УДК 517.538

М. С. Султанмахмедов, Т. Н. Шах-Эмиров

Алгоритм быстрого дискретного преобразования для сумм Фурье по ортогональным по Соболеву полиномам, порожденным полиномами Чебышева первого рода

Рассмотрена задача о численной реализации линейных комбинаций вида $S_N(x) = \sum_{k=0}^{N-1} p_k T_{1,k+1}(x)$, где $T_{1,n}(x)$ ($n = 0, 1, \dots$) — ортогональные по Соболеву полиномы, порожденные полиномами Чебышева первого рода $T_0 = 1/\sqrt{2}$, $T_n(x) = \cos(n \arccos x)$ ($n \in \mathbb{N}$) посредством равенств $T_{1,0} = 1$, $T_{1,n+1}(x) = \int_{-1}^x T_n(t) dt$ ($n = 1, \dots$). Для решения этой задачи на сетке $x_j = \cos \frac{(2j+1)\pi}{2M}$ ($0 \leq j \leq M-1$) осуществлен ряд преобразований выражения $S_N(x)$, которые в итоге позволяют свести рассматриваемую задачу к применению быстрого дискретного преобразования Фурье. Разработаны соответствующий алгоритм и программа на языке C#. С их помощью проведены численные эксперименты, которые показывают, что алгоритм, основанный на быстром преобразовании значительно выигрывает в смысле скорости вычислений по сравнению с методом непосредственного вычисления суммы $S_N(x)$ пользуясь явным видом полиномов $T_{1,n}(x)$.

Библиография: 3 названия.

We consider the problem of numerical realization of linear combinations $S_N(x) = \sum_{k=0}^{N-1} p_k T_{1,k+1}(x)$, where $T_{1,n}(x)$ ($n = 0, 1, \dots$) are Sobolev orthogonal polynomials generated by Chebyshev polynomials of the first kind $T_0 = 1/\sqrt{2}$, $T_n(x) = \cos(n \arccos x)$ ($n \in \mathbb{N}$) as follows $T_{1,0} = 1$, $T_{1,n+1}(x) = \int_{-1}^x T_n(t) dt$ ($n = 1, \dots$). To solve this problem on the grid $x_j = \cos \frac{(2j+1)\pi}{2M}$ ($0 \leq j \leq M-1$) a number transformations of expression $S_N(x)$ are applied, which allows to reduce the problem under consideration to the application of a fast discrete Fourier transform. The corresponding algorithm and program in the language C# have been developed. The numerical experiments with using this program show that algorithm based on fast transform is significantly faster than method based on direct calculation of $S_N(x)$ with using explicit formulas for $T_{1,n}(x)$.

Bibliography: 3 items.

Ключевые слова: полиномы Чебышева; полиномы, ортогональные по Соболеву; быстрое преобразование Фурье; дискретное косинусное преобразование.

Keywords: Chebyshev polynomials, Sobolev orthogonal polynomials, fast Fourier transform, discrete cosine transform.

Введение

В последние десятилетия получила интенсивное развитие теория полиномов, ортогональных относительно скалярного произведения типа Соболева, которых принято называть полиномами, ортогональными по Соболеву [1, 2]. В

работе [2] были введены и исследованы полиномы $T_{r,k}(x)$ ($k = 0, 1, \dots$), ортогональные по Соболеву, порожденные полиномами Чебышева первого рода $T_0(x) = \frac{1}{\sqrt{2}}$, $T_k(x) = \cos(k \arccos x)$, $k = 1, 2, \dots$, посредством следующих равенств

$$T_{r,k}(x) = \frac{(x+1)^k}{k!}, \quad k = 0, 1, \dots, r-1, \quad (0.1)$$

$$T_{r,r}(x) = \frac{(x+1)^r}{\sqrt{2}r!}, \quad T_{r,r+n}(x) = \frac{1}{(r-1)!} \int_{-1}^x (x-t)^{r-1} T_n(t) dt, \quad n = 1, \dots \quad (0.2)$$

Эти новые полиномы $T_{r,k}(x)$ ортонормированы [2] относительно скалярного произведения

$$\langle f, g \rangle = \sum_{\nu=0}^{r-1} f^{(\nu)}(-1) g^{(\nu)}(-1) + \int_{-1}^1 f^{(r)}(x) g^{(r)}(x) \mu(x) dx, \quad (0.3)$$

где $\mu(x) = \frac{2}{\pi}(1-x^2)^{-\frac{1}{2}}$.

