u}_N) = \Psi(s^{(p-1)}), \text{ если } v_p = v_{p+1} = ... = v_N = 0. \qquad (23)$$

В итоге получается, что входные данные по пути на выход проходят через память, как это представлено на коммутативной диаграмме

$$\begin{array}{c}(v_0, v_1, ..., v_{p-1})\\ \downarrow \Phi\\ s^{(p-1)}\\ \downarrow \Psi\\ (\tilde{u}_p, \tilde{u}_{p+1}, ..., \tilde{u}_N).\end{array} \tag{24}$$

В пространстве входных последовательностей построим такую m-мерную сферу B^m радиуса δ с центром в начале координат, что

$$\| Gv \| \geq \delta\sigma_{m+1} \text{ для любого } v \in B^m. \tag{25}$$

Для этого приведём квадратичную форму $\| Gv \|^2$ к главным осям. В этих координатах $\| Gv \|^2 = \sum_{i=1}^{p} \sigma_i^2\xi_i^2$, где $(\xi_1, ..., \xi_p)$ – координаты вектора v. Сферу B^m зададим системой: $\xi_1^2 + ... + \xi_{m+1}^2 = \delta^2$, $\xi_{m+2} = ... = \xi_p = 0$. Проверим (25). Для $v \in B^m$ имеем $\| Gv \|^2 = \sum_{i=1}^{m+1} \sigma_i^2\xi_i^2 \geq \sum_{i=1}^{m+1} \sigma_{m+1}^2\xi_i^2 = \sigma_{m+1}^2 \sum_{i=1}^{m+1} \xi_i^2 = \delta^2\sigma_{m+1}^2$, что и требовалось проверить.

Функция Φ непрерывна и, в частности, непрерывно отображает m-мерную сферу B^m в m-мерное пространство состояний. По теореме Борсука-Улама об антиподах функция Φ склеивает пару диаметрально противоположных точек, скажем v' и v'', сферы B^m. Это значит, что для них $\Phi(v') = \Phi(v'')$. Следовательно

$$\Psi\Phi(v') = \Psi\Phi(v''). \tag{26}$$

Другими словами, согласно (24), для входных последовательностей $v = v'$ и $v = v''$ на выход алгоритма поступают, начиная с шага p, одинаковые значения:

$$(\tilde{u}'_p, \tilde{u}'_{p+1}, ..., \tilde{u}'_N) = \Psi\Phi(v') = \Psi\Phi(v'') = (\tilde{u}''_p, \tilde{u}''_{p+1}, ..., \tilde{u}''_N). \tag{27}$$

Диаметрально противоположные точки v' и v'' сферы B^m связаны между собой равенством $v'' = -v'$, так как они симметричны относительно её центра, расположенного в начале координат. Тогда $Gv' - Gv'' = Gv' - G(-v') = 2Gv'$ и из неравенства (25) при $v = v'$ следует, что

$$\| Gv' - Gv'' \| \geq 2\delta\sigma_{m+1}. \tag{28}$$

Отсюда, обозначая через $\tilde{w}$ любой из равных векторов равенства (27), по неравенству треугольника получим

$$\| Gv' - \tilde{w} \| + \| \tilde{w} - Gv'' \| \geq \| Gv' - Gv'' \| \geq 2\delta\sigma_{m+1}. \tag{29}$$

Рассмотрим случай, когда $\| Gv' - \tilde{w} \| \geq \| \tilde{w} - Gv'' \|$ (противоположный случай, когда $\| Gv' - \tilde{w} \| < \| \tilde{w} - Gv'' \|$ рассматривается аналогично). Тогда

из (29) следует, что

$$\| Gv' - \tilde{w} \| \geq \delta\sigma_{m+1}. \tag{30}$$

Вспомним, что вектор Gv' совпадает с вектором, составленным из последних $N - p + 1$ координат свёртки $u' = K * v'$, а вектор $\tilde{w}$ состоит из последних $N - p + 1$ выходных значений нашего алгоритма, применённого к входной последовательности v'. Поэтому неравенство (30) означает, что векторы $K * v'$ и $\tilde{K}(v')$ отличаются друг от друга на σ_{m+1} уже в последних $N - p + 1$ координатах. Это отличие может только возрасти, если сравнивать их во всех координатах. Следовательно,

$$\| K * v' - \tilde{K}(v') \| \geq \delta\sigma_{m+1},$$

что и требовалось доказать.

Численные эксперименты показывают, что оценка снизу — величина $\sigma_{m+1} = \sigma_{m+1}(p)$ принимает своё наибольшее значение когда $p \approx N/2$. В этом случае σ_{m+1} можно выразить через собственные числа некоторой симметричной матрицы.

