ПРИМЕНЕНИЕ ГИПЕР-ДУАЛЬНЫХ ЧИСЕЛ ДЛЯ ЧИСЛЕННОГО РЕШЕНИЯ СИСТЕМ НЕЛИНЕЙНЫХ УРАВНЕНИЙ МЕТОДАМИ ЧЕБЫШЕВА-ХЭЛЛИ

Олифер В. И.

Одной из важнейших задач численного анализа является решение систем нелинейных уравнений $\vec{f}(\vec{x}) = 0$, где $\vec{f} \equiv \{f_k(\vec{x})\},\ \vec{x} \equiv \{x_k\},\ k = \overline{1,K}$. Общего аналитического решения таких систем уравнений не найдено. Поэтому часто используются различные итерационные методы. Грубо говоря, итерационный метод начинается с начального предположения $\vec{x}_0$ (называемого порной точкой), которое улучшается посредством итераций $\vec{x}_{i+1} = \vec{f}(\vec{x}_i),\ i = \overline{0,n}$.

Нахождение корней таких систем уравнений бывает целесообразно свести к нахождению минимума некоторого функционала $\varphi(\vec{x})$. Следует отметить интересный факт о том, что любая система уравнений над действительными числами может быть представлена одним равносильным уравнением, если взять каждое уравнение исходной системы возвести его в квадрат и сложить полученные результаты, т.е. $\varphi(\vec{x}) = \left\|\vec{f}(\vec{x})\right\|^2$. Тогда решение исходной системы уравнений сводится к задаче поиска минимума $\vec{x}^* = min(\left\|\vec{f}(\vec{x})\right\|^2)$, которая является задачей многомерной оптимизации. Из множества методов решения последней, возпользуемся идеями итерационного метода Гаусса-Зейделя безусловнной оптимизации (метод покоординатного спуска) [1]. Суть метода заключается в том, чтобы на каждой итерации по очереди минимизировать целевую функцию $\varphi(\vec{x})$ вдоль каждого из компонентов x_k вектора $\vec{x}$. Слабым местом этого метода являются медленная сходимость, обусловленная использованием только значений финкции в итерационных точках.

Для устранения названного недостатка воспользуемся двухшаговой итерационной процедурой одномерного случая, полученной на основе метода Чебышева-Хелли 5-го порядка сходимости [2], который учитывает первую, вторую и третью производные целевой функции:

■ Первый шаг:

$$x_{i+1} = x_i - \left[1 + \frac{1}{2}\cdot\frac{L(x_i)}{1 - \alpha L(x_i)}\right] D(x_i), \qquad \alpha \in \mathbb{R}, \qquad i = 0, 1, 2, \ldots \tag{1}$$

где: $L(x_i) = D(x_i)\cdot\varphi'''(x_i)/\varphi''(x_i),\ D(x_i) = \varphi'(x_i)/\varphi''(x_i)$ – ньютоновская корректировка.

■ Второй шаг:

$$\tilde{x}_{i+1} = x_{i+1} - \left[1 + \frac{M(x_i, x_{i+1})}{1 - \beta M(x_i, x_{i+1})}\right] D(x_i, x_{i+1}), \qquad \tilde{x}_{i+1} \Rightarrow x_{i+1}, \tag{2}$$

где: $M(x_i, x_{i+1}) = L(x_i)(1 - \varphi(x_{i+1})/\varphi(x_i))$, $D(x_i, x_{i+1}) = \varphi(x_{i+1})/\varphi'(x_i)$, $\beta \in \mathbb{R}$

Формула (1) первого шага суть – семейство методов Чебышева-Хэлли 3-го порядка сходимости, а формула (2) второго шага была получена на основе классического метода Ньютона с гипер-касательной в виде гиперболы. Величины α и β являются параметрами 1-го и 2-го шагов. При $\alpha = 0\,, 0.5,\ 1$ имеем формулы: метода Чебышева, метода Хэлли и супер-метода Хэлли соответственно. Параметр β входит в формулу, описывающую гипер-касательную (гиперболу).