Обозначим через $W_{L_\mu^2(-1,1)}^r$ пространство Соболева, состоящее из функций $f = f(x)$, непрерывно дифференцируемых $r-1$ -раз, причем $f^{(r-1)}$ абсолютно непрерывна, а $f^{(r)} \in L_\mu^2(-1,1)$. В [2] было показано, что система полиномов $\{T_{r,k}(x)\}_{k=0}^\infty$ образует базис в $W_{L_\mu^2(-1,1)}^r$, кроме того в [2] показано, что ряд Фурье функции $f \in W_{L_\mu^2(-1,1)}^r$ имеет следующий вид

$$f(x) = \sum_{\nu=0}^{r-1} f^{(\nu)}(-1) \frac{(1+x)^\nu}{\nu!} + \sum_{k=r}^\infty \hat{f}_{r,k} T_{r,k}(x), \quad (0.4)$$

где

$$\hat{f}_{r,k} = \int_{-1}^1 f^{(r)}(t) T_{k-r}(t) \mu(t) dt, \quad (k \geq r). \quad (0.5)$$

При этом, как показано в [2], ряд Фурье, фигурирующий в правой части равенства (0.4), сходится к $f(x)$ равномерно относительно $x \in [-1, 1]$. В частности, при $r = 1$ ряд (0.4) принимает следующий вид:

$$f(x) = f(-1) + \sum_{k=1}^\infty \hat{f}_k T_{1,k}(x), \quad (0.6)$$

где

$$\hat{f}_k = \int_{-1}^1 f'(t) T_{k-1}(t) \mu(t) dt, \quad (k \geq 0). \quad (0.7)$$

Ряды Фурье вида (0.6) встречаются в различных прикладных задачах, в том числе при приближенном решении систем дифференциальных уравнений так называемыми спектральными методами путем применения некоторых итерационных процедур. При этом, в качестве промежуточной возникает необходимость многократного вычисления выражения вида

$$S_N(x) = S_N(p, x) = \sum_{k=0}^{N-1} p_k T_{1,k+1}(x), \quad (0.8)$$

где $p = \{p_k\}_{k=0}^{N-1}$ – произвольные действительные числа.

В настоящей работе для решения этой задачи на сетке $x_j = \cos \frac{(2j+1)\pi}{2M}$ ($0 \leq j \leq M-1$) осуществлен ряд преобразований выражения $S_N(x)$, которые в итоге позволяют свести рассматриваемую задачу к применению быстрого дискретного преобразования Фурье. Разработаны соответствующий алгоритм и программа на языке C#. С их помощью проведены численные эксперименты, которые показывают, что алгоритм, основанный на быстром преобразовании значительно выигрывает в смысле скорости вычислений по сравнению с методом непосредственного вычисления суммы $S_N(x)$ пользуясь явным видом полиномов $T_{1,n}(x)$.

1. Некоторые вспомогательные результаты

При разработке алгоритма решения этой промежуточной задачи нам понадобятся некоторые преобразования правой части равенства (0.8), направленные на то, чтобы показать, что при вычислении $S_N(x)$ на сетке $\{x_j\}_{j=0}^{M-1}$ может быть использовано быстрое дискретное преобразование Фурье. С этой целью воспользуемся следующими свойствами полиномов $T_{1,k}(x)$, установленными в работе [2]:

$$T_{1,0}(x) = 1, \quad T_{1,1}(x) = \frac{1+x}{\sqrt{2}}, \quad T_{1,2}(x) = \frac{1}{2}(x^2 - 1), \quad (1.1)$$

$$T_{1,k+1}(x) = \frac{T_{k+1}(x)}{2(k+1)} - \frac{T_{k-1}(x)}{2(k-1)} - \frac{(-1)^k}{k^2 - 1}, \quad (k \geq 2) \quad (1.2)$$