Следствие. Пусть p некоторое целое, удовлетворяющее условию $p > m$, и $N = 2p - 1$. Погрешность ε_0 в диапазоне $n \in [0, N]$ любого алгоритма (2) с непрерывными F_n и памятью из m чисел удовлетворяет оценке

$$\varepsilon_0 \geq |\lambda_{m+1}|, \tag{31}$$

где $|\lambda_1| \geq \cdots \geq |\lambda_m| \geq |\lambda_{m+1}| \geq \ldots$ и $\lambda_1, \lambda_2, \lambda_3, ...$ — собственные числа матрицы

$$H = \begin{pmatrix} K_1 & K_2 & \ldots & K_p \\ K_2 & K_3 & \ldots & K_{p+1} \\ \ldots & \ldots & \ldots & \ldots \\ K_p & K_{p+1} & \ldots & K_{2p-1} \end{pmatrix}. \tag{32}$$

Доказательство. При указанных N и p матрица H получается из матрицы G перестановкой столбцов в обратном порядке. Поэтому сингулярные числа матриц G и H совпадают. Так как матрица H симметрична, то её сингулярные числа равны абсолютным значениям от её собственных значений, в чём легко убедиться, рассматривая матрицу H в собственном базисе. Следовательно $\sigma_1 = |\lambda_1|$, $\sigma_2 = |\lambda_2|, ..., \sigma_{m+1} = |\lambda_{m+1}|, ...$ Но, тогда доказываемое неравенство совпадает с неравенством (17), доказанным в теореме 2.

5. Пример

В качестве примера возьмём ядро $K_n = n^{-0.5}$ при $n = 1, 2, ..., N$ ($K_0 = 0$), где $N = 299$. Рассмотрим его аппроксимацию суммой восьми экспонент

$$K_n \approx \sum_{i=1}^{8} \alpha_i e^{-\Omega_i(n-1)}. \tag{33}$$

Параметры α_i и Ω_i даны в таблице 1. В ней же даны: ε — погрешность линейного алгоритма (6)–(7), основанного на аппроксимации (33), и ε_C — точность аппроксимации (33) ядра K_n при $n = 1, 2, ..., N$ в равномерной норме.

Таблица 1. $n^{-0.5} \approx \sum \alpha_i e^{-\Omega_i(n-1)}$

i	α_i	Ω_i
1	8.952499173210060e-2	2.232463244689552e+0
2	1.804081904319615e-1	9.179972722734009e-1
3	1.944914494036348e-1	4.041244919835693e-1
4	1.629629526827911e-1	1.793269924622576e-1
5	1.241581429276527e-1	7.824580079115573e-2
6	9.543291834658668e-2	3.173731896889229e-2
7	7.967320482469276e-2	9.976547211762939e-3
8	7.334814273122857e-2	1.043396121553825e-3
ε	8.3e-5	
ε_C	7.3e-6	

Таблица 2. σ_j

j	σ_j
1	1.4e+1
2	2.0e+0
3	4.8e-1
4	1.1e-1
5	2.4e-2
6	4.9e-3
7	9.4e-4
8	1.7e-4
9	3.1e-5
10	5.2e-6

В таблице 2 даны десять наибольших сингулярных чисел матрицы G, определённой в (18), при $N = 299$ и $p = 150$.

Предоставим читателю судить, насколько близки погрешность ε и её оценка снизу σ_9. Мы же отметим, что $\varepsilon < \sigma_8$. Отсюда по теореме 2 следует, что рассматриваемый линейный алгоритм с памятью из восьми чисел заведомо более точен, чем любой алгоритм с непрерывными F_n и памятью из семи чисел.

6. Континуальный аналог

Вкратце рассмотрим континуальный аналог развитой теории. В своей простейшей форме вектор $\{K_n\}_{n=0,1,2,...}$ определяется равенствами $K_n = \Delta t K(t_n)$, где $K(t_n)$ представляют собой значения непрерывной функции $K(t)$, взятые в точках равномерной сетки $t_n = n\Delta t$ с шагом Δt. Связь предыдущих результатов и понятий и их предельная форма при $\Delta t \to 0$, $p\Delta t \to T$ приведена в таблице соответствия:

1.	$n; \quad K_n = \Delta t K(t_n); \quad \|\| \cdot \|\|_{l_2}$	$t = n\Delta t; \quad K(t); \quad \|\| \cdot \|\|_{L_2}$
2.	свёртка $u_n = \sum\limits_{k=0}^{n} K_{n-k}\, v_k$	свёртка $u(t) = \int_0^t K(t-\tau)v(\tau)d\tau$
3.	$\begin{pmatrix} s^{(n)} \\ \tilde{u}_n \end{pmatrix} = F_n \begin{pmatrix} s^{(n-1)} \\ v_n \end{pmatrix}, \; s^{(-1)} = 0$	$\begin{cases} \dfrac{ds}{dt} = f(t,s,v(t)) \\ \tilde{u}(t) = g(t,s,v(t)) \end{cases}, \quad s(0) = 0$
4.	$\begin{aligned} s^{(n)} &= As^{(n-1)} + v_n B, \quad s^{(-1)} = 0, \\ \tilde{u}_n &= Cs^{(n-1)} + v_n d, \end{aligned}$	$\begin{aligned} \dfrac{ds}{dt} &= As + v(t)B, \quad s(0) = 0, \\ \tilde{u}(t) &= Cs + v(t)d, \end{aligned}$
5.	нормальная форма (6),(7)	$\dfrac{dq_i(t)}{dt} = \lambda_i q_i(t) + v(t), \;\; q_i(0) = 0, \;\; i = 1, ..., m,$ $\tilde{u}(t) = \sum\limits_{i=1}^{m} c_i q_i(t) + c_0 v(t)$
6.	матрица H (см. (32))	оператор $\mathrm{H}v(t) = \int_0^T K(t+\tau)v(\tau)d\tau, \; 0 \le t \le T.$

