К ВЫЧИСЛЕНИЮ ПРОИЗВОДНЫХ ВЫСОКОГО ПОРЯДКА НА ОСНОВЕ ГИПРЕ-ДУАЛЬНЫХ ЧИСЕЛ

Олифер В.И.

Эта короткая публикация описывает реализацию схемы вычисления значений высших производных (от первого до восьмого порядка) аналитической функции $y = f(x)$. Названная схема основана на методе автоматического дифференцирования (AD) [1] с использованием гипер-дуальных чисел четвертого порядка [2].

Суть AD заключается в создании нового типа данных и переопределении операций и базовых функций над ними. В нашем случае, в качестве такого нового типа данных будут исползованы гипер-дуальные числа 4-го порядка, которые согласно [2], имеют представление в виде $Z = z + z_1\boldsymbol{\varepsilon} + z_2\boldsymbol{\omega} + z_3\boldsymbol{\gamma} + z_4\boldsymbol{\eta}$, $\forall\, z,\ z_1,\ z_2,\ z_3,\ z_4 \in \mathbb{R}$. Параметр z называется главной (*Re* – действительной) частью гипер-дуального числа, а $z_1,\ z_2,\ z_3,\ z_4$ – его мнимыми (*Im1*, *Im2*, *Im3*, *Im4* – инфинитезимальными) частями. Абстрактные элементы $\boldsymbol{\varepsilon}$, $\boldsymbol{\omega}$, $\boldsymbol{\gamma}$, $\boldsymbol{\eta}$ образуют базис мнимых частей гипер-дуального числа, отвечающий следующими правилам: $\boldsymbol{\varepsilon}^2 = 2\boldsymbol{\omega}$, $\boldsymbol{\varepsilon\omega} = 3\boldsymbol{\gamma}$, $\boldsymbol{\varepsilon\gamma} = 4\boldsymbol{\eta}$, $\boldsymbol{\omega}^2 = 6\boldsymbol{\eta}$, $\boldsymbol{\varepsilon\eta} = \boldsymbol{\omega\gamma} = \boldsymbol{\omega\eta} = \boldsymbol{\gamma\eta} = \boldsymbol{\gamma}^2 = \boldsymbol{\eta}^2 = 0$. Для краткости будем называть такие числа просто `sdn4` числами.

Алгебраические операции над `sdn4` числами представляются формулами:

$$
\begin{aligned}
&X = x + x_1\boldsymbol{\varepsilon} + x_2\boldsymbol{\omega} + x_3\boldsymbol{\gamma} + x_4\boldsymbol{\eta}, \qquad Y = y + y_1\boldsymbol{\varepsilon} + y_2\boldsymbol{\omega} + y_3\boldsymbol{\gamma} + y_4\boldsymbol{\eta},\\
&X + Y = x + y + (x_1 + y_1)\boldsymbol{\varepsilon} + (x_2 + y_2)\boldsymbol{\omega} + (x_3 + y_3)\boldsymbol{\gamma} + (x_4 + y_4)\boldsymbol{\eta},\\
&X\cdot Y = x\cdot y + (x\cdot y_1 + y\cdot x_1)\boldsymbol{\varepsilon} + (x\cdot y_2 + 2x_1\cdot y_1 + y\cdot x_2)\boldsymbol{\omega} + (x\cdot y_3 + y\cdot x_3 + 3(x_1\cdot y_2 +\\
&\qquad + y_1\cdot x_2))\boldsymbol{\gamma} + (xy_4 + x_4 y + 6x_2 y_2 + 4(x_1 y_3 + x_3 y_1))\boldsymbol{\eta},\\
&X/Y = X\cdot Y^{-1},\\
&Y^{-1} = z + z_1\boldsymbol{\varepsilon} + z_2\boldsymbol{\omega} + z_3\boldsymbol{\gamma} + z_4\boldsymbol{\eta},\\
&\text{где: } z = y^{-1},\ z_1 = -z^2 y_1,\ z_2 = -z(zy_2 + 2z_1 y_1),\ z_3 = -z(zy_3 + 3(z_1 y_2 +\\
&\qquad + z_2 y_1)),\ z_4 = -z(zy_4 + 6z_2 y_2 + 4(z_1 y_3 + z_3 y_1))
\end{aligned}
\tag{1}
$$

Для определения функции $F(X)$, разложим её в ряд Тейлора:

$$
F(x + \Delta) = f(x) + \Delta\cdot f'(x) + \Delta^2\cdot f''(x)/2 + \Delta^3\cdot f'''(x)\ /\ 6 + \Delta^4\cdot f^{IV}(x)\ /\ 24,
$$

$$
\begin{aligned}
&\text{где: } \Delta = x_1\boldsymbol{\varepsilon} + x_2\boldsymbol{\omega} + x_3\boldsymbol{\gamma} + x_4\boldsymbol{\eta},\quad \Delta^2 = 2x_1^2\boldsymbol{\omega} + 6x_1x_2\boldsymbol{\gamma} + (8x_1x_3 + 6x_2^2)\boldsymbol{\eta},\\
&\qquad \Delta^3 = 6x_1^3\boldsymbol{\gamma} + 36x_1^2x_2\boldsymbol{\eta},\quad \Delta^4 = 24x_1^4\boldsymbol{\eta}
\end{aligned}
\tag{2}
$$

