АВТОМАТИЧЕСКОЕ ДИФФЕРЕНЦИРОВАНИЕ НА ОСНОВЕ СУПЕР-ДУАЛЬНЫХ ЧИСЕЛ

Автоматическое дифференцирование

Точность вычислений является основным критерием качества результатов вычислений. Любой вычислительный процесс содержит погрешность метода или усечения и погрешность конечного округления. Так, например, вычисление базовой функции e^x осуществляется путем ее разложения в ряд Тейлора с остаточным членом $\mathcal{O}(k)$

$$e^x = \sum_{n=0}^{\infty} x^n/n! = \sum_{n=0}^{k} x^n/n! + \sum_{n=k+1}^{\infty} x^n/n! = \sum_{n=0}^{k} x^n/n! + \mathcal{O}(k),$$

где остаточный член $\mathcal{O}(k)$ и есть погрешность метода. Величина k определяется из условия: $\mathcal{O}(k)$ меньше точности представления данных.

Погрешность округления связана с точностью представления данных (количеством значащих цифр после запятой). Для данных с плавающей точкой компьютер выделяет определенный размет дробной части числа, не вмещающиеся цифры отбрасываются без округления. Например, пусть размер дробной части равен 15-ти цифрам и $x = 0.000000000000001$, а $y = 0.1$, тогда $x{\cdot}y = 0.000000000000000$, что приводит к потери результата.

Традиционные способы вычисления производной $f'(x)$ на практике сопряжены с трудностями и имеют существенные недостатки. Реализация вычислений $f'(x)$ в виде отдельной программной процедуры является избыточным решением и нуждается в предварительном анализа $f(x)$. Использование метода конечных разностей требует компромисса между погрешностями усечения и округления, т. е. нахождения оптимальной величины приращения аргумента x. Более того, задача численного дифференцирования является некорректной и ее нужно решать методами регуляции [1]. Еще большие проблемы возникают при вычислении производных высших порядков.

Обойти указанные трудности позволяет метод автоматическое дифференцирование (AD), связанный с вычислением точных значений производных функции, представленной компьютерным кодом [2].

Автоматическое дифференцирование позволяет избежать дублирование функциональности программного кода (изменение кода функции не требует изменения кода ее производных) и осуществлять точное дифференцирование вплоть до погрешности округления.

Для компьютерной реализации AD необходимо создать новый тип данных, перезагрузить базовые функции и операции над ними. Если новый тип данных строится на основе обычных дуальных чисел [3], то за одно обращение к перезагруженной базовой функции точно (вплоть до погрешности округления) вычисляются значение самой функции и ее производной. При использовании гипер-дуального числа [4, 5] становится возможным одновременно получить точные значения функции, ее первой и второй производных и т. д.

Дуальные числа и их расширение

Классические дуальные числа это – гиперкомплексные параболические числа вида $X = x + x_1\boldsymbol{\varepsilon}_1$, где x и x_1 – вещественные числа, а $\boldsymbol{\varepsilon}_1$ – абстрактный элемент, квадрат которого равен нулю [3]. Число x называется главной (*Re* - действительной) частью дуального числа, а x_1 – мнимой (*Im* - инфинитезимальной) его частью. Абстрактный элемент $\boldsymbol{\varepsilon}_1$ является *основным базисом* мнимой части дуального числа. Расширение дуального числа достигается за счет увеличения его основного базиса мнимой части. Так, например, гипер-дуальное число [4, 5] образуется путем введения дополнительного основного базиса мнимой части $\boldsymbol{\varepsilon}_2$. Полная структура базиса мнимой части состоит из *основной* и *комбинированной* частей. Комбинированная часть образуется из неповторяющихся комбинаций (combinations) основных базисных элементов. Мнимые символы с одним индексом образуют основной базис, например, для гипер-дуальных чисел – ($\boldsymbol{\varepsilon}_1$, $\boldsymbol{\varepsilon}_2$), а с двумя и более – комбинированный ($\boldsymbol{\varepsilon}_{12} = \boldsymbol{\varepsilon}_1 \cdot \boldsymbol{\varepsilon}_2$). Общее количество неповторяющихся комбинаций базисных элементов мнимой части определится по формуле

$$N(n) = n + \sum_{m=2}^{n} \frac{n!}{(n-m)!\,m!}, \qquad (1)$$

n =1, 2, 3, 4, … размер основного базиса. При *N(1)* = 1 -дуальное число, *N(2)* = 3 -гипер-дуальное число, *N(3)* = 7, *N(4)* = 15, … .

Будем называть размер основного базиса *классом* (*K*) дуального числа[*)]. Как показано далее, класс дуального числа указывает на максимальный порядок производной, входящей в определение функции соответствующего дуального аргумента. Правило умножение базисных элементов определяется следующим образом: *если у двух сомножителей совпадает хотя бы один индекс, то результат равен нулю*.

Приведем базисы для дуальных чисел некоторых классов:

Класс	Базис	Кол-во
1	$\boldsymbol{\varepsilon}_1$	1
2	$\boldsymbol{\varepsilon}_1, \boldsymbol{\varepsilon}_2, \boldsymbol{\varepsilon}_{12}$	3
3	$\boldsymbol{\varepsilon}_1, \boldsymbol{\varepsilon}_2, \boldsymbol{\varepsilon}_3, \boldsymbol{\varepsilon}_{12}, \boldsymbol{\varepsilon}_{13}, \boldsymbol{\varepsilon}_{23}, \boldsymbol{\varepsilon}_{123}$	7
4	$\boldsymbol{\varepsilon}_1, \boldsymbol{\varepsilon}_2, \boldsymbol{\varepsilon}_3, \boldsymbol{\varepsilon}_4, \boldsymbol{\varepsilon}_{12}, \boldsymbol{\varepsilon}_{13}, \boldsymbol{\varepsilon}_{14}, \boldsymbol{\varepsilon}_{23}, \boldsymbol{\varepsilon}_{24}, \boldsymbol{\varepsilon}_{34}, \boldsymbol{\varepsilon}_{123}, \boldsymbol{\varepsilon}_{124}, \boldsymbol{\varepsilon}_{134}, \boldsymbol{\varepsilon}_{234}, \boldsymbol{\varepsilon}_{1234}$	15

Таблица 1. Базисы некоторых классов дуальных чисел.

Числа класса 1 – это классические дуальные числа [3], класса 2 – гипер-дуальные [4, 5], класс 3 – числа типа чисел Паули [6], но с другим правилом умножения элементов базиса:

×	$\boldsymbol{\varepsilon}_1$	$\boldsymbol{\varepsilon}_2$	$\boldsymbol{\varepsilon}_3$	$\boldsymbol{\varepsilon}_{12}$	$\boldsymbol{\varepsilon}_{13}$	$\boldsymbol{\varepsilon}_{23}$	$\boldsymbol{\varepsilon}_{123}$
$\boldsymbol{\varepsilon}_1$	0	$\boldsymbol{\varepsilon}_{12}$	$\boldsymbol{\varepsilon}_{13}$	0	0	$\boldsymbol{\varepsilon}_{123}$	0

[*)] В [5] используется термин гипер-дуальное число *k*-й размерности.

