ЧИСЛЕННОЕ ОПРЕДЕЛЕНИЕ КОЭФФИЦИЕНТОВ ФУРЬЕ С ИСПОЛЬЗОВАНИЕМ УСЕЧЕННЫХ ГИПЕР-ДУАЛЬНЫХ ЧИСЕЛ

Олифер В. И.

1. Вводная часть

Разложение функции в ряд Фурье является мощным инструментом при решении самых разных задач благодаря тому, что ряд Фурье прозрачным образом ведёт себя при дифференцировании, интегрировании, сдвиге функции по аргументу и свёртке функций. Особенно это актуально для исходных функций, заданных сложным программным кодом. Тригонометрический ряд Фурье функции $f(x)$ определяется соотношением [1]:

$$f(x) \approx T_N(x) = \frac{a_0}{2} + \sum_{n=1}^{N} (a_n \cos(c_n x) + b_n \sin(c_n x)), \quad x \in [a, b] \tag{1.1}$$

где $(n = 0, 1, 2, \dots, N)$:

$$a_n = \frac{1}{L}\int_{-L}^{L} \varphi_n(x)dx, \quad b_n = \frac{1}{L}\int_{-L}^{L} \psi_n(x)dx, \quad c_n = \frac{n\pi}{L}, \quad L = \frac{b-a}{2},$$
$$\varphi_n(x) = f(x) \cdot \cos(c_n x), \quad \psi_n(x) = f(x) \cdot \sin(c_n x) \tag{1.2}$$

От коэффициентов Фурье a_n и b_n нетрудно перейти к комплексным коэффициентам Фурье $\hat{f}_k$ используя следующую формулу: $\hat{f}_k = (a_{|k|} - i{\cdot}sgn(k){\cdot}b_{|k|})/2$; где $i^2 = -1$, $k = 0, \pm1, \pm2, \dots, \pm N$. Обратный переход осуществляется соотношениями: $a_n = \hat{f}_n + \hat{f}_{-n}$; $b_n = i{\cdot}(\hat{f}_n + \hat{f}_{-n})$.

Правая часть выражения (1.1) $T_N(x)$ является тригонометрическим многочленом степени N. При $N \to \infty$: $T_N(x) \underset{[a,b]}{\Longrightarrow} f(x)$. В практических расчетах естественно ограничиваются несколькими первыми членами ряда Фурье, что и определяет приближённое выражение исходной функции $f(x)$ на интервале $[a,\ b]$. В численном (практическом) гармоническом анализе коэффициенты Фурье a_n и b_n разложения (1.1) определяются методами численного интегрирования, например, методом прямоугольников, трапеций и т. д. Большинство этих методов состоит в следующем. Заданный отрезок интегрирования $[a, b]$ делится опорными точками $x_0, x_1, x_2, \dots, x_k$ на k равных отрезков длиной $\Delta x = 2h = (b-a)/k$. На каждом из этих отрезков $[x_i, x_{i+1}]$ исходная подынтегральная функция $\tau(x)$ обычно аппроксимируется многочленами той либо иной степени. Коэффициенты аппроксимирующих многочленов определяются из условий сопряжения соседних отрезков. При этом, если условия сопряжения требуют помимо значений самой функции ещё и значения её производных, то используются разностные схемы для их вычисления, что сопряжено с оптимизацией Δx. Применение разностных схем нуждается в компромиссе между погрешностями усечения и округления, т.е. нахождении оптимальной величины приращения аргумента x. Более того, задача численного дифференцирования является некорректной и её нужно решать методами регуляции [2].

