К РАСЧЕТУ КРИВЫХ ОТКЛИКА С ИСПОЛЬЗОВАНИЕМ КРИГИНГ ИНТЕРПОЛЯЦИИ, АВТОМАТИЧЕСКОГО ДИФФЕРЕНЦИРОВАНИЯ И УСЕЧЕННЫХ ГИПЕР-ДУАЛЬНЫХ ЧИСЕЛ

Олифер В.И.

1. Кривые отклика и кригинг интерполяция

Кривая (функция) отклика – зависимость математического ожидания отклика от заданных факторов. Поверхность отклика (response curve line) определяется некоторой функцией, зависящей от входных факторов (входных данных): $Z = \psi(x_1, \ldots, x_l)$, где l – количество входных факторов. Используя входные данные $Z_i = Z(x_i)$, строится соответствующая интерполяционная модель, позволяющая определять $Z(x)$ в любой точки заданного интервала. На основании полученной интерполяционной модели рассчитываются $Z_j(x_j)$ в разных точках x_j. Найденный набор $Z_j(x_j)$ и определяет кривую отклика. Ключевым моментом является выбор интерполяционной модели. На практике считается, что кригинг интерполяция дает весьма хороший результат.

Кригинг интерполяция является базовой интерполяционной моделью геостатистики [1, 2]. При правильно выбранных априорных предположениях кригинг дает наилучшее линейное несмещенное предсказание промежуточных значений. Этот метод широко используется в области пространственного анализа и компьютерных (численных) экспериментах. Выделяют несколько вариантов моделей кригинга (простой, обычный, универсальный, логнормальный, невязок и др.), которые различаются принятыми предположениями и используемой информацией о моделируемой переменной. Наиболее часто используется обычный кригинг (ordinary kriging). Исходными данными служат значения непрерывной переменной Z в произвольных точках x_i, т. е. набор интерполяционных (опорных) узлов $\{z(x_i), i = 1, \ldots, n\}$ из n измерений. Далее делается предположение, что каждая точка $z(x_i)$ имеет разную степень значимости и интерполяционная функция $f(x)$ имеет вид:

$$f(x) = \sum_{i=1}^{n} w_i(x) \cdot z(x_i), \qquad (1.1)$$

где набор весов $w_i(x)$, удовлетворяющих условию $\sum w_i(x) = 1$, которое называется условием несмещенности.

Из (1.1) видно, что для каждой расчетной точки x необходимо иметь значения весов $w_i(x)$ всех интерполяционных узлов относительно этой точки, т. е. все интерполяционные узлы по-разному влияют на расчетную точку x. Значения $z(x_i)$ интерпретируются как реализации случайной переменной $Z(x_i)$.

Корреляция между двумя интерполяционными узлами вычисляется с использованием корреляционной функции, основанной на расстоянии между интерполяционными узлами и

параметром, связанным с длиной корреляции. Когда длина корреляции велика, выборки влияют на более широкую область.

Ковариация (мера взаимосвязи) между значениями функции $f(x)$ в двух разных точках x_i и x_j определяется как [1]:

$$Cov\big[f(x_i), f(x_j)\big] = \sigma^2 R\big(x_j - x_i\big), \tag{1.2}$$

где: σ^2 – квадрат стандартного отклонения, $R(x_j - x_i)$ – корреляционная функция.

Стандартная корреляционная функция для кригинг приближений является гауссовой корреляционной функцией. Для одномерного случая она имеет вид [2]:

$$R\big(d_{ji}\big) = exp\big(-\theta \cdot d_{ji}^2\big), \tag{1.3}$$

где: $d_{ji} = x_j - x_i$, $\theta > 0$ – параметр, связанный с длиной корреляции.

Заметим, что гауссова корреляционная функция (1.3) является четной функцией.