$\boldsymbol{\varepsilon}_2$	$\boldsymbol{\varepsilon}_{12}$	0	$\boldsymbol{\varepsilon}_{23}$	0	$\boldsymbol{\varepsilon}_{123}$	0	0
$\boldsymbol{\varepsilon}_3$	$\boldsymbol{\varepsilon}_{13}$	$\boldsymbol{\varepsilon}_{23}$	0	$\boldsymbol{\varepsilon}_{123}$	0	0	0
$\boldsymbol{\varepsilon}_{12}$	0	0	$\boldsymbol{\varepsilon}_{123}$	0	0	0	0
$\boldsymbol{\varepsilon}_{13}$	0	$\boldsymbol{\varepsilon}_{123}$	0	0	0	0	0
$\boldsymbol{\varepsilon}_{23}$	$\boldsymbol{\varepsilon}_{123}$	0	0	0	0	0	0
$\boldsymbol{\varepsilon}_{123}$	0	0	0	0	0	0	0

Таблица 2. Умножение элементов базиса дуальных чисел класса 3.

Общее представление дуального числа k-го класса имеет вид:

$$X_{[k]} = x + \sum_{i=1}^{k} x_i \boldsymbol{\varepsilon}_i + \sum_{j=k+1}^{N} x_j \boldsymbol{\varepsilon}_{j[k]}, \tag{2}$$

где $\boldsymbol{\varepsilon}_{j[k]}$ – элементы комбинированной мнимой чисти дуального числа k-го класса.
Приведем развернутые выражения формулы (2) для классов $k = 1, 2, 3, 4$:

$$\begin{aligned}
X_{[1]} &= x + x_1\boldsymbol{\varepsilon}_1, \\
X_{[2]} &= x + x_1\boldsymbol{\varepsilon}_1 + x_2\boldsymbol{\varepsilon}_2 + x_3\boldsymbol{\varepsilon}_{12}, \\
X_{[3]} &= x + x_1\boldsymbol{\varepsilon}_1 + x_2\boldsymbol{\varepsilon}_2 + x_3\boldsymbol{\varepsilon}_3 + x_4\boldsymbol{\varepsilon}_{12} + x_5\boldsymbol{\varepsilon}_{13} + x_6\boldsymbol{\varepsilon}_{23} + x_7\boldsymbol{\varepsilon}_{123}, \\
X_{[4]} &= x + x_1\boldsymbol{\varepsilon}_1 + x_2\boldsymbol{\varepsilon}_2 + x_3\boldsymbol{\varepsilon}_3 + x_4\boldsymbol{\varepsilon}_4 + \\
&\quad + x_5\boldsymbol{\varepsilon}_{12} + x_6\boldsymbol{\varepsilon}_{13} + x_7\boldsymbol{\varepsilon}_{14} + x_8\boldsymbol{\varepsilon}_{23} + x_9\boldsymbol{\varepsilon}_{24} + x_{10}\boldsymbol{\varepsilon}_{34} + x_{11}\boldsymbol{\varepsilon}_{123} + \\
&\quad + x_{12}\boldsymbol{\varepsilon}_{124} + x_{13}\boldsymbol{\varepsilon}_{134} + x_{14}\boldsymbol{\varepsilon}_{234} + x_{15}\boldsymbol{\varepsilon}_{1234}
\end{aligned} \tag{3}$$

Алгебра дуальных чисел соответствует алгебре многочленов с учетом правил умножения базисных элементов мнимой части, например, $X_{[1]}{\cdot}Y_{[1]} = (x + x_1\boldsymbol{\varepsilon}_1){\cdot}(y + y_1\boldsymbol{\varepsilon}_1) = x{\cdot}y + (x{\cdot}y_1 + y{\cdot}x_1)\boldsymbol{\varepsilon}_1$.
Рассмотрим теперь дуальную функции k-го класса $F(X_{[k]})$. Ее разложение в ряд Тейлора имеет вид:

$$F\big(X_{[k]}\big) = F\big(x + e_{[k]}\big) = F(x) + \sum_{i=1}^{\infty} F(x)^{(i)} \cdot e_{[k]}^{i}/i! \tag{4}$$

где: $F(x)^{(i)} = d^i F(x)/dx^i$, $e_{[k]} = Im(X_{[k]})$ – мнимая часть $X_{[k]}$.
Количество членов в (4) зависит от последовательности значений $e_{[k]}^{i}$. Для наглядности рассмотрим $e_{[k]}^{i}$ для второго класса ($k = 2$):

$$\begin{aligned}
&e_{[2]} = x_1\boldsymbol{\varepsilon}_1 + x_2\boldsymbol{\varepsilon}_2 + x_3\boldsymbol{\varepsilon}_{12}, \\
&e_{[2]}^{2} = 2x_1{\cdot}x_2\boldsymbol{\varepsilon}_{12}, \\
&e_{[2]}^{3},\ e_{[2]}^{4},\ e_{[2]}^{5}, \ldots = 0
\end{aligned} \tag{5}$$

Тогда

$$F\left(X_{[2]}\right) = f(x) + f(x)'(x_1\boldsymbol{\varepsilon}_1 + x_2\boldsymbol{\varepsilon}_2) + (x_3 f(x)' + f(x)'' x_1 x_2)\boldsymbol{\varepsilon}_{12}, \quad (6)$$

где $f(x)$ равна $F\left(X_{[2]}\right)$ при $X_{[2]}$ с нулевой мнимой частью.
Если положить $x_1 = x_2 = 1,\ x_3 = 0$, то соотношение (6) упрощается и принимает вид:

$$F\left(X_{[2]}\right) = f(x) + f'(x)\boldsymbol{\varepsilon}_1 + f'(x)\boldsymbol{\varepsilon}_2 + f''(x)\boldsymbol{\varepsilon}_{12} \quad (7)$$

Для итерационных процессов, использующих значения функции, ее первой и второй производных, выражение (7) является избыточным т. к. первая и вторая мнимые части содержат одну и туже величину – значение первой производной. Чтобы избавиться от указанной избыточности введем новый базис $\{\boldsymbol{\varepsilon}, \boldsymbol{\omega}\}$ с правилом умножения $\boldsymbol{\varepsilon}^2 = 2\boldsymbol{\omega}$ и $\boldsymbol{\varepsilon}\cdot\boldsymbol{\omega} = \boldsymbol{\omega}\cdot\boldsymbol{\varepsilon} = \boldsymbol{\omega}\cdot\boldsymbol{\omega} = 0$[*)]. Тогда

$$X_{[2]} = x + x_1\boldsymbol{\varepsilon} + x_2\boldsymbol{\omega} \quad (8)$$

и разложение $F\left(X_{[2]}\right)$ в ряд Тейлора дает

$$F\left(X_{[2]}\right) = f(x) + x_1 \cdot f'(x)\boldsymbol{\varepsilon} + (x_2 \cdot f'(x) + x_1^2 \cdot f''(x))\boldsymbol{\omega} \quad (9)$$

При $x_1 = 1$ и $x_2 = 0$ выражение (9) принимает вид:

$$F\left(X_{[2]}\right) = F(x) = f(x) + f'(x)\boldsymbol{\varepsilon} + f''(x)\boldsymbol{\omega} \quad (10)$$

Полученное соотношение (10) свободно от избыточности, присутствующей в (7).
Выражения (8) и (10) являются *компактным* представлением второго класса дуальных чисел и их функций соответственно.
Приведенную выше процедуру нетрудно применить и для других классов дуальных чисел (см. таблицу 3).