2. Аппроксимация многочленами третьей степени

В работе [3] представлен метод численного интегрирования, согласно которому на каждом i-ом отрезке исходная подынтегральная функция $\tau(x)$ аппроксимируется многочленом третьей степени относительно средины рассматриваемого отрезка. Коэффициенты аппроксимирующих многочленнов находятся из условий сопряжения соседних отрезков, а также значений функции в центре каждого из отрезков. Тогда формула численного интегрирования для функции $\tau(x)$ имеет вид

$$J[\tau(x)] = \int_a^b \tau(x)\,dx \approx \sum_{i=1}^{k} S_i, \tag{2.1}$$

где:

$$\begin{gathered} S_i = [2h{\cdot}\tau_i + r_1{\cdot}\Delta\tau_i + r_2{\cdot}\Delta\tau_i' + r_3{\cdot}\Delta\tau_i''], \\ \tau_i = \tau(x_i), \quad \Delta\tau_i = \tau(x_{1i}) + \tau(x_{2i}) - 2{\cdot}\tau(x_i), \\ \Delta\tau_i' = \tau'(x_{1i}) - \tau'(x_{2i}), \quad \Delta\tau_i'' = \tau''(x_{1i}) + \tau''(x_{2i}), \\ x_{1i} = x_i - h, \quad x_{2i} = x_i + h, \quad x_i \text{ - центр } i\text{-го участка}, \\ r_1 = \frac{3}{4}h, \quad r_2 = \frac{1}{4}h^2, \quad r_3 = \frac{1}{24}h^3, \\ (\dots)', \ (\dots)'' - \text{первая и вторая производные соответственно} \end{gathered} \tag{2.2}$$

Если исходная функция $\tau(x)$ на участке интегрирования $[a, b]$ имеет разрыв (discontinuity) первого (removable or jump) или второго рода (essential) в точке $x_d \in [a, b]$ (point of discontinuity), то участок интегрирования $[a, b]$ разбивается на два последовательных участка $[a, x_d - 0]$ и $[x_d + 0, b]$, а затем производится последовательное интегрирование на этих участках

$$J[\tau(x)] = \int_a^b \tau(x)\,dx = \int_a^{x_d-0} \tau(x)\,dx + \int_{x_d+0}^{b} \tau(x)\,dx \tag{2.3}$$

В случае разрыва второго рода, когда $\tau(x_d) \to \pm\infty$ следует воспользоваться следующмими соотношениями

$$J[\tau(x)] = \int_a^b \tau(x)\,dx = \int_a^{x_d-\delta} \tau(x)\,dx + \int_{x_d+\delta}^{b} \tau(x)\,dx, \tag{2.4}$$

где: δ – малая величина, зависимая от заданной точности вычисления.
Применяя (2.1) к (1.2) получим $(n = 0, 1, 2, \dots, N)$

$$a_n = I[\varphi_n(x)], \quad b_n = I[\psi_n(x)], \tag{2.5}$$

где:

$$\begin{gathered} I[\tau_n(x)] \approx \frac{1}{L}\sum_{i=1}^{k} S_{in} = \frac{1}{L}\sum_{i=1}^{k}[2h{\cdot}\tau_{in} + r_1{\cdot}\Delta\tau_{in} + r_2{\cdot}\Delta\tau_{in}' + r_3{\cdot}\Delta\tau_{in}''], \\ \tau_{in} = \tau_n(x_i), \quad \Delta\tau_{in} = \tau_n(x_{1i}) + \tau_n(x_{2i}) - 2{\cdot}\tau_n(x_i), \\ \Delta\tau_{in}' = [\tau_n(x_{1i}) - \tau_n(x_{2i})]', \quad \Delta\tau_{in}'' = [\tau_n(x_{1i}) + \tau_n(x_{2i})]'', \\ x_{1i} = x_i - h, \quad x_{2i} = x_i + h, \quad x_i \text{ - центр } i\text{-го участка}, \end{gathered} \tag{2.6}$$

$$r_1 = \frac{3}{4}h, \qquad r_2 = \frac{1}{4}h^2, \qquad r_3 = \frac{1}{24}h^3$$

Из (2.5) и (2.6) видно, что для получения $n-$го коэффициента Фурье a_n или b_n необходимо располагать значениями исходной функции, её первой и второй производных в $(2k + 1)$ точках интервала интегрирования $[-L, L]$.