Ошибка интерполяции определяется по формуле $\delta(x) = f(x) - Z(x)$. Её дисперсия имеет вид [4]

$$Var[\delta(x)] = Cov[f(x), f(x)] - Cov[f(x), Z(x)] - Cov[Z(x), f(x)] + Var[Z(x)] \tag{1.4}$$

где:

$$\begin{aligned} Cov[f(x), f(x)] &= \sigma^2 \cdot w_i(x) \cdot w_j(x) \cdot R\big(d_{ji}\big), \\ Cov[f(x), Z(x)] &= \sigma^2 \cdot w_i(x) \cdot R(x - x_i), \\ Cov[Z(x), f(x)] &= \sigma^2 \cdot w_i(x) \cdot R(x_i - x), \\ Var[Z(x)] &= \sigma^2 \end{aligned}$$

С учетом четности $R(\dots)$ выражение (1.4) принимает вид:

$$Var[\delta(x)] = \sigma^2\{w_i(x) \cdot w_j(x) \cdot R(d_{ji}) - 2 \cdot w_i(x) \cdot R(x - x_i) + 1\} \tag{1.5}$$

Оптимальное решение для w_i можно найти, продифференцировав дисперсию $Var[\delta(x)]$ по w_i и приравняв результат нулю. При этом следует учесть ограничение $\sum w_i(x) \equiv 1$. В результате получим следующее матричное уравнение и его решение:

$$\boldsymbol{K} \cdot \boldsymbol{w} = \boldsymbol{r} \implies \boldsymbol{w} = \boldsymbol{K}^{-1} \cdot \boldsymbol{r}, \tag{1.6}$$

где $\boldsymbol{K}$ – симметричная матрица (матрица Криге) размером $(n + 1) \times (n + 1)$, $\boldsymbol{w}$ и $\boldsymbol{r}$ – столбцы размером $(n + 1)$:

$$K_{ij} = \begin{cases} R\big(d_{ji}\big) & \text{при } i \le n \text{ и } j \le n \\ 1 & \text{при } i = n+1 \text{ или } j = n+1 \\ 0 & \text{при } i = j = n+1 \end{cases}$$

$$\boldsymbol{w}^T = [\, w_1 \quad w_2 \quad \dots \quad w_n \quad \mu], \qquad \boldsymbol{r}^T = [R(x - x_1) \quad R(x - x_2) \quad \dots \, R(x - x_n) \quad 1]$$

и μ – множитель Лагранжа.

Это матричное уравнение затем решается в каждой точке x ($\boldsymbol{w} = \boldsymbol{K}^{-1}\cdot\boldsymbol{r}$), где требуется определить интерполяционное приближение. Заметим, что $\boldsymbol{w}(x)$ зависит только от взаимного расположения точек x, x_i и параметра длины корреляции θ.

Стандартный метод определения параметра θ для оптимизации кригинга заключается в выполнении оптимизации максимального правдоподобия с использованием данных выборки [3]. Этот метод применяется для одномерных тестовых функций. Уменьшение θ приводит к увеличению радиуса корреляции, и наоборот рост θ уменьшает радиуса корреляции. Для малых значений θ кривые отклика очень гладкие и плоские, в то время как для больших значений θ кривые отклика становятся ошибочными (появляются локальные флуктуации).

Включение информации о первой и второй производных может повысить точность создаваемой кривой отклика. Последнее особенно актуально при компьютерных экспериментах. Однако это требует вычисления производных фактической функции, что может быть вычислительно дорогим. Преимущество этой дополнительной информации может превысить преимущество простого добавления большего количества выборок значений функции, если вычислительные затраты на вычисление производной минимизированы. Далее будет показано, что это обстоятельство (трудность вычисления производной) можно обойти посредством использования усеченных гипер-дуальных чисел, что и будет показано в разделе 2.