В [2] показано, что (1) и (2) реализуют метода Чебышева-Хэлли 5-го порядка сходимости.

С учётом (1) и (2), $(i+1)$-я итерация для задачи $\vec{x}^* = min(\varphi(\vec{x}))$, будет состоять из K шагов, каждый из которых осуществляет дифференцирование по x_k в предположении, что $x_n \equiv const$ при $n \neq k$:

$$\begin{aligned} x_{k,i+1} &= x_{k,i} - \left[1 + \frac{1}{2} \cdot \frac{L_k(\vec{x}_{i,k})}{1 - \alpha L_k(\vec{x}_{i,k})}\right] D_k(\vec{x}_{i,k}), \\ \tilde{x}_{k,i+1} &= x_{k,i+1} - \left[1 + \frac{M_k(\vec{x}_{i,k}, \vec{x}_{i+1,k})}{1 - \beta M_k(\vec{x}_{i,k}, \vec{x}_{i+1,k})}\right] D_k(\vec{x}_{i,k}, \vec{x}_{i+1,k}), \qquad \tilde{x}_{k,i+1} \Rightarrow x_{k,i+1}, \end{aligned} \tag{3}$$

где:

$$\begin{aligned} &L_k(\vec{x}_{i,k}) = \frac{D(\vec{x}_{i,k}) \cdot \varphi_k'''(\vec{x}_{i,k})}{\varphi_k''(\vec{x}_{i,k})}, \quad D_k(\vec{x}_{i,k}) = \frac{\varphi_k'(\vec{x}_{i,k})}{\varphi_k''(\vec{x}_{i,k})}, \\ &M_k(\vec{x}_{i,k}, \vec{x}_{i+1,k}) = L_k(\vec{x}_{i,k})\left[1 - \frac{\varphi(\vec{x}_{i+1,k})}{\varphi(\vec{x}_{i,k})}\right], \ D_k(\vec{x}_{i,k}, \vec{x}_{i+1,k}) = \frac{\varphi(\vec{x}_{i+1,k})}{\varphi'(\vec{x}_{i,k})} \end{aligned} \tag{4}$$

$\vec{x}_{i,k}$ – арифметический вектор, содержащий все самые последние обновления координат экстремума функции $\varphi(\vec{x})$.

Для наглядности, пусть $\vec{x} = [x, y, z]$, тогда $(\varphi(\vec{x}) = f_1^2(x, y, z) + f_2^2(x, y, z) + f_3^2(x, y, z))$

$$k = 1: \begin{cases} x_{i+1} = x_i - \left[1 + \frac{1}{2} \cdot \frac{L_x}{1 - \alpha L_x}\right] D_x, \\ \tilde{x}_{i+1} = x_{i+1} - \left[1 + \frac{M_x}{1 - \beta M_x}\right] Q_x, \qquad \tilde{x}_{i+1} \Rightarrow x_{i+1} \end{cases} \tag{5}$$

$$k=2:\begin{cases} y_{i+1}=y_i-\left[1+\frac{1}{2}\cdot\frac{L_y}{1-\alpha L_y}\right]D_y, \\ \tilde{y}_{i+1}=y_{i+1}-\left[1+\frac{M_y}{1-\beta M_y}\right]Q_y, \qquad \tilde{y}_{i+1}\Rightarrow y_{i+1} \end{cases}$$

$$k=3:\begin{cases} z_{i+1}=y_i-\left[1+\frac{1}{2}\cdot\frac{L_z}{1-\alpha L_z}\right]D_z, \\ \tilde{z}_{i+1}=y_{i+1}-\left[1+\frac{M_z}{1-\beta M_z}\right]Q_z, \qquad \tilde{z}_{i+1}\Rightarrow z_{i+1} \end{cases}$$