Выводы

1. Введено понятие алгоритма с малой памятью и выделен его важный подкласс: линейные автономные алгоритмы с малой памятью (ЛААМП).
2. Найдена экономичная с точки зрения вычислений форма ЛААМП (нормальная форма).
3. Необходимым и достаточным условием применимости ЛААМП для задачи о свёртке является аппроксимация ядра суммами экспонент. Гладкие функции, как известно, хорошо аппроксимируются тригонометрическими функциями, а тем самым и суммами экспонент. Это означает, что существуют ЛААМП для вычисления сверток с гладкими ядрами.
4. Дана оценка снизу для погрешности нелинейных алгоритмов с малой памятью в применении к вычислению свёртки. Эта оценка может быть использована для проверки качества ЛААМП в задаче о свёртке, а также для того, чтобы решить вопрос о целесообразности поиска подходящих нелинейных алгоритмов.

Благодарности

Эта работа была выполнена в Институте Прикладной математики им. М.В. Келдыша, Российской Академии Наук. Мы благодарны профессору В.С. Рябенькому за глубокое внимание к данной работе и А.Л. Плешкевичу (ЦГЭ) за полезные замечания, способствовавшие улучшению этой работы.

Список литературы

[Biot62] M.A. Biot, *Mechanics of deformation and acoustic propagation in porous media*, J. Appl. Phys.,**33**,4,(1962) 1482-1498.

[Joh87] J. Dashen, R. Johnson, D.L. Koplik, *Theory of dynamic permeability and tortuosity in fluid-saturated porous media*, J. Fluid Mech. **176** (1987), 379–402.

[Ply00] B.D. Plyushchenkov, V.I. Turchaninov, *Acoustic logging modeling by refined biot's equations*, Int. J. Mod. Phys. C **11** (2000), no. 2, 365–396.

[Ply02fd] B.D. Plyushchenkov, and V.I. Turchaninov, *Construction principles of the efficient finite difference scheme for the refined Biot's equations*, Poromechanics II, Auriault et al (eds.), 2002 Swets & Zeitlinger, Lisse, ISBN 90 5809 394 8, 757-764 (2002).

[Ply02] B.D. Plyushchenkov, and V.I. Turchaninov, *Optimum approximation of convolution of arbitrary grid function with the power kernel*, Poromechanics II, Auriault et al (eds.), 2002 Swets & Zeitlinger, Lisse, ISBN 90 5809 394 8, 753-756 (2002).

[Sofr03] A. Arnold, M. Ehrhardt, I. Sofronov, *Discrete transparent boundary conditions for the Schroedinger equation: Fast calculation, approximation, and stability*, Comm. Math. Sci. 1 (2003), 501-556.

[Ham68] Р.В. Хемминг, Численные методы для научных работников и инженеров, М., Наука, 1968.

Приложение 1.

Приведём независимый вывод формул для вычисления свёртки в случае, когда ядро аппроксимировано суммой экспонент.

Будем исходить из квадратуры для свёртки

$$(K * Lv)^{n+1} = \int_0^{t_{n+1}} K(t_{n+1} - t) Lv(t) dt, \quad Lv(t) = av(t) + b\frac{dv(t)}{dt},\ v_0 = 0, \quad (34)$$

где $v(t)$ — кусочно-линейная интерполяция сеточной функции v_j от времени, определенной на равномерной сетке $t_{j+1} = t_j + \Delta t,\ t_0 = 0, \quad j = 0, 1, \ldots$. Предполагается, что $v_0 = 0$, a и b — некоторые постоянные.

Отметим, что кусочно-линейная интерполяция сеточной функции v_j по времени для $t \in [0, t_{n+1}]$ может быть определена как

$$v(t) = \delta(t - t_{n+1})\, v_{n+1} + \sum_{k=1}^{n} \delta(t - t_k)\, v_k\,, \quad (35)$$

где

$$\delta(t) = \begin{cases} 1 - |t|/\Delta t, & \text{if } |t| \le \Delta t; \\ 0, & \text{if } |t| > \Delta t. \end{cases} \tag{36}$$

Ядро свёртки $K(t)$ может иметь интегрируемую особенность при $t = 0$, а может и не иметь. Поэтому следует рассмотреть два способа аппроксимации свёртки. В первом случае мы исходим из того, что при $t \in [\Delta t, T]$ ядро $K(t)$ свёртки аппроксимируется суммой экспонент

$$K(t) \approx \tilde{K}(t) = \sum_{i=1}^{m} \beta_i e^{-\omega_i t}, \tag{37}$$

Во втором случае используем такую же аппроксимацию для $t \in [0, T]$.