и запишем (0.8) в следующем виде

$$\begin{aligned} S_N(x) &= \frac{p_0}{\sqrt{2}}(1+x) + \frac{p_1}{2}(x^2 - 1) + \sum_{k=2}^{N-1} p_k \left[\frac{T_{k+1}(x)}{2(k+1)} - \frac{T_{k-1}(x)}{2(k-1)} - \frac{(-1)^k}{k^2 - 1} \right] \\ &= - \sum_{k=2}^{N-1} \frac{(-1)^k p_k}{k^2 - 1} + \frac{p_0}{\sqrt{2}}(1+x) + \frac{p_1}{2}(x^2 - 1) + \\ &\quad \sum_{k=2}^{N-1} p_k \left[\frac{T_{k+1}(x)}{2(k+1)} - \frac{T_{k-1}(x)}{2(k-1)} \right]. \quad (1.3) \end{aligned}$$

Поскольку

$$\begin{aligned} \sum_{k=2}^{N-1} p_k \left[\frac{T_{k+1}(x)}{2(k+1)} - \frac{T_{k-1}(x)}{2(k-1)} \right] &= \sum_{k=2}^{N-1} p_k \frac{T_{k+1}(x)}{2(k+1)} - \sum_{k=2}^{N-1} p_k \frac{T_{k-1}(x)}{2(k-1)} = \\ &= \sum_{k=2}^{N-1} p_k \frac{T_{k+1}(x)}{2(k+1)} - \sum_{k=0}^{N-3} p_{k+2} \frac{T_{k+1}(x)}{2(k+1)} = \\ &= -\frac{p_2}{2}T_1(x) - \frac{p_3}{4}T_2(x) + \sum_{k=3}^{N-2} \frac{p_{k-1} - p_{k+1}}{2k}T_k(x) + \frac{p_{N-2}}{2(N-1)}T_{N-1}(x) + \frac{p_{N-1}}{2N}T_N(x), \end{aligned}$$

то мы можем переписать (1.3) в виде

$$S_N(x) = \bar{p}(N) + \frac{p_0}{\sqrt{2}}(1+x) + \frac{p_1}{2}(x^2-1) - \frac{p_2}{2}T_1(x) - \frac{p_3}{4}T_2(x) + \\ \sum_{k=3}^{N-2} \frac{p_{k-1} - p_{k+1}}{2k} T_k(x) + \frac{p_{N-2}}{2(N-1)} T_{N-1}(x) + \frac{p_{N-1}}{2N} T_N(x), \quad (1.4)$$

где

$$\bar{p}(N) = - \sum_{k=2}^{N-1} \frac{(-1)^k p_k}{k^2 - 1}. \quad (1.5)$$

Далее имеем

$$\frac{p_0}{\sqrt{2}}(1+x) = p_0 T_0 + \frac{p_0}{\sqrt{2}} T_1(x), \quad \frac{p_1}{2}(x^2-1) = \frac{p_1}{4} T_2(x) - \frac{p_1}{2\sqrt{2}} T_0,$$

поэтому из (1.4) и (1.5) окончательно получаем

$$S_N(x) = \bar{p}(N) + \left(p_0 - \frac{p_1}{2\sqrt{2}}\right) T_0 + \left(\frac{p_0}{\sqrt{2}} - \frac{p_2}{2}\right) T_1(x) + \\ \sum_{k=2}^{N-2} \frac{p_{k-1} - p_{k+1}}{2k} T_k(x) + \frac{p_{N-2}}{2(N-1)} T_{N-1}(x) + \frac{p_{N-1}}{2N} T_N(x). \quad (1.6)$$

Полагая $x = \cos \theta$, это равенство можно переписать

$$Q_N(\theta) = S_N(\cos \theta) = \bar{p}(N) + \frac{p_0}{\sqrt{2}} - \frac{p_1}{4} + \left(\frac{p_0}{\sqrt{2}} - \frac{p_2}{2}\right) \cos \theta + \\ \sum_{k=2}^{N-2} \frac{p_{k-1} - p_{k+1}}{2k} \cos k\theta + \frac{p_{N-2}}{2(N-1)} \cos(N-1)\theta + \frac{p_{N-1}}{2N} \cos N\theta. \quad (1.7)$$

Таким образом, вычисление значений алгебраического полинома $S_N(x)$ на сетке x_j ($0 \leq j \leq M-1$) сводится к вычислению четного тригонометрического полинома $Q_N(\theta)$ на сетке $\theta_j = \frac{(2j+1)\pi}{2M}$ ($0 \leq j \leq M-1$), которое можно осуществлять путем быстрого дискретного косинусного преобразования Фурье. Перейдем к более подробному обсуждению этого вопроса.

