ВЫЧИСЛЕНИЕ ЭКСТРЕМУМА ФУНКЦИЙ МНОГИХ ПЕРЕМЕННЫХ НА ОСНОВЕ ГИПЕР-ДУАЛЬНЫХ ЧИСЕЛ

Олифер В. И.

Большинство методов поиска экстремума функций действительных переменных приспособлено только для тех функций, у которых каждый локальный экстремум является одновременно и глобальным. Этим свойством обладают унимодальные функции. Именно такие функции рассматриваются далее.

Задачу безусловной минимизации или максимизации $(extr)$ скалярного поля размерностью K, описываемого функцией $f(\vec{x})$, где $\vec{x} = \{x_k\} \in \mathbb{R}^K$, $(k = 1, 2, \dots, K)$ – арифметический вектор, можно записать в виде:

$$\vec{x}^* = extr(f(\vec{x})), \;\; \vec{x}^* \in \mathbb{R}^K \qquad (1)$$

Необходимые и достаточные условия оптимальности функции $f(\vec{x})$ служат следующие соотношения:

$$f_k'(\vec{x}) = 0, \qquad f_k''(\vec{x}) > 0 \;\; - min,$$

$$f_k'(\vec{x}) = 0, \qquad f_k''(\vec{x}) < 0 \;\; - max,$$

$$f_k'(\vec{x}) = 0, \;\; sign(f_k''(\vec{x})) \neq sign\left(f_n''(\vec{x})\right) \;\; - min/max,$$

где: $f_k'(\vec{x})$ и $f_k''(\vec{x})$ – операции дифференцирования по компоненте x_k.

Для поиска экстремуму функции $f(\vec{x})$ будем использовать идеи итерационного метода Гаусса-Зейделя безусловнной оптимизации (метод покоординатного спуска) [1]. Суть метода заключается в том, чтобы на каждой итерации по очереди минимизировать целевую функцию $f(\vec{x})$ 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 итерационной формулой одномерного случая [2], полученной на основе метода Чебышева-Хелли, который учитывает первую, вторую и третью производные целевой функции:

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

где:

$L(x_i) = \frac{f'(x_i)f'''(x_i)}{f''(x_i)^2} = D(x_i)\cdot f^{'''}(x_i)/f^{''}(x_i)$, $D(x_i) = f'(x_i)/f''(x_i)$ и $i -$ номер итерации.

Тогда $(i+1)$-я итерация для задачи (1) будет состоять из K шагов, каждый из которых осуществляет дифференцирование по x_k в предположении, что $x_n \equiv const$ при $n \neq k$:

$$x_{k,i+1} = x_{k,i} - \left[1 + \frac{1}{2}\cdot\frac{L_k(\vec{x}_{i,k})}{1 - \alpha L_k(\vec{x}_{i,k})}\right] D_k(\vec{x}_{i,k}), \tag{3}$$

где:

$$L_k(\vec{x}_{i,k}) = \frac{D(\vec{x}_{i,k})\cdot f_k'''(\vec{x}_{i,k})}{f_k''(\vec{x}_{i,k})} \in \mathbb{R}, \quad D_k(\vec{x}_{i,k}) = \frac{f_k'(\vec{x}_{i,k})}{f_k''(\vec{x}_{i,k})} \in \mathbb{R}, \tag{4}$$

$\vec{x}_{i,k} -$ арифметический вектор, содержащий все самые последние обновления координат экстремума функции $f(\vec{x})$.
Для наглядности, пусть $\vec{x} = [x, y, z]$, тогда

$$\begin{aligned}
x_{i+1} &= x_i - \left[1 + \frac{1}{2}\cdot\frac{L_x(x_i, y_i, z_i)}{1 - \alpha L_x(x_i, y_i, z_i)}\right] D_x(x_i, y_i, z_i),\\
y_{i+1} &= y_i - \left[1 + \frac{1}{2}\cdot\frac{L_y(x_{i+1}, y_i, z_i)}{1 - \alpha L_y(x_{i+1}, y_i, z_i)}\right] D_y(x_{i+1}, y_i, z_i),\\
z_{i+1} &= z_i - \left[1 + \frac{1}{2}\cdot\frac{L_z(x_{i+1}, y_{i+1}, z_i)}{1 - \alpha L_z(x_{i+1}, y_{i+1}, z_i)}\right] D_z(x_{i+1}, y_{i+1}, z_i),
\end{aligned} \tag{5}$$