В выше цитированой работе [3] для вычисления значений исходной функции и её производных использован метод автоматическим дифференцированием на основе усеченных гипер-дуальных чисел. Последнее обеспечивает точное (с точностью представления чисел в компьютерной системе) вычисление значений сложных математических функций и их производных, а также позволяет избежать дублирование функциональности программного кода (изменение кода функции не требует изменения кода ее производных).

3. Усеченные гипер-дуальные числа.

Согласно [4] усеченное гипер-дуальное число (truncated hyper-dual number) определяется выражением $X = x + x_1\boldsymbol{\varepsilon} + x_2\boldsymbol{\omega}$, где x, x_1 и x_2 – вещественные числа, $\boldsymbol{\varepsilon}$ и $\boldsymbol{\omega}$ – мнимые символы. Пространство усеченных гипер-дуальных чисел отвечает трехмерной алгебре с правилом умножения элементов базиса $\{\mathbf{1}, \boldsymbol{\varepsilon}, \boldsymbol{\omega}\}$:

×	$\mathbf{1}$	$\boldsymbol{\varepsilon}$	$\boldsymbol{\omega}$
$\mathbf{1}$	1	$\boldsymbol{\varepsilon}$	$\boldsymbol{\omega}$
$\boldsymbol{\varepsilon}$	$\boldsymbol{\varepsilon}$	$2\boldsymbol{\omega}$	0
$\boldsymbol{\omega}$	$\boldsymbol{\omega}$	0	0

(3.1)

Таблица 3.1. Правила умножения элементов базиса усечённых гипер-дуальных чисел

Число $x = Re(X) = X.Re$ называется главной частью X, а $x_1 = Im_1(X) = X.Im1$ и $x_2 = Im_2(X) = X.Im2$ – мнимыми частями X.

Алгебраические операции сложения, умножения, обращения и деления (с учетом табл. 3.1) определены по правилам:

$$\begin{aligned}
&A = a + a_1\boldsymbol{\varepsilon} + a_2\boldsymbol{\omega}, \qquad B = b + b_1\boldsymbol{\varepsilon} + b_2\boldsymbol{\omega}\\
&A + B = a + b + (a_1 + b_1)\boldsymbol{\varepsilon} + (a_2 + b_2)\boldsymbol{\omega},\\
&A{\cdot}B = a{\cdot}b + (a{\cdot}b_1 + b{\cdot}a_1)\boldsymbol{\varepsilon} + (a{\cdot}b_2 + 2{\cdot}a_1{\cdot}b_1 + b{\cdot}a_2)\boldsymbol{\omega},\\
&A^{-1} = a^{-1} - a_1{\cdot}a^{-2}\boldsymbol{\varepsilon} + (2{\cdot}a_1^2{\cdot}a^{-3} - a_2{\cdot}a^{-2})\boldsymbol{\omega},\\
&{}^{A}/_{B} = A{\cdot}B^{-1} = a{\cdot}b^{-1} + (a_1{\cdot}b^{-1} - a{\cdot}b_1{\cdot}b^{-2})\boldsymbol{\varepsilon} + [2{\cdot}(a{\cdot}b_1^2{\cdot}b^{-3} - a_1{\cdot}b_1{\cdot}b^{-2}) -\\
&\qquad a{\cdot}b_2{\cdot}b^{-2} + a_2{\cdot}b^{-1}]\boldsymbol{\omega}
\end{aligned} \tag{3.2}$$

Отметим одно интересное свойство числа $A = 1 + 1\boldsymbol{\varepsilon} + 0\boldsymbol{\omega}$ в степени $n = 1, 2, 3, \ldots$: $A^n = 1 + n\boldsymbol{\varepsilon} + n(n-1)\boldsymbol{\omega}$: мнимая часть при $\boldsymbol{\omega}$ даёт последовательность прямоугольных чисел (pronic numbers) [5], а именно 0, 2, 6, 12, 20, 30, 42, 56, 72, … .