где:

$$\begin{aligned} D_x &= \frac{\varphi'_x(x_i,y_i,z_i)}{\varphi''_x(x_i,y_i,z_i)}, & L_x &= \frac{D_x\cdot\varphi'''_x(x_i,y_i,z_i)}{\varphi''_x(x_i,y_i,z_i)}, \\ D_y &= \frac{\varphi'_y(x_{i+1},y_i,z_i)}{\varphi''_y(x_{i+1},y_i,z_i)}, & L_y &= \frac{D_y\cdot\varphi'''_y(x_{i+1},y_i,z_i)}{\varphi''_y(x_{i+1},y_i,z_i)}, \\ D_z &= \frac{\varphi'_z(x_{i+1},y_{i+1},z_i)}{\varphi''_z(x_{i+1},y_{i+1},z_i)}, & L_z &= \frac{D_z\cdot\varphi'''_z(x_{i+1},y_{i+1},z_i)}{\varphi''_z(x_{i+1},y_{i+1},z_i)}, \\ Q_x &= \frac{\varphi'_x(x_{i+1},y_i,z_i)}{\varphi''_x(x_i,y_i,z_i)}, & M_x &= L_x\cdot\left(1-\frac{\varphi'_x(x_{i+1},y_i,z_i)}{\varphi'_x(x_i,y_i,z_i)}\right), \\ Q_y &= \frac{\varphi'_y(x_{i+1},y_{i+1},z_i)}{\varphi''_y(x_{i+1},y_i,z_i)}, & M_y &= L_y\cdot\left(1-\frac{\varphi'_y(x_{i+1},y_{i+1},z_i)}{\varphi'_y(x_i,y_i,z_i)}\right), \\ Q_z &= \frac{\varphi'_z(x_{i+1},y_{i+1},z_{i+1})}{\varphi''_z(x_{i+1},y_{i+1},,z_i)}, & M_z &= L_z\cdot\left(1-\frac{\varphi'_z(x_{i+1},y_{i+1},z_{i+1})}{\varphi'_z(x_i,y_i,z_i)}\right) \end{aligned} \qquad (6)$$

Признаком окончания такой итерационной процедуры служит выполнение следующих неравенств: $\varphi'_x(x_i,y_i,z_i)\le\delta$, $\varphi'_y(x_i,y_i,z_i)\le\delta$ и $\varphi'_z(x_i,y_i,z_i)\le\delta$ или $f_1(x_i,y_i,z_i)\le\delta$, $f_2(x_i,y_i,z_i)\le\delta$, и $f_3(x_i,y_i,z_i)\le\delta$, где δ – заданная малая величина.

Как видно, для решения оптимизационной задачи $\vec{x}^*=min(\varphi(\vec{x}))$, предлогаемым итерационным методом (3-4), необходимо наличие простого и точного способа определения значений частных производных целевой функции (особенно, когда целевая функция является сложной композицией базовых функций).

При компьютерной реализации рассмотренной выше итерационной процедуры, вопрос о простом и точном определении значений первых трех производных целевой функции может быть успешно решен, если воспользоваться методом автоматического дифференцирования (АД) [3], суть которого заключается в создании нового типа данных и переопределения операций и базовых функций над ними. Для рассматриваемого случая в

качестве такого нового типа данных могут быть исползованы гипер-дуальные числа 3-го класса [3].

Гипер-дуальное число 3-го класса имеет представление в виде $Z = z + z_1\boldsymbol{\varepsilon} + z_2\boldsymbol{\omega} + z_3\boldsymbol{\gamma}$, $z,\ z_1,\ z_2,\ z_3 \in \mathbb{R}$. Параметр z называется главной (*Re* - действительной) частью гипер-дуального числа, а z_1, z_2, z_3 – его мнимыми (*Im1*, *Im2*, *Im3* - инфинитезимальными) частями. Абстрактные элементы $\boldsymbol{\varepsilon},\ \boldsymbol{\omega},\ \boldsymbol{\gamma}$ образуют базис мнимых частей гипер-дуального числа, отвечающий следующими правилам: $\boldsymbol{\varepsilon^2} = 2\boldsymbol{\omega},\ \boldsymbol{\varepsilon\omega} = 3\boldsymbol{\gamma},\ \boldsymbol{\varepsilon\gamma} = \boldsymbol{\omega\gamma} = \boldsymbol{\gamma^2} = \boldsymbol{\omega^2} = 0$.