где:

$$\begin{aligned}
D_x(x,y,z) &= \frac{f_x'(x,y,z)}{f_x''(x,y,z)}, \qquad L_x(x,y,z) = \frac{D_x(x,y,z)\cdot f_x'''(x,y,z)}{f_x''(x,y,z)},\\
D_y(x,y,z) &= \frac{f_y'(x,y,z)}{f_y''(x,y,z)}, \qquad L_y(x,y,z) = \frac{D_y(x,y,z)\cdot f_y'''(x,y,z)}{f_y''(x,y,z)},\\
D_z(x,y,z) &= \frac{f_z'(x,y,z)}{f_z''(x,y,z)}, \qquad L_z(x,y,z) = \frac{D_z(x,y,z)\cdot f_z'''(x,y,z)}{f_z''(x,y,z)}
\end{aligned} \tag{6}$$

Признаком окончания такой итерационной процедуры служит выполнение следующих неравенств: $f_x'(x_i, y_i, z_i) \le \delta$, $f_y'(x_i, y_i, z_i) \le \delta$ и $f_z'(x_i, y_i, z_i) \le \delta$ где δ – заданная малая величина.

Как видно, для решения оптимизационной задачи (1), предлогаемым итерационным методом (3), необходимо наличие простого и точного способа определения значений

первых трех производных целевой функции (особенно, когда целевая функция является сложной композицией базовых функций).

При компьютерной реализации итерационной процедуры (3), 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 над ними. Для рассматриваемого случая в качестве такого нового типа данных могут быть исползованы гипер-дуальные числа 3-го класса [3].

Гипер-дуальные число 3-го класса имеет представление в виде $Z = z + z_1\boldsymbol{\varepsilon} + z_2\boldsymbol{\omega} + z_3\boldsymbol{\gamma}$, $z,\ z_1,\ z_2,\ z_3 \in \mathbb{R}$. Число 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

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

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

$$D_k(\vec{x}_{i,k}) = \frac{F\left(\vec{X}_{i,k}\right).Im1}{F\left(\vec{X}_{i,k}\right).Im2}, \quad L_k(\vec{x}_{i,k}) = \frac{D_k(\vec{x}_{i,k}){\cdot}F_k'''\left(\vec{X}_{i,k}\right).Im3}{F_k''\left(\vec{X}_{i,k}\right).Im2}, \tag{9}$$

где: $\vec{X}_{i,k} = \left\{x_{l,k} + \delta_k^l\,\boldsymbol{\varepsilon} + 0\boldsymbol{\omega} + 0\boldsymbol{\gamma}\right\}_i$, $l -$ индекс компонента вектора $\vec{x}$, $k -$ номер шага в i-й итерации, $i -$ номер итерации и $\delta_k^l = \begin{cases}1, & l = k\\ 0, & l \neq k\end{cases}$.