Класс	X	$F(x)$	Умножение базиса [*)]
1	$x + x_1\boldsymbol{\varepsilon}$	$f(x) + f'(x)\boldsymbol{\varepsilon}$	$\boldsymbol{\varepsilon}^2 = 0$
2	$x + x_1\boldsymbol{\varepsilon} + x_2\boldsymbol{\omega}$	$f(x) + f'(x)\boldsymbol{\varepsilon} + f''(x)\boldsymbol{\omega}$	$\boldsymbol{\varepsilon}^2 = 2\boldsymbol{\omega},\ \boldsymbol{\omega}^2 = \boldsymbol{\varepsilon}\boldsymbol{\omega} = 0$
3	$x + x_1\boldsymbol{\varepsilon} + x_2\boldsymbol{\omega} + x_3\boldsymbol{\gamma}$	$f(x) + f'(x)\boldsymbol{\varepsilon} + f''(x)\boldsymbol{\omega} + f'''(x)\boldsymbol{\gamma}$	$\boldsymbol{\varepsilon}^2 = 2\boldsymbol{\omega},\ \boldsymbol{\varepsilon}\boldsymbol{\omega} = 3\boldsymbol{\gamma},$ $\boldsymbol{\varepsilon}\boldsymbol{\gamma} = \boldsymbol{\omega}\boldsymbol{\gamma} = \boldsymbol{\gamma}^2 = \boldsymbol{\omega}^2 = 0$
4	$x + x_1\boldsymbol{\varepsilon} + x_2\boldsymbol{\omega} + x_3\boldsymbol{\gamma} + x_4\boldsymbol{\eta}$	$f(x) + f'(x)\boldsymbol{\varepsilon} + f''(x)\boldsymbol{\omega} + f'''(x)\boldsymbol{\gamma} + f''''(x)\boldsymbol{\eta}$	$\boldsymbol{\varepsilon}^2 = 2\boldsymbol{\omega},\ \boldsymbol{\varepsilon}\boldsymbol{\omega} = 3\boldsymbol{\gamma},$ $\boldsymbol{\varepsilon}\boldsymbol{\gamma} = 4\boldsymbol{\eta},\ \boldsymbol{\omega}^2 = 6\boldsymbol{\eta},$ $\boldsymbol{\varepsilon}\boldsymbol{\eta} = \boldsymbol{\omega}\boldsymbol{\gamma} = \boldsymbol{\omega}\boldsymbol{\eta} =$ $= \boldsymbol{\gamma}\boldsymbol{\eta} = \boldsymbol{\gamma}^2 = \boldsymbol{\eta}^2 = 0$

Таблица 3. Компактное представление дуальных чисел классов 1, 2, 3 и 4 при $x_1 = 1$ и $x_2 = x_3 = x_4 = 0$.

[*)] Соотношение $\boldsymbol{\varepsilon}^2 = 2\boldsymbol{\omega}$ получено из $\boldsymbol{\varepsilon}^2 = (\boldsymbol{\varepsilon}_1 + \boldsymbol{\varepsilon}_2)^2 = 2\boldsymbol{\varepsilon}_1\boldsymbol{\varepsilon}_2 = 2\boldsymbol{\varepsilon}_{12} = 2\boldsymbol{\omega}$
[*)] Таблицы умножения базисов см. Приложение 1.

Для компьютерной реализации итерационных процессов на основе градиентных методов с использованием дуальных функций, очевидно, что их компактное представление предпочтительно (точность, экономия памяти и сокращение количества операций).
Всякая компьютерная система содержит библиотеки для вычисления базовых элементарных функций. В случае использования дуальных функций необходимо расширение этих библиотек за счет включения вычисления производных различных порядков, также введения новых алгебраических операций.
Воспользовавшись [7], приведем компоненты некоторых элементарных (базовых) функций для первого, второго и третьего классов дуальных чисел:

$f(X)$	$f(x)$	$f'(x)$	$f''(x)$	$f'''(x)$
X^n	x^n	$n{\cdot}x^{n-1}$	$n{\cdot}(n-1){\cdot}x^{n-2}$	$n{\cdot}(n-1){\cdot}(n-2){\cdot}x^{n-3}$
e^X	e^x	e^x	e^x	e^x
a^X	a^x	$a^x{\cdot}\ln(a)$	$a^x{\cdot}\ln^2(a)$	$a^x{\cdot}\ln^3(a)$
$\ln(X)$	$\ln(x)$	x^{-1}	$-x^{-2}$	$2{\cdot}x^{-3}$
$\log_a(X)$	$\log_a(x)$	$x^{-1}{\cdot}\ln(a)^{-1}$	$-x^{-2}{\cdot}\ln(a)^{-1}$	$2{\cdot}x^{-3}{\cdot}\ln(a)^{-1}$
$\sin(X)$	$\sin(x)$	$\cos(x)$	$-\sin(x)$	$-\cos(x)$
$\cos(X)$	$\cos(x)$	$-\sin(x)$	$-\cos(x)$	$\sin(x)$
$\mathrm{tg}(X)$	$\mathrm{tg}(x)$	$\cos^{-2}(x)$	$2{\cdot}\mathrm{tg}(x){\cdot}\cos^{-2}(x)$	$2{\cdot}\cos^{-2}(x){\cdot}(3{\cdot}\mathrm{tg}^2(x)+1)$
$\mathrm{ctg}(X)$	$\mathrm{ctg}(x)$	$-\sin^{-2}(x)$	$-2{\cdot}\mathrm{ctg}(x){\cdot}\sin^{-2}(x)$	$-2{\cdot}\sin^{-2}(x){\cdot}(3{\cdot}\mathrm{ctg}^2(x)+1)$
$\arcsin(X)$	$\arcsin(x)$	$(1-x^2)^{-1/2}$	$x{\cdot}(1-x^2)^{-3/2}$	$(2{\cdot}x^2+1){\cdot}(1-x^2)^{-5/2}$
$\arccos(X)$	$\arccos(x)$	$-(1-x^2)^{-1/2}$	$-x{\cdot}(1-x^2)^{-3/2}$	$-(2{\cdot}x^2+1){\cdot}(1-x^2)^{-5/2}$
$\mathrm{arctg}(X)$	$\mathrm{arctg}(x)$	$(1+x^2)^{-1}$	$-2{\cdot}x{\cdot}(1+x^2)^{-2}$	$(6{\cdot}x^2-2){\cdot}(1+x^2)^{-3}$
$\mathrm{arcctg}(X)$	$\mathrm{arcctg}(x)$	$-(1+x^2)^{-1}$	$2{\cdot}x{\cdot}(1+x^2)^{-2}$	$-(6{\cdot}x^2-2){\cdot}(1+x^2)^{-3}$
$\mathrm{sh}(X)$	$\mathrm{sh}(x)$	$\mathrm{ch}(x)$	$\mathrm{sh}(x)$	$\mathrm{ch}(x)$
$\mathrm{ch}(X)$	$\mathrm{ch}(x)$	$\mathrm{sh}(x)$	$\mathrm{ch}(x)$	$\mathrm{sh}(x)$

Таблица 4. Компоненты элементарных функций для первого, второго и третьего классов дуальных чисел

Для практических целей (численной реализации) целесообразно использовать тот класс, который обеспечивает нужный порядок производной, входящей в определение функций соответствующего класса.
Вычисление сложной дуальной функции вида $F = f_1(f_2(\dots f_n(X)\dots), X)$ (где значение f_n используется в качестве аргумента для f_{n-1}) необходимо начать с вычисления $F_n = f_n(X)$, продолжить вычислениями $F_{n-1} = f_{n-1}(F_n, X)$, $F_{n-2} = f_{n-2}(F_{n-1}, X)$, … , $F = f_1(F_2, X)$.