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

$$
\begin{aligned}
&X = x + x_1\boldsymbol{\varepsilon} + x_2\boldsymbol{\omega} + x_3\boldsymbol{\gamma}, \qquad Y = y + y_1\boldsymbol{\varepsilon} + y_2\boldsymbol{\omega} + y_3\boldsymbol{\gamma},\\
&X + Y = x + y + (x_1 + y_1)\boldsymbol{\varepsilon} + (x_2 + y_2)\boldsymbol{\omega} + (x_3 + y_3)\boldsymbol{\gamma},\\
&X\cdot Y = x\cdot y + (x\cdot y_1 + y\cdot x_1)\boldsymbol{\varepsilon} + (x\cdot y_2 + 2x_1\cdot y_1 + y\cdot x_2)\boldsymbol{\omega} + (x\cdot y_3 + y\cdot x_3 + 3(x_1\cdot y_2\\
&\qquad + y_1\cdot x_2))\boldsymbol{\gamma},\\
&Y^{-1} = y^{-1} - y^{-2}\cdot y_1\boldsymbol{\varepsilon} + y^{-2}(2\cdot y_1^2\cdot y^{-1} - y_2)\boldsymbol{\omega} + a^{-2}(6y_1\cdot y^{-1}(y_2 - y_1^2\cdot y^{-1})\\
&\qquad - y_3)\boldsymbol{\gamma},\\
&X/Y = X\cdot Y^{-1},
\end{aligned}
\tag{7}
$$

а функция гипер-дуального аргумента 3-го класса, после ее разложения в ряд Тейлора, имеет вид

$$
\begin{aligned}
F(X) = f(x) + x_1 f'(x)\boldsymbol{\varepsilon} + [x_2 f'(x) + x_1^2 f''(x)]\boldsymbol{\omega} +\\
+ [x_3 f'(x) + 3x_1x_2 f''(x) + x_1^3 f'''(x)]\boldsymbol{\gamma}
\end{aligned}
\tag{8}
$$

При $x_1 = 1$ и $x_2 = x_3 = 0$ соотношение (8) принимает форму:

$$
F(x + \boldsymbol{\varepsilon} + 0\boldsymbol{\omega} + 0\boldsymbol{\gamma}) = f(x) + f'\boldsymbol{\varepsilon} + f''(x)\boldsymbol{\omega} + f'''(t)\boldsymbol{\gamma}, \quad f(x) = F(x + 0\boldsymbol{\varepsilon} + 0\boldsymbol{\omega} + 0\boldsymbol{\gamma}),
$$

что позволяет представить (4) следующим образом

$$
\begin{aligned}
&L_k(\vec{x}_{i,k}) = \frac{D(\vec{x}_{i,k})\cdot\Phi(\vec{X}_{i,k}).Im3}{\Phi(\vec{X}_{i,k}).Im2}, \quad D_k(\vec{x}_{i,k}) = \frac{\Phi(\vec{X}_{i,k}).Im1}{\Phi(\vec{X}_{i,k}).Im2},\\
&M_k(\vec{x}_{i,k}, \vec{x}_{i+1,k}) = L_k(\vec{x}_{i,k})[1 - \frac{\Phi(\vec{X}_{i+1,k}).Re}{\Phi(\vec{X}_{i,k}).Re}], \quad D_k(\vec{x}_{i,k}, \vec{x}_{i+1,k}) = \frac{\Phi(\vec{X}_{i+1,k}).Re}{\Phi(\vec{X}_{i+1,k}).Im1},
\end{aligned}
\tag{9}
$$

где: $\Phi(\vec{X})$ – целевая функция $\varphi(\vec{x})$, записанная в терминах гипер-дуальных чисел 3-го класса, $\vec{X}_{i,k} = \{x_{l,k} + \delta_k^l\ \boldsymbol{\varepsilon} + 0\boldsymbol{\omega} + 0\boldsymbol{\gamma}\}_i$, $l -$ индекс компонента вектора $\vec{x}$, $k -$ номер шага в i-й итерации, $i -$ номер итерации и $\delta_k^l = \begin{cases}1, & l = k\\ 0, & l \neq k\end{cases}$.

