ВЫЧИСЛЕНИЕ ГРАДИЕНТА И ГЕССИАНА МЕТОДОМ АВТОМАТИЧЕСКОГО ДИФФЕРЕНЦИРОВАНИЯ С ИСПОЛЬЗОВАНИЕМ УСЕЧЕННЫХ ГИПЕР-ДУАЛЬНЫХ ЧИСЕЛ

Олифер В.И.

Есть много приложений - например, численная оптимизация, анализ чувствительности и устойчивости, анализ рисков, численное интегрирование и дифференцирование, где ищутся первая и вторая производные исследуемой математической функции $f(\boldsymbol{x})$, описывающей соответствующее скалярное поле. Направление скорейшего возрастания или убывания скалярного поля и изменение его кривизны определяются значениями производных первого и второго порядков.

Введем некоторые обозначения для производных. Запишем

$$\nabla f(\boldsymbol{x}) = \left(\frac{\partial f}{\partial x_i}(\boldsymbol{x})\right) \tag{1}$$

для градиента функции $f(\boldsymbol{x})$, и

$$\nabla^2 f(\boldsymbol{x}) = \left(\frac{\partial^2 f}{\partial x_i \partial x_j}(\boldsymbol{x})\right) \tag{2}$$

для ее гессиана.

Рассмотрим функцию $f(\boldsymbol{x})$, которая реализуется компьютерным кодом (программой). Для вычисления градиента и гессиана этой функции будем использовать подход, называемый автоматическим дифференцированием. Автоматическое дифференцирование направлено на использование алгоритмического подхода, который автоматически генерирует и определяет аналитическое выражение градиента и / или гессиана функции $f(\boldsymbol{x})$ в заданной точке.

Усеченные гипер-дуальные числа [1] или усеченные гипер-дуальные векторы являются усечением гипер-дуальных чисел [2]. Одна из их целей состоит в том, чтобы сделать возможным напрямую вычислить производные первого и второго порядков с машинной точностью.

Согласно [1] усеченное гипер-дуальное число имеет вид $X = x + x_1\boldsymbol{\varepsilon} + x_2\boldsymbol{\omega}$, где: x, x_1 и x_2 – вещественные числа; $\boldsymbol{\varepsilon}$ и $\boldsymbol{\omega}$ – мнимые символы, причем $\boldsymbol{\varepsilon} \neq 0$ и $\boldsymbol{\omega} \neq 0$, и $\boldsymbol{\varepsilon}\cdot\boldsymbol{\varepsilon} = 2\boldsymbol{\omega}$, $\boldsymbol{\varepsilon}\cdot\boldsymbol{\omega} = \boldsymbol{\omega}\cdot\boldsymbol{\omega} = 0$. Число $x = Re(X)$ называется главной частью X, а $x_1 = Im_1(X)$, $x_2 = Im_2(X)$ мнимыми частями X.

Алгебраические операции сложения, умножения, обращения и деления определены по правилам:

$$\begin{aligned}
&A = a + a_1\boldsymbol{\varepsilon} + a_2\boldsymbol{\omega}, \qquad B = b + b_1\boldsymbol{\varepsilon} + b_2\boldsymbol{\omega},\\
&A + B = a + b + (a_1 + b_1)\boldsymbol{\varepsilon} + (a_2 + b_2)\boldsymbol{\omega},\\
&A\cdot B = a\cdot b + (a\cdot b_1 + b\cdot a_1)\boldsymbol{\varepsilon} + (a\cdot b_2 + 2\cdot a_1\cdot b_1 + b\cdot a_2)\boldsymbol{\omega},\\
&A^{-1} = a^{-1} - a_1\cdot a^{-2}\boldsymbol{\varepsilon} + (2\cdot a_1^2\cdot a^{-3} - a_2\cdot a^{-2})\boldsymbol{\omega},
\end{aligned} \tag{3}$$

$$A/_B = A{\cdot}B^{-1} = a{\cdot}b^{-1} + (a_1{\cdot}b^{-1} - a{\cdot}b_1{\cdot}b^{-2})\boldsymbol{\varepsilon} + [2{\cdot}(a{\cdot}b_1^2{\cdot}b^{-3} - a_1{\cdot}b_1{\cdot}b^{-2}) - a{\cdot}b_2{\cdot}b^{-2} + a_2{\cdot}b^{-1}]\boldsymbol{\omega}$$