Рассмотрим первый случай. Для вывода квадратуры заменим ядро в (34) на его аппроксимацию при $\zeta = t_{n+1} - t \ge \Delta t$. Имеем

$$\begin{aligned} (K * Lv)^{n+1} &= \int_0^{t_{n+1}} K(t_{n+1} - t) Lv(t) dt \\ &\approx \int_0^{t_n} \tilde{K}(t_{n+1} - t) Lv(t) dt + \int_{t_n}^{t_{n+1}} K(t_{n+1} - t) Lv(t) dt. \end{aligned} \tag{38}$$

Правая часть (38) представляет собой свёртку функции $Lv(t)$ с ядром, составленном из $K(t)$ при $t \le \Delta t$ и $\tilde{K}(t)$ при $t \ge \Delta t$. Во избежание систематических ошибок функции $K(t)$ и $\tilde{K}(t)$ склеиваются в точке $t = \Delta t$, то есть рассматриваются такие аппроксимации $\tilde{K}(t)$, что $\tilde{K}(t)\Big|_{t=\Delta t} = K(t)\Big|_{t=\Delta t}$ и $\frac{d\tilde{K}(t)}{dt}\Big|_{t=\Delta t} \approx \frac{dK(t)}{dt}\Big|_{t=\Delta t}$.

Интеграл $\int_0^{t_n}$ в (38) не зависит от v_{n+1}. Поэтому в нем можно заменить вектор $v = (v_0, v_1, ..., v_n, v_{n+1})$ на вектор $\breve{v} = (v_0, v_1, ..., v_n, 0)$. Представим правую часть (38) в виде суммы трёх слагаемых:

$$\begin{aligned} (K * Lv)^{n+1} \approx &\int_0^{t_{n+1}} \tilde{K}(t_{n+1} - t) L\breve{v}(t) dt + \int_{t_n}^{t_{n+1}} [K(t_{n+1} - t) - \tilde{K}(t_{n+1} - t)] L\breve{v}(t) dt \\ &+ \int_{t_n}^{t_{n+1}} K(t_{n+1} - t) L[v(t) - \breve{v}(t)] dt = A_n + I_n + J_n, \end{aligned} \tag{39}$$

где слагаемые A_n, I_n и J_n означают соответствующие интегралы. Их основная особенность состоит в следующем: A_n вычисляется по рекуррентным формулам, а I_n и J_n линейно зависят только от v_n и v_{n+1} соответственно.

В интеграле A_n выразим $\tilde{K}$ через сумму экспонент (37). Получим

$$A_n = \int_0^{t_{n+1}} \sum_{i=1}^{m} \beta_i e^{-\omega_i(t_{n+1}-t)} L\breve{v}(t)dt = \sum_{i=1}^{m} \beta_i S_n^i, \tag{40}$$

$$\text{где} \qquad S_n^i = \int_0^{t_{n+1}} e^{-\omega_i(t_{n+1}-t)} L\breve{v}(t)dt \tag{41}$$

Вычислим S_n^i:

$$S_n^i = \sum_{k=1}^{n} v_k \int_{t_{k-1}}^{t_{k+1}} e^{-\omega_i(t_{n+1}-t)} L\delta(t-t_k)dt = \sum_{k=1}^{n} v_k e^{-\omega_i(t_{n+1}-t_k)} \int_{-\Delta t}^{\Delta t} e^{\omega_i\xi} L\delta(\xi)d\xi$$
$$= \sum_{k=1}^{n} e^{-\omega_i\Delta t(n+1-k)} c_i\, v_k = \sum_{k=1}^{n} \lambda_i^{n+1-k} c_i\, v_k \tag{42}$$

где

$$\lambda_i = e^{-\omega_i\Delta t}\,, \tag{43}$$

$$c_i = \int_{-\Delta t}^{\Delta t} e^{\omega_i\xi} L\delta(\xi)d\xi = c_i^+ + c_i^- = \frac{1}{\omega_i\Delta t}(a\,\omega_i^{-1} - b)(\lambda_i + \lambda_i^{-1} - 2)\,, \tag{44}$$

$$c_i^+ = \int_0^{\Delta t} e^{\omega_i\xi} L\delta(\xi)d\xi = \frac{1}{\omega_i\Delta t}\left((a\omega_i^{-1} - b)(\lambda_i^{-1} - 1) - a\Delta t\right), \tag{45}$$

$$c_i^- = \int_{-\Delta t}^{0} e^{\omega_i\xi} L\delta(\xi)d\xi = \frac{1}{\omega_i\Delta t}\left((a\omega_i^{-1} - b)(\lambda_i - 1) + a\Delta t\right). \tag{46}$$

Вычисляя интегралы I_n и J_n, получим:

$$I_n = C_I v_n, \text{ где } C_I = \Delta t^{-1}(bE_0 + aE_1) - \sum_{i=1}^{m} \beta_i\lambda_i c_i^+, \tag{47}$$

$$J_n = C_J v_{n+1}, \text{ где } C_J = \Delta t^{-1}((a\Delta t + b)E_0 - aE_1), \tag{48}$$

$$E_0 = \int_0^{\Delta t} K(t)dt, \quad E_1 = \int_0^{\Delta t} K(t)tdt. \tag{49}$$

Итоговые расчетные формулы следуют из (39), (40), (47), (48):

$$(K * Lv)^{n+1} = \sum_{i=1}^{m} \beta_i S_n^i \; + \; C_I v_n \; + \; C_J v_{n+1}, \tag{50}$$

причём переменные S_n^i удовлетворяют рекуррентным соотношениям

$$S_n^i = \lambda_i (c_i \, v_n + S_{n-1}^i), \quad n = 1, 2, ...; \;\; S_0^i = 0, \tag{51}$$

вытекающим непосредственно из (42).

Пример 1. $K(t) = c_0 e^{-\omega_d t}/\sqrt{\pi \omega_d t}$.

В таблице 3 приложения 2 приводится аппроксимация:

$$\frac{1}{\sqrt{1+\xi}} \approx \sum_i \alpha_i e^{-\Omega_i \xi}, \quad \xi \geq 0.$$

По ней вычисляется следующая аппроксимация

$$c_0 \frac{e^{-\omega_d t}}{\sqrt{\pi \omega_d t}} = c_0 \frac{e^{-\omega_d t}}{\sqrt{\pi \omega_d \Delta t}} \frac{1}{\sqrt{t/\Delta t}} \approx c_0 \frac{e^{-\omega_d t}}{\sqrt{\pi \omega_d \Delta t}} \sum_i \alpha_i e^{-\Omega_i (\frac{t}{\Delta t} - 1)}$$
$$= \sum_i \frac{c_0 \alpha_i e^{\Omega_i}}{\sqrt{\pi \omega_d \Delta t}} e^{-\left(\omega_d + \frac{\Omega_i}{\Delta t}\right) t}, \;\; t \geq \Delta t.$$

Ее точность в равномерной норме не хуже, чем $c_0 \varepsilon_C / \sqrt{\pi \omega_d \Delta t} \varepsilon_C$, где ε_C — точность исходной аппроксимации.

Еще нужно вычислить два интеграла

$$E_0 = \int_0^{\Delta t} K(t) dt = \int_0^{\Delta t} c_0 \frac{e^{-\omega_d t}}{\sqrt{\pi \omega_d t}} dt = \frac{c_0}{\omega_d} \frac{2}{\sqrt{\pi}} \int_0^{\Delta t} e^{-\omega_d t} d\sqrt{\omega_d t}$$
$$= \frac{c_0}{\omega_d} \frac{2}{\sqrt{\pi}} \int_0^{\sqrt{\omega_d \Delta t}} e^{-y^2} dy = \frac{c_0}{\omega_d} \mathrm{erf}\,(\sqrt{\omega_d \Delta t}).$$

где $\mathrm{erf}\,(\cdot)$ – функция ошибок.

$$E_1 = \int_0^{\Delta t} K(t) t dt = \int_0^{\Delta t} c_0 \frac{e^{-\omega_d t}}{\sqrt{\pi \omega_d t}} t dt = -\frac{c_0}{\omega_d \sqrt{\pi \omega_d}} \int_0^{\Delta t} \sqrt{t} de^{-\omega_d t}$$
$$= -\frac{c_0}{\omega_d \sqrt{\pi \omega_d}} \left(\sqrt{\Delta t} e^{-\omega_d \Delta t} - \frac{1}{2} \int_0^{\Delta t} \frac{e^{-\omega_d t}}{\sqrt{t}} dt \right) = \frac{1}{2\omega_d} E_0 - c_0 \frac{\sqrt{\Delta t}}{\omega_d \sqrt{\pi \omega_d}} e^{-\omega_d \Delta t}.$$

Рассмотрим второй случай, когда ядро $K(t)$ свёртки аппроксимируется суммой экспонент на всем отрезке $t \in [0, T]$. Заменим ядро в (34) на его аппроксимацию (37) на всем отрезке интегрирования. Имеем

$$(K * lv)^{n+1} \approx \int_0^{t_{n+1}} \tilde{K}(t_{n+1} - t) Lv(t) dt = \int_0^{t_{n+1}} \tilde{K}(t_{n+1} - t) L\breve{v}(t) dt + \int_{t_n}^{t_{n+1}} \tilde{K}(t_{n+1} - t) L[v(t) - \breve{v}(t)] dt = A_n + D_n. \tag{52}$$

Первый интеграл A_n такой же, как в предыдущем случае. Вычисление второго интеграла D_n приводит к формуле

$$D_n = C_D v_{n+1}, \text{ где } C_D = \sum_{i=1}^{m} \beta_i c_i^-. \tag{53}$$

Итоговые расчетные формулы следуют из (52), (40), (53):

$$(K * Lv)^{n+1} = \sum_{i=1}^{m} \beta_i S_n^i + C_D v_{n+1}, \tag{54}$$

причём переменные S_n^i удовлетворяют рекуррентным соотношениям (51).