А для функции трех переменных (6) получим

$$D_x(x,y,z) = \frac{F(X,\hat{Y},\hat{Z}).Im1}{F(X,\hat{Y},\hat{Z}).Im2},\quad L_x(x,y,z) = \frac{D_x(x,y,z)\cdot F(X,\hat{Y},\hat{Z}).Im3}{F(X,\hat{Y},\hat{Z}).Im2},$$

$$D_y(x,y,z) = \frac{F(\hat{X},Y,\hat{Z}).Im1}{F(\hat{X},Y,\hat{Z}).Im2},\quad L_y(x,y,z) = \frac{D_y(x,y,z)\cdot F(\hat{X},Y,\hat{Z}).Im3}{F(\hat{X},Y,\hat{Z}).Im2}, \qquad (10)$$

$$D_z(x,y,z) = \frac{F(\hat{X},\hat{Y},Z).Im1}{F(\hat{X},\hat{Y},Z).Im2},\quad L_z(x,y,z) = \frac{D_z(x,y,z)\cdot F(\hat{X},\hat{Y},Z).Im3}{F(\hat{X},\hat{Y},Z).Im2}$$

где:

$$X = x + \boldsymbol{\varepsilon} + 0\boldsymbol{\omega} + 0\boldsymbol{\gamma},\quad Y = y + \boldsymbol{\varepsilon} + 0\boldsymbol{\omega} + 0\boldsymbol{\gamma},\quad Z = z + \boldsymbol{\varepsilon} + 0\boldsymbol{\omega} + 0\boldsymbol{\gamma},$$

$$\hat{X} = x + 0\boldsymbol{\varepsilon} + 0\boldsymbol{\omega} + 0\boldsymbol{\gamma},\quad \hat{Y} = y + 0\boldsymbol{\varepsilon} + 0\boldsymbol{\omega} + 0\boldsymbol{\gamma},\quad \hat{Z} = z + 0\boldsymbol{\varepsilon} + 0\boldsymbol{\omega} + 0\boldsymbol{\gamma}$$

Следуя духу экспериментальной математики [5] – любой абстрактный математический формализм требует экспериментальной проверки, была выполнена компьютерная реализация гипер-дуальных чисел 3-го класса (Sdn3) 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)-(5) и (9) в Приложение 1 приводится код итерационной процедуры MinMax_N(...), реализирующей названные формулы. Там же даны процедуры, описывающие некоторые из тестируемых функций.

С целью проверки адекватности работы прикладный программы MinMax_N(...) была проведена верификация вычислительных алгоритмов на основе сравнения численных и аналитических решений, полученных для тестовых задач. Расчеты проводились в операционной среде MacOS Ventura 13.5.2 с CPU 2.3 GHz.

В таблице 1 представлены результаты тестируемых функций при K=3 и $\alpha = 3/4$.

$\boldsymbol{f(\vec{x}) = f(x,y,z)}$	$\boldsymbol{\vec{x}_0}$	$\boldsymbol{\vec{x}_*/f(\vec{x}_*)}$	***Min/Max***	***i***	***t(мс)***
1	2	3	4	5	6
$f_1(\vec{x}) = x^2 + y^2 + z^2$	[1, 1, 1]	[0.0000000000, 0.0000000000, 0.0000000000] 0.0000000000	*min*	2	0.39

$f_2(\vec{x}) = xy + 2yz + 11x + 2y + + 18z + 10 - x^2 - 5y^2 - 3z^2 - - 2xz$	[1, 1, 1]	[4.0000000000, 1.0000000000, 2.0000000000] 51.0000000000	*max*	30	0.51
$f_3(\vec{x}) = 2z^2 + 3xz + 8y + z + 4 - - 2x^2 - y^2 - 2xy - 2yz$	[1, 1, 1]	[-0.3513513514, 2.8918918919, 1.4594594595] 16.2972972973	*min/max*	72	1.4
$f_4(\vec{x}) = x + \frac{y^2}{4x} + \frac{z^2}{y} + 2$	[1, 2, 2]	[0.5000000000, 1. 0000000000, 1. 0000000000] 4. 0000000000	*min*	57	0.55
$f_5(\vec{x}) = x^2 + y^2 - 3x + 2y - z^3$	[1, 1, 1]	[1.5000000000, -1.0000000000, 0.0000000000] -3.2500000000	*min/max*	15	0.49
$f_6(\vec{x}) = xy^2 + z^2 - xyz$	[1, 1, 1]	[1.0000000000, 0.0000000000, 0.0000000000] 0.0000000000	*min/max*	26	0.47
$f_7(\vec{x}) = z^3 - x^2 - 3y^2 - 1.5z - - 4x + 6y + 2$	[1, 2, 2]	[-2.0000000000, 1.0000000000, 0.7071067812] 8.2928932188	*min/max*	5	0.38
$f_8(\vec{x}) = (2x - 3z)^2 + + 2(x + y + z)^2 + 3(y - z)^2$	[1, 2, 2]	[0.0000000000, 0.0000000000, 0.0000000000] 0.0000000000	*min*	26	0.51
$f_9(\vec{x}) = \exp(x^2 + y^2 + z^2)$	[1, 2, 2]	[0.0000000000, 0.0000000000, 0.0000000000] 1.0000000000	*min*	7	0.43