Компьютерная реализация гипер-дуальных чисел 3-го класса (Sdn3) была выполнена на языке программирования SWIFT 5 для macOS 13.3 в виде статической библиотеки (static library) SdnLibrary3, которую можно скачать с интернет ресурса (см. Приложение 1), а затем добавить в свой Xcode проект.

Для выполнения численных экспериментов по формулам (3) и (9) в Приложение 1 приводится код итерационной процедуры EXTR_N(...), реализирующей названные формулы. Там же даны процедуры, описывающие некоторые из тестируемых функций.

С целью проверки адекватности работы прикладный программы EXTR_N(...) была проведена верификация вычислительных алгоритмов на тестовых задачах, приведенных в табл 1. Расчеты проводились в операционной среде MacOS Ventura 13.5.2 с CPU 2.3 GHz.

#	$\vec{f}(\vec{x}) = 0$	$\vec{x}_0$	$\vec{x}^*$	$\vec{f}(\vec{x}^*)$	*i*	*мс*
1	$\begin{cases} sin(x-0.5)-y-1.5 \\ 2x-\cos(y)-0.6 \end{cases}$	$\begin{bmatrix} 0.13 \\ -1.80 \end{bmatrix}$	$\begin{bmatrix} 0.1787450 \\ -1.8157532 \end{bmatrix}$	$\begin{bmatrix} -4\cdot10^{-6} \\ 4\cdot10^{-6} \end{bmatrix}$	55	0.72
2	$\begin{cases} x^2-2y^2-xy+2x-y+1 \\ 2x^2-y^2+xy+3y-5 \end{cases}$	$\begin{bmatrix} 0.50 \\ 0.50 \end{bmatrix}$	$\begin{bmatrix} 1.0000000 \\ 1.0000000 \end{bmatrix}$	$\begin{bmatrix} 8\cdot10^{-8} \\ 2\cdot10^{-7} \end{bmatrix}$	6	0.25
3	$\begin{cases} (x-2y)(2x-y+1)-6 \\ 0.5x-(7/3)y+1 \end{cases}$	$\begin{bmatrix} -1.00 \\ 1.00 \end{bmatrix}$	$\begin{bmatrix} -1.9999977 \\ 1.8121007\cdot10^{-6} \end{bmatrix}$	$\begin{bmatrix} -9\cdot10^{-7} \\ -3\cdot10^{-6} \end{bmatrix}$	280	3.1
4	$\begin{cases} (x+y)^2+xy-5(x+y)-15 \\ (x+y)^2-xy-(x+y)-1 \end{cases}$	$\begin{bmatrix} 1.10 \\ 1.00 \end{bmatrix}$	$\begin{bmatrix} 1.0000000 \\ 0.9999999 \end{bmatrix}$	$\begin{bmatrix} 1\cdot10^{-8} \\ 1\cdot10^{-8} \end{bmatrix}$	1	0.2
5	$\begin{cases} x^3-y^3-27 \\ x^2-y^2-9 \end{cases}$	$\begin{bmatrix} 2.90 \\ 0.10 \end{bmatrix}$	$\begin{bmatrix} 3.0000000 \\ 0.0003807 \end{bmatrix}$	$\begin{bmatrix} -2\cdot10^{-7} \\ 1\cdot10^{-7} \end{bmatrix}$	6	0.3
6	$\begin{cases} 2x^2-xy-5x-y+1 \\ x-y^2-ln(x) \end{cases}$	$\begin{bmatrix} 2.00 \\ 2.00 \end{bmatrix}$	$\begin{bmatrix} 2.1452763 \\ -1.1755881 \end{bmatrix}$	$\begin{bmatrix} 5\cdot10^{-7} \\ 4\cdot10^{-7} \end{bmatrix}$	15	0.4
7	$\begin{cases} x^2+x-2yz-0.1 \\ y^2-y-3xz-0.2 \\ z^2+z+2xy-0.3 \end{cases}$	$\begin{bmatrix} 0.00 \\ 0.00 \\ 0.00 \end{bmatrix}$	$\begin{bmatrix} 0.0128239 \\ -0.1778006 \\ 0.2446880 \end{bmatrix}$	$\begin{bmatrix} -3\cdot10^{-7} \\ 1\cdot10^{-7} \\ 7\cdot10^{-8} \end{bmatrix}$	6	0.3
8	$\begin{cases} x^2+y^2+z^2-1 \\ 2x^2+y^2-4z \\ 3x^2-4y+z^2 \end{cases}$	$\begin{bmatrix} 0.50 \\ 0.50 \\ 0.50 \end{bmatrix}$	$\begin{bmatrix} 0.7851974 \\ 0.4966119 \\ 0.3699232 \end{bmatrix}$	$\begin{bmatrix} 2\cdot10^{-6} \\ 4\cdot10^{-7} \\ 2\cdot10^{-7} \end{bmatrix}$	39	0.9
9	$\begin{cases} (3-x)^3-y-2 \\ (2z-y)\cdot(y+2)-4y-9 \\ x^2+z^2-4x \end{cases}$	$\begin{bmatrix} 1.00 \\ 1.00 \\ 1.00 \end{bmatrix}$	$\begin{bmatrix} 2.5000000 \\ 16.1023254 \\ 11.1023254 \end{bmatrix}$	$\begin{bmatrix} -6\cdot10^{-7} \\ -2\cdot10^{-7} \\ 7\cdot10^{-7} \end{bmatrix}$	527	11.2
10	$\begin{cases} x+y+z-6 \\ xyz-6 \\ z^2-9 \end{cases}$	$\begin{bmatrix} 1.00 \\ 1.00 \\ 1.00 \end{bmatrix}$	$\begin{bmatrix} 1.9999795 \\ 1.0000104 \\ 3.0000001 \end{bmatrix}$	$\begin{bmatrix} -6\cdot10^{-7} \\ -2\cdot10^{-7} \\ 7\cdot10^{-7} \end{bmatrix}$	423	8.1