Использование дуальных чисел в градиентных методах

Рассмотрим применение дуальных чисел класса 3 для численного решения нелинейных (алгебраических или трансцендентных) уравнений и задач оптимизации функций одной и двух переменных. Алгебраические операции над дуальными числами 3-го класса определяются следующими формулами:

$$
\begin{aligned}
&X_{[3]} = x + x_1\boldsymbol{\varepsilon} + x_2\boldsymbol{\omega} + x_3\boldsymbol{\gamma}, \qquad Y_{[3]} = y + y_1\boldsymbol{\varepsilon} + y_2\boldsymbol{\omega} + y_3\boldsymbol{\gamma}\\
&X_{[3]} + Y_{[3]} = x + y + (x_1 + y_1)\boldsymbol{\varepsilon} + (x_2 + y_2)\boldsymbol{\omega} + (x_3 + y_3)\boldsymbol{\gamma},\\
&X_{[3]}\cdot Y_{[3]} = x\cdot y + (x\cdot y_1 + y\cdot x_1)\boldsymbol{\varepsilon} + (x\cdot y_2 + 2x_1\cdot y_1 + y\cdot x_2)\boldsymbol{\omega} + (x\cdot y_3 + y\cdot x_3 + 3(x_1\cdot y_2\\
&\qquad + y_1\cdot x_2))\boldsymbol{\gamma},\\
&Y_{[3]}^{-1} = y^{-1} - y^{-2}\cdot y_1\boldsymbol{\varepsilon} + y^{-2}(2\cdot y_1^2\cdot y^{-1} - y_2)\boldsymbol{\omega} + a^{-2}(6y_1\cdot y^{-1}(y_2 - y_1^2\cdot y^{-1}) - y_3)\boldsymbol{\gamma},\\
&X_{[3]}/Y_{[3]} = \mathrm{X}_{[3]}\cdot Y_{[3]}^{-1},
\end{aligned}
\tag{11}
$$

а функция дуального аргумента после ее разложения в ряд Тейлора имеет вид

$$
\begin{aligned}
&F\left(X_{[3]}\right) = f(x) + x_1 f'(x)\boldsymbol{\varepsilon} + [x_2 f'(x) + x_1^2 f''(x)]\boldsymbol{\omega} + [x_3 f'(x) + 3x_1 x_2 f''(x) +\\
&x_1^3 f'''(x)]\boldsymbol{\gamma},\\
&F(x + \boldsymbol{\varepsilon} + 0\boldsymbol{\omega} + 0\boldsymbol{\gamma}) = f(x) + f'(x)\boldsymbol{\varepsilon} + f''(x)\boldsymbol{\omega} + f'''(x)]\boldsymbol{\gamma}
\end{aligned}
\tag{12}
$$

Если дуальную функцию $F(X_{[3]}, Y_{[3]})$ двух дуальных аргументов третьего класса разложить в ряд Тейлора (с учетом правил умножения базиса), то получим:

$$
\begin{aligned}
&F(X_{[3]}, Y_{[3]}) = f +\\
&+ \left[x_1 f'_x + y_1 f'_y\right]\boldsymbol{\varepsilon} +\\
&+ \left[x_2 f'_x + y_2 f'_y + x_1^2 f''_{xx} + y_1^2 f''_{yy} + 2x_1 y_1 f''_{xy}\right]\boldsymbol{\omega} +\\
&+ \left[x_3 f'_x + y_3 f'_y + x_1^3 f'''_{xxx} + y_1^3 f'''_{yyy} + 3\left(x_1 x_2 f''_{xx} + y_1 y_2 f''_{yy} +\right.\right.\\
&\left.\left. + (x_1 y_2 + x_2 y_1) f''_{xy} + x_1^2 y_1 f'''_{xxy} + x_1 y_1^2 f'''_{xyy}\right)\right]\boldsymbol{\gamma}
\end{aligned}
\tag{13}
$$

где f, $f'_{\dots}$, $f''_{\dots}$, $f'''_{\dots}$– функция и ее соответствующие частные производные в точке (x, y). Разные комбинации x_1, x_2, x_3, y_1, y_2 и y_3 позволяют сохранить в (13) те либо иные члены. Наиболее актуальные комбинации (фильтры) x_k и y_k (dX, dY и dXY) приведены в следующей таблице:

	$X_{[3]}, Y_{[3]}$	$F\left(X_{[3]}, Y_{[3]}\right)$
dX	$X_{[3]} = (x, 1, 0, 0), \quad Y_{[3]} = (y, 0, 0, 0)$	$f + f'_x\boldsymbol{\varepsilon} + f''_{xx}\boldsymbol{\omega} + f'''_{xxx}\boldsymbol{\gamma}$
dY	$X_{[3]} = (x, 0, 0, 0), \quad Y_{[3]} = (y, 1, 0, 0)$	$f + f'_y\boldsymbol{\varepsilon} + f''_{yy}\boldsymbol{\omega} + f'''_{yyy}\boldsymbol{\gamma}$
dXY	$X_{[3]} = (x, 1, 0, 0), \quad Y_{[3]} = (y, 1, 0, 0)$	$f + \left[f'_x(x, y) + f'_y\right]\boldsymbol{\varepsilon} + \left[f''_{xx} + 2f''_{xy} + f''_{yy}\right]\boldsymbol{\omega} +$ $+ \left[f'''_{xxx} + f'''_{yyy} + 3(f'''_{xxy} + f'''_{xyy})\right]\boldsymbol{\gamma}$

Таблица 5. Актуальные фильтры dX, dY и dXY для третьего класса

Из таблицы 5 видно, что главные части dX, dY и dXY содержать значение функции в точке (x, y), а мнимые части – частные производные:

- $Im1(dX) = f'_x$, $Im2(dX) = f''_{xx}$, $Im3(dX) = f'''_{xxx}$
- $Im1(dY) = f'_y$, $Im2(dY) = f''_{yy}$, $Im3(dX) = f'''_{yyy}$
- $Im1(dXY) = f'_x + f'_y$, $Im2(dXY) = f''_{xx} + 2f''_{xy} + f''_{yy}$, $Im3(dX) = f'''_{xxx} + f'''_{yyy} + 3(f'''_{xxy} + f'''_{xyy})$

Для определения f''_{xy} и $(f'''_{xxy} + f'''_{xyy})$ можно использовать формулы:

$$f''_{xy} = (Im2(dXY) - Im2(dX) - Im2(dY))/2,$$
$$(f'''_{xxy} + f'''_{xyy}) = (Im3(dXY) - Im3(dX) - Im3(dY))/3$$

В случае, если $F(X_{[3]}, Y_{[3]}) = \Phi(X_{[3]})\cdot\Psi(Y_{[3]})$, то нахождение значений частных производных упрощается:

$f'_x = \varphi'_x\psi = Im1(dX)$, $\quad f'_y = \varphi\psi'_y = Im1(dY)$,
$f''_{xx} = \varphi''_{xx}\psi = Im2(dX)$, $\quad f''_{yy} = \varphi\psi''_{yy} = Im2(dY)$,
$f'''_{xxx} = \varphi'''_{xxx}\psi = Im3(dX)$, $\quad f'''_{yyy} = \varphi\psi'''_{yyy} = Im3(dY)$,
$f''_{xy} = f''_{yx} = \varphi'_x\psi'_y = Im1(dX)\cdot Im1(dY)/Re(dX) = Im1(dX)\cdot Im1(dY)/Re(dY)$,
$f'''_{xxy} = \varphi''_{xx}\psi'_y = Im2(dX)\cdot Im1(dY)/Re(dX)$, $\quad f'''_{xyy} = \varphi'_x\psi''_{yy} = Im1(dX)\cdot Im2(dY)/Re(dY)$

Используя соотношения (11), (12) и (13), а также таблицы 4 и 5, нетрудно реализовать новый тип данных Tdn3 (см. Приложение 2), который применим для многих градиентных методов.

Рассмотрим применение дуальных чисел класса 3 для численного решения нелинейных уравнений вида $f(x) = 0$ на основе итерационной процедуры Ньютона-Чебышева (Newton-Chebyshev) [9]. Каждое новое приближение x_{n+1} вычисляется по формуле $x_{n+1} = x_n - g(x,\ f(x),\ f'(x)\ , f''(x), f'''(x)\ \ldots)|_{x=x_n}$, где x_n решение полученное на n-ом шаге итерационного процесса и

$$g(\ldots) = \frac{f(x)}{f'(x)}\left[1 + \frac{1}{2}\cdot\frac{f(x)\cdot f''(x)}{f'(x)^2} + \frac{f(x)^2}{f'(x)^2}\left[\frac{1}{2}\cdot\left(\frac{f''(x)}{f'(x)}\right)^2 - \frac{1}{6}\cdot\frac{f'''(x)}{f'(x)}\right] + \cdots\right] \qquad (14)$$

При этом на интервале поиска решения функция $f(x)$ должна быть гладкой и $f'(x) \neq 0$.
Если ограничится только первым членом в квадратных скобках выражения (14), то получим итерационную формулу классического метода Ньютона (первого порядка): $x_{n+1} = x_n - f(x)/\,f'(x)$. Количество удерживаемых членов определяет порядок метода, а он – максимальный порядок используемой производной. Наличие производных функции $f(x)$ различных порядков в $g(\ldots)$ определяет скорость сходимости, но требует вычисления значений этих производных, что может оказаться неприемлемым или труднодостижимым.
Нетрудно видеть, что для реализации итерационного метода третьего порядка необходимо вычисление значений $f(x)$, $f'(x)$, $f''(x)$ и $f'''(x)$ в точке x_n. Поэтому в этом случае дуальные числа класса 3 являются самым подходящим кандидатом для процедуры автоматического дифференцирования. При этом надо заменить функцию действительного аргумента $f(x)$ на соответствующую эквивалентную дуальную функцию $F_{[3]}(X)$ третьего класса:

$$F_{[3]}(x + \boldsymbol{\varepsilon} + 0\boldsymbol{\omega} + 0\boldsymbol{\gamma}) = f(x) + f'(x)\boldsymbol{\varepsilon} + f''(x)\boldsymbol{\omega} + f'''(x)\boldsymbol{\gamma} \qquad (15)$$

В этом случае выражение (14) можно представить следующим образом:

$$g(X_n) = \alpha_1\left[1 + \frac{1}{2}\alpha_1\cdot\alpha_2 + \frac{1}{2}\alpha_1^2\cdot\left(\alpha_2^2 - \frac{1}{3}\alpha_3\right)\right], \tag{16}$$

где: $\alpha_1 = Re(X_n)/Im_1\big(F(X_n)\big)$, $\alpha_2 = Im_2\big(F(X_n)\big)/Im_1\big(F(X_n)\big)$,

$\alpha_3 = Im_3\big(F(X_n)\big)/Im_1\big(F(X_n)\big)$.

Тогда следующее приближение определяется по формуле $x_{n+1} = x_n - g(X_n)$ *).

Пусть теперь требуется решить оптимизационную задачу вида $x = argmin(f)$, что соответствует решению уравнения $f'(x) = 0$. Используя метод Ньютона-Чебышева второго порядка, получим выражение для (n+1)-го приближения:

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)}\left[1 + \frac{1}{2}\cdot\frac{f'(x_n)\cdot f'''(x_n)}{f''(x_n)^2}\right] \tag{17}$$

Заменив функцию действительного аргумента $f(x)$ на соответствующую эквивалентную функцию $F\big(X_{[3]}\big)$ дуальных чисел класса 3, получим (n+1)-е приближение в виде:

$$x_{n+1} = x_n - \tau(X_n) - \mu(X_n), \tag{18}$$

где: $\tau(X_n) = Im_1\big(F(X_n)\big)/Im_2\big(F(X_n)\big)$ и

$\mu(X_n) = 0.5\cdot Im_1\big(F(X_n)\big)^2\cdot Im_3\big(F(X_n)\big)/Im_2\big(F(X_n)\big)^3$.*)

Если отбросить $\mu(X_n)$, то имеем классический метод Ньютона. На интервале поиска решения функция

$f(x)$ должна быть гладкой и $f''(x) \neq 0$. За начальное приближение следует выбирать точку, удовлетворяющую неравенству $f'(x_0)\cdot f''(x_0) > 0$.

Для поиска минимума функции двух переменных $f(x,y)$ метод Ньютона определяется формулой:

$$\begin{bmatrix}x\\y\end{bmatrix}_{n+1} = \begin{bmatrix}x\\y\end{bmatrix}_n - G^{-1}(x_n,y_n)\cdot\nabla F(x_n,y_n),$$

где:

$G(x,y) = \begin{bmatrix}F''_x(x,y) & F''_{xy}(x,y)\\ F''_{yx}(x,y) & F''_y(x,y)\end{bmatrix}$ и $\nabla F(x,y) = \begin{bmatrix}F'_x(x,y)\\ F'_y(x,y)\end{bmatrix}$ – матрица Гессе и градиент функции $F(x,y)$ соответственно. Матрица обратная $G(x,y)$ имеет вид:

$$G^{-1}(x,y) = \det\big(G(x,y)\big)\cdot\begin{bmatrix}F''_y(x,y) & -F''_{xy}(x,y)\\ -F''_{yx}(x,y) & F''_x(x,y)\end{bmatrix},$$

где: $\det\big(G(x,y)\big) = 1/(F''_x(x,y)\cdot F''_y(x,y) - F''_{xy}(x,y)\cdot F''_{yx}(x,y))$.

Тогда окончательно имеем:

*) Компьютерную реализацию см. Приложение 2 и 3

$$\begin{bmatrix} x \\ y \end{bmatrix}_{n+1} = \begin{bmatrix} x \\ y \end{bmatrix}_n - \det(x_n, y_n) \begin{bmatrix} F_y''(x,y)\cdot F_x'(x,y) - F_{xy}''(x,y)\cdot F_y'(x,y) \\ F_x''(x,y)\cdot F_y'(x,y) - F_{yx}''(x,y)\cdot F_x'(x,y) \end{bmatrix}_n . \quad (19)$$

Как видно, для вычисления следующего приближения $n+1$ необходимо знать: $F_x'(x_n, y_n)$, $F_y'(x_n, y_n)$, $F_x''(x_n, y_n)$, $F_y''(x_n, y_n)$, $F_{xy}''(x_n, y_n)$. Если $F(x,y)$ представима в виде $F(x,y) = \varphi(x)\cdot\psi(y)$, то $F_x'(x,y) = \varphi'(x)\cdot\psi(y)$, $F_x''(x,y) = \varphi''(x)\cdot\psi(y)$, $F_y'(x,y) = \varphi(x)\cdot\psi'(y)$, $F_y''(x,y) = \varphi(x)\cdot\psi''(y)$, $F_{xy}''(x,y) = \varphi'(x)\cdot\psi'(y)$ и можно воспользоваться таблицей элементарных (базовых) функций одного гипер-дуального аргумента (см. табл. 4) *).

И, наконец, рассмотрим метод Ньютона-Рафсона (метод Ньютона с регулировкой шага) для поиска минимума функции двух переменных $f(x,y)$. Итерационный процесс (19) в этом случае определяется выражением:

$$\begin{bmatrix} x \\ y \end{bmatrix}_{n+1} = \begin{bmatrix} x \\ y \end{bmatrix}_n - \beta\cdot \begin{bmatrix} v \\ u \end{bmatrix}_n \quad (20)$$

где: $v_n = \det(x_n, y_n)\cdot(F_y''(x_n, y_n)\cdot F_x'(x_n, y_n) - F_{xy}''(x_n, y_n)\cdot F_y'(x_n, y_n))$ и

$u_n = \det(x_n, y_n)\cdot(F_x''(x_n, y_n)\cdot F_y'(x_n, y_n) - F_{yx}''(x_n, y_n)\cdot F_x'(x_n, y_n))$ – направления спуска, а

β – величина шага, которая определяется в результате решения задачи одномерной минимизации относительно β:

$$\min_{\beta \geq 0} F(x_n - \beta\cdot v_n,\ y_n - \beta\cdot u_n), \quad (21)$$

что обеспечивает наискорейший спуск [10]. При $\beta = 1$ соотношения (20) переходят в (19). Сходимость метода Ньютона-Рафсона, независима от начального приближения, что не присуще методу Ньютона для невыпуклых функций *).

Компьютерная реализация

Компьютерная реализация компактных дуальных чисел класса 3 была выполнена на языке SWIFT операционной системы macOS (см. Приложение 2). Был создан новый тип данных Tdn3 (truncated-dual number 3) и его расширение, которое включает перезагрузку элементарных функции (см. табл. 4) и операций.
Отдельно добавлены процедуры решения выше рассмотренных вычислительных задач.
Использование этого типа данных приведено в Приложениях 2 и 3.

*) Компьютерную реализацию см. Приложение 2 и 3

Приложение 1

Таблицы умножения базисов мнимых частей компактных дуальных чисел.

1)

×	ε
ε	0

2)

×	ε	ω
ε	2ω	0
ω	0	0

3)

×	ε	ω	γ
ε	2ω	3γ	0
ω	3γ	0	0
γ	0	0	0

4)

×	ε	ω	γ	η
ε	2ω	3γ	4η	0
ω	3γ	6η	0	0
γ	4η	0	0	0
η	0	0	0	0

Таблицы умножения базисов мнимых частей для первого (1), второго (2), третьего (3) и четвертого (4) классов компактных дуальных чисел.