Функция усеченного гипер-дуального аргумента реализуется выражением

$$F(X) = f(x) + x_1{\cdot}f'(x)\boldsymbol{\varepsilon} + (x_2{\cdot}f'(x) + x_1^2{\cdot}f''(x))\boldsymbol{\omega}, \tag{3.3}$$

которое при $X = x + 1\boldsymbol{\varepsilon} + 0\boldsymbol{\omega}$ принимает вид:

$$F(X) = f(x) + f'(x)\boldsymbol{\varepsilon} + f''(x)\boldsymbol{\omega} \tag{3.4}$$

Подстановка $A = F(X)$ и $B = \Phi(X)$ в (3.2) при $X = x + 1\boldsymbol{\varepsilon} + 0\boldsymbol{\omega}$ дает:

$$F(X) + \Phi(X) = [f(x) + \varphi(x)] + [f'(x) + \varphi'(x)]\boldsymbol{\varepsilon} + [f''(x) + \varphi''(x)]\boldsymbol{\omega},$$

$$\begin{aligned} F(X)\cdot\Phi(X) \;=\; & [f(x)\cdot\varphi(x)] + [f(x)\cdot\varphi'(x) + \varphi(x)\cdot f'(x)]\boldsymbol{\varepsilon} \\ & + [f(x)\cdot\varphi''(x) + 2\cdot f'(x)\cdot\varphi'(x) + \varphi(x)\cdot f''(x)]\boldsymbol{\omega}, \end{aligned}$$

$$F(X)^{-1} = f(x)^{-1} - [f'(x)/f(x)^2]\boldsymbol{\varepsilon} + [\,2\cdot\varphi'(x)^2/f(x)^3 - f''(x)/f(x)^2]\boldsymbol{\omega}, \tag{3.5}$$

$$\begin{aligned} F(X)\cdot\Phi(X)^{-1} = {} & f(x)/\varphi(x) + [f'(x)/\varphi(x) - f(x)\cdot\varphi'(x)/\cdot\varphi(x)^2]\boldsymbol{\varepsilon} \\ & + [2\cdot(f(x)\cdot\varphi'(x)^2/\varphi(x)^3 - f'(x)\cdot\varphi'(x)/\varphi(x)^2) - f(x)\cdot\varphi''(x)/\varphi(x)^2 \\ & + f''(x)/\varphi(x)]\boldsymbol{\omega} \end{aligned}$$

Нетрудно видеть, что мнимые части всех соотношений (3.5) соответствуют классическим правилам дифференцирования (мнимая часть $\boldsymbol{\varepsilon}$ для первой производной и мнимая часть $\boldsymbol{\omega}$ для второй производной).

Вычисление сложной усеченной гипер-дуальной функции (function composition) вида $F = f_1(f_2(\dots f_n(X)\dots), X), X)$ (где значение f_n используется в качестве аргумента для f_{n-1}) необходимо начать с вычисления $F_n = f_n(X)$, продолжить вычислениями $F_{n-1} = f_{n-1}(F_n, X)$, $F_{n-2} = f_{n-2}(F_{n-1}, X), \dots, F = f_1(F_2, X)$.

Описание элементарных (базовых) функций усечённого гипер-дуального аргумента приведены в [4]. Например, $\ln(X) = \ln(x) + x^{-1}\boldsymbol{\varepsilon} - x^{-2}\boldsymbol{\omega}$, откуда $Re(\ln(X)) = f(x) = \ln(x)$, $Im_1(\ln(X)) = f'(x) = x^{-1}$ и $Im_2(\ln(X)) = f''(x) = -x^{-2}$.