Пример 2. $K(t) = c_0 e^{-(\omega_d + \omega_2)t} \mathrm{I}_0(\omega_2 t)$, где $\mathrm{I}_0(\xi)$ — модифицированная функция Бесселя.

В таблице 3 приложения 2 приводится аппроксимация:

$$e^{-\xi} \mathrm{I}_0(\xi) \approx \sum_i \alpha_i e^{-\Omega_i \xi}, \quad \xi \geq 0.$$

Тогда

$$c_0 e^{-(\omega_d + \omega_2)t} \mathrm{I}_0(\omega_2 t) = c_0 e^{-\omega_d t} e^{-\omega_2 t} \mathrm{I}_0(\omega_2 t) \approx \sum_i c_0 \alpha_i e^{-(\Omega_i \omega_2 + \omega_d)t}, \ \omega_d \geq 0, \ t \geq 0.$$

Точность этой аппроксимации в равномерной норме не хуже, чем $c_0 \varepsilon_C$, где ε_C — точность исходной аппроксимации.

Приложение 2.

Приведем таблицу аппроксимаций суммой экспонент указанных в таблице функций $K(t)$:

$$K(t) \approx \tilde{K}(t) = \sum_i \alpha_i e^{-\Omega_i t}, \quad t \geq 0.$$

Точность ε_C аппроксимации дается в равномерной норме:

$$\varepsilon_C = \sup_{t\geq 0} |K(t) - \tilde{K}(t)|.$$

Особо отметим, что для приводимых аппроксимаций $K(0) \equiv \tilde{K}(0)$.

Таблица 3. Приближение $K(t) \approx \sum_i \alpha_i e^{-\Omega_i t}, t \geq 0$, для следующих функций

$K(t)$	$(1+t)^{-0.5}$	$e^{-t}I_0(t)$	$e^{-t}I_1(t)$	
i	α_i	α_i	α_i	Ω_i
1	9.540853988228605e-3	-1.888936303924003e-3	5.629745186581524e-3	4.581698614290183
2	2.643808354434316e-2	4.805945260292661e-2	-7.194580866498679e-2	2.548512146355390
3	9.893857731350460e-2	4.329092558711587e-1	-2.619857819769025e-1	1.600716848071002
4	2.141865537517109e-1	2.146145753353252e-1	6.458364658758935e-2	7.019150616142205e-1
5	2.294809830604571e-1	1.279422178466515e-1	9.383685781628216e-2	2.649314294615471e-1
6	1.739649605050017e-1	7.777201030111280e-2	7.084532097489001e-2	9.010047654713393e-2
7	1.111593189180471e-1	4.590959030808642e-2	4.458798417552870e-2	2.805928304793382e-2
8	6.535548682266647e-2	2.634369036425807e-2	2.614742881993943e-2	7.984611165067145e-3
9	3.628827525969136e-2	1.451026840275643e-2	1.447489728920669e-2	2.040785280166425e-3
10	1.906613855432791e-2	7.612001158607920e-3	7.610752897778071e-3	4.542637130459330e-4
11	9.348150866819713e-3	3.729277689291453e-3	3.728194735377377e-3	8.369199468138241e-5
12	4.166561097454345e-3	1.662388801222656e-3	1.662603494868761e-3	1.163822819125711e-5
13	1.598581265910818e-3	6.377067168849195e-4	6.376492797359850e-4	9.977782167834097e-7
14	4.674750518357016e-4	1.865009056405130e-4	1.865093841111135e-4	2.680497437744414e-8
ε_C	7.4e-5	6e-5	4.6e-5	

Здесь $I_0(t)$ и $I_1(t)$ – модифицированные функции Бесселя нулевого и первого порядков соответственно.

Отметим, что приближение $(1+t)^{-0.5} \approx \sum \alpha_i e^{-\Omega_i t},\ t \geq 0,$ данное в таблице 3 можно почленно дифференцировать. Получающаяся при этом аппроксимация $(1+t)^{-1.5} \approx \sum 2\alpha_i \Omega_i e^{-\Omega_i t}$ имеет погрешность $\varepsilon_C = 6.8 \cdot 10^{-5}$.

Отметим также, что приведенный в таблице 1 набор $\{\Omega_i\}$ можно использовать для аппроксимации суммой экспонент функций $K_\lambda(t) = (1+t)^\lambda$ для каждого $\lambda \in [-0.5, -1.5]$.

Приложение 3.

