ИПМ им.М.В.Келдыша РАН • Электронная библиотека

Препринты ИПМ • Препринт № 79 за 2009 г.

Турчанинов В.И.

# Рекуррентное вычисление свёртки с заданным ядром и экспоненциальная аппроксимация

***Рекомендуемая форма библиографической ссылки:*** Турчанинов В.И. Рекуррентное вычисление свёртки с заданным ядром и экспоненциальная аппроксимация // Препринты ИПМ им. М.В.Келдыша. 2009. № 79. 18 с. URL: http://library.keldysh.ru/preprint.asp?id=2009-79

РОССИЙСКАЯ АКАДЕМИЯ НАУК

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

В.И. Турчанинов

# Рекуррентное вычисление свёртки с заданным ядром и экспоненциальная аппроксимация

МОСКВА 2009

**В.И. Турчанинов**

## Рекуррентное вычисление свёртки с заданным ядром и экспоненциальная аппроксимация

**Аннотация**. Изучается семейство рекуррентных алгоритмов вычисления дискретной односторонней свёртки произвольной последовательности с заданным ядром, которые классифицируются по объему используемой ими памяти. Излагается методика анализа их эффективности. Показывается, что линейные с постоянными коэффициентами рекуррентные алгоритмы приводят к методу аппроксимации ядра суммой экспонент. Предлагается метод вычисления этих коэффициентов, что, по-существу, дает метод решения классической задачи аппроксимации ядра, заданного на равномерной сетке, суммой экспонент с неизвестными априори показателями. Возможности предлагаемой методики демонстрируются на численных экспериментах, в том числе и для данных, искаженных случайными помехами.

Работа выполнена при финансовой поддержке РФФИ грант № 08-01-00099

Ключевые слова: экспоненциальная аппроксимация, свертка, рекуррентный алгоритм, пористые среды, цифровая обработка сигнала, метод Прони.

**V.I. Turchaninov**

## Recurrent calculation of convolution with given kernel and exponential approximation

**Abstract.** Recurrent algorithms of calculation of discrete one-sided convolution of arbitrary sequence with given kernel are on the study. They are classified on the used memory involved. We are showing methods of analysis of their effectiveness. Also we are showing that calculations of linear recurrent algorithms with constant coefficients lead to method of kernel approximation by sum of exponents. We are recommending calculation method of mentioned coefficients that in real gives solution method of classical task kernel approximation determinate on equidistant grid by sum of exponents. Potentials of proposed method are demonstrated on numerical experiments also for the noisy sampled data.

Keywords: exponential sums, convolution, recurrent algorithm, porous medium, digital signal processing, Prony's method.

## Введение

Распространение упругих волн в многофазных средах (в средах "с памятью") описывается системой уравнений в частных производных, содержащих интегральные слагаемые, имеющие вид свёртки найденного решения во все предыдущие данному моменты времени с известным ядром, зависящим от свойств изучаемой среды [Biot62], [Joh87], [Лок82].

Построение экономичных разностных схем для расчета волновых полей во временном представлении сталкивается с трудностью вычисления свёртки, поскольку само по себе количество арифметических операций, приходящееся на вычисление одного значения свёртки, пропорционально числу шагов по времени. Здесь даётся методика построения и анализа эффективности алгоритмов вычисления свёртки.

Так как процесс вычислений эволюционных задач является рекуррентным и состоит в переходе на следующий временной слой с нескольких предыдущих, то вычисление свёртки также должно быть рекуррентным. Поэтому в данной работе для вычисления свёртки изучается семейство рекуррентных 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.

Предлагается методика вычисления этих коэффициентов, что, по-существу, дает метод решения классической задачи аппроксимации ядра, заданного на равномерной сетке, суммой экспонент с неизвестными априори показателями [Ham68], [Marp].

Для рекуррентных алгоритмов, заданных непрерывными функциями, указывается оценка снизу их точности в зависимости от объема используемой памяти. Эта оценка является основным инструментом анализа эффективности конкретных рекуррентных алгоритмов вычисления свёртки.

Возможности предлагаемой методики демонстрируются на численных экспериментах.

Эта работа является продолжением совместных с Б.Д. Плющенковым работ [Ply02], [Ply09] и включает в себя в переработанном виде их основные результаты, потому что на них опирается изложение предлагаемой методики вычисления коэффициентов линейных алгоритмов.