4. Автоматическое дифференцирование

Автоматическое дифференцирование [6, 7] связанно с точным (точностью представления чисел в компьютерной системе) вычислением производных сложных математических функций. Автоматическое дифференцирование (АД) позволяет избежать дублирования функциональности программного кода (изменение кода функции не требует изменения кода ее производной). При компьютерной реализации АД необходимо создать новый тип данных, перезагрузить базовые математические функции и операции над ними. Если новый тип данных строится на основе усеченных гипер-дуальных чисел [4], то за одно обращение к перезагруженной функции точно вычисляются значения самой функции и ее первой и второй производных. Для применения АД (на базе усеченных гипер-дуальных чисел и усеченных гипер-дуальных функций) в соотношениях (2.5) и (2.6) необходимо выполнить следующую замену исходных переменных, базовых опереций и функций на соответствующие им перезагруженные усеченные гипер-дуальные числа, функции и операции:

$$x_i \;\to\; X_i = x_i + 1\cdot\boldsymbol{\varepsilon} + \;0\cdot\boldsymbol{\omega},$$
$$\varphi_n(x) \to \Phi_n(X), \qquad \psi_n(x) \to \Psi_n(X), \qquad \tau_n(x) \to T_n(X) \tag{4.1}$$

Тогда (2.5) и (2.6) запишутся в следующем виде:

$$a_n = \; I[\Phi_n(X)], \qquad b_n = \; I[\Psi_n(X)],$$
$$I[T_n(X)] \approx \frac{1}{L}\sum_{i=1}^{k} S_{in} = \frac{1}{L}\sum_{i=1}^{k}[2h\cdot\tau_{in} + r_1\cdot\Delta\tau_{in} + \; r_2\cdot\Delta\tau'_{in} + \; r_3\cdot\Delta\tau''_{in}] \tag{4.2}$$

и

$$\tau_{in} = T_n(X_i).Re, \quad \Delta\tau_{in} = [T_n(X_{1i}) + T_n(X_{2i}) - 2\cdot T_n(X_i)].Re,$$
$$\Delta\tau'_{in} = [T_n(X_{1i}) - \; T_n(X_{2i})].Im1, \quad \Delta\tau''_{in} = [T_n(X_{1i}) + \; T_n(X_{2i})].Im2, \tag{4.3}$$
$$X_{1i} = \; X_i - h, \qquad X_{2i} = \; X_i + h, \qquad X_i \text{ - центр } i\text{-го участка},$$
$$r_1 = \frac{3}{4}h, \qquad r_2 = \frac{1}{4}h^2, \qquad r_3 = \frac{1}{24}h^3$$

При вычислении (4.3) для a_n следует положить $T_n(X) = \Phi_n(X)$, а для b_n: $T_n(X) = \Psi_n(X)$.

5. Компьютерная реализация

Используя программный код на языке SWIFT, описывающий тип данных усеченные гипер-дуальные числа Thdn (truncated hyper-dual number) [4], была выполнена компьютерная реализация представленного выше метода (численного определения коэффициенты Фурье) на языке программирования SWIFT 5 для macOS 11.2.1. Для метода (4.2) были написаны компьютерные процедуры Fourier_AB(…), Fourier_ABd(…) и Integral(…) (см. Приложение 1), которые использовались в численных экспериментах. Процедура Fourier_AB(…) вычисляет коэффициенты Фурье для гладких функций, процедура Fourier_ABd(…) для функций, имеющих разрыв внутри интервала интегрирования, и процедура Integral(…) вычисляющая интеграл от любых гладких функций. В том же приложении даны некоторые примеры исходных функций в терминах усеченных гипер-дуальных чисел и обращения к процедурам Fourier_AB(…), Fourier_ABd(…).

6. Численные эксперименты

Для проведения численного анализа предлагаемого подхода были рассмотрены самые разнообразные исходные функции с разными пределами интегрирования и количеством расчетных точек. Некоторые результаты вычислений, сопоставленные с точными аналитическими решениями, для $L = \pi, N = 5, k = 60$, приведены в таблице 6.1.

n	Численное решение		Точное решение	
	a_n	b_n	a_n	b_n
$f(x) = x^2$				

0	6.57974…	0	6.5797362…	0
1	-4.00000…	0	-4.0	0
2	1.00000…	0	1.0	0
3	-0.44444…	0	-0.(4)	0
4	0.25000…	0	0.25	0
5	-0.16000…	0	-0.16	0
$f(x)=\begin{cases}-1, & x<0\\ 1, & x\geq 0\end{cases}$				
0	0	0	0	0
1	0.000…	1.27323…	0	1.2732339…
2	0	0	0	0
3	0.000…	0.42441…	0	0.4244131…
4	0	0	0	0
5	0.000…	0.25464…	0	0.2546479…
$f(x)=\begin{cases}0, & x<0\\ 1, & x\geq 0\end{cases}$				
0	1.00000…	0	1.0	0
1	0	0.63661...	0	0.6366197…
2	0	0	0	0
3	0	0.21220…	0	0.2122065…
4	0	0	0	0
5	0	0.12732...	0	0.1273239…
$f(x)=\lvert x\rvert=\begin{cases}-x, & x<0\\ x, & x\geq 0\end{cases}$				
0	3.14159...	0	3.1415926...	0
1	-1.27323...	0	-1.2732395...	0
2	0	0	0	0
3	-0.14147...	0	-0.1414710…	0
4	0	0	0	0
5	-0.05092…	0	-0.0509295…	0
$f(x)=\lvert\sin(x)\rvert=\begin{cases}-\sin(x), & x<0\\ \sin(x), & x\geq 0\end{cases}$				
0	1.27323…	0	1.2732395…	0
1	0	0	0	0
2	-0.42441…	0	-0.4244131…	0
3	0	0	0	0
4	-0.08488…	0	-0.0848826…	0
5	0	0	0	0

Таблица 6.1. Результаты вычисления коэффициентов Фурье

Общий анализ полученных вычислительных результатов показал, что описанный выше подход численного определения коэффициентов Фурье даёт заметно лучшие результаты, чем методы Симпсона и Рунге-Кутты 4-го порядка.

7. Заключение

Полученные формулы и процедуры численного определения коэффициентов Фурье, основанные на использовании типа данных Thdn и автоматического дифференцирования дают заметно лучшие результаты, чем известные методы численного интегрирования 4-го порядка погрешности, которые в некоторых случаях вызывают «пробуксовку».
Как видно из компьютерных процедур Fourier_AB(…), Fourier_ABd(…) и Integral(…) (см. Приложение 1), соотношения (4.2) и (4.3) относительно просто программируются.
Описанные выше процедуры численного определения коэффициентов Фурье обладают универсальностью, позволяющей рассматривать функцию *F(X)* как некий «черной ящик», поставляющий значение функции *f(x)* и точные величины ее первой и второй производных по данному значению $X = x + 1\cdot\boldsymbol{\varepsilon} + 0\cdot\boldsymbol{\omega}$.

Приложение 1.

Код для численного эксперимента на языке Swift 5 (macOS 11.2). Тип данных Thdn (truncated-dual number) приведен в [4].

```
/* FUNCTION Fourier_AB (...) - calculating Fourier coefficients for continuous functions
    INPUT:  N - maximum index of Fourier coefficients,
            K - number of sections forming the integration interval [-L, L],
            L - half period,
            F(X:Thdn) -> Thdn - original function
    OUTPUT: (A[ ], B[ ]) - Fourier coefficients A[a0, a1, ... aN], B[0, b1, b2, ... bN] */

func Fourier_AB(N:Int, K:Int, L:Double, F:(_ X:Thdn) -> Thdn)->(A:[Double], B:[Double]){
        var A:[Double] = [], B:[Double] = [];
        let Δ = 2.0*L/Double(K), h = Δ/2.0, h2 = h*h, h3 = h*h2,
            r1 = h*3.0/4.0, r2 = h2/4.0, r3 = h3/24.0, pl=Double.pi/L;
        var cn:Double,  F0, F1, F2:(Fa:Thdn, Fb:Thdn), Sx,  S:(Sa:Double, Sb:Double);
        func Q(X:Thdn)->(Fa:Thdn, Fb:Thdn){
            let Fx = F(X),  Xcn = cn*X;
            return (Fa:Fx*Thdn.cos(X: Xcn), Fb:Fx*Thdn.sin(X: Xcn));
        }
        func Sab(x:Double)->(Sa:Double, Sb:Double){
            F0 = Q(X:Thdn(re:x - h, im1:0, im2:0));
            F2 = Q(X:Thdn(re:x,     im1:1, im2:0));
            let S0 = (Sa: F0.Fa.re*Δ, Sb: F0.Fb.re*Δ);
            let S1 = (Sa: (F1.Fa.re + F2.Fa.re - 2.0*F0.Fa.re)*r1,
                      Sb: (F1.Fb.re + F2.Fb.re - 2.0*F0.Fb.re)*r1);
            let S2 = (Sa: (F1.Fa.im1 - F2.Fa.im1)*r2,
                      Sb: (F1.Fb.im1 - F2.Fb.im1)*r2);
            let S3 = (Sa: (F1.Fa.im2 + F2.Fa.im2)*r3,
                      Sb: (F1.Fb.im2 + F2.Fb.im2)*r3);
            return   (Sa: S0.Sa + S1.Sa + S2.Sa + S3.Sa,
                      Sb: S0.Sb + S1.Sb + S2.Sb + S3.Sb);
        }
        F0 = (Fa:Thdn(), Fb:Thdn());  F2 = (Fa:Thdn(), Fb:Thdn());
        for n in 0...N {
            S = (Sa: 0.0, Sb: 0.0);
            cn = Double(n)*pl;   F1 = Q(X: Thdn(re: -L, im1:1, im2:0));
```

```
            for x in stride(from: -L + Δ, through: L, by: Δ){
                Sx = Sab(x: x);    S = (Sa:S.Sa + Sx.Sa, Sb:S.Sb + Sx.Sb);
                F1 = F2;
            }
            A.append(S.Sa/L);    B.append(S.Sb/L);
        }
        return (A:A, B:B);
}

/* FUNCTION Fourier_ABd (...) - calculating Fourier coefficients for discontinuous functions
    INPUT:  N - maximum index of Fourier coefficients,
            K - number of sections forming the integration interval [-L, L],
            L - half period,
            F(X:Thdn) -> Thdn - original function,
            Xd - point of discontinuity
    OUTPUT: (A[ ], B[ ]) - Fourier coefficients A[a0, a1, ... aN], B[0, b1, b2, ... bN] */

func Fourier_ABd(N:Int, K:Int, L:Double, F:(_ X:Thdn) -> Thdn, Xd:Double)->(A:[Double], B:[Double]){
        let pl = Double.pi/L, δ = 1E-15;
        var A:[Double] = [], B:[Double] = [], a, b: Double;
        var cn:Double = 0;
        func Qc(X:Thdn)->Thdn{return F(X)*Thdn.cos(X: cn*X)}
        func Qs(X:Thdn)->Thdn{return F(X)*Thdn.sin(X: cn*X)}
        for n in 0...N {
            cn = Double(n)*pl;
             a =   Integral (xo: -L,  xn: Xd - δ, n: K, F: Qc);
             a += Integral (xo: Xd, xn: L,       n: K, F: Qc);
             b =   Integral (xo: -L,  xn: Xd - δ, n: K, F: Qs);
             b += Integral (xo: Xd, xn: L,       n: K, F: Qs);
               A.append(a/L);    B.append(b/L);
        }
        return (A:A, B:B);
}

/* FUNCTION Integral (…) - calculating the integral of continuous functions
     INPUT:    xo – start point, xn – end point, n –number of intervals, F– function pointer
    OUTPUT:   J – integral value */

func Integral(xo: Double, xn: Double, n: Int, F:(Thdn) -> Thdn) -> Double{
        let  Δ = (xn - xo)/Double(n), h = Δ/2.0, h2 = h*h, h3 = h*h2,
             r1 = h*3.0/4.0, r2 = h2/4.0, r3 = h3/24.0;
        var J = 0.0, S = 0.0, Δf = 0.0, f0 = 0.0, F2:Thdn,
              F1:Thdn = F(Thdn(re: xo, im1:1, im2:0));
        for x in stride(from: xo + Δ, through: xn, by: Δ){
            f0  = F(Thdn(re:x - h, im1:0, im2:0)).re;
            F2 = F(Thdn(re:x,      im1:1, im2:0));
            Δf = F1.re + F2.re - 2.0*f0;
            S = f0*Δ + Δf*r1 + (F1.im1 - F2.im1)*r2 + (F1.im2 + F2.im2)*r3;
            J += S; F1 = F2;
        }
        return J;
}

// TESTED FUNCTIONS:

//-- f(x) = x*x
func test1(X:Thdn)->Thdn{return X*X}

//-- f(x) = (x >= 0)?1: -1
```

```
func test2(X:Thdn)->Thdn{return X.re >= 0.0? Thdn(re: 1.0): -Thdn(re: -1.0))}

//-- f(x) = (x >= 0)?1 : 0
func test3(X:Thdn)->Thdn{ return X.re >= 0.0? Thdn(re: 1.0): Thdn(re: 0.0)}

//-- f(x) = (x >= 0)?x: -x
func test4(X:Thdn)->Thdn{ return X.re >= 0.0? X: -X }

//-- f(x) = (x >= 0)?sin(x): -sin(x)
func test5(X:Thdn)->Thdn{ return X.re >= 0.0? Thdn.sin(X: X): -Thdn.sin(X: X)}

// CALCULATING FOURIER COEFFICIENTS:

    let AB1 = Fourier_AB(N:5, K:60, L: Double.pi, F: test1);
    let AB2 = Fourier_ABd(N:5, K:60, L: Double.pi, F: test2, Xd:0.0);
    let AB3 = Fourier_ABd(N:5, K:60, L: Double.pi, F: test3, Xd:0.0);
    let AB4 = Fourier_ABd(N:5, K:60, L: Double.pi, F: test4, Xd:0.0);
    let AB5 = Fourier_ABd(N:5, K:60, L: Double.pi, F: test5, Xd:0.0);
```