Постановку Δ, Δ^2, Δ^3 и Δ^4 в (3) даёт:

$$\begin{aligned} F(X) &= f(x) + x_1 f'(x)\boldsymbol{\varepsilon} + [x_2 f'(x) + x_1^2 f''(x)]\boldsymbol{\omega} + \\ &+[x_3 f'(x) + 3x_1x_2 f''(x) + x_1^3 f'''(x)]\boldsymbol{\gamma} + \\ &+[x_4 f'(x) + (4x_1x_3 + 3x_2^2) f''(x) + 6x_1^2x_2 f'''(x) + x_1^4 f^{IV}(x)]\boldsymbol{\eta} \end{aligned} \qquad (3)$$

При $x_1 = 1$ и $x_2 = x_3 = x_4 = 0$ соотношение (3) принимает форму:

$$F(x + \boldsymbol{\varepsilon} + 0\boldsymbol{\omega} + 0\boldsymbol{\gamma} + 0\boldsymbol{\eta}) = f(x) + f'\boldsymbol{\varepsilon} + f''(x)\boldsymbol{\omega} + f'''(x)\boldsymbol{\gamma} + f^{IV}(x)\boldsymbol{\eta},$$

$$\forall\, f(x) = F(x + 0\boldsymbol{\varepsilon} + 0\boldsymbol{\omega} + 0\boldsymbol{\gamma} + 0\boldsymbol{\eta})$$

Как видно, при автоматическом дифференцировании на базе `sdn4` чисел можно вычислять (с машинной точностью) значения как самой функции, так и её перевых четырех производных. Если же в качестве целевой функции взять 4-ю производную исходной функции и переопределить базовые функции, то естественно появляется возможность вычисления и производных с 5-го до 8-го порядков исходной функции. В этом случае схема, реализующая такой подход, имеет вид:

$$\begin{aligned} &\{\bar{x}, f(x)\} \xrightarrow{1} \{\bar{X}, F(X)\} \xrightarrow{2} \{f, f', f'', f''', f^{IV}\}_{X=\bar{X}}\,, \\ &\{\bar{x},\ f^{IV}(x)\} \xrightarrow{3} \{\bar{X}, F^{IV}(X)\} \xrightarrow{4} \{f^{IV}, f^{V}, f^{VI}, f^{VII}, f^{VIII}\}_{X=\bar{X}}, \end{aligned} \qquad (4)$$

где: $\bar{x}$ – заданная точка;

$\xrightarrow{1}$ – представление $\bar{x}$ и $f(x)$ в терминах `sdn4`;

$\xrightarrow{2}$ – вычисление $\{, f', f'', f''', f^{IV}\}_{X=\bar{X}}$;

$\xrightarrow{3}$ – представление $\bar{x}$ и $f^{IV}(x)$ в терминах `sdn4x`;

$\xrightarrow{4}$ – вычисление $\{f^{IV}, f^{V}, f^{VI}, f^{VII}, f^{VIII}\}_{X=\bar{X}}$;

При этом `sdn4x` наследует `sdn4`, но переопределяет только базовые функции.

Для поддержки `sdn4` и `sdn4x` на языке C# были разработаны статические библиотеки `Sdn4.dll` и `Sdn4Ex.dll` соответственно, которые включают переопределение операций +, –, ×, ÷ и основных базовых функций `pow(…)`, `powX(…)`, `exp(…)`, `ln(…)`, `log(…)`, `sqrt(…)`, `sin(…)`, `cos(…)`, `sh(…)`, `ch(…)`, возвращающих `Sdn4`. Кроме того, библиотека `Sdn4Ex.dll`, имеет метод `Derivatives(x, f, fx)`, возвращающая кортеж (tuple) `(x, f, f1, f2, ..., f8),` состоящий из значений $\bar{x}$, f и её восьми производных. Входными параметрами метода `Derivatives(...)` служат $\bar{x}$ и ссылки на f и f^{IV}.

Таким образом, для практической реализации схемы (4) необходимо описать исходную функцию как в терминах `sdn4`, так и в терминах `sdn4x`.

С целью сокращения письма программного кода введены псевдонимы (alias) `T` и `S` для чисел `Sdn4` и `Sdn4x` соответственно. В этом случае исходная функция описывается в терминах `T`, а функция f^{IV} получается из описания f путем замены префиксов `T.` на префиксы `S.`(см. Приложение 1).

Численный эксперимент проводился на компьютере с операционной системой Microsoft Windows 10 Pro, CPU 3.40 GHz, RAM 16 MB в среде MS VS 2021, при этом использовались тестируемые функции представленные в следующей таблице.

	Тестируемые функции	$\bar{x}$
1	$f_1(x) = (sin(x) - x/2)^2$	2.0
2	$f_2(x) = x^6 - 6x^5 + 50x^3 - 45x^2 - 108x + 108$	1.0
3	$f_3(x) = (x\cdot exp(x^2) - sin^2(x) + 3\,cos(x) + 5)^2$	0.0
4	$f_4(x) = ln^2(x) \cdot (exp(x-3) - 1)\cdot \sin(\pi x/3)$	4.0
5	$f_6(x) = x^8$	1.0
6	$f_7(x) = sin(cos(tg(sh(ch(th(x))))))$	0.0

Таблица 1. Список тестируемых функций

Релультаты расчета тестируемых функций представлены в таблице 2.

	$f_1(2)$	$f_2(1)$	$f_3(0)$	$f_4(4)$	$f_5(1)$	$f_6(0)$
f	0.008226956780442	0	64	-0.714950461877138	1	-0.66963611460373
f'	0.166193750960674	-72	16	-2.594749360823758	8	0
f''	1.843601285018424	120	-78	-6.453673834514070	56	-5.243581082878374
f'''	4.922808588614473	60	66	-7.306004014294588	336	0
f^{IV}	1.745966767068962	-360	374	10.70381188612373	1680	185.1268486079934
f^{V}	0.166193750960674	-108	16	0.379478176540554	6720	0
f^{VI}	1.843601285018424	720	-78	-0.608041882779440	20160	-5.243581082878374
f^{VII}	4.922808588614473	0	66	0.443710322007873	40320	0
f^{VIII}	1.745966767068962	0	374	3.869036919767793	40320	185.1268486079934

Таблица 2. Результаты расчета тестируемых функций

Общее время вычисления всего пакета тестируемых функций не превысило 0.05 ms.

Приложение 1.

Исходный код на C#, использующий статические библиотеки Sdn4Lib.dll и Sdn4Exdll.

```
using Sdn4Lib;
using T = Sdn4Lib.sdn4;
using S = Sdn4Ex.sdn4Ex;

//******ПРИМЕРЫ  ТЕСТИРУЕМЫХ ФУНКЦИИ******
    //---------------------------5)x^8
    public static sdn4 x8( sdn4 X){return T.pow(X, 8);}
    public static sdn4 x8x(sdn4 X){return S.pow(X, 8);}

    //---------------------------6)sin(cos(tg(sh(ch(th(x))))))
    public static sdn4 sinh4(sdn4 X)
    {   var th = T.sh(X) / T.ch(X);
        var ch = T.ch(th);
        var sh = T.sh(ch);
        var tg = T.sin(sh) / T.cos(sh);
        return T.sin(T.cos(tg));
    }
    public static sdn4 sinh4x(sdn4 X)
    {   var th = S.sh(X) / S.ch(X);
        var ch = S.ch(th);
        var sh = S.sh(ch);
        var tg = S.sin(sh) / S.cos(sh);
        return S.sin(S.cos(tg));
    }


//******ВЫЗОВ МЕТОДА DERIVATIVES(...)******

      var R = sdn4Ex.Derivatives(x: 1.0, f: x8,    fx: x8x);
          R = sdn4Ex.Derivatives(x: 0.0, f: sinh4, fx: sinh4x);

      //where R = tuple10(x, f, f1, f2, f3, f4, f5, f6, f7, f8)
```

ЛИТЕРАТУРА

1. *Naumann U.* The Art of Differentiating Computer Programs. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2012.

2. *Олифер В. И.* Автоматическое дифференцирование на основе супер-дуальных чисел. – URL:http://viosolutions.amerihomesrealty.com/pdf/Автоматическое_дифференцирование _на__основе_супер-дуальных_чисел.pdf, 2019.

Абстракт

В данной публикации рассматривается схема вычисления значений производных высших порядков на основе автоматического дифференцирования с использованием гипер-дуальных чисел 4-го порядка. Представлена компьютерная реализация этого подхода для языка C# операционной системы Windows 10 Pro. *Проведены численные эксперименты*.

This publication discusses a scheme for calculating the values of higher-order derivatives based on automatic differentiation using hyper-dual numbers of the 4th order. A computer implementation of this approach for the C# language of the Windows 10 Pro operating system is presented. Numerical experiments were conducted.

Ключевые слова: *производные высших порядков, автоматическое дифференцирование, гипер-дуальные числа, higher order derivatives, automatic differentiation,* hyper*-dual numbers.*

---

20 июня 2025 г