Для удобства читателя мы излагаем элементарное доказательство теоремы Борсука-Улама об антиподах, приведённое в книге М.А. Красносельского [Крас56]. С требуемым нам понятием триангуляции можно ознакомиться по книге [Прас04]. Там же дано доказательство теоремы Борсука-Улама, основанное на понятии степени симплициального отображения.

Теорема Борсука-Улама об антиподах [Bor33].

Теорема A. Непрерывное отображение сферы в евклидово пространство той же размерности склеивает пару диаметрально противоположных точек сферы.

Доказательство достаточно провести для n-мерной единичной сферы $S^n = \{x \ : \ ||x|| = 1, \ x \in R^{n+1}\}$, вложенной в $(n+1)$-мерное евклидово пространство R^{n+1}.

Если взять антисимметричную часть $\varphi(x) = \frac{1}{2}(f(x) - f(-x))$ отображения f, то теорему Борсука-Улама можно сформулировать в следующем виде:

Теорема A1. Нечетное непрерывное отображение $\varphi : S^n \to R^n$ имеет нуль.

Теорема Борсука-Улама эквивалентна следующей теореме Люстерника-Шнирельмана о категории проективного пространства [Люс30]:

Теорема B Пусть сфера S^n покрыта $n+1$ замкнутым множеством. Тогда одно из них содержит пару диаметрально противоположных точек сферы.

Выведем теорему A1 из теоремы B. Запишем $\varphi(x)$ $(x \in S^n)$ в координатном виде $\varphi(x) = (\varphi_1(x), \varphi_2(x), \ldots, \varphi_n(x))$. Пусть $\varepsilon > 0$. Множества $F_i = \{x \ : \ \varphi_i(x) \geq \varepsilon\}$ для $i = 1, 2, \ldots, n$ и $F_{n+1} = \{x : \varphi_1(x) \leq \varepsilon, \varphi_2(x) \leq \varepsilon, ...\varphi_n(x) \leq \varepsilon\}$ замкнуты и покрывают сферу S^n. По теореме B найдется такая точка x_ε, что $x_\varepsilon \in F_{n+1}$ и $-x_\varepsilon \in F_{n+1}$ (другие F_i не годятся). Соотношения $\pm x_\varepsilon \in F_{n+1}$ запишем в виде системы неравенств $|\varphi_i(x_\varepsilon)| \leq \varepsilon, \ \ i = 1, 2, \ldots, n$. Пусть x_0 – предельная точка семейства x_ε при $\varepsilon \longrightarrow 0$. Для нее неравенства $|\varphi_i(x_\varepsilon)| \leq \varepsilon$ переходят в предельные неравенства $|\varphi_i(x_0)| \leq 0$, то есть $\varphi(x_0) = 0$, что и т.д.

Докажем теорему B методом от противного. Предположим, что замкнутые множества $F_1, F_2, ..., F_{n+1}$ покрывают сферу S^n и ни одно из них не содержит диаметрально противоположные точки. Для каждого $i = 1, 2, \ldots, n+1$ введем множество $F_{-i} = -F_i$, центрально-симметричное множеству F_i. Так как F_i и F_{-i} замкнуты и, по предположению, не пересекаются, то между F_i и F_{-i} существует некоторый просвет $\varepsilon_i > 0$, то есть расстояние между любыми двумя точками из F_i и F_{-i} не меньше чем ε_i. Положим $\varepsilon = \min \varepsilon_i$.

Рассмотрим центрально-симметричную триангуляцию $\mathcal{T}$ сферы S^n с симплексами, диаметры которых меньше ε, и такую, что сферы S^k полностью состоят из k-мерных симплексов триангуляции $\mathcal{T}$. Здесь и ниже сферы S^k равны пересечению S^n с подпространствами, натянутыми на первые $k+1$ базисных векторов; $k = 0, 1, \ldots, n$. Триангуляцию $\mathcal{T}$ можно получить, например, если взять мелкую триангуляцию сферического симплекса, образованного точками сферы S^n с неотрицательными декартовыми координатами, а затем всевозможными отражениями относительно координатных гиперплоскостей распространить ее на всю сферу S^n. В дальнейшем, говоря о симплексах, мы будем иметь ввиду, что это симплексы построенной триангуляции.

Каждой вершине x триангуляции $\mathcal{T}$ присвоим такой номер $i = I(x)$, что F_i – самое первое множество в последовательности $F_1, F_{-1}, F_2, F_{-2}, ..., F_{n+1}$,

$F_{-(n+1)}$, содержащее x.

Следующие свойства функции $i = I(x)$ вытекают из свойств множеств F_i:

1) $I(x)$ определена для всех вершин триангуляции, так как F_i образуют покрытие;

2) $x \in F_{I(x)}$, что следует из определения функции $I(x)$;

3) $I(x)$ – нечетная функция, так как F_{-i} и F_i — центрально-симметричны друг другу;

4) Смежные вершины не могут иметь номера i и $-i$, потому что иначе было бы ребро, соединяющее множества F_{-i} и F_i и меньшее, чем ε (в силу мелкости триангуляции).