Экспоненциальный алгоритм для вычисления свёртки был предложен в [Ply02], [Sofr03]. Методы аппроксимации сеточных функций суммой экспонент предлагались и изучались, начиная с метода Prony (1795), многими авторами, см., например, [Ham68], [Marp], [Beyl05], [Potts].

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

Таким образом, наш алгоритм полностью определяется некоторыми отображениями $F_n$:

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

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 $\tilde{K}(v)$: $\tilde{K}(v) = \tilde{u} = (\tilde{u}_0, \tilde{u}_1, \tilde{u}_2, \ldots)$.

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 же цель — добиться того, 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.

Для линейных $F_n$ независящих от $n$ с помощью несложной линейной алгебры мы 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.

Нами будет также рассматриваться нетривиальная задача определения параметров экспоненциальной аппроксимации.

## 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] $A$

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

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

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

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

**Отступление.** 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

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

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

где $s^{(n)} = (q_i^{(n)})_{i=1,...,m}$ — вектор состояния памяти алгоритма, $\lambda_i$– собственные значения матрицы $A$ и

$$\alpha_i = (C\,W)_i \cdot (W^{-1}B)_i, \tag{8}$$

где $W$ – матрица собственных векторов матрицы $A$.

**3.** Алгоритм (6)-(7) вычисляет свёртку

$$\tilde{u} = \tilde{K} * v, \tag{9}$$

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

$$\tilde{K}_n = \begin{cases} \sum\limits_{i=1}^{m} \alpha_i \lambda_i^{n-1}, & n > 0, \\ d, & n = 0. \end{cases} \tag{10}$$

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

$$|\lambda_i| \leq 1. \tag{11}$$

Доказательство.

Пункт **1** теоремы есть следствие предположений.

Умножая первое равенство (3) на $W^{-1}$ слева и учитывая, что $W^{-1}A = \Lambda W^{-1}$, где $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, ..., \lambda_m)$, получим $W^{-1}s^{(n)} = \Lambda W^{-1}s^{(n-1)} + v_n W^{-1}B$. Второе равенство (3) представим в виде: $\tilde{u}_n = CWW^{-1}s^{(n-1)} + v_n d$. Формулы пункта 2 следуют из двух предыдущих равенств в результате замены $q^{(n-1)} = W^{-1}s^{(n-1)}$ и $q^{(n)} = W^{-1}s^{(n)}$ и указанной выше перенормировки.

Докажем третье утверждение теоремы.

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

$$q_i^{(n)} = v_n + \lambda_i(v_{n-1} + \lambda_i(v_{n-2} + ... + \lambda_i(v_1 + \lambda_i v_0)...)) = \sum_{k=0}^{n} \lambda_i^{n-k} v_k. \tag{12}$$

Подставляя (12) (с заменой $n$ на $n-1$) в (7) получим

$$\tilde{u}_n = \sum_{k=0}^{n-1} v_k \sum_{i=1}^{m} \alpha_i \lambda_i^{n-1-k} + dv_n. \tag{13}$$

Формула (13) есть в точности развёрнутое выражение соотношения (9).

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 (10) (это сумма экспонент) и считать свёртку по формулам (6)-(7).

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 при-

ложений 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 $u$:

$$\| w \| = \sqrt{w_0^2 + w_1^2 + ... + w_N^2}\,, \text{ где } w = v \text{ либо } w = u. \tag{14}$$

Погрешность алгоритма (2) будем вычислять по формуле

$$\varepsilon = \max_{||v|| \le 1} ||\tilde{K}(v) - K * v||. \tag{15}$$

Альтернативное определение погрешности:

$$\varepsilon_\delta = \sup_{0 < ||v|| \le \delta} \frac{||\tilde{K}(v) - K * v||}{||v||}, \tag{16}$$

где $\delta$ – положительный параметр, и его предельный вариант:

$$\varepsilon_0 = \lim_{\delta \to +0} \varepsilon_\delta \tag{17}$$

дают то же самое $\varepsilon$, как в определении (15), 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 (15) используется какое-нибудь из упомянутых альтернативных определений.

В общем случае величина $\varepsilon_\delta$ возрастает с увеличением $\delta$ вследствие вложенности шаров $||v|| \le \delta$ друг в друга. Это обосновывает существование предела в (17).

## 4 *Алгоритмы с непрерывными* $F_n$

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 примеры исследования "на оптимальность"нескольких линейных алгоритмов вычисления свёртки.