Усеченная гипер-дуальная функция определяется выражением:

$$F(X) = f(x) + x_1{\cdot}f'(x)\boldsymbol{\varepsilon} + (x_2{\cdot}f'(x) + x_1^2{\cdot}f''(x))\boldsymbol{\omega}, \quad (4)$$

где $f'(x)$ и $f''(x)$ – первая и вторая производные функции $f(x)$ соответственно.
При $x_1 = 1$ и $x_2 = 0$ выражение (4) принимает вид:

$$F(X) = f(x) + f'(x)\boldsymbol{\varepsilon} + f''(x)\,\boldsymbol{\omega} \quad (5)$$

Определим векторную форму усеченного гипер-дуального числа в виде $\boldsymbol{X} = \boldsymbol{x} + \boldsymbol{x}_1{\cdot}\boldsymbol{\varepsilon} + \boldsymbol{x}_2{\cdot}\boldsymbol{\omega}$, где $\boldsymbol{x}$, $\boldsymbol{x}_1$ и $\boldsymbol{x}_2$ – матрицы столбцы одного и того же размера с вещественными компонентами. Нетрудно проверить, что алгебра усеченных гипер-дуальных чисел справедлива и для их векторной формы. При этом выражение скалярной функции от аргумента $\boldsymbol{X}$ принимает вид:

$$F(\boldsymbol{X}) = f(\boldsymbol{x}) + \boldsymbol{x}_1^T{\cdot}\nabla f(\boldsymbol{x})\boldsymbol{\varepsilon} + (\boldsymbol{x}_2^T{\cdot}\nabla f(\boldsymbol{x}) + \boldsymbol{x}_1^T{\cdot}\nabla^2 f(\boldsymbol{x}){\cdot}\boldsymbol{x}_1)\boldsymbol{\omega} \quad (6)$$

или в развернутой форме

$$F(\boldsymbol{X}) = f(\boldsymbol{x}) + \sum_i \left\{ \frac{\partial f(\boldsymbol{x})}{\partial x_i} x_{1i}\boldsymbol{\varepsilon} + \left(\frac{\partial f(\boldsymbol{x})}{\partial x_i} x_{2i} + \sum_{\boldsymbol{j}} \frac{\partial^2 f(\boldsymbol{x})}{\partial x_i \partial x_j} x_{1i}\, x_{1j} \right) \boldsymbol{\omega} \right\} \quad (7)$$

При $\boldsymbol{X} = \boldsymbol{X}_\alpha = \boldsymbol{x} + \boldsymbol{x}_{1\alpha}{\cdot}\boldsymbol{\varepsilon} + \mathbf{0}\cdot\boldsymbol{\omega}$, где: $\boldsymbol{x}_{1\alpha}$ – вектор с нулевыми элементами исключая α-й элемент, который равен 1; $\mathbf{0}$ – нулевой вектор, имеем

$$F(\boldsymbol{X}_\alpha) = f(\boldsymbol{x}) + f'_\alpha(\boldsymbol{x})\cdot\boldsymbol{\varepsilon} + f''_{\alpha\alpha}(\boldsymbol{x})\cdot\boldsymbol{\omega} \quad (8)$$

и

$$f'_\alpha(\boldsymbol{x}) = \frac{\partial f(\boldsymbol{x})}{\partial x_\alpha} = Im1\big(F(\boldsymbol{X}_\alpha)\big), \qquad f''_{\alpha\alpha}(\boldsymbol{x}) = \frac{\partial^2 f(\boldsymbol{x})}{\partial x_\alpha^2} = Im2\big(F(\boldsymbol{X}_\alpha)\big) \quad (9)$$

Если же положить $\boldsymbol{X} = \boldsymbol{X}_{\alpha\beta} = \boldsymbol{x} + (\boldsymbol{x}_{1\alpha} + \boldsymbol{x}_{1\beta})\cdot\boldsymbol{\varepsilon} + \mathbf{0}\cdot\boldsymbol{\omega}$, то получим

$$F\big(\boldsymbol{X}_{\alpha\beta}\big) = f(\boldsymbol{x}) + (f'_\alpha(\boldsymbol{x}) + f'_\beta(\boldsymbol{x}))\cdot\boldsymbol{\varepsilon} + (f''_{\alpha\alpha}(\boldsymbol{x}) + f''_{\beta\beta}(\boldsymbol{x}) + 2f''_{\alpha\beta}(\boldsymbol{x}))\cdot\boldsymbol{\omega} \quad (10)$$

и

$$f''_{\alpha\beta}(\boldsymbol{x}) = \Big[Im2\left(F\big(\boldsymbol{X}_{\alpha\beta}\big)\right) - Im2\big(F(\boldsymbol{X}_\alpha)\big) - Im2\left(F\big(\boldsymbol{X}_\beta\big)\right)\Big]/2 \quad (11)$$

Таким образом последовательно обращаясь к функции $F(\boldsymbol{X})$ с аргументами $\boldsymbol{X}_\alpha$ и $\boldsymbol{X}_{\alpha\beta}$ можно получить все значения $f'_\alpha(\boldsymbol{x})$, $f''_{\alpha\alpha}(\boldsymbol{x})$ и $f''_{\alpha\beta}(\boldsymbol{x})$, которые являются компонентами градиента и гессиана функции $f(\boldsymbol{x})$.

Рассмотрим компьютерный алгоритм описанного выше подхода. Пусть вектор (массив) аргументов $\boldsymbol{x}$ содержит n элементов. Тогда для хранения элементов градиента (первых производных) будем использовать одномерный массив gV действительных чисел размером n, причем $gV[i] = \partial f(\boldsymbol{x})/\partial x_i$, $i = 0, 1, \dots, n-1$.

С учетом того, что если смешанные производные функции $f(\boldsymbol{x})$ непрерывны, то гессиан является квадратной симметричной матрицей размером $n \times n$, то целесообразно хранить элементы гессиана в виде нижней треугольной матрицы, представленной одномерным массивом hV размером $n(n+1)/2$, где элементы гессиана расположены по строкам (см. Табл. 1).

i/j	0	1	2	3	4	5	6
0	0						
1	1	2					
2	3	4	5				
3	6	7	8	9			
4	10	11	12	13	14		
5	15	16	17	18	19	20	
6	21	22	23	24	25	26	27

Нижняя треугольная матрица Гессе hV

i
0
1
2
3
4
5
6

Вектор г gV

Табл.1. Графическое представление hV и gV

Порядковый индекс любого элемента нижней треугольной матрицы равен $p = i(i+1)/2 + j$, $j \le i$. При $j = i$ получим индекс диагональных элементов k, где $hV[k] = \partial^2 f(\boldsymbol{x})/\partial x_i^2$. Индекс для смешанных производных q определяется при $j < i$ и $hV[q] = \partial^2 f(\boldsymbol{x})/\partial x_i \partial x_j$.

Для вычисления значений $\partial f(\boldsymbol{x})/\partial x_i$, $\partial^2 f(\boldsymbol{x})/\partial x_i^2$ и $\partial^2 f(\boldsymbol{x})/\partial x_i \partial x_j$ исследуемая функция $f(\boldsymbol{x})$ должна быть переписана в терминах усеченных гипер-дуальных векторов т.е. $f(\boldsymbol{x}) \to F(\boldsymbol{X})$.

Тогда при $\boldsymbol{X}_i = \boldsymbol{x} + \boldsymbol{x}_{1i} \cdot \boldsymbol{\varepsilon} + \boldsymbol{0} \cdot \boldsymbol{\omega}$ имеем $F(\boldsymbol{X}_i) = f(\boldsymbol{x}) + f_i'(\boldsymbol{x}) \cdot \boldsymbol{\varepsilon} + f_{ii}''(\boldsymbol{x}) \cdot \boldsymbol{\omega}$ и $gV[i] = Im1(F(\boldsymbol{X}_i))$, $hV[i(i+1)/2 + i] = Im2(F(\boldsymbol{X}_i))$, т.е. за одно обращение к функции $F(\boldsymbol{X}_i)$ определяются величины одного элемента градиента и одного диагонального элемента гессиана. В Табл. 1 эти элементы имеют желтый фон, например элементы 4 и 14 для градиента и гессиана соответственно. Пусть теперь $\boldsymbol{X}_{ij} = \boldsymbol{x} + (\boldsymbol{x}_{1i} + \boldsymbol{x}_{1j}) \cdot \boldsymbol{\varepsilon} + \boldsymbol{0} \cdot \boldsymbol{\omega}$, тогда

$$F(\boldsymbol{X}_{ij}) = f(\boldsymbol{x}) + (f_i'(\boldsymbol{x}) + f_j'(\boldsymbol{x})) \cdot \boldsymbol{\varepsilon} + (f_{ii}''(\boldsymbol{x}) + f_{jj}''(\boldsymbol{x}) + 2f_{ij}''(\boldsymbol{x})) \cdot \boldsymbol{\omega}$$