$f_{10}(\vec{x}) = 2x^2 - xy + 2xz - y + + y^3 + z^2$	[1, 2, 2]	[0.3333333333, 0.6666666666, -0.3333333333] -0.4814814814	*min*	59	0.79
	[-1, -1, 1]	[-0.2500000000, -0.5000000000, 0.2500000000] 0.3125000000	*min/max*	40	0.58

Таблица 1.Результаты тестируемых функций при K=3 и $\alpha = 3/4$.
1-я колонка: целевая функция; 2-я: начальное приближение;
3-я: точка экстремума и значение целевой функции в ней;
4-я: тип экстремума; 5-я: количество итераций;
5-я: время поиска экстремума в милисекундах (мс).

В выше приведенной таблице даны результаты численного эксперимента (при $\delta = 10^{-14}$) с точностью 10 знаков после запятой, полученные по процедуре MinMax_N(...). Значения координат экстремума практически на 100% совпадают с соответствующими точными аналитическими решениями.

В отличие от градиентных методов, требующих вычисление градиента и гессиана, а также обращения матриц [4], предлагаемый подход свободен от этого. Более того, требуется только кодирование целевой функции в терминах гипер-дуальных чисел 3-го класса (Sdn3). Сама же процедура MinMax_N (...) достаточно компактна и обладает внушительным быстродействием.

Описанный выше метод поиска экстремума функции многих переменных может быть с успехом применен для поиска корней системы нелинейных уравнений вида $\vec{\varphi}(\vec{x}) = 0$, где $\vec{\varphi}$ – арифметический вектор функций многих переменных. Вычисление корней такой системы уравнений можно свести к нахождению минимума функционала $f(\vec{x}) = \|\vec{\varphi}(\vec{x})\|^2$. Для примера, рассмотрим следующую систему уравнений

$$\begin{cases} \varphi_1(x,y) = \sin(x - 0.5) - y - 1.5 = 0, \\ \varphi_2(x,y) = 2x - \cos(y) - 0.6 = 0 \end{cases}$$

Функционал, который соответствует ей, имеет вид $f(x,y) = [\varphi_1(x,y) = \sin(x - 0.5) - y - 1.5]^2 + [2x - \cos(y) - 0.6]^2$. Применяя процедуру MinMax_N (...) к $f(x,y)$, которая должна быть описана в терминах гипер-дуальных чисел 3-го класса (Sdn3), получим (при $x_0 = 0.13$, $y_0 = -1.8$, $\alpha = 3/4$, $\delta = 10^{-5}$):
$x_* \approx 0.1787440$, $y_* \approx -1.8157547$, $f(x_*, y_*) \approx 3.9441611 \times 10^{-11}$, $i = 56$, $t = 0.5$ мс.,
$\varphi_1(x_*, y_*) \approx -3.8373155 \times 10^{-6}$, $\varphi_2(x_*, y_*) \approx 3.9553943 \times 10^{-6}$.