**Теорема 2**. Пусть $N$ и $p$ некоторые целые, удовлетворяющие условию $N - m > p > m$. Погрешность $\varepsilon_0$ в диапазоне $n \in [0, N]$ любого алгоритма (2) с непрерывными $F_n$ и памятью из $m$ чисел удовлетворяет оценке

$$\varepsilon_0 \geq \sigma_{m+1}, \tag{18}$$

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

$$G = \begin{pmatrix} K_p & K_{p-1} & \ldots & K_1 \\ K_{p+1} & K_p & \ldots & K_2 \\ \ldots & \ldots & \ldots & \ldots \\ K_N & K_{N-1} & \ldots & K_{N-p+1} \end{pmatrix}. \tag{19}$$

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

$$v_p = v_{p+1} = ... = v_N = 0. \tag{20}$$

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

$$Gv = (u_p, u_{p+1}, ..., u_N). \tag{21}$$

Запишем алгоритм (2) в более удобном "координатном"виде:

$$\begin{array}{rcll} s^{(n)} & = & f_n(s^{(n-1)}, v_n), & s^{(-1)} = 0, \\ \tilde{u}_n & = & g_n(s^{(n-1)}, v_n), & \end{array} \tag{22}$$

где $f_n$ и $g_n$ непрерывны.

Выясним, какие функциональные зависимости возникают между входными, выходными и промежуточными данными в процессе работы алгоритма.

Прослеживая по алгоритму (22) историю перехода в состояние $s^{(p-1)}$, убеждаемся, что $s^{(p-1)}$ является непрерывной функцией от $v_0, v_1, ..., v_{p-1}$:

$$s^{(p-1)} = \Theta(v_0, v_1, ..., v_{p-1}). \tag{23}$$

Из алгоритма (22) (разумеется с учётом условия (20)) 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

$$(\tilde{u}_p, \tilde{u}_{p+1}, ..., \tilde{u}_N) = \Upsilon(s^{(p-1)}), \text{ если } v_p = v_{p+1} = ... = v_N = 0. \tag{24}$$

В итоге получается, что алгоритм (22) устроен так, что входные данные по пути на выход проходят через узкое место – память нашего алгоритма, как это представлено на диаграмме

$$\begin{array}{c}(v_0, v_1, ..., v_{p-1})\\ \downarrow \Theta\\ s^{(p-1)}\\ \downarrow \Upsilon\\ (\tilde{u}_p, \tilde{u}_{p+1}, ..., \tilde{u}_N).\end{array} \tag{25}$$

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

$$\| Gv \| \geq \delta\sigma_{m+1} \text{ для любого } v \in B^m. \tag{26}$$

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: $\xi_1^2 + ... + \xi_{m+1}^2 = \delta^2, \quad \xi_{m+2} = ... = \xi_p = 0$. Проверим (26). 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 $\Theta$ непрерывна и, в частности, непрерывно отображает $m$-мерную сферу $B^m$ в $m$-мерное пространство состояний. По теореме Борсука-Улама об антиподах [Bor33] функция $\Theta$ склеивает пару диаметрально противоположных точек, скажем $v'$ и $v''$, сферы $B^m$. Это значит, что для них $\Theta(v') = \Theta(v'')$. Следовательно

$$\Upsilon\Theta(v') = \Upsilon\Theta(v''). \tag{27}$$

Другими словами, согласно (25), для входных последовательностей $v = v'$ и $v = v''$ на выход алгоритма поступают, начиная с шага $p$, одинаковые значения:

$$(\tilde{u}'_p, \tilde{u}'_{p+1}, ..., \tilde{u}'_N) = \Upsilon\Theta(v') = \Upsilon\Theta(v'') = (\tilde{u}''_p, \tilde{u}''_{p+1}, ..., \tilde{u}''_N). \tag{28}$$

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 (26) при $v = v'$ следует, что

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

Отсюда, обозначая через $\tilde{w}$ любой из равных векторов равенства (28), по неравенству треугольника получим

$$\| Gv' - \tilde{w} \| + \| \tilde{w} - Gv'' \| \geq \| Gv' - Gv'' \| \geq 2\delta\sigma_{m+1}. \tag{30}$$

Рассмотрим случай, когда $\| Gv' - \tilde{w} \| \geq \| \tilde{w} - Gv'' \|$ (противоположный случай, когда $\| Gv' - \tilde{w} \| < \| \tilde{w} - Gv'' \|$ рассматривается аналогично). Тогда из (30) следует, что

$$\| Gv' - \tilde{w} \| \geq \delta\sigma_{m+1}. \tag{31}$$

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 (31) означает, что векторы $K * v'$ и $\tilde{K}(v')$ отличаются друг от друга на $\sigma_{m+1}$ 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 $\sigma_{m+1}$ можно выразить через собственные числа некоторой симметричной матрицы.