11	$\begin{cases} c_x \cdot p_1 - s_x \cdot q_1 + s_y \cdot s_z - 1 \\ s_x \cdot p_2 + c_x \cdot q_2 - c_z \cdot s_y - 1 \\ s_y \cdot (s_x + c_x) - 1 \end{cases}$ где: $c_u = cos(u)\,,\ s_u = sin(u),$ $p_1 = c_z + s_z \cdot c_y,\ \ q_1 = c_y \cdot s_z + c_z,$ $p_2 = c_y \cdot c_z - s_z,\ \ q_2 = c_y \cdot c_z - s_z$	$\begin{bmatrix} 1.00 \\ 1.00 \\ 1.00 \end{bmatrix}$	$\begin{bmatrix} -0.5488752 \\ 0.6399783 \\ 1.0219130 \end{bmatrix}$	$\begin{bmatrix} -2 \cdot 10^{-6} \\ -2 \cdot 10^{-6} \\ 4 \cdot 10^{-6} \end{bmatrix}$	18	0.6

*Таблица 1.Результаты тестируемых задач при α=3/4, β=1, $\delta = 10^{-5}$, где: $\vec{x}_0$, $\vec{x}^*$, $\vec{f}(\vec{x}^*)$, **i, мс** – начальное приближение, полученные корни, значение правой части исходной системы при найденных корнях, кол-во итеракций, затраченное время в миллисекундах соответственно.*

Любая система уравнений может не иметь корней, либо иметь один или более кореней. Итерационный процесс (если он сходится) дает только один корень, зависящий от начального приближения и величин α, β. Для поиска других корней используются различные методы отделения корней, которые тут не рассматриваются.

На примере системы #5 из табл. 1, в табл. 2 показано влияние величин α и β на количество итераций.