Для сравнения, метод Ньютона этой же задачи дал заметно худшие результаты:
$x_* \approx 0.180366, \quad y_* \approx -1.8142,$
$\varphi_1(x_*, y_*) \approx -1.9119391 \times 10^{-5}, \ \varphi_2(x_*, y_*) \approx 1.7393601 \times 10^{-3}.$

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-14;
let Nmax = 1000; // max number of iterations

enum minmax:Int{
   case Min = 0
   case Max
   case MaxMin
}

// INPUT DATA:
// f – function pointer, xo – start points array, α – parameter of method
// OUTPUT DATA:
// (x, f, extr, i) – extremum points array, extremum value, extremum type and number of iterations

func MinMax_N(Fn:([Sdn3])->Sdn3, x0:[Double], α:Double)->
   (x:[Double],f:Double,extr:minmax, i:Int){
   var i = 0, X:[Sdn3],
   xi = x0,  D = 0.0, L = 0.0, N = x0.count - 1,
   F:[Sdn3] = Array(repeating: Sdn3(), count: N + 1);
         //--- Nested functions:
          func CH( F:Sdn3)->Double{
             if F.im1 == 0.0 || F.im2 == 0.0  {return 0.0}
             D = F.im1/F.im2; L = D*(F.im3/F.im2);
             return  (1.0 + 0.5*L/(1.0 - α*L))*D;
          }
```

```
            func isNotEnd()->Bool{
               for k in 0...N {if abs(F[k].im1) >= δ {return true}}
              return false;
            }
            func isMinMax()->minmax{
               if F.filter({$0.im2 > 0}).count == F.count {return minmax.Min}
               if F.filter({$0.im2 < 0}).count == F.count {return minmax.Max}
               return minmax.MaxMin;
            }
           //--- End of Nested functions
   X = xi.map{Sdn3(re: $0)}
  repeat {
      for k in 0...N {
         X[k].im1 = 1;  F[k] = Fn(X);
         xi[k] -=  CH(F: F[k]);
         X[k].re = xi[k];  X[k].im1 = 0; X[k].im2 = 0;
      }
      i += 1;
      if i > Nmax {break}
   }while (isNotEnd());
   return (x:xi,f:F[0].re, extr:isMinMax(),i:i);
}


//----- MinMax_N( ) function call example

let Extr = MinMax_N(Fn: Fn10, x0: [1, 2, 2], α: 3/4);


//----- some functions under study for MinMax_N( )

func Fn2(X:[Sdn3])->Sdn3{
   let R = X[0]*X[1] + 2*X[1]*X[2] + 11*X[0] + 2*X[1] + 18*X[2]
         + Sdn3(re: 10) - X[0]**2 - 5*(X[1]**2) - 3*(X[2]**2) - 2*X[0]*X[2];
   return R;
}

func Fn5(X:[Sdn3])->Sdn3{
   var R =  X[0]**2 + X[1]**2 - 3*X[0] + 2*X[1] - X[2]**3;
   return R;
}

func Fn9(X:[Sdn3])->Sdn3{
   var R = Sdn3.exp(X: X[0]**2 + X[1]**2 + X[2]**2);
   return R;
}
```

ЛИТЕРАТУРА

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

2. *Olifer V. I.* Implementation of Chebyshev-Halley type methods based on hyper-dual numbers.– URL:http://viosolutions.amerihomesrealty.com/pdf/Implementation_of_Chebyshev-Halley_type_methods_based_on_hyper-dual_numbers.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.

4. *Олифер В. И.* Вычисление градиента и гессиана методом автоматического дифференцирования с использованием усеченных гипер-дуальных чисел. – URL:http://viosolutions.amerihomesrealty.com/pdf/градиен_гессиан.pdf, 2020.

5. *Арнольд В. И.* Экспериментальная математика.– М.: ФАЗИС, 2005.

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 ноября 2023 г