**Следствие 1**. Пусть $p$ некоторое целое, удовлетворяющее условию $p > m$, и $N = 2p - 1$. Погрешность $\varepsilon_0$ в диапазоне $n \in [0, N]$ любого алгоритма (2) с непрерывными $F_n$ и памятью из $m$ чисел удовлетворяет оценке

$$\varepsilon_0 \geq |\zeta_{m+1}|, \tag{32}$$

где $|\zeta_1| \geq \cdots \geq |\zeta_m| \geq |\zeta_{m+1}| \geq \ldots$ и $\zeta_1, \zeta_2, \zeta_3, \ldots$ – собственные числа матрицы

$$H = \begin{pmatrix} K_1 & K_2 & \ldots & K_p \\ K_2 & K_3 & \ldots & K_{p+1} \\ \ldots & \ldots & \ldots & \ldots \\ K_p & K_{p+1} & \ldots & K_{2p-1} \end{pmatrix}. \tag{33}$$

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 $\sigma_1 = |\zeta_1|$, $\sigma_2 = |\zeta_2|, \ldots, \sigma_{m+1} = |\zeta_{m+1}|, \ldots$. Но тогда доказываемое неравенство совпадает с неравенством (18), доказанным в теореме 2.

## 5 Экспоненциальная аппроксимация

Построим линейный автономный алгоритм вида (3), аппроксимирующий вычисление свертки.

Пусть $p$ и $N$ фиксированы и $q = N - p + 1$. Пусть $K_0 = 0$. Рассмотрим две входные последовательности:

$$v^{(p-1)} = (v_0, v_1, ..., v_{p-1}, 0, 0, ...) \text{ и } v^{(p)} = (v_0, v_1, ..., v_{p-1}, v_p, 0, ...)$$

и их свертки с ядром $K$: $u = K * v^{(p-1)}$ и $u^{'} = K * v^{(p)}$. Определим векторы $u^{(p-1)}$ и $u^{(p)}$ как сужения $u$ и $u^{'}$ на множества индексов $\{p, p+1, ..., p+q-1\}$ и $\{p+1, p+2, ...p+q\}$ соответственно:

$$u^{(p-1)} = (u_p, u_{p+1}, ..., u_{p+q-1}), \qquad u^{(p)} = (u^{'}_{p+1}, u^{'}_{p+2}, ..., u^{'}_{p+q}). \tag{34}$$

Из линейности свёртки вытекает, что

$$(u^{'}_{p+1}, u^{'}_{p+2}, ..., u^{'}_{p+q}) = (u_{p+1}, u_{p+2}, ..., u_{p+q}) + v_p \cdot (K_1, K_2, ..., K_q). \tag{35}$$

Напомним, что матрица $G$ определялась так, чтобы выполнялось равенство:

$$u^{(p-1)} = Gv^{[p-1]}, \text{ где } v^{[p-1]} = (v_0, v_1, ..., v_{p-1}). \tag{36}$$

Пусть $\psi_1, ..., \psi_m$ - левые сингулярные векторы матрицы $G$, соответствующие её $m$ наибольшим сингулярным числам. Эти векторы идут вдоль главных осей эллипсоида $GB^m$ (см.(26)).

Из доказательства теоремы 2 следует, что на $(p-1)$-ом шаге вычисления свёртки наиболее выгодно хранить в памяти коэффициенты разложения выходной последовательности $u^{(p-1)}$ по векторам $\psi_1, ..., \psi_m$. Поэтому для построения добротного алгоритма необходимо выполнить условие, при котором числа $s_1^{(p-1)}, ..., s_m^{(p-1)}$, определяющие текущее состояние алгоритма после $(p-1)$-ого шага, были как можно ближе к этим коэффициентам и, следовательно, удовлетворяли равенству:

$$s_1^{(p-1)}\psi_1 + ... + s_m^{(p-1)}\psi_m \approx u^{(p-1)}. \tag{37}$$

На $p$-ом шаге определим числа $s_1^{(p)}, ..., s_m^{(p)}$ как коэффициенты разложения вектора $u^{(p)}$ по векторам $\psi_1, ..., \psi_m$. Напишем соответствующую систему уравнений:

$$s_1^{(p)}\psi_1 + ... + s_m^{(p)}\psi_m = u^{(p)}. \tag{38}$$

Наша цель – найти $s_1^{(p)}, ..., s_m^{(p)}$ по данным $s_1^{(p-1)}, ..., s_m^{(p-1)}$ и $v_p$, но никак не по входной последовательности $v^{[p-1]}$, так как после $(p-1)$-ого шага она считается забытой. Для этого займемся восстановлением вышеупомянутых

величин, а именно: $u_p, u_{p+1}, ..., u_{p+q-1}, u_{p+q}$ и $u^{(p)}$. Для произвольной величины $X$ обозначим через $\hat{X}$ её восстановленную величину.