2. Описание алгоритма вычисления суммы $S_N(x)$

В настоящем пункте будет описан алгоритм быстрого дискретного преобразования сумм $S_N(x)$. Полагая $x = \cos \theta$ и используя (1.7), мы можем переписать сумму (0.8) в виде следующей суммы по косинусам

$$Q_N(\theta) = d_0 + \sum_{k=1}^N d_k \cos k\theta = S_N(\cos \theta), \quad (2.1)$$

где

$$d_0 = \bar{p}(N) + \frac{p_0}{\sqrt{2}} - \frac{p_1}{4}, \quad d_1 = \frac{p_0}{\sqrt{2}} - \frac{p_2}{2}, \quad (2.2)$$

$$d_k = \frac{p_{k-1} - p_{k+1}}{2k}, \quad (2 \leq k \leq N-2), \quad (2.3)$$

$$d_{N-1} = \frac{p_{N-2}}{2(N-1)}, \quad d_N = \frac{p_{N-1}}{2N}. \quad (2.4)$$

Тогда значения частичной суммы (0.8) в узлах $\{x_j = \cos \theta_j\}_{j=0}^{M-1}$ могут быть вычислены с помощью формулы

$$Q_N(\theta_j) = d_0 + \sum_{k=1}^N d_k \cos k\theta_j, \quad (0 \leq j \leq M-1). \quad (2.5)$$

Как хорошо известно, эта задача в случае, когда $M \geq N+1$ и $M = 2^m$, $m \in \mathbb{N}$, может быть решена путем применения быстрого косинусного преобразования Фурье, основанного на алгоритме *DCT-II* (см. [3]).

Подводя итоги отметим, что алгоритм вычисления суммы $S_N(x)$, определяемой равенством (0.8), сводится к последовательному осуществлению следующих шагов:

1. На вход алгоритма подаются коэффициенты p_k ;
2. Отправляясь от коэффициентов p_k , по формулам (2.2) – (2.4) вычисляются новые коэффициенты d_k . Тогда исходное выражение (0.8) принимает вид (2.5);
3. Осуществляется вычисление значений $Q_N(\theta)$ в узлах сетки $\{\theta_j\}_{j=0}^{M-1}$ с помощью быстрого дискретного косинусного преобразования Фурье *DCT-II*.

3. Численные эксперименты

Авторами была разработана компьютерная программа, реализующая предложенный алгоритм.

Принцип работы с программой

1. Пользователь задает количество узлов сетки N и количество M экспериментов, которые он хочет провести.
2. Программа генерирует M наборов случайных коэффициентов в задаваемом пользователем диапазоне. Есть возможность отображения сгенерированных коэффициентов.
3. Пользователь выбирает — осуществление обычных либо быстрых преобразований над сгенерированными наборами коэффициентов.
4. После осуществления преобразований, программа выводит количество затраченного времени (всего и в среднем на один эксперимент), также средние по всем экспериментам погрешности двух видов — максимальное отклонение и отклонение в среднем. Имеется возможность отображения подробных сведений об ошибках для каждого эксперимента.

Далее мы приведем несколько скриншотов с пояснениями.

Составные части интерфейса:

1. Текстовое поле для вывода сообщений программы («консоль»).
2. Задание количества узлов сетки N . Для осуществления быстрых преобразований требуется чтобы это число было степенью двойки.
3. Задание количества экспериментов M .
4. Задание разброса случайных коэффициентов (дисперсия).

Рис. 1. Главный экран программы.