β \ α	0	1/4	1/2	3/4	1
0	8	8	7	7	6
1/4	8	8	7	7	6
1/2	8	7	7	7	6
3/4	7	7	7	7	6
1	7	7	7	6	6

Таблица 2. Влияние α и β на количество итераций для системы уравнений #5.

Приложение 1.

Код для численного эксперимента на языке Swift 5 (macOS 13.3). Тип данных Sdn3 (супер-дуальные числа 3-го класса) приведен в [3], или можно скачать статическую библиотеку SdnLibrary3.zip и после распаковки добавить SdnLibrary3 в свой проект (как это сделать см. ReadMe.txt). Download SdnLibrary3.zip

```
import Foundation;
import SdnLibrary3;   // add this one if you use static library SdnLibrary3

let δ:Double = 1E-5;
let Nmax = 1000; // max number of iterations


// INPUT DATA:
// Fn –function pointers array, xo –start points array, α, β – parameters of first and second steps
// OUTPUT DATA:
// (x, f, extr, i, ms) – extremum points array, function values array, number of iterations, milliseconds


func EXTR_N(Fn:[([Sdn3])->Sdn3], x0:[Double], α:Double, β:Double)->
        (x:[Double],f:[Double],i:Int, ms:Double){
   var i = 0, D = 0.0,L = 0.0, M = 0.0, F = Sdn3(), Φ = Sdn3();
   let N = x0.count - 1;
   var X = Array(repeating: Sdn3(), count: N + 1);
   X = x0.map{Sdn3(re: $0)};
//--- Nested functions:
   func F2(X:[Sdn3])->Sdn3{
      var R = Sdn3();
      for k in 0...N {R += Fn[k](X)**2}
      return R;
   }

   func CH(k:Int){
      F = F2(X:X);
      D = F.im1/F.im2;  L = D*(F.im3/F.im2);
      if F.im2 == 0 {return}
      X[k].re -= (1.0 + 0.5*L/(1.0 - α*L))*D;
      Φ = F2(X:X);
      if F.im1 == 0 {return}
      D = Φ.im1/F.im2;  M = L*(1.0 - Φ.im1/F.im1);
      X[k].re -= (1.0 + M/(1.0 - β*M))*D;
   }
```

```
    func isNotEnd()->Bool{
        for k in 0...N {
            X[k].im1 = 1;
            if abs(F2(X:X).im1) >= δ {return true}
            X[k].im1 = 0; X[k].im2 = 0; X[k].im3 = 0;
        }
        return false;
    }
//--- End of Nested functions
    let start = CFAbsoluteTimeGetCurrent();
    repeat{
        for k in 0...N {
            X[k].im1 = 1;
            CH(k: k);
            X[k].im1 = 0; X[k].im2 = 0; X[k].im3 = 0;
        }
        i += 1; if i > Nmax {break}
    }while(isNotEnd());
    let end = CFAbsoluteTimeGetCurrent() - start;
    var f = Array(repeating: 0.0, count: N + 1),        x = Array(repeating: 0.0, count: N + 1);
    for k in 0...N {f[k] = Fn[k](X).re;  x[k] = X[k].re}
    return (x,f,i,end*1000);
}


//----- EXTR_N(…) FUNCTION CALL EXAMPLE

let R = EXTR_N(Fn: [FS11_1, FS11_2, FS11_3], x0: [1, 1, 1], α: 3/4, β: 1);


//----- SOME FUNCTIONS UNDER STUDY FOR EXTR_N(…)

//-------------FS6-----------
func FS6_1(X:[Sdn3])->Sdn3{
    return 2.0*X[0]**2 - X[0]*X[1] - 5.0*X[0] - Sdn3(re: 1.0);
}
func FS6_2(X:[Sdn3])->Sdn3{
    return X[0] - X[1]**2 - Sdn3.ln(X: X[0]);
}
// x0: [2, 2], α: 3/4, β: 1, δ: = 1E-5
// x: [2.1452763, -1.1755881],
// f: [4.6855676e-07, 4.2752374e-07],
//i: 15,
//ms: 0.4
//-----------End FS6-----------

//-------------FS7---------------
```

```
func FS7_1(X:[Sdn3])->Sdn3{
   return X[0]**2 + X[0] - 2.0*X[1]*X[2] - Sdn3(re: 0.1);
}
func FS7_2(X:[Sdn3])->Sdn3{
   return X[1]**2 - X[1] - 3.0*X[0]*X[2] - Sdn3(re: 0.2);
}
func FS7_3(X:[Sdn3])->Sdn3{
   return X[2]**2 + X[2] + 2.0*X[0]*X[1] - Sdn3(re: 0.3);
}
// x0: [0, 0, 0], α: 3/4, β: 1, δ: = 1E-5
// x: [0.0128239, -0.1778006, 0.2446880],
// f: [-2.8401678e-07, 1.1615888e-07, 7.0811900e-08],
// i: 6,
// ms: 0.3
//-------------End FS7---------------

//---------------FS11------------------
func FS11_1(X:[Sdn3])->Sdn3{
   let cz = Sdn3.cos(X: X[2]), sz =  Sdn3.sin(X: X[2]),
       cy = Sdn3.cos(X: X[1]), sy =  Sdn3.sin(X: X[1]),
       cx = Sdn3.cos(X: X[0]), sx =  Sdn3.sin(X: X[0]);
   let p = cz - sz*cy, q = cy*sz + cz;
   return cx*p - sx*q + sy*sz - Sdn3(re: 1.0);
}
func FS11_2(X:[Sdn3])->Sdn3{
   let cz = Sdn3.cos(X: X[2]), sz =  Sdn3.sin(X: X[2]),
       cy = Sdn3.cos(X: X[1]), sy =  Sdn3.sin(X: X[1]),
       cx = Sdn3.cos(X: X[0]), sx =  Sdn3.sin(X: X[0]);
   let p = cy*cz - sz, q = cy*cz + sz;
   return sx*p + cx*q - sy*cz - Sdn3(re: 1.0);
}
func FS11_3(X:[Sdn3])->Sdn3{
   let cy = Sdn3.cos(X: X[1]), sy =  Sdn3.sin(X: X[1]),
        cx = Sdn3.cos(X: X[0]), sx =  Sdn3.sin(X: X[0]);
   return sy*(sx + cx) + cy - Sdn3(re: 1.0);
}
// x0: [1, 1, 1], α: 3/4, β: 1, δ: = 1E-5
// (x: [-0.5488752, 0.6399783, 1.0219130],
// f: [-1.9e-06, -1.9e-06, 3.9e-06],
//  i: 18,
// ms: 0.6
//------------End FS11---------------
```

ЛИТЕРАТУРА

1. *Гилл Ф., Мюррей У., Райт М.* Практическая оптимизация. Пер. с англ. – М.: Мир, 1985.

2. *Олифер В. И.* Реализация методов типа Чебышева-Хэлли 5-го порядка сходимости на основе гипер-дуальных чисел.– URL:http://viosolutions.amerihomesrealty.com/pdf/Реализация_методов_ Чебышева- Хэлли_5-го_порядка.pdf, 2023.

3. *Олифер В. И.* Автоматическое дифференцирование на основе супер-дуальных чисел. – URL:http://viosolutions.amerihomesrealty.com/pdf/Автоматическое_дифференцирование_ на_основе_супер-дуальных_чисел.pdf, 2020.

Абстракт

В данной публикации рассматривается метод численной многомерной оптимизации, основанный на итерационных формулах Гаусса-Зейделя, Чебышева-Хелли, и автоматического дифференцирования с использованием специальных дуальных чисел (супер-дуальных чисел 3-го класса). Представлена компьютерная реализация этого подхода на языке SWIFT операционной системы macOS. Проведены численные эксперименты. Показана возможность сведения задачи нахождения корней системы нелинейных уравнений к поиску экстремума соответствующего функционала.

Ключевые слова: *итерационные методы оптимизации, автоматическое дифференцирование, супер-дуальные числа, корни системы нелинейных уравнений, iterative optimization methods, automatic differentiation, super-dual numbers, roots of a system of nonlinear equations.*

20 декабря 2023 г