Определим $\hat{u}^{(p-1)}$ по левой части формулы (37). Тем самым восстановлены значения $u_p, u_{p+1}, ..., u_{p+q-1}$.

Восстановим $u_{p+q}$. Для этого найдём $\hat{v}^{[p-1]}$ как нормальное решение уравнения $\hat{u}^{(p-1)} = G\hat{v}^{[p-1]}$, аналогичного равенству (36). Затем дополним вектор $\hat{v}^{[p-1]}$ нулями справа и свернем его с ядром $K$. Получим последовательность $\hat{u}$. Тогда восстановленное значение $u_{p+q}$ определяется как $p+q$-ый член последовательности $\hat{u}$. При этом отметим, что в силу уравнения $\hat{u}^{(p-1)} = G\hat{v}^{[p-1]}$ ранее вычисленный вектор $\hat{u}^{(p-1)}$ равен сужению последовательности $\hat{u}$ на множество индексов $\{p, p+1, ..., p+q-1\}$.

Наконец, $\hat{u}^{(p)}$ находится по формуле, аналогичной (35):

$$\hat{u}^{(p)} = (\hat{u}_{p+1}, \hat{u}_{p+2}, ..., \hat{u}_{p+q}) + v_p \cdot (K_1, K_2, ..., K_q)\,. \tag{39}$$

Решая систему (38) с восстановленной правой частью $\hat{u}^{(p)}$ методом наименьших квадратов получим коэффициенты $s_1^{(p)}, ..., s_m^{(p)}$.

Для вывода расчетных формул воспользуемся разложением матрицы $G$ по сингулярным числам:

$$G = \Psi\Sigma\Phi^T, \tag{40}$$

где $\Psi$ и $\Phi$ – ортогональные матрицы, $T$ – операция транспонирования,
$\Sigma = \mathrm{diag}(\sigma_1, \sigma_2, ...)$ – диагональная матрица, в которой на диагонали стоят сингулярные числа, расположенные в убывающем порядке: $\sigma_1 \geq \sigma_2 \geq ... \geq 0$. Столбцы матрицы $\Psi$ – это левые сингулярные векторы матрицы $G$.

Из сингулярного разложения (40) следует, что оператор восстановления вектора $(\hat{u}_{p+1}, \hat{u}_{p+2}, ..., \hat{u}_{p+q})$ по состоянию $s^{(p-1)} = (s_1^{(p-1)}, ..., s_m^{(p-1)})$ имеет следующую матрицу:

$$Q = \begin{pmatrix} \psi_{21} & \psi_{22} & \dots & \psi_{2m} \\ \psi_{31} & \psi_{32} & \dots & \psi_{3m} \\ \dots & \dots & \dots & \dots \\ \psi_{q1} & \psi_{q2} & \dots & \psi_{qm} \\ \psi^*_{q+1,1} & \psi^*_{q+1,2} & \dots & \psi^*_{q+1,m} \end{pmatrix}, \tag{41}$$

в которой $\psi^*_{q+1,j} = \sigma_j^{-1}(k, \phi_j)$, $j = 1, 2, ..., m$, где $(k, \phi_j)$ означает скалярное произведение вектора $k = (K_{p+q}, K_{p+q-1}, K_{p+q-2}, ..., K_{q+1})$ и $j$-ого столбца матрицы $\Phi$. $\psi_{ij}$ – элементы матрицы $\Psi$.

Запишем систему (38) с восстановленной правой частью $\hat{u}^{(p)}$:

$$\Psi_m\, s^{(p)} = Qs^{(p-1)} + v_p \cdot (K_1, K_2, ..., K_q)^T\,. \tag{42}$$

где $\Psi_m$ – матрица, составленная из $m$ первых столбцов матрицы $\Psi$. Умножая обе части (42) на $\Psi_m^T$ получим для определения $s^{(p)}$ искомую нормальную систему уравнений:

$$s^{(p)} = \Psi_m^T Q\, s^{(p-1)} + v_p B, \quad B = \Psi_m^T \left(K_1, K_2, ..., K_q\right)^T, \tag{43}$$

причём сразу в разрешенном виде, так как в силу ортогональности матрицы $\Psi$ матрица $\Psi_m^T \Psi_m$ – единичная.

