УДК 519.688

Г. Г. Акниев, Р. М. Гаджимирзаев

Алгоритм численной реализации полиномов по функциям, ортогональным по Соболеву и порожденным косинусами

Разработан алгоритм, основанный на быстром дискретном преобразовании Фурье для численной реализации на сетке $\{t_j = \frac{j}{N}\}_{j=0}^{N-1}$ полиномов по функциям $\xi_{1,0}(t) = 1$, $\xi_{1,1}(t) = t$, $\xi_{1,n+1}(t) = \frac{\sqrt{2}}{\pi n} \sin(\pi nt)$, $n = 1, 2, \dots$, ортогональным по Соболеву относительно скалярного произведения $\langle f, g \rangle = f(0)g(0) + \int_0^1 f'(t)g'(t)dt$, порожденным косинусами $\xi_0(x) = 1$, $\{\xi_n(t) = \sqrt{2} \cos(\pi nt)\}_{n=1}^\infty$.

Библиография: 2 названия.

In this paper we developed an algorithm for numerical computation of polynomials by the functions $\xi_{1,0}(t) = 1$, $\xi_{1,1}(t) = t$, $\xi_{1,n+1}(t) = \frac{\sqrt{2}}{\pi n} \sin(\pi nt)$, ($n = 1, 2, \dots$) on the grid $\{t_j = \frac{j}{N}\}_{j=0}^{N-1}$. These functions are orthogonal on Sobolev with respect to the inner product $\langle f, g \rangle = f(0)g(0) + \int_0^1 f'(t)g'(t)dt$ and generated by functions $\xi_0(x) = 1$, $\{\xi_n(t) = \sqrt{2} \cos(\pi nt)\}_{n=1}^\infty$. The algorithm is based on the fast Fourier transform.

Bibliography: 2 items.

Ключевые слова: быстрое преобразование Фурье, дискретное синус-преобразование, скалярное произведение типа Соболева, ортогональные по Соболеву функции.

Keywords: fast Fourier transform, discrete sine transform, inner product of Sobolev type, Sobolev orthogonal function.

Введение

В работе [1] была введена система функций

$$\xi_{1,0}(t) = 1, \quad \xi_{1,1}(t) = t, \quad \xi_{1,n+1}(t) = \frac{\sqrt{2}}{\pi n} \sin(\pi nt), \quad n = 1, 2, \dots, \quad (0.1)$$

порожденная системой косинусов

$$\xi_0(x) = 1, \quad \{\xi_n(t) = \sqrt{2} \cos(\pi nt)\}_{n=1}^\infty, \quad (0.2)$$

посредством равенств

$$\xi_{1,0}(t) = 1, \quad \xi_{1,n}(t) = \int_0^t \xi_{n-1}(\tau) d\tau, \quad n = 1, 2, \dots,$$

которая образует ортогональную систему в смысле Соболева относительно скалярного произведения

$$\langle \xi_{1,n}, \xi_{1,m} \rangle = \xi_{1,n}(0)\xi_{1,m}(0) + \int_0^1 \xi'_{1,n}(t)\xi'_{1,m}(t)dt.$$

В [1] было показано, что системы (0.1) и (0.2), взятые вместе являются весьма эффективным инструментом решения систем линейных и нелинейных дифференциальных уравнений. При численной реализации итерационного алгоритма приближенного нахождения неизвестных коэффициентов разложения искомого решения задачи Коши для ОДУ по системе (0.1) ключевую роль играет умение быстро вычислять значения полиномов по этой системе вида

$$S_{N-1}(t) = \frac{\sqrt{2}}{\pi} \sum_{k=1}^{N-1} \frac{p_k}{k} \sin \pi kt \quad (0.3)$$

на сетке $\{t_j = \frac{j}{N}\}_{j=0}^{N-1}$, где p_k заданные действительные числа. В настоящей статье мы предлагаем один из возможных подходов к численной реализации полинома на указанной сетке, основанный на алгоритме быстрого дискретного преобразования Фурье (БПФ).

1. Описание алгоритма

Опишем подробнее алгоритм работы программы для быстрого вычисления значений полинома (0.3) в узлах сетки $\{t_j\}_{j=0}^{N-1}$.

Входными данными являются: число N — степень двойки и массив вещественных чисел p длины N . В массиве p мы будем хранить значения $0, p_1, p_2, \dots, p_{N-1}$. После этого создадим массив f длины $2N$, который мы заполним следующим образом:

$$f[k] = \begin{cases} 0, & k = 0 \text{ или } k = N, \\ p[k]/k, & 1 \leq k \leq N-1, \\ -f[2N-k], & N+1 \leq k \leq 2N-1. \end{cases}$$

Перед тем, как применить к массиву f обратное БПФ отметим, что алгоритм обратного БПФ принимает на вход комплексный массив и возвращает комплексный массив. В том случае, когда входной массив является вещественным, возможен вариант преобразования его в комплексный путем добавления нулевой мнимой части. Однако возможен более оптимальный алгоритм, учитывающий тот факт, что входной массив состоит из вещественных чисел [2; с. 618]. Мы будем пользоваться этим алгоритмом в данной работе. Создадим массив комплексных чисел h длины N . Массив h заполним следующим образом:

$$h[k] = f[2k] + if[2k+1], \quad k = 0, 1, \dots, N-1.$$

Теперь к массиву h применим обратное БПФ и запишем результат в массив q длины N . Как нетрудно убедиться [2; с. 621], вектор $(q_0, q_1, \dots, q_{N-1})$ представляют собой результат обратного дискретного синус-преобразования вектора

$\{p_k\}_{k=1}^N$, умноженное на $2i$ на сетке $\{t_j\}_{j=0}^{N-1}$. Поделив значения $q_0, q_1, \dots, q_{N-1}$ на $2i$ и умножив на $\frac{\sqrt{2}}{\pi}$ мы получим значения (0.3) в узлах $\{t_j\}_{j=0}^{N-1}$. Теперь перейдем к листингу программы.