5. Задание математического ожидания наборов случайных коэффициентов.
6. Служит для выбора – отображать подробные данные о сгенерированных коэффициентах или нет.
7. Генерирование коэффициентов с выбранными параметрами.
8. Служит для выбора – отображать подробные об ошибках в экспериментах или нет.
9. Осуществление преобразований по формуле (0.8).
10. Осуществление быстрых преобразований по формуле (2.5).
11. Очистка «консоли».
12. Выход из программы.

Рис. 2. Сгенерирован набор случайных коэффициентов. Наборов – 50, количество коэффициентов в каждом – 1023. Справа – подробные данные о коэффициентах.

Проведено 500 экспериментов с 1023 коэффициентами в каждом. Средняя максимальная погрешность составила $3.29007932009517E-14$, отклонение в среднем – $4.12676406860645E-15$. На осуществление всех преобразований затрачено 14808.8839 миллисекунд (мс), в среднем на эксперимент затрачено 29.6177678 мс.

Далее, на том же наборе коэффициентов произведены быстрые преобразования. Средняя максимальная погрешность составила $5.59753976975058E-14$,

Рис. 3. Осуществлены обычное (слева) и быстрое (справа) преобразования суммы Фурье по системе полиномов $T_{1,k}(x)$.

отклонение в среднем – $1.2463832175076E-14$. На осуществление всех преобразований затрачено 593.7384 мс, в среднем на эксперимент затрачено 1.1874768 мс. Также приведем сравнительную таблицу среднего времени вычисления $S_N(x_j)$ при различных N посредством быстрой и обычной формул в миллисекундах:

Число узлов	Обычная формула	С использованием БПФ
8	0.00225264	0.00704127
16	0.00647647	0.01316473
32	0.02033089	0.02625329
64	0.06989277	0.0542377
128	0.28020358	0.12493092
256	1.15891625	0.27047539
512	6.80259062	0.61383475
1024	37.28032723	1.0818284
2048	147.88811337	2.42418507
4096	714.4442895	4.8854531

4. Листинг программы

В этом разделе мы приводим листинг основных функциональных частей разработанной программы.

*Основные функции **SobCheb1Benchmark** и **SobCheb1Benchmark_Fast**, осуществляющие обычное и быстрое преобразования соответственно и вычисляющие погрешности и затраченное время*

```
public static bool SobCheb1Benchmark(double[] F, out double avg,
                                     out double max, out TimeSpan time)
{
    //Вычисление преобразования по явным формулам
    //Используется для сравнения и нахождения ошибок
    double[] f_expl = SobCheb1RestoreExplicit(F);

    //Вычисление по формуле (2.6), без быстрых преобразований
    Stopwatch sw = new Stopwatch();
    sw.Start();
```

```

double[] f_cheb = SobCheb1Restore(F);
sw.Stop();
time = sw.Elapsed;

//Вычисляем погрешности
return
    MathTools.ArrayDifference(f_expl, f_cheb, out avg, out max);
}

public static bool SobCheb1Benchmark_Fast(double[] F, out double avg,
                                          out double max, out TimeSpan time)
{
    //Вычисление преобразования по явным формулам
    //Используется для сравнения и нахождения ошибок
    double[] f_expl = SobCheb1RestoreExplicit(F);

    //Быстрое преобразование
    Stopwatch sw = new Stopwatch();
    sw.Start();
    double[] f_cheb = SobCheb1Restore_Fast(F);
    sw.Stop();
    time = sw.Elapsed;

    //Вычисляем погрешности
    return
        MathTools.ArrayDifference(f_expl, f_cheb, out avg, out max);
}

```

Для перехода к новым коэффициентам используется функция

```

static double[] prepareChebCoeffs(double[] F)
{
    int N = F.GetLength(0)+1;
    int n = N - 1;
    double pN = 0.0;
    for (int k = 2; k < n; k++) {
        pN += MathTools.MinusOne(k) * F[k] / (k * k - 1.0);
    }
    pN = -pN;

    double[] D = new double[N];
    double Sqrt2 = Math.Sqrt(2);
    D[0] = F[0] / Sqrt2 - F[1] * 0.25 + pN;

    if (n < 3) {
        D[1] = F[0] / Sqrt2;
    } else {
        D[1] = F[0] / Sqrt2 - 0.5 * F[2];
    }
    if (n > 1) {
        for (int k = 2; k < n - 1; k++)
            D[k] = (F[k-1] - F[k + 1]) / (2.0 * k);
    }
    if (n > 2) {
        D[n-1] = F[n-2] / (2.0 * n - 2.0);
    }
    D[n] = F[n-1] / (2.0 * n);

    return D;
}