Нумерация k-мерного симплекса – это последовательность $k+1$ номеров его вершин, расположенных по неубыванию своих абсолютных значений.

Замечание. В определении нумерации симплекса есть некоторый произвол, ибо не указано, как упорядочены между собой номера с одинаковыми абсолютными значениями. Однако указывать это не требуется, так как из 4-ого свойства функции $i = I(x)$ вытекает, что одинаковые по абсолютной величине номера, присутствующие в нумерации, равны между собой. Таким образом, нумерация симплекса единственна.

Симплексы, нумерация которых знакопеременная и начинается с положительного (соответственно, отрицательного) числа назовем A-симплексами (соответственно, B-симплексами). Остальные симплексы назовем C-симплексами.

Введем следующие обозначения:

S^k_+ и S^k_- – две замкнутые полусферы, на которые экватор S^{k-1} разрезает сферу S^k.

a^k_+, a^k_- и a^k – количества A-симплексов, принадлежащих, соответственно, S^k_+, S^k_- и S^k.

b^k_+ – количество B-симплексов, принадлежащих S^k_+.

Центрально-симметричные симплексы имеют противоположные номера в силу нечетности $I(x)$. Поэтому их нумерации состоят из противоположных чисел. Следовательно, B-симплексы полусферы S^k_+ центрально-симметричны A-симплексам полусферы S^k_- и наоборот. Значит $a^k_- = b^k_+$ и поэтому

$$a^k_+ + b^k_+ = a^k. \tag{55}$$

Симплекс Ω^k размерности k имеет несколько $(k-1)$-мерных A-граней. Их

число N_{Ω^k}, как легко подсчитать, дается формулой: [3]

$$N_{\Omega^k} = \begin{cases} 1, & \text{если } \Omega^k \text{ суть } A\text{-симплекс или } B\text{-симплекс,} \\ 0 \text{ или } 2, & \text{если } \Omega^k \text{ суть } C\text{-симплекс.} \end{cases}$$

Рассмотрим полусферу S^k_+. Из предыдущей формулы следует, что

$$\sum N_{\Omega^k} \equiv a^k_+ + b^k_+ \mod 2, \tag{56}$$

где суммирование ведется по всем k-мерным симплексам, лежащим в S^k_+.

На сумму $\sum N_{\Omega^k}$ можно смотреть как на подсчет с повторениями $(k-1)$-мерных A-симплексов Ω^{k-1}_A из S^k_+, причем Ω^{k-1}_A считается столько раз, гранью скольких k-мерных симплексов из S^k_+ он служит. Но каждый Ω^{k-1}_A, лежащий на крае полусферы S^k_+ (то есть на экваторе S^{k-1}), служит A-гранью ровно одного k-мерного симплекса, а все другие Ω^{k-1}_A – ровно двух. Поэтому $\sum N_{\Omega^k} \equiv a^{k-1} \mod 2$. Отсюда и соотношений (55) и (56) следует, что

$$a^{k-1} \equiv a^k \mod 2. \tag{57}$$

Но $a^0 = 1$, так как нуль-мерная сфера S^0, состоящая из двух точек, покрыта одним A-симплексом и одним B-симплексом. Поэтому из сравнений (57) при $k = 1, 2, ..., n$ следует, что a^n нечетно. Следовательно, существует n-мерный A-симплекс. Его нумерация состоит из $n+1$ чисел и равна $-1, 2, ..., (-1)^n(n+1)$. Таким образом для одной из его вершин $|I(x)| = n+1$. Отсюда, заменяя, в случае необходимости x на $-x$, следует, что найдется вершина y, получившая номер $I(y) = -(n+1)$. Но тогда y не принадлежит ни одному из множеств $F_1, F_2, ..., F_{n+1}$, а это противоречит тому, что они образуют покрытие сферы. Что и требовалось доказать.

Список литературы

[Крас56] М.А. Красносельский, Топологические методы в теории нелинейных интегральных уравнений, М., Гос. Изд. Технико-Теорет. Лит-ры, (1956)

[Прас04] В.В. Прасолов, Элементы комбинаторной и дифференциальной топологии, М., МЦНМО, 2004.

[3] Нумерации граней получаются всевозможными вычеркиваниями одного числа из нумерации симплекса. Если нумерация знакопеременная, то знакопеременость сохранится лишь при вычеркивании крайних чисел. Это даст 1 в формуле для N_{Ω^k}.

Если знакопеременность нарушается в одном месте, то, чтобы добиться знакопеременности, необходимо убрать одного из нарушителей. Убираем одного, убираем другого — знак первого числа не изменится. Два и более нарушений не исправить.

[Bor33] K. Borsuk, *Drei sätze über die n-dimensionale euklidische sphär*, Fund. Math. **20** (1933).

[Люс30] Л.Г. Люстерник, Л.А. Шнирельман, Топологические методы в вариационных задачах, М., Госиздат, 1930.