Выражение для вектора $B$ можно упростить. Для этого заметим, что вектор $\left(K_1, K_2, ..., K_q\right)^T$ совпадает с последним столбцом матрицы $G$. Тогда из сингулярного разложения (40) следует, что $B = (\sigma_1\phi_{p1}, \sigma_2\phi_{p2}, ..., \sigma_m\phi_{pm})^T$, где $\phi_{ij}$ – элементы матрицы $\Phi$.

Наконец выходное значение $\tilde{u}_p$ вычисляется по первой строке системы (37):

$$\tilde{u}_p = s_1^{(p-1)}\psi_{11} + ... + s_m^{(p-1)}\psi_{1m}. \tag{44}$$

В итоге получается следующий линейный автономный алгоритм:

$$\begin{array}{rcl} s^{(n)} & = & As^{(n-1)} + v_n B, \quad s^{(-1)} = 0, \\ \tilde{u}_n & = & Cs^{(n-1)} + v_n d, \end{array} \tag{45}$$

где $A = \Psi_m^T Q,\ B = (\sigma_1\phi_{p1}, \sigma_2\phi_{p2}, ..., \sigma_m\phi_{pm})^T,\ C = (\psi_{11}, \psi_{12}, ..., \psi_{1m}),\ d = K_0$. Соответственно аппроксимация ядра K вычисляется по формуле (10) теоремы 1 .

Замечания и дополнения.

Если $p \geq m,\ q \geq m$, то алгоритм (45) точен на ядрах, являющихся суммами $m$ экспонент. Действительно, для таких ядер все элементы конструкции: равенства (37),(38),(39),(44), процедуры восстановления величин $u_p, ..., u_{p+q}$ и $u^{(p)}$, а также вычисления $B$ и $C$, являются точными.

Наиболее тонким элементом конструкции является способ восстановления $u_{p+q}$. По своему смыслу восстановление $u_{p+q}$ – это экстраполяция последовательности $u_p, u_{p+1}, ..., u_{p+q-1}$ на один шаг вперёд, причем "внутренним"образом – через сдвиги ядра $K$, опираясь лишь на то, что значение $K_{p+q}$ дано и, следовательно, его экстраполяция уже известна. Именно это восстановление $u_{p+q}$ придает построенному методу высокое разрешение, а систематическое применение метода наименьших квадратов дает ему высокую помехоустойчивость.

В случае $p = q$ построения можно немного упростить, если использовать симметричную матрицу $H$ в (33) вместо матрицы $G$. Действительно, в этом случае $H = GJ_p$, где $J_p$ – матрица, в которой на кросс-диагонали находятся единицы, а на остальных местах – нули. Тогда собственные числа $\zeta_i$ матрицы $H$ связаны с $\sigma_i$ равенствами $\sigma_1 = |\zeta_1|$, $\sigma_2 = |\zeta_2|, ..., \sigma_{m+1} = |\zeta_{m+1}|, ...$, а матрица собственных векторов $H$ равна $\Psi$. Соответственно изменятся формулы:

$\psi^*_{q+1,j} = \zeta_j^{-1}(J_p k, \psi_j)$ и $B = (\zeta_1\psi_{11}, \zeta_2\psi_{12}, ..., \zeta_m\psi_{1m})^T$. Именно этот вариант метода применялся в численных экспериментах.

## 6 Численные эксперименты.

В качестве примера рассмотрим два ядра:

$$K^1_n = n^{-0.5} \text{ и } K^2_n = n^{-0.5}\cos(0.1 \cdot n^{0.5}) \text{ при } n = 1, 2, ..., N, \quad K^1_0 = K^2_0 = 0,$$

где $N = 15999$.

Для каждого ядра при $p = 8000$ был построен линейный автономный алгоритм по вышеизложенной методике (45). По теореме 1 это приводит, в частности, к аппроксимации соответствующих ядер суммой $m$ экспонент (10).

Таблица 1. $K^1$

| $m$ | $\varepsilon_C$ | $\varepsilon$ | $\sigma_m$ |
|---|---|---|---|
| 9 | 1.3e-3 | 2.2e-2 | 1.3e-2 |
| 10 | 4.2e-4 | 8.5e-3 | 4.8e-3 |
| 11 | 1.4e-4 | 3.2e-3 | 1.7e-3 |
| 12 | 5.6e-5 | 1.1e-3 | 6.3e-4 |
| 13 | 1.8e-5 | 4.1e-4 | 2.3e-4 |
| 14 | 6.3e-6 | 1.5e-4 | 8.1e-5 |
| 15 | 2.3e-6 | 5.2e-5 | 2.9e-5 |
| 16 | 7.1e-7 | 1.9e-5 | 1.0e-5 |
| 17 | 2.6e-7 | 6.6e-6 | 3.5e-6 |

Таблица 2. $K^2$

| $m$ | $\varepsilon_C$ | $\varepsilon$ | $\sigma_m$ |
|---|---|---|---|
| 9 | 5.6e-3 | 9.4e-2 | 6.4e-2 |
| 10 | 1.8e-3 | 2.7e-2 | 2.1e-2 |
| 11 | 5.2e-4 | 5.9e-3 | 6.4e-3 |
| 12 | 1.4e-4 | 2.3e-3 | 1.9e-3 |
| 13 | 1.4e-4 | 2.7e-2 | 1.1e-3 |
| 14 | 4.4e-5 | 3.6e-3 | 5.7e-4 |
| 15 | 1.1e-5 | 2.5e-4 | 1.7e-4 |
| 16 | 4.0e-6 | 8.2e-5 | 5.3e-5 |
| 17 | 1.2e-6 | 2.8e-5 | 1.7e-5 |

В таблице 1 для первого ядра и в таблице 2 для второго ядра приводятся следующие величины:

$\varepsilon_C$ – точность аппроксимации (10) ядра в равномерной норме,

$\varepsilon$ – погрешность (15) построенного алгоритма,

сингулярное число $\sigma_m$ матрицы $G$, определённой в (19), при $N = 15999$ и $p = 8000$.

Сравнивая погрешность $\varepsilon(m)$ и её оценку снизу – величину $\sigma_{m+1}$ видно, что $\varepsilon(m)$ невозможно существенно уменьшить. Следовательно, для данного примера предложенная методика даёт алгоритм вычисления свёртки, неулучшаемый по порядку величины.

Следующий тест предназначен для того, чтобы выяснить применимость предложенной методики для задач цифровой обработки сигнала.

Рассмотрим следующую модельную задачу: определить частоты $\omega_j$ и веса $\rho_j$ вещественной функции

$$f(x) = \sum_{j=-J}^{J} \rho_j \exp(i\omega_j x),\ i = \sqrt{-1},\ \omega_j = -\omega_{-j},\ \rho_j = \bar{\rho}_{-j}, \tag{46}$$

по её наблюдениям в точках $x = 0, 1, 2, ..., 2p$ на фоне случайного шума, то есть по данным

$$K_{j+1} = f(j) + n(j), \quad j = 0, 1, 2, ..., 2p, \tag{47}$$

где шум $n(j)$ - случайная величина.

Напомним, что рассматриваемая методика находит приближения $\tilde{\rho}_j$ и $\tilde{\lambda}_j$ для $\rho_j$ и $\lambda_j = \exp(i\omega_j)$, причём в ней никак не используется информация о том, что $\lambda_j$ лежат на единичной окружности. Пусть $\tilde{f}(x) = \sum_{j=-J}^{J} \tilde{\rho}_j \tilde{\lambda}_j{}^x$ – восстановленная функция. Для измерения ошибок определим величины:

$$e(\omega) = \max|\tilde{\lambda}_j - \exp(i\omega_j)|,\ \ e(\rho) = \max|\tilde{\rho}_j - \rho_j|,\ j = -J, ..., J,$$

$$e(f) = \max|\tilde{f}(x) - f(x)| / \max|f(x)|,\ x \in [0, 2p],$$

аналогичные предложенным в [Potts].

Для примера возьмём функцию

$$f(x) = 34 + 300\ \cos(\pi x/4) + \cos(\pi x/2), \tag{48}$$

рассмотренную в работах [Mhas05], [Potts]. В таблицах 3 и 4 приведены ошибки восстановления функции $f(x)$ по данным (47) при наличии шума, равномерно распределенного на отрезках $[0, 1]$ и $[0, 3]$ соответственно. Ошибка определения коэффициента при $\cos(\pi x/2)$ менялась от 16% до 3% и, соответственно, от 60% до 4% при увеличении $p$ от 32 до 1024.