Согласно [4], кригинг аппроксимация, которая включает в себя информацию о первой производной, имеет вид:

$$f(x) = \boldsymbol{w}_0^T(x)\cdot\boldsymbol{z} + \boldsymbol{w}_1^T(x)\cdot\boldsymbol{z}', \tag{1.7}$$

где: $\boldsymbol{w}_0^T = [\,w_{01} \quad w_{02} \quad \ldots \quad w_{0n}]$, $\boldsymbol{w}_1^T = [\,w_{11} \quad w_{12} \quad \ldots \quad w_{1n}]$ – весовые параметры значений непрерывной переменной Z и ее первой производной соответственно; $\boldsymbol{z} = [\,z_1 \quad z_2 \quad \ldots \quad z_n]^T$, $\boldsymbol{z}' = [\,z_1' \quad z_2' \quad \ldots \quad z_n']^T$ – значения переменной Z и ее первой производной соответственно; весовые параметры w_{0i}, и w_{1i} должны удовлетворять условиям: $\sum w_{0i}\,(x) = 1$, $\sum w_{1i}\,(x) = 0$.

Используя процедуру вывода соотношений (1.4) и (1.5), можно получить систему разрешающих уравнений с учетом первой производной $\boldsymbol{z}'$. В матрично-блочном виде она имеет следующий вид:

$$\begin{bmatrix} \boldsymbol{R} & \boldsymbol{R}' & \boldsymbol{I}_{10}^T \\ \boldsymbol{R}' & \boldsymbol{R}'' & \boldsymbol{I}_{01}^T \\ \boldsymbol{I}_{10} & \boldsymbol{I}_{01} & \boldsymbol{O}_2 \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{w}_0 \\ \boldsymbol{w}_1 \\ \boldsymbol{\mu}_2 \end{bmatrix} = \begin{bmatrix} \boldsymbol{r} \\ \boldsymbol{r}' \\ \boldsymbol{\nu}_2 \end{bmatrix}, \tag{1.8}$$

где:

$$R_{ij} = R\big(d_{ji}\big), \quad R'_{ij} = -2\cdot\theta\cdot d_{ji}\cdot R_{ij}, \quad R''_{ij} = 2\cdot\theta\cdot\big[2\cdot\theta\cdot d_{ji}^2 - 1\big]\cdot R_{ij}$$

$$r_i = R(x - x_i), \quad r_i' = -2\cdot\theta\cdot(x - x_i)\cdot r_i$$

$$\boldsymbol{I}_{10} = \begin{bmatrix} 1 & \ldots & 1 \\ 0 & \ldots & 0 \end{bmatrix}_{2\times n}, \qquad \boldsymbol{I}_{01} = \begin{bmatrix} 0 & \ldots & 0 \\ 1 & \ldots & 1 \end{bmatrix}_{2\times n}, \qquad \boldsymbol{O}_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}_{2\times 2},$$

$$\boldsymbol{\mu}_2 = [\mu_0 \quad \mu_1]^T, \qquad \boldsymbol{\nu}_2 = [1 \quad 0]^T$$

Элементы матриц $\boldsymbol{R}$, $\boldsymbol{R}''$ и $\boldsymbol{r}$ – четные функции, а элементы матриц $\boldsymbol{R}'$, $\boldsymbol{R}'''$ и $\boldsymbol{r}'$ – нечетные функции.

Кригинг аппроксимация, включающая информацию о первой и второй производных [4], представима в виде:

$$f(x) = \boldsymbol{w}_0^T(x)\cdot \boldsymbol{z} + \boldsymbol{w}_1^T(x)\cdot \boldsymbol{z}' + \boldsymbol{w}_2^T(x)\cdot \boldsymbol{z}'', \tag{1.9}$$

где: $\boldsymbol{w}_0^T = [\,w_{01} \quad w_{02} \quad \dots \quad w_{0n}]$, $\boldsymbol{w}_1^T = [\,w_{11} \quad w_{12} \quad \dots \quad w_{1n}]$, $\boldsymbol{w}_2^T = [\,w_{21} \quad w_{22} \quad \dots \quad w_{2n}]$ – весовые параметры значений непрерывной переменной Z, ее первой и второй производных соответственно; $\boldsