# К ЧИСЛЕННОМУ РЕШЕНИЮ ЗАДАЧИ КОШИ С ИСПОЛЬЗОВАНИЕМ ГИПЕР-ДУАЛЬНЫХ ЧИСЕЛ

## 1. Задача Коши

Задача Коши – одна из основных задач теории дифференциальных уравнений, которая состоит в нахождении решения дифференциального уравнения, удовлетворяющего начальным условиям. Подавляющее большинство важных для современной практики дифференциальных уравнений, описывающие реальные явления, не поддается аналитическому интегрированию. Поэтому приходится прибегать к использованию приближенных численных методов. В этом случаи область интегрирования разбивается определенным количеством опорных точек (узлов), в которых и осуществляется интегрирование.
Рассмотрим задачу Коши в параметрическом виде:

$$\begin{cases} y'(t) = \varphi(t) - \alpha x(t) \\ x'(t) = \psi(t) - \beta y(t) \end{cases} \tag{1.1}$$

с начальными условиями $y(t_0) = y_o,\ x(t_0) = x_o$. Здесь штрих означает производную по *t*, а *t* – вещественный параметр (например, время), $\alpha$ и $\beta$ – некоторые вещественные числа, $\varphi(t)$ и $\psi(t)$ – произвольные аналитические функции, $y(t)$ и $x(t)$ – искомое решение. Уравнения (1.1) подобны уравнениям гармонического осциллятора [1] и фактически эквивалентны дифференциальному уравнению второго порядка $y''(t) - \alpha\beta y(t) = \varphi'(t) - \alpha\psi(t)$.
Одним из старейших методов интегрирования дифференциальных уравнений является метод разложения в ряд Тейлора. Если разложить функции $y(t)$ и $x(t)$ в ряды Тейлора в окрестности точки $t_0$, то для вычисления $y(t_0 + \Delta t)$ и $x(t_0 + \Delta t)$ (при достаточно малом шаге интегрирования $\Delta t$) требуются значения производных различных порядков этих функций в точке $t_0$. Из (1.1) можно получить следующие рекуррентные соотношения:

$$\begin{array}{ll} y'(t_0) = \varphi(t_0) - \alpha x(t_0), & x'(t_0) = \psi(t_0) - \beta y(t_0), \\ y''(t_0) = \varphi'(t_0) - \alpha x'(t_0), & x''(t_0) = \psi'(t_0) - \beta y'(t_0), \\ y'''(t_0) = \varphi''(t_0) - \alpha x''(t_0), & x'''(t_0) = \psi''(t_0) - \beta y''(t_0), \\ \dots & \dots \\ y^{(n)}(t_0) = \varphi^{(n-1)}(t_0) - \alpha x^{(n-1)}(t_0), & x^{(n)}(t_0) = \psi^{(n-1)}(t_0) - \beta y^{(n-1)}(t_0), \end{array} \tag{1.2}$$

где *n=1,2,3, … N* – максимальный порядок производных, удерживаемых в разложении Тейлора. Тогда для точки $t_0 + \Delta t$ будем иметь:

$$\begin{aligned} y_1 = y(t_0 + \Delta t) &\approx y(t_0) + y'(t_0)\Delta t + \frac{1}{2}y''(t_0)\Delta t^2 + \frac{1}{6}y'''(t_0)\Delta t^3 + \cdots, \\ x_1 = x(t_0 + \Delta t) &\approx x(t_0) + x'(t_0)\Delta t + \frac{1}{2}x''(t_0)\Delta t^2 + \frac{1}{6}x'''(t_0)\Delta t^3 + \cdots, \end{aligned} \tag{1.3}$$

Получив значения $y_1$ и $x_1$, далее можно определить значения соответствующих производных по формулам (1.2) и перейти в следующую точку, добавив шаг интегрирования $\Delta t$:

$$\begin{aligned} y_{i+1} &= y_i + y_i'\Delta t + \frac{1}{2}y_i''\Delta t^2 + \frac{1}{6}y_i'''\Delta t^3 + \cdots, \\ x_{i+1} &= x_i + x_i'\Delta t + \frac{1}{2}x_i''\Delta t^2 + \frac{1}{6}x_i'''\Delta t^3 + \cdots \end{aligned} \tag{1.4}$$

Сохраняя разное количество членов в (1.4) будем получать методы разных порядков.
Если функции $\varphi(t)$ и $\psi(t)$ достаточно сложные или более того заданы композитным программным кодом, то вычисление значений их производных становится весьма трудоемкой, а иногда и невыполнимой задачей, что является главным недостатком метода разложения в ряд Тейлора. Однако этот метод является основой для многих других методов, где вычисление значений производных осуществляется на базе той либо иной разностной схемы. В свою очередь, использование разностных схем приводить к потери точности, повышенной трудоемкости, к серьезным проблемам определения оптимального шага интегрирования.

## 2. Автоматическое дифференцирование

С развитием вычислительной техники возникло новое направление в вопросах численного дифференцирования – автоматическое дифференцирование [2, 3], связанное с точным (точностью представления чисел в компьютерной системе) вычислением производных сложных математических функций. Автоматическое дифференцирование (АД) позволяет избежать дублирование функциональности программного кода (изменение кода функции не требует изменения кода ее производной). Для компьютерной реализации АД необходимо создать новый тип данных, перезагрузить базовые математические функции и операции над ними. Если новый тип данных строится на основе гипер-дуальных чисел [4, 5], то за одно обращение к перезагруженной функции точно вычисляются значения самой функции и ее первой и второй производных.

## 3. Гипер-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 [6]. 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 $\boldsymbol{\varepsilon}_2$. Будем

называть размер основного базиса *классом* ($K$) дуального числа[*]. Класс дуального числа указывает на максимальный порядок производной, входящей в определение функции соответствующего дуального аргумента. Так, например, классические дуальные числа [6] относятся к первому, а гипер-дуальные числа [4, 5] ко второму классам соответственно. Расширим гипер-дуальные числа [4, 5] до дуальных чисел третьего класса. Дуальное число третьего класса имеет вид $A_{[3]} = a + a_1\boldsymbol{\varepsilon} + a_2\boldsymbol{\omega} + a_3\boldsymbol{\gamma}$ с правилом умножения базиса мнимых частей ( $\boldsymbol{\varepsilon}$, $\boldsymbol{\omega}$ и $\boldsymbol{\gamma}$) представленном в следующей таблице.

| × | $\boldsymbol{\varepsilon}$ | $\boldsymbol{\omega}$ | $\boldsymbol{\gamma}$ |
|---|---|---|---|
| $\boldsymbol{\varepsilon}$ | $2\boldsymbol{\omega}$ | $3\boldsymbol{\gamma}$ | 0 |
| $\boldsymbol{\omega}$ | $3\boldsymbol{\gamma}$ | 0 | 0 |
| $\boldsymbol{\gamma}$ | 0 | 0 | 0 |

*Таблица 3.1. Правила умножения базисов мнимых частей дуальных чисел третьего класса*

Алгебраические операции над дуальными числами 3-го класса определяются следующими формулами:

$$
\begin{aligned}
&X_{[3]} = x + x_1\boldsymbol{\varepsilon} + x_2\boldsymbol{\omega} + x_3\boldsymbol{\gamma}, \qquad Y_{[3]} = y + y_1\boldsymbol{\varepsilon} + y_2\boldsymbol{\omega} + y_3\boldsymbol{\gamma}\\
&X_{[3]} + Y_{[3]} = x + y + (x_1 + y_1)\boldsymbol{\varepsilon} + (x_2 + y_2)\boldsymbol{\omega} + (x_3 + y_3)\boldsymbol{\gamma},\\
&X_{[3]}\cdot Y_{[3]} = x\cdot y + (x\cdot y_1 + y\cdot x_1)\boldsymbol{\varepsilon} + (x\cdot y_2 + 2x_1\cdot y_1 + y\cdot x_2)\boldsymbol{\omega} + (x\cdot y_3 + y\cdot x_3 + 3(x_1\cdot y_2\\
&\qquad + y_1\cdot x_2))\boldsymbol{\gamma},\\
&Y_{[3]}^{-1} = y^{-1} - y^{-2}\cdot y_1\boldsymbol{\varepsilon} + y^{-2}(2\cdot y_1^2\cdot y^{-1} - y_2)\boldsymbol{\omega} + a^{-2}(6y_1\cdot y^{-1}(y_2 - y_1^2\cdot y^{-1}) - y_3)\boldsymbol{\gamma},\\
&X_{[3]}/Y_{[3]} = \mathrm{X}_{[3]}\cdot Y_{[3]}^{-1},
\end{aligned}
\tag{3.1}
$$

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

$$
\begin{aligned}
&F\left(X_{[3]}\right) = f(x) + x_1 f'(x)\boldsymbol{\varepsilon} + [x_2 f'(x) + x_1^2 f''(x)]\boldsymbol{\omega} + [x_3 f'(x) + 3x_1 x_2 f''(x) +\\
&x_1^3 f'''(x)]\boldsymbol{\gamma},\\
&F(x + \boldsymbol{\varepsilon} + 0\boldsymbol{\omega} + 0\boldsymbol{\gamma}) = f(x) + f'(x)\boldsymbol{\varepsilon} + f''(x)\boldsymbol{\omega} + f'''(x)]\boldsymbol{\gamma}
\end{aligned}
\tag{3.2}
$$

Воспользовавшись [7], приведем компоненты некоторых элементарных (базовых) функций для первого, второго и третьего классов дуальных чисел:

| $f(X)$ | $f(x)$ | $f'(x)$ | $f''(x)$ | $f'''(x)$ |
|---|---|---|---|---|
| $X^n$ | $x^n$ | $n\cdot x^{n-1}$ | $n\cdot(n-1)\cdot x^{n-2}$ | $n\cdot(n-1)\cdot(n-2)\cdot x^{n-3}$ |
| $e^X$ | $e^x$ | $e^x$ | $e^x$ | $e^x$ |
| $a^X$ | $a^x$ | $a^x\cdot\ln(a)$ | $a^x\cdot\ln^2(a)$ | $a^x\cdot\ln^3(a)$ |
| $\ln(X)$ | $\ln(x)$ | $x^{-1}$ | $-x^{-2}$ | $2\cdot x^{-3}$ |
| $\log_a(X)$ | $\log_a(x)$ | $x^{-1}\cdot\ln(a)^{-1}$ | $-x^{-2}\cdot\ln(a)^{-1}$ | $2\cdot x^{-3}\cdot\ln(a)^{-1}$ |
| $\sin(X)$ | $\sin(x)$ | $\cos(x)$ | $-\sin(x)$ | $-\cos(x)$ |

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

| | | | | |
|---|---|---|---|---|
| $\cos(X)$ | $\cos(x)$ | $-\sin(x)$ | $-\cos(x)$ | $\sin(x)$ |
| $\mathrm{tg}(X)$ | $\mathrm{tg}(x)$ | $\cos^{-2}(x)$ | $2{\cdot}\mathrm{tg}(x){\cdot}\cos^{-2}(x)$ | $2{\cdot}\cos^{-2}(x){\cdot}(3{\cdot}\mathrm{tg}^2(x)+1)$ |
| $\mathrm{ctg}(X)$ | $\mathrm{ctg}(x)$ | $-\sin^{-2}(x)$ | $-2{\cdot}\mathrm{ctg(x)}{\cdot}\sin^{-2}(x)$ | $-2{\cdot}\sin^{-2}(x){\cdot}(3{\cdot}\mathrm{ctg}^2(x)+1)$ |
| $\arcsin(X)$ | $\arcsin(x)$ | $(1-x^2)^{-1/2}$ | $x{\cdot}(1-x^2)^{-3/2}$ | $(2{\cdot}x^2+1){\cdot}(1-x^2)^{-5/2}$ |
| $\arccos(X)$ | $\arccos(x)$ | $-(1-x^2)^{-1/2}$ | $-x{\cdot}(1-x^2)^{-3/2}$ | $-(2{\cdot}x^2+1){\cdot}(1-x^2)^{-5/2}$ |
| $\mathrm{arctg}(X)$ | $\mathrm{arctg}(x)$ | $(1+x^2)^{-1}$ | $-2{\cdot}x{\cdot}(1+x^2)^{-2}$ | $(6{\cdot}x^2-2){\cdot}(1+x^2)^{-3}$ |
| $\mathrm{arcctg}(X)$ | $\mathrm{arcctg}(x)$ | $-(1+x^2)^{-1}$ | $2{\cdot}x{\cdot}(1+x^2)^{-2}$ | $-(6{\cdot}x^2-2){\cdot}(1+x^2)^{-3}$ |
| $\mathrm{sh}(X)$ | $\mathrm{sh}(x)$ | $\mathrm{ch}(x)$ | $\mathrm{sh}(x)$ | $\mathrm{ch}(x)$ |
| $\mathrm{ch}(X)$ | $\mathrm{ch}(x)$ | $\mathrm{sh}(x)$ | $\mathrm{ch}(x)$ | $\mathrm{sh}(x)$ |