2. Листинг программы

Входные данные: число $N = 2^m$ и массив p длины N .

```
//Заполнение массива f
var f = new double[2 * N];
f[0] = 0;
for (var i = 1; i < N; i++)
{
    f[i] = p[i] / i;
}
f[N] = 0;
for (var i = 1; i < N; i++)
{
    f[2 * N - i] = -p[i] / i;
}

//Заполнение массива h
var h = new Complex[N];
for (var i = 0; i < N; i++)
{
    h[i] = new Complex(f[2*i], f[2*i+1]);
}

var q = RealFFT(h);
var a = new Complex[N];
var imagNum = new Complex(0, 2);
for (var i = 0; i < N; i++)
{
    a[i] = Sqrt(2.0) / PI * q[i] / imagNum;
}
```

Функция *RealFFT*, используемая в программе принимает массив комплексных чисел и возвращает результат применения обратного БПФ к данному массиву. Листинг кода данной функции приведен ниже.

```
private static Complex[] RealFFT(Complex[] a)
{
    var N = a.Length;
    var expArray = new Complex[N];

    for (var k = 0; k < N; k++)
    {
        expArray[k] = new Complex(Cos(PI * k / N),
```

```

        Sin(PI * k / N));
    }
    var imag = new Complex(0, 1);
    var b = new Complex[N];
    var a1 = IFFT(a);

    b[0] = new Complex(a1[0].Real, -a1[0].Imaginary);
    for (int i = 1; i < N; i++)
    {
        b[i] = new Complex(a1[N - i].Real,
                           -a1[N - i].Imaginary);
    }
    var g1 = new Complex[N];
    var g2 = new Complex[N];
    var b1 = new Complex[N];
    for (int i = 0; i < N; i++)
    {
        g1[i] = (a1[i] + b[i]) / 2;
        g2[i] = -imag * (a1[i] - b[i]) / 2;
    }
    for (int i = 0; i < N; i++)
    {
        b1[i] = g1[i] + expArray[i] * g2[i];
    }
    return b1;
}

private static Complex[] IFFT(Complex[] array)
{
    var N = array.Length;

    if (N == 1)
    {
        return array;
    }

    var omegaN = new Complex(Cos(2 * PI / N),
                              Sin(2 * PI / N));

    var omega = new Complex(1, 0);

    var even = new Complex[N / 2];
    var odd = new Complex[N / 2];
    for (var i = 0; i < N / 2; i++)
    {
        even[i] = array[2 * i];
        odd[i] = array[2 * i + 1];
    }

```

```

}

var y0 = IFFT(even);
var y1 = IFFT(odd);

var y = new Complex[N];

for (var k = 0; k < N/2; k++)
{
    y[k] = y0[k] + omega * y1[k];
    y[k + n / 2] = y0[k] - omega * y1[k];
    omega = omega * omegaN;
}
return y;
}

```

3. Численный эксперимент

Были проведены численные эксперименты на компьютере. Использовался CPU Intel i7 6600k, 4.0ГГц, вычисления производились в однопоточном режиме. Ниже приведена таблица, в которой сравнивается время выполнения вышеописанного алгоритма численной реализации полинома (0.3) на сетке $\{t_j\}_{j=0}^{N-1}$ при различных значениях N , с временем вычисления этих же значений при использовании явной формулы (0.3).

N	G1S	G1F
32	0.0000412	0.0000158
64	0.0001723	0.0000278
128	0.0006547	0.0000527
256	0.0026109	0.0001207
512	0.0104431	0.0002641
1024	0.0431023	0.0005400
2048	0.1709378	0.0011593
4096	0.6581820	0.0023930
8192	2.6002868	0.0050222
16384	10.3937830	0.0104718
32768	41.3140566	0.0217692

Таблица 1. Время вычисления (0.3) на ЭВМ в секундах, где G1S — это время вычисления (0.3) на сетке $\{t_j\}_{j=0}^{N-1}$ по формуле, G1F — время вычисления (0.3) на этой же сетке с использованием обратного БПФ.

Список литературы

- [1] Шарапудинов И.И., Магомед-Касумов М.Г. Численный метод решения задачи

Коши для систем обыкновенных дифференциальных уравнений с помощью ортогональной в смысле Соболева системы, порожденной системой косинусов // Дагестанские электронные математические известия. 2017. Вып. 8. С. 53–60.

- [2] William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery Numerical Recipes: The Art of Scientific Computing (3rd ed.). New York: Cambridge University Press. 2007. P. 1262.

Г. Г. Акниев (G. G. Akniyev)

Дагестанский научный центр РАН

E-mail: hasan.akniyev@gmail.com

Поступила в редакцию

27.03.2018

Р. М. Гаджимирзаев (R. M. Gadzhimirzaev)

Дагестанский научный центр РАН

E-mail: ramis3004@gmail.com