Таблица 3. $f(x)$, $0 < n(x) < 1$

| $p$ | $e(\omega)$ | $e(\rho)$ | $e(f)$ |
|---|---|---|---|
| 32 | 3.072e-3 | 5.120e-1 | 2.200e-3 |
| 64 | 6.058e-4 | 5.212e-1 | 2.080e-3 |
| 128 | 4.397e-4 | 5.654e-1 | 2.026e-3 |
| 256 | 3.512e-4 | 4.840e-1 | 1.901e-3 |
| 512 | 9.233e-5 | 5.134e-1 | 1.761e-3 |
| 1024 | 1.976e-5 | 5.048e-1 | 1.667e-3 |

Таблица 4. $f(x)$, $0 < n(x) < 3$.

| $p$ | $e(\omega)$ | $e(\rho)$ | $e(f)$ |
|---|---|---|---|
| 32 | 1.165e-2 | 1.534 | 6.865e-3 |
| 64 | 1.523e-3 | 1.563 | 6.278e-3 |
| 128 | 1.419e-3 | 1.696 | 6.134e-3 |
| 256 | 1.138e-3 | 1.452 | 5.815e-3 |
| 512 | 2.940e-4 | 1.540 | 5.301e-3 |
| 1024 | 5.964e-5 | 1.515 | 5.001e-3 |

По сравнению с результатами работы [Potts], в которой $e(\omega)$ менялась в пределах от 3.992e-03 до 6.521e-04 при $n(x) \in [0, 1]$, восстановление частоты по предложенной методике выглядит более оптимистично.

Таблица 5. $f(x)$, $0 < n(x) < 10$.

| $p$ | $e(\omega)$ | $e(\rho)$ | $e(f)$ |
|---|---|---|---|
| 32 | 1.027e-01 | 5.099e+00 | 2.460e-02 |
| 64 | 9.706e-03 | 5.204e+00 | 2.144e-02 |
| 128 | 6.284e-03 | 5.651e+00 | 2.130e-02 |
| 256 | 5.830e-03 | 4.836e+00 | 1.993e-02 |
| 512 | 1.553e-03 | 5.135e+00 | 1.781e-02 |
| 1024 | 2.200e-04 | 5.049e+00 | 1.665e-02 |

В таблице 5 приведены ошибки восстановления функции $f(x)$ по данным (47) при шуме $n(x) \in [0, 10]$. Ошибка определения коэффициента при $\cos(\pi x/2)$ равнялась 14% при $p$ =1024. В остальных случаях этот коэффициент отличался от истинного от 1.5 до 5 раз.

## 7 Выводы

Главные результаты можно сформулировать следующим образом:

1. Разработана методика построения линейного автономного алгоритма вычисления свёртки с заданным ядром.

2. Эта методика позволяет определить коэффициенты линейных алгоритмов весьма близкие к теоретически неулучшаемым по своим характеристикам.

3. Данная методика дает решение классической задачи аппроксимации суммой экспонент искаженных шумами экспериментальных данных в задачах цифровой обработки сигнала.

4. Численные эксперименты подтверждают устойчивость данной методики.

## 8 Благодарности

Эта работа была выполнена в Институте Прикладной математики им. М.В. Келдыша, Российской Академии Наук. Автор благодарен профессору В.С. Рябенькому за глубокое внимание к данной работе и А.Л. Плешкевичу (ЦГЭ) за полезные замечания, способствовавшие улучшению этой работы. Автор особенно признателен Б.Д. Плющенкову за плодотворное многолетнее сотрудничество по данной тематике.

## 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.

[Лок82] А.А.Локшин, Ю.В.Суворова, Математическая теория распространения волн в средах с памятью, Изд. Московского университета, 1982.

[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.

[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).

[Ply09] Б.Д. Плющенков, В.И.Турчанинов, *Пошаговая свёртка*, препринт ИПМ им. М.В. Келдыша РАН, 2009, № 24, ISSN 2071-2898.

[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.

[Marp] С.Л. Марпл-мл. Цифровой спектральный анализ и его приложения. - М.: "Мир", 1990, С.584.

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

[Beyl05] G. Beylkin, L. Monzon, On approximations of functions by exponential sums, Appl. Comput. Harmon. Anal. 19 (2005) 17-48

[Mhas05] H.N. Mhaskar, J. Prestin, On local smoothness classes of periodic functions, J.Fourier Anal.Appl. 11(2005) 353-373.

[Potts] D. Potts, M. Tasche, *Parameter estimation for exponential sums by approximate Prony method*, Signal Process., in print.