*Таблица 3.2. Компоненты элементарных функций для третьего классов дуальных чисел*

Вычисление сложной дуальной функции 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 $F_{n-1} = f_{n-1}(F_n, X)$, $F_{n-2} = f_{n-2}(F_{n-1}, X)$, … , $F = f_1(F_2, X)$.
Для практических целей (численной реализации) целесообразно использовать тот класс, который обеспечивает нужный порядок производной, входящей в определение функций соответствующего класса.

## 4. Использование дуальных чисел в задаче Коши

Для численного интегрирования уравнений (1.1) по формулам (1.4) ограничивающихся производными до третьего порядка включительно будем использовать дуальные числа 3-го класса (см. секцию 3). Далее для компактности изложения будем опускать индекс класса дуального числа. На первом шаге следует заменить функции $\varphi(t)$ и $\psi(t)$ на их дуальные эквиваленты: $\varphi(t) \Longrightarrow \Phi(T) \to Z$, $\psi(t) \Longrightarrow \Psi(T) \to Z$, где $T = t + 1\boldsymbol{\varepsilon} + 0\boldsymbol{\omega} + 0\boldsymbol{\gamma}$.
Начальные условия запишутся в виде:

$$T_0 = t_0 + 1\boldsymbol{\varepsilon} + 0\boldsymbol{\omega} + 0\boldsymbol{\gamma}, \quad Y_0(t_0) = y_0 + 0_1\boldsymbol{\varepsilon} + 0\boldsymbol{\omega} + 0\boldsymbol{\gamma}, \quad X_0(t_0) = x_0 + 0_1\boldsymbol{\varepsilon} + 0\boldsymbol{\omega} + 0\boldsymbol{\gamma}$$

Тогда процедура интегрирования в *i*-й точке (*i* = 0, 1, 2, …) состоит из последовательного применения следующих формул:

$$T = T_0 + i{\cdot}\Delta t = (t_0 + i{\cdot}\Delta t) + 1\boldsymbol{\varepsilon} + 0\boldsymbol{\omega} + 0\boldsymbol{\gamma},$$
$$Y_i(T).Im1 = \Phi(T).Re - \alpha\, X_i(T).Re, \quad \longrightarrow \quad X_i(T).Im1 = \Psi(T).Re - \beta\, Y_i(T).Re,$$
$$Y_i(T).Im2 = \Phi(T).Im1 - \alpha\, X_i(T).Im1, \quad \longrightarrow \quad X_i(T).Im2 = \Psi(T).Im1 - \beta\, Y_i(T).Im1,$$
$$Y_i(T).Im3 = \Phi(T).Im2 - \alpha\, X_i(T).Im2, \quad \longrightarrow \quad X_i(T).Im3 = \Psi(T).Im2 - \beta\, Y_i(T).Im2,$$
$$Y_{i+1}(T).Re = YX(Y_i), \qquad X_{i+1}(T).Re = YX(X_i),$$

где: $YX(Z) = Z.Re + \Delta t{\cdot}Z.Im1 + \frac{1}{2}\Delta t^2{\cdot}Z.Im2 + \Delta t^3 Z.Im3$

Таким образом, при *i* = 0 вычисляются значения производных функций $y(t_0)$ и $x(t_0)$, т.е. $Y.Im1$, $Y.Im2$, $Y.Im3$ и $X.Im1$, $X.Im2$, $X.Im3$, а затем используя полученные величины определяются $y(t)$ и $x(t)$ в следующем узле при $t = t_0 + \Delta t$ т.е. $Y.Re$, $X.Re$ и т. д.
Графическая схема этого процесса представлена на рис. 4.1.

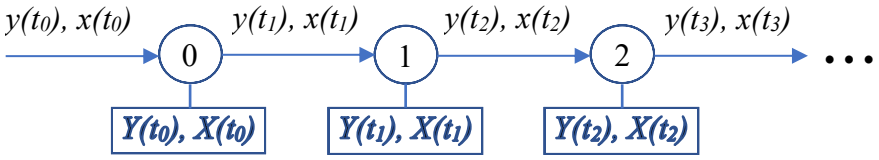


*Рисунок 4.1. Схема последовательного точечного интегрирования*

Заметим, что эта процедура напоминает простую линейную нейронную сеть, где узлы интегрирования ассоциируются с нейронами, которые получают входной сигнал, обрабатывают его, а затем передают результаты следующему нейрону.

## 5. Компьютерная реализация

Компьютерная реализация компактных дуальных чисел класса 3 была выполнена на языке SWIFT операционной системы macOS (см. Приложение 1). Был создан новый тип данных Tdn3 (truncated-dual number 3) и его расширение, которое включает перезагрузку элементарных функции (см. табл. 3) и операций.
Отдельно добавлены процедура интегрирования уравнений (1.1) *Taylor( )* с использованием созданного типа данных Tdn3 (см. Приложениях 2) и соотношений секции 4.

## 6. Численный эксперимент

Для проведения численного анализа предлагаемого подхода была рассмотрена параметри – ческая задача Коши (1.1) с $\varphi(t) = 2e^{3t}$, $\psi(t) = e^{3t}$, $\alpha = \beta = 1$, $y(0) = 3$ и $x(0) = 1$. В этом случае точное аналитическое решение имеет вид $y(t) = 0.75e^{t} + 1.625e^{-t} + 0.625e^{3t}$, $x(t) = -0.75e^{t} + 1.625e^{-t} + 0.125e^{3t}$, которое использовалось в качестве эталона. Была проведена серия расчетов по процедуре *Taylor( )* (см. Приложениях 2) при разных значениях $\Delta t$ на интервале $t = [0, 1]$. Некоторые результаты этого численного эксперимента приведены в табл. 6.1. Полученные численные решения весьма близки к точному решению. Максимальная локальная погрешность на конце интервала интегрирования $t = 1$ при $\Delta t = 0.1$ составила: ~0.08% для $y(t)$ и ~0.04% для $x(t)$. При $\Delta t = 0.05$ максимальная погрешность уже получилась ~0.01% для $y(t)$ и $x(t)$. На рис. 6.1 представлены графики $y(t)$ и $x(t)$ при $\Delta t = 0.1$. В тоже время метод Рунге-Кутты четвертого порядка для $\Delta t = 0.1$ уже при $t = 0.5$ дает локальную погрешность более 3%.

| t/n | *n*=5 | | *n*=10 | | Точное решение | |
|---|---|---|---|---|---|---|
| | *y* | *x* | *y* | *x* | *y* | *x* |
| 0 | 3.0 | 1.0 | 3.0 | 1.0 | 3.0 | 1.0 |
| 0.1 | - | - | 3.1426667 | 0.8101667 | 3.1429007 | 0.8102149 |
| 0.2 | 3.3813333 | 0.6413333 | 3.3847711 | 0.6420621 | 3.3853138 | 0.6421503 |
| 0.3 | - | - | 3.7525212 | 0.4987684 | 3.7534757 | 0.4988859 |

| | | | | | | |
|---|---|---|---|---|---|---|
| 0.4 | 4.2721989 | 0.3839689 | 4.2817034 | 0.3852828 | 4.2832116 | 0.3854162 |
| 0.5 | - | - | 5.0209527 | 0.3091509 | 5.0232089 | 0.3092825 |
| 0.6 | 6.0156516 | 0.2796453 | 6.0361690 | 0.2813297 | 6.0394376 | 0.2814357 |
| 0.7 | - | - | 7.4164813 | 0.3173595 | 7.4211218 | 0.3174078 |
| 0.8 | 9.2416118 | 0.4372743 | 9.2823006 | 0.4389541 | 9.2888005 | 0.4389009 |
| 0.9 | - | - | 11.7961905 | 0.6761540 | 11.8052103 | 0.6759398 |
| 1.0 | 15.099853 | 1.0692643 | 15.1775423 | 1.0702414 | 15.1899760 | 1.0697848 |

*Таблица 6.1. Результаты численного интегрирования уравнения*
$\boldsymbol{y}'(t) = 2e^{3t} - x(t),\ \ x'(t) = e^{3t} - y(t)$ *при* $y(0) = 3,\ \ x(0) = 1$

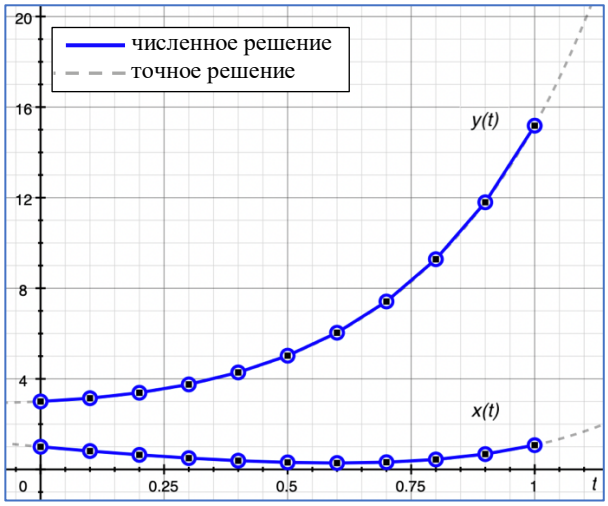


*Рисунок 6.2. Графики результатов численного интегрирования уравнения*
$\boldsymbol{y}'(t) = 2e^{3t} - x(t),\ \ x'(t) = e^{3t} - y(t)$ *при* $y(0) = 3,\ \ x(0) = 1,\ \Delta t = 0.1$

Следует заметить, что рассмотренная выше процедура численного интегрирования была также реализована для дуальных чисел 4-го класса, включающего вычисление производных до 4-го порядка включительно (здесь не приводиться). Численный эксперимент с использованием этого класса дуальных чисел для тестируемой функции уже при $\Delta t = 0.1$ дал совпадение с точным решением до пятого знака после запятой.

## 7. Заключение

Представленный новый тип гипер-дуальных чисел (дуальные числа третьего класса) позволяет за одно обращение к функциям этого класса получать точные значения самой функции и её производных до 3-го порядка включительно. Это обстоятельство позволяет реанимировать старейший метод интегрирования дифференциальных уравнений – метод разложения в ряд Тейлора. Процедура последовательного поточечного интегрирования (см. секцию 4) основанная на дуальных числах 3-го класса (см. секцию 3) и ее реализация в виде программного кода оказалась достаточно компактной (см. Приложение 2). Эта процедура использовалась для проведения численных экспериментов. Результаты проведенных

испытаний оказались весьма впечатляющими. Однако заметим, что целью работы не было фундаментальное исследование вопросов сходимости, определения оптимального шага интегрирования, оценки глобальной погрешности, производительности и т.д. Основная цель – продемонстрировать сравнительную простату, компактность и универсальность применения дуальных числах 3-го класса в вычислительных задачах, требующих многократного вычисления точных значений функций и их производных до третьего порядка включительно.

## Приложение 1

Ниже приведено описание типа данных `Tdn3` (truncated-dual number 3) для Swift 5 (macOS 10.15.3)

```
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 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 {...}
}
```

## Приложение 2

Ниже приводятся результаты численного интегрирования уравнения (1.1) при $\varphi(t) = 2e^{3t}$, $\psi(t) = e^{3t}$, $a = b = 1$ и $y(0) = 3$, $x(0) = 1$, $t_0 = 0$, $t_n = 1$, $n = 10$.

```
 let R = Taylor(φ: φ, ψ: ψ, a: 1.0, b: 1.0, t0: 0.0, tn: 1.0, n: 10, y0: 3.0, x0: 1.0);
// Result (R) :
// t : 0.0   y : 3.0                 x : 1.0
// t : 0.1   y : 3.14266666666666    x : 0.8101666666666667
// t : 0.2   y : 3.384771137214955   x : 0.6420621362227533
// t : 0.3   y : 3.752521168945022   x : 0.49876842456743287
```

```
  // t : 0.4   y : 4.281703379754168     x : 0.3852827916798613
  // t : 0.5   y : 5.0209527092498165    x : 0.30915093860204673
  // t : 0.6   y : 6.0361690136381165    x : 0.28132974201758915
  // t : 0.7   y : 7.416481343522962     x : 0.31735954326789384
  // t : 0.8   y : 9.282300614403239     x : 0.43895408050266516
  // t : 0.9   y : 11.796190569134417    x : 0.6761540435835838
  // t : 1.0   y : 15.177542303264662    x : 1.0702413773881008

func φ(t:Double)->Tdn3{return 2.0*Tdn3.exp(X: 3.0*Tdn3(re: t, im1: 1, im2: 0, im3: 0))}
func ψ(t:Double)->Tdn3{return Tdn3.exp(X: 3.0*Tdn3(re: t, im1: 1, im2: 0, im3: 0))}

//----------Taylor solution for y'= φ - a*x,   x'= ψ - b*y
//      φ, ψ   - predefined functions
//      a, b   - real numbers
//      t0, tn - start and end of integration interval
//      n      - number of parts into which the interval [t0, tn] is divided
//      y0, x0 - initial values
//      output:  array [(t, y, x)]
func Taylor(φ:(Double) -> Tdn3, ψ:(Double) -> Tdn3,
            a:Double,  b:Double, t0:Double, tn:Double, n:Int,
            y0:Double, x0:Double)->[(t:Double, y:Double, x:Double)]{
   let Δt:Double = (tn - t0)/Double(n);
   var f1:Tdn3, f2:Tdn3, tXY:[(t:Double, y:Double, x:Double)] = [];
   var Yi:(y:Double,y1:Double,y2:Double,y3:Double) = (y:y0,y1:0,y2:0,y3:0);
   var Xi:(x:Double,x1: Double,x2: Double,x3:Double) = (x:x0,x1:0,x2:0,x3:0);
   func yx(z:(_:Double,_:Double,_:Double,_:Double))-> Double {
        return  z.0 + Δt*z.1 + Δt*Δt*z.2/2.0 + Δt*Δt*Δt*z.3/6.0;
   }
   tXY.append((t:t0, y:y0, x:x0));
   for t in stride(from: t0, to: tn, by: Δt) {
        f1 = φ(t);                        f2 = ψ(t);
        Yi.y1 = f1.re  -  a*Xi.x;         Xi.x1 = f2.re  - b*Yi.y;
        Yi.y2 = f1.im1 -  a*Xi.x1;        Xi.x2 = f2.im1 - b*Yi.y1;
        Yi.y3 = f1.im2 -  a*Xi.x2;        Xi.x3 = f2.im2 - b*Yi.y2;
        Yi.y =  yx(z: Yi);                Xi.x  = yx(z: Xi)
        tXY.append ((t:t + Δt, y:Yi.y, x:Xi.x));
   }
    return tXY;
}
```

## Литература


1. Авдюшев В. А. Численные методы интегрирования обыкновенных дифференциальных уравнений. Томск: Изд-во ТГУ, 2009, 52с.
2. Naumann U. The art of differentiating computer programs. Society for industrial and applied mathematics, Philadelphia, USA, 2012.
3. Corliss G., Faure C., Griewank A., Hascoлt L., Naumann U. Automatic Differentiation Bibliography // Automatic Differentiation of Algorithms: From Simulation to Optimization. Springer, 2002. p. 383—425.
4. Fike J.A., 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. Диментберг Ф. М.  Теория винтов и ее приложения. – М.: Наука, 1978. – 328 с.
7. Двайт Г. Б.  Таблицы интегралов и другие математические формулы. - М.: Наука,1973. – 228 с.

**Аннотация**

В данной публикации рассматривается применение расширенных гипер-дуальных чисел в численном интегрировании уравнений задачи Коши с использованием разложения в ряды Тейлора и компьютерного автоматического дифференцирования. Вводится новый тип чисел – дуальные числа третьего класса. Описаны основные операции и базовые функции указанного класса дуальных чисел. Представлена процедура последовательного поточечного интегрирования с использованием автоматического дифференцирования и дуальных чисел третьего класса. Дана компьютерная реализация для языка SWIFT операционной системы macOS. Проведены численные эксперименты на базе полученного программного обеспечения.

**Ключевые слова:** Задача Коши, дуальные числа, гипер-дуальные числа, расширение гипер-дуальных чисел, автоматическое дифференцирование

---

*20 февраля 2020 г.*