Приложение 2

Ниже приведено описание типа данных `Tdn3` (truncated-dual number 3) для Swift 5 (macOS 10.14)

```
import Foundation

precedencegroup SuperPowerPrecedence {
    associativity: left
    assignment: true
    higherThan: MultiplicationPrecedence}
infix operator **: SuperPowerPrecedence

struct Tdn3{
   var re, im1, im2, im3: Double;
   // Initializers:
   init() {...}
   init(re:Double) {...}
   init(re:Double, im1:Double, im2:Double, im3:Double) {...}
   // Methods:
   func norm() -> Double{return self.re.magnitude}
   func string() -> String {...}
}

extension Tdn3{
    // CONSTANTS:
    static let accuracy:Double = 1e-15, PI:Double = Double.pi;
    // OPERATORS (overloading):
    static prefix func - (A:Tdn3) -> Tdn3 {...}
    static prefix func + (A:Tdn3) -> Tdn3 {...}
    static func + (A:Tdn3, B:Tdn3) -> Tdn3 {...}
    static func += (lhs: inout Tdn3, rhs: Tdn3) {...}
    static func - (A:Tdn3, B:Tdn3) -> Tdn3 {...}
    static func -= (lhs: inout Tdn3, rhs:Tdn3) {...}
    static func * (A:Tdn3, B:Tdn3) -> Tdn3 {...}
    static func * (A:Tdn3, b:Double) -> Tdn3 {...}
    static func * (a:Double, B:Tdn3) -> Tdn3 {...}
    static func *= (lhs: inout Tdn3, rhs:Tdn3) {...}
```

```
 static func *= (lhs: inout Tdn3, rhs:Double) {...}
 static func / (A:Tdn3, B:Tdn3) -> Tdn3 {...}
 static func / (A:Tdn3, B:Double) -> Tdn3 {...}
 static func /= (lhs: inout Tdn3, rhs: Tdn3) {...}
 static func /= (lhs: inout Tdn3, rhs: Double) {...}
 static func ** (left:Tdn3, right:Double) -> Tdn3 {...}
 static func == (left:Tdn3, right:Tdn3) -> Bool {...}
 static func != (left:Tdn3, right:Tdn3) -> Bool {...}
 static func <  (left:Tdn3, right:Tdn3) -> Bool {...}
 static func >  (left:Tdn3, right:Tdn3) -> Bool {...}
 static func <= (left:Tdn3, right:Tdn3) -> Bool {...}
 static func >= (left:Tdn3, right:Tdn3) -> Bool {...}

// FUNCTIONS:  F(X) = f(X.re) + X.im1·f'(X.re)•ε + X.im2·f''(X.re)•ω +  X.im3·f'''(X.re)•γ
 static func inverse (A: Tdn3) -> Tdn3 {...}

 static func pow (x:Double, n:Double) -> Tdn3 {...}
 static func pow (X:Tdn3, n:Double) -> Tdn3 {...}

 static func powX (a:Double, x:Double) -> Tdn3 {...}
 static func powX (A:Tdn3, x:Double) -> Tdn3 {...}

 static func exp (x:Double) -> Tdn3 {...}
 static func exp (X:Tdn3) -> Tdn3 {...}

 static func log (x:Double)-> Tdn3 {...}
 static func log (X:Tdn3)-> Tdn3 {...}

 static func ln (x:Double)-> Tdn3 {...}
 static func ln (X:Tdn3)-> Tdn3 {...}

 static func sqrt (x:Double) -> Tdn3 {...}
 static func sqrt (X:Tdn3) -> Tdn3 {...}

 static func sin (x:Double) -> Tdn3 {...}
 static func sin (X:Tdn3) -> Tdn3 {...}

 static func cos (x:Double) -> Tdn3 {...}
 static func cos (X:Tdn3) -> Tdn3 {...}

 static func tg (x:Double)-> Tdn3 {...}
 static func tg (X:Tdn3)-> Tdn3 {...}

 static func ctg (x:Double) -> Tdn3 {...}
 static func ctg (X:Tdn3) -> Tdn3 {...}

 static func asin (x:Double) -> Tdn3 {...}
 static func asin (X:Tdn3) -> Tdn3 {...}

 static func acos (x:Double) -> Tdn3 {...}
 static func acos (X:Tdn3) -> Tdn3 {...}

 static func atg (x:Double) -> Tdn3 {...}
 static func atg (X:Tdn3) -> Tdn3 {...}

 static func actg (x:Double) -> Tdn3 {...}
 static func actg (X:Tdn3) -> Tdn3 {...}

 static func sh (x:Double) -> Tdn3 {...}
 static func sh (X:Tdn3) -> Tdn3 {...}

 static func ch (x:Double) -> Tdn3 {...}
 static func ch (X:Tdn3) -> Tdn3 {...}

 static func max (X1:Tdn3, X2:Tdn3)-> Tdn3 {...}
 static func min (X1:Tdn3, X2:Tdn3)-> Tdn3 {...}

 static func dX  (x:Double, y:Double, F:(Tdn3,Tdn3) -> Tdn3) -> Tdn3 {...}
 static func dY  (x:Double, y:Double, F:(Tdn3,Tdn3) -> Tdn3) -> Tdn3 {...}
 static func dXY (x:Double, y:Double, F:(Tdn3,Tdn3) -> Tdn3) -> Tdn3 {...}

// SOLUTIONS:
```

```
// Newton-Chebyshev's method Xn+1 = Xn - a1*(1 + 0.5*(a1*a2 + a1*a1*(a2*a2 - a3/3)))
 static func NCH (x0:Double, F:(Tdn3)->Tdn3, E:Double, maxIteraction:Int=100) -> Double{
     var Xn = x0, Xnn = x0, Fn:Tdn3,  i:Int = 1, a1:Double=0, a2:Double=0, a3:Double=0
     while true {
         Fn = F(Tdn3(re: Xn, im1: 1, im2: 0, im3: 0));
         if abs(Fn.re) <= E || i > maxIteraction || Xnn.isNaN {return Xnn}
         if Fn.im1 == 0 {return Double.infinity}
         a1=Fn.re/Fn.im1; a2=Fn.im2/Fn.im1; a3=Fn.im3/Fn.im1;
         Xnn = Xn - a1*(1 + a1*a2/2.0 + a1*a1*(a2*a2/2.0 - a3/6.0))
         i += 1;          Xn = Xnn;
     }
     return Double.infinity
 }

// Newton-Chebyshev's method (minimum of function)  Xn+1 = Xn - f'(Xn)/f''(Xn)- [f'''(Xn)*f'(Xn)^2)]/[2*f''(Xn)^3]
 static func NCH_Min (x0:Double, F:(Tdn3)->Tdn3, E:Double, maxIteraction:Int=100) -> Double{
     var Xn = x0, Xnn = x0, Fn:Tdn3,  i:Int = 1, τ:Double=0, μ:Double;
     while true {
         Fn = F(Tdn3(re: Xn, im1: 1, im2: 0, im3: 0));
         if Fn.im2 == 0 {return Double.infinity}
         τ = Fn.im1/Fn.im2;      μ = 0.5*Fn.im1*Fn.im3/(Fn.im2*Fn.im2)
         Xnn = Xn - τ - τ*μ;
         if abs(Xnn - Xn) <= E || i > maxIteraction || Xnn.isNaN {return Xnn}
         i += 1;          Xn = Xnn;
     }
     return  Double.infinity
 }

// Minimum of function  F(X ,Y)  (Newton)
 static func XY_Min (x0:Double,y0:Double,F:(Tdn3, Tdn3) -> Tdn3, E:Double,
                     maxIteraction:Int=100) -> (x:Double, y:Double){
     var Xn, Yn, Xnn, Ynn, det, Fx, Fy, Fxx, Fyy, Fxy:Double, i:Int = 1;
     var dX, dY, dXY: Tdn3;
     Xn = x0;  Yn = y0;
     while true {
         dX  = Tdn3.dX(x: Xn, y: Yn, F: F);
         dY  = Tdn3.dY(x: Xn, y: Yn, F: F);
         dXY = Tdn3.dXY(x: Xn, y: Yn, F: F);
         Fx = dX.im1;    Fxx = dX.im2;   Fy = dY.im1;    Fyy = dY.im2;
         Fxy = (dXY.im2 - dX.im2 - dY.im2)/2;
         det = (Fxx*Fyy - Fxy*Fxy)
         if det == 0 {return (x:Double.infinity, y: Double.infinity)}
         Xnn = Xn - (Fyy*Fx - Fxy*Fy)/det;   Ynn = Yn - (Fxx*Fy - Fxy*Fx)/det;
         if (abs(Xnn - Xn) <= E && abs(Ynn - Yn) < E)  || i > maxIteraction  {return (x:Xnn,
y:Ynn)}
         i += 1; Xn = Xnn; Yn = Ynn;
     }
    return (x:Double.infinity, y: Double.infinity)
 }

// Minimum of function F(X ,Y)  (Newton-Raphson)
 static func XYR_Min (x0:Double,y0:Double,F:@escaping (Tdn3, Tdn3) -> Tdn3, E:Double,
                      maxIteraction:Int=100) -> (x:Double, y:Double){
     var Xn, Yn, Xnn, Ynn, Vn, Un, bn, det, Fx, Fy, Fxx, Fyy, Fxy:Double, i:Int = 1;
     var dX, dY, dXY: Tdn3;
     Xn = x0;  Yn = y0; bn = 1;
     func getB(b:Tdn3)->Tdn3{
         let X = Xn - b.re*Vn, Y = Yn - b.re*Un;
         let R = F(Tdn3(re: X, im1: -Vn, im2: 0, im3: 0), Tdn3(re: Y, im1: -Un, im2: 0, im3:
0));
        return R
     }
     while true {
         dX  = Tdn3.dX(x: Xn, y: Yn, F: F);
         dY  = Tdn3.dY(x: Xn, y: Yn, F: F);
         dXY = Tdn3.dXY(x: Xn, y: Yn, F: F);
         Fx  = dX.im1;    Fxx = dX.im2;   Fy = dY.im1;    Fyy = dY.im2;
         Fxy = (dXY.im2 - dX.im2 - dY.im2)/2;
         det = (Fxx*Fyy - Fxy*Fxy)
         if det == 0 {return (x:Double.infinity, y: Double.infinity)}
         Vn  = (Fyy*Fx - Fxy*Fy)/det;  Un = (Fxx*Fy - Fxy*Fx)/det;
         bn  = NCH_Min(x0: bn, F: getB, E: 0.0001)
```

```
            Xnn = Xn - bn*Vn;   Ynn = Yn - bn*Un;
            if (abs(Xnn - Xn) <= E && abs(Ynn - Yn) < E)  || i > maxIteraction  {return (x:Xnn,
y:Ynn)}
            i += 1; Xn = Xnn; Yn = Ynn;
        }
        return (x:Double.infinity, y: Double.infinity)
    }
}
```