Т.к. $f_{ii}''(\boldsymbol{x}) = hV[i(i+1)/2 + i]$ и $f_{jj}''(\boldsymbol{x}) = hV[j(j+1)/2 + j]$, то $hV[i(i+1)/2 + j] = \left[Im2\left(F(\boldsymbol{X}_{ij})\right) - hV[i(i+1)/2 - i]\, hV[j(j+1)/2 + j]\right]/2$.

Таким образом, для вычисления всех элементов массивов gV и hV потребуется $n(n+1)/2$ обращений к функции $F(\boldsymbol{X})$. Далее приводится программный код, реализующий вышеприведенный алгоритм (язык программирования SWIFT 5).

```
// INPUT DATA:
// x –array of argument values of investigated function F
// F – pointer to the function being investigated
// OUTPUT DATA:
// tuple (f:Double, gV:[Double],hV:[Double]),
// where: f  – value of function,
//        gV – array of gradient values, hV – array of hessian values
func GHV(x:[Double],F:([Thdn])->Thdn)->(f:Double,gV:[Double],hV:[Double]){
    let n  = x.count;
    var gV = [Double](repeating: 0.0, count:n);
    var hV = [Double](repeating: 0.0, count:n*(n + 1)/2);
    var X:[Thdn] = [], R:Thdn;
    for i in 0...(n - 1) {X.append(Thdn(re: x[i]))}
    var k = 0, p = 0, q = 0;
    // i - index for gradient in gV
    // k - index for second derivative in hV
    // p - index for mixed derivative in hV
    for i in 0...(n - 1){
        X[i].im1 = 1.0;
        R = F(X);
        gV[i] = R.im1;
        k = i*(i + 1)/2 + i;
        hV[k] = R.im2;
        if i > 0 {
            for j in 0...(i - 1){
                X[j].im1 = 1.0;
                R = F(X);
                p = k - i + j;
                q = j*(j + 1)/2 + j;
                hV[p] = (R.im2 - hV[q] - hV[k])/2.0;
                X[j].im1 = 0.0;
            }
       }
       X[i].im1 = 0.0;
  return (f:R.re, gV:gV, hV:hV);
}
```

Для использования приведенного кода необходимо добавить файл, содержащий описание типа данных `Thdn` (см. [1]).

Обратные матрицы Гессе наиболее часто используются в задачах оптимизации методом Ньютона. В SWIFT 5 быстрое обращение матрицы можно осуществить, используя системную библиотеку (framework) `Accelerate`. Ниже дана процедура `invertS(…)`, заимствованная из [3], обеспечивающая обращение квадратной матрицы. Входным параметром этой процедуры является одномерный массив, соответствующий квадратной матрице. Поэтому перед обращением к процедуре `invertS(…)` необходимо преобразовать нижнею треугольную матрицу `hV` в квадратную симметричную матрицу используя процедуру `convert_LtoS(…)`.

```
import Accelerate
// INPUT DATA:
```

```
// S – square matrix
// OUTPUT DATA:
// S⁻¹ – inverse square matrix
func invertS(S:[Double])->[Double]{
    var inMatrix = S;
    var N = __CLPK_integer(sqrt(Double(S.count)));
    var pivots = [__CLPK_integer](repeating: 0, count: Int(N));
    var workspace = [Double](repeating: 0.0, count: Int(N));
    var error : __CLPK_integer = 0; withUnsafeMutablePointer(to: &N) {
        dgetrf_($0, $0, &inMatrix, $0, &pivots, &error);
        dgetri_($0, &inMatrix, $0, &pivots, &workspace, $0, &error)
    }
    return inMatrix;
}


// INPUT DATA:
// L – lower triangular matrix
// OUTPUT DATA:
// S – square matrix
func convert_LtoS(L:[Double])->[Double]{
    let n = Int(ceil(sqrt(Double(L.count))));
    var S = [Double](repeating: 0.0, count:n*n);
    var p, q:Int;
    for i in 0...(n - 1){
        for j in 0...(n - 1){
            q = i*n + j;  p = i < j ? j*(j+1)/2 + i : i*(i+1)/2 + j;
            S[q] = L[p];
        }
    }
    return S;
}
```

Далее приводятся некоторые примеры численной реализации процедур `GHV(…)`, `convert_LtoS(…)` и `invertS(…)`. Для наглядности в качестве исследуемых функций использовались простейшие композиции из элементарных функций.

```
// function f1(x) = sin(x²(0) + x(1)); n = 2
func F1(X:[Thdn])->Thdn{return Thdn.sin(X: X[0]*X[0] + X[1])}
let _F1 = GHV(x: [2.0, 3.0], F: F1);
//result:   f  = 0.6569865987187891
//          gV = [3.0156090173732184, 0.7539022543433046]
//          hV = [-9.003981070814016,
                  -2.627946394875156, -0.6569865987187891]
//-------------------------------------------------------------------

// function f2(x) = x³(0) * x²(1) + 2 * x(0) + 3 * x(0) * x(1); n = 2
func F2(X:[Thdn])->Thdn{
   return Thdn.pow(X: X[0], n: 3)*Thdn.pow(X: X[1], n: 2) + 2.0*X[0] + 3.0*X[0]*X[1];
}
let _F2 = GHV(x: [1.0, 2.0], F: F2);
```

```
//result:   f  = 12.0
//          gV = [20.0, 7.0]
//          hV = [24.0,
                  15.0, 2.0]
//-------------------------------------------------------------

// function f3(x) = sin(x(0) + 2 * x(1)) + sqrt(x(0) * x(1)x * (2));  n = 3
func F3(X:[Thdn])->Thdn{
   return Thdn.sin(X: X[0]+ 2.0*X[1]]) + Thdn.pow(X: X[0]*X[1]*X[2], n: 0.5);
}
let _F3 = GHV(x: [1.0, 2.0, 3.0], F: F3);
//result:   f  = 1.4905654681200393
//          gV = [1.5084070568548151, 1.179696806622247, 0.408248290463863]
//          hV = [0.346551838967344,
                  2.2240347671741745,  3.682603989728605,
                  0.20412414523193148, 0.1020620726159657, -0.06804138174397717]
var S = convert_LtoS(L: RRR.hV);
//result: S = [0.346551838967344,   2.2240347671741745,  0.20412414523193148,
               2.2240347671741745,  3.682603989728605,   0.1020620726159657,
               0.20412414523193148, 0.1020620726159657, -0.06804138174397717]
var S_1 = invertS(S: S);
//result: S⁻¹ = [-1.4081765324664723,  0.9289048956222148, -2.831172253966094,
//               0.9289048956222148, -0.352044133116618,    2.258648487191717,
//              -2.831172253966094,   2.2586484871917167, -19.802482487809772]
```

Как видно из изложенного, процедура автоматического дифференцирования на основе векторной формы усеченных гипер-дуальных чисел позволяет относительно просто вычислять с машинной точностью значение функции многих переменных, элементы ее градиента и гессиана.
Другие примеры использования усеченных гипер-дуальных чисел можно найти в [4].

Литература

1. *Олифер В. И.* Усеченные гипер-дуальные числа в автоматическом дифференцировании. – URL: http://viosolutions.amerihomesrealty.com/pdf/Усеченные_гипер-дуальные_числа_в_автоматическом_дифференцировании.pdf.
2. *Fike J.A.* Multi-objective optimization using hyper-dual numbers. Ph.D. Dissertation, Stanford University, 2013.
3. *Олифер В. И.* Гипер-дуальные матричные уравнения и их чувствительность. –URL: http:// viosolutions.amerihomesrealty.com/pdf/гипер-дуальные_матрицы.pdf.
4. URL: http:// viosolutions.amerihomesrealty.com/

Абстракт

В статье рассматривается применение усеченных гипер-дуальных чисел, функций от них и компьютерного автоматического дифференцирования для точного (с машинной точностью) вычисления величин компонент градиента и гессиана функций многих переменных. Дан компьютерный код на языке SWIFT операционной системы macOS реализующий предлагаемый алгоритм.

Ключевые слова*: гипер-дуальные числа, усеченные гипер-дуальные числа, автоматическое дифференцирование, градиент, гессиан*

21 июля 2020 г.