ЛИТЕРАТУРА

1. *Жук В.В., Натансон Г.И.* Тригонометрические ряды Фурье и элементы теории аппроксимации. – Л.: Изд-во Ленингр. ун-та, 1983. –188 с.
2. *Тихонов А. Н., В. Я. Арсенин В. Я.* Методы решения некорректных задач. – М.: Наука,1979.– 285с.
3. *Олифер В. И.* Численное интегрирование с использованием гипер-дуальных чисел. – URL: http://viosolutions.amerihomesrealty.com/pdf/Численное_интегрирование_с_использованием_гипер-дуальных_чисел.pdf
4. *Олифер В. И.* Усеченные гипер-дуальные числа в автоматическом дифференцировании. – URL: http://viosolutions.amerihomesrealty.com/pdf/Усеченные_гипер_дуальные_числа_в_автоматическом_дифференцировании.pdf
5. The on-line encyclopedia of integer sequences. – https://oeis.org/A002378
6. *Naumann U.* The art of differentiating computer programs. Society for industrial and applied mathematics, Philadelphia, USA, 2012.
7. *Corliss G., Faure C., Griewank A., Hascoet L., Naumann U.* Automatic Differentiation Bibliography // Automatic Differentiation of Algorithms: From Simulation Optimization. Springer, 2002. p. 383 – 425.

Абстракт

В данной публикации рассматривается метод численного определения коэффициентов Фурье на основе автоматического дифференцирования с использованием специальных дуальных чисел (усеченных гипер-дуальных чисел). Представлена компьютерная реализация этого метода для языка SWIFT операционной системы macOS. Проведены численные эксперименты.

Ключевые слова*: коэффициенты Фурье, численное интегрирование, усеченные гипер-дуальные числа, автоматическое дифференцирование, Fourier coefficients, numerical integration, truncated hyper-dual numbers, automatic differentiation*

20 августа 2021 г.