Приложение 3

Ниже приводятся численные результаты манипуляций с дуальными числами типа TDN3 (см. Приложение 2).

```
    let A:Tdn3 = Tdn3(re: 1, im1: 2, im2: 3, im3: 4)        //  1.0 + 2.0ε + 3.0ω + 4.0γ
    let B:Tdn3 = Tdn3(re: 10, im1: 20, im2: 30, im3: 40)    //  10.0 + 20.0ε + 30.0ω+ 40.0γ
    let C = A + B                                           //  11.0 + 22.0ε + 33.0ω + 44.0γ
    let A1 = Tdn3.inverse(A: A)                             //  1.0 - 2.0ε + 5.0ω -16.0γ
    let E = A*A1                                            //  1.0 + 0.0ε + 0.0ω + 0.0γ
    let M = A / B                                           //  0.1 + 0.0ε + 0.0ω + 0.0 γ
    let N = M*B                                             //  1.0 + 2.0ε + 3.0ω + 4.0γ
// EQUATION SOLUTION (Newton-Chebyshev's method for F(X)=0)
    var x = Tdn3.NCH(x0: 1.5, F: f_x, E:0.001)
        //  x0=1.5
        //  x1=0.9463570157614446
        //  x2=0.7148696891295812
        //  x3=0.7065126646557145
// MINIMUM CALCULATION (Newton-Chebyshev's method for min F(X))
    x = Tdn3.NCH_Min(x0: 1.5, F: f_x, E:0.001)
        //  x0= 1.5
        //  x1=1.005144379423351
        //  x2=0.43306354857263607
        //  x3=0.07990305891534454
        //  x4=0.0013514819485346396
        //  x5=8.598517941619242e-09
        //  x6=3.329495684666667e-24
// MINIMUM CALCULATION (Newton's method for min F(X, Y))
    var xy = Tdn3.XY_Min(x0: 1, y0: 1, F: f_xy, E: 0.0000000001, maxIteraction: 100)
        //  x0= 1.0                 y0= 1.0
        //  x1= 0.564757792293217   y1= 0.28237889614660894
        //  x2=-0.064847300998348   y2= -0.0324236504991740
        //  x3=-0.296214032477046   y3= -0.1481070162385230
        //  x4=-0.312700620210325   y4= -0.1563503101051627
        //  x5=-0.312766806080772   y5= -0.1563834030403864
        //  x6=-0.312766807129992   y6= -0.1563834035649960
        //  x7=-0.312766807129992   y7= -0.1563834035649960
// MINIMUM CALCULATION (Newton-Raphson's method for min F(X, Y))
    xy = Tdn3.XYR_Min(x0: 1, y0: 1, F: f_xy, E: 0.0000000001, maxIteraction: 100)
        //  x0= 1.0                 y0= 1.0                 b0=1.0
                //  b1=1.8881398171007115
                //  b2=1.9170744702880769
                //  b3=1.9170743626716933
        //  x1=  0.16560832205269926        y1= -0.3757330203094966
                //  b1=1.0079897625027896
                //  b2=1.0083091294685174
                //  b3=1.0083091294685016
        //  x2=  -0.308907101474343         y2= -0.1506435059313302
                //  b1=1.0012675578115906
                //  b2=1.0012675578118921
        //  x3=  -0.31276184522550854       y3= -0.15638575206481156
                //  b1=1.000000073862478
                //  b2=1.00000007386528
        //  x4=  -0.3127668071296317        y4= -0.15638340356445912
                //  b1=1.00000007386528
        //  x5=  -0.31276680712999216       y5= -0.15638340356499608
```

```
//CUSTOM FUNCTIONS:
//--  X^3 + exp(X^2)  - 2
func f_x (X:Tdn3) -> Tdn3 {return Tdn3.pow(X: X, n: 3) + Tdn3.exp(X: X*X) -  Tdn3(re:2)}

//--  X^2 + 2Y^2 + exp(X+Y)
func f_xy (X:Tdn3, Y:Tdn3)->Tdn3{return X*X + 2.0*(Y*Y) + Tdn3.exp(X: X + Y)}
```

Литература

1. *Тихонов А. Н., Арсенин В. Я.* Методы решения некорректных задач. – М.: Наука, 1979. – 285 с.
2. *Naumann U.* The Art of Differentiating Computer Programs. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2012.
3. *Дименберг Ф. М.* Теория винтов и ее приложения. – М.: Наука, 1978. – 328 с.
4. *Fike* J. *A. and Alonso J. J.* The Development of Hyper-Dual Numbers for Exact Second Derivative Calculations. AIAA paper 2011-886, 49th AIAA Aerospace Sciences Meeting, January 4-7, 2011.
5. *Fike* J. *A.* Multi-Objective Optimization Using Hyper-Dual Numbers. Ph.D. Dissertation, Stanford University, 2013.
6. *Сильвестров В. В.* Системы чисел. – Соросовский образовательный журнал, № 8, 1998, 121-127 с.
7. *Двайт Г. Б.* Таблицы интегралов и другие математические формулы. - М.: Наука, 1973. - 228 с.
8. *Шарый С. П* Курс вычислительных методов. – Новосибирск: Новосиб. гос. ун-т., 2014. - 507 с.
9. *Antoni G.* An Efficient and Straightforward Numerical Technique Coupled to Classical Newton's Method for Enhancing the Accuracy of Approximate Solutions Associated with Scalar Nonlinear Equations. International Journal of Engineering Mathematics, vol. 2016, Article ID 8565821, 12 p.
10. *Кононюк А. Е.* Основы теории оптимизации. Безусловная оптимизация К.2. ч.1. Киев: "Освіта України", 2011. - 544 с.

Абстракт

В данной публикации рассматриваются вопросы, связанные с применением специальных дуальных чисел (супер-дуальных чисел) в методе компьютерного автоматического дифференцирования. Вводятся новые типы дуальных чисел разных классов. Рассматриваются основные операции и базовые функции пространства некоторых классов супер-дуальных чисел. В приложениях 1, 2 и 3 приведена компьютерная реализация компактных дуальных чисел третьего класса для языка SWIFT операционной системы macOS.

Ключевые слова: *супер-дуальные числа, автоматическое дифференцирование*

---

2 августа 2019 г