```

Для вычисления полиномов, ортогональных в смысле Соболева, применяется функция

```

public static double[][] SobCheb1Normed(double[] t, int m = -1)
{
    int N = t.Length;
    if (m < 2) { m = N; }

    double tmp = 1.0 / Math.Sqrt(2);

```



```

double [][] T1r = new double[m + 1][];
T1r[0] = new double[N];
T1r[1] = new double[N];
T1r[2] = new double[N];

//Вычисляем T10, T11 and T12
for (int j = 0; j < N; j++) {
    T1r[0][j] = 1.0;
    T1r[1][j] = tmp * (1.0 + t[j]);
    T1r[2][j] = 0.5 * (t[j] * t[j] - 1.0);
}

//Вычисляем T1k, k >= 3
double [][] T = Cheb1Normed(t, m); //полиномы Чебышева Tn
double c1, c2, c3;

for (int k = 3; k <= m; k++) {
    T1r[k] = new double[N];
    //расчет коэффициентов
    c1 = 0.5 / (double)k; c2 = 0.5 / (k - 2.0);
    if (k % 2 == 0) {
        c3 = 1.0 / (double)k / (k - 2.0);
    } else {
        c3 = -1.0 / (double)k / (k - 2.0);
    }

    //основной цикл вычислений
    for (int j = 0; j < N; j++) {
        T1r[k][j] = c1 * T[k][j] - c2 * T[k - 2][j] + c3;
    }
}

return T1r;
}

```

Для вычисления значений частичной суммы $S_N(f, x)$ быстрым и обычным способом служат функции:

```

public static double[] SobCheb1Restore(double[] F, double[] t = null)
{
    int N = F.GetLength(0)+1;
    int n = N - 1;

    //Генерирование сетки
    if (t == null) {
        t = GridGenerator.GetChebZeroes(N);
    } else {
        N = t.Length;
    }

    //Подготовка коэффициентов
    double[] D = prepareChebCoeffs(F);
    D[0] *= Math.Sqrt(2);

    //Восстановление
    double[] Result = Cheb.RestoreCheb1NormedSum(D, t);

    return Result;
}

public static double[] SobCheb1Restore_Fast(double[] F)
{
    //Подготовка коэффициентов
    double[] D = SobCheb.prepareChebCoeffs(F);

    //Восстановление
    double[] Result = Cheb.RestoreCheb1NormedSum_Fast(D);

    return Result;
}

```

Кроме того, программа включает классы ***Cheb***, ***FFT***, ***DCT***, ***IDCT*** и ряд других, используемые внутри указанных функций в качестве служебных, а также класс ***GridGenerator*** для генерации случайных коэффициентов и различных сеток (нулей многочлена Чебышева и его производных и др.).

Список литературы

- [1] Marcellan F., Yuan Xu. On Sobolev orthogonal polynomials // *Expositiones Mathematicae*. 2015. Vol. 33. Issue 3. Pp. 308–352.
- [2] Шарапудинов И.И. Системы функций, ортогональные по Соболеву, ассоциированные с ортогональной системой // *Изв. РАН. Сер. матем.* 2018. Том. 82, вып. 1. С. 225–258.
- [3] K. Rao, P. Yip. *Discrete Cosine Transform. Algorithms, Advantages, Applications*. 1st Edition. Academic Press, 2014. 512 p.

М. С. Султанакмедов (M. S. Sultanakhmedov)

Дагестанский научный центр РАН

E-mail: sultanakhmedov@gmail.com

Поступила в редакцию

15.03.2018

Т. Н. Шах-Эмиров (T. N. Shakh-Emirov)

Дагестанский научный центр РАН

E-mail: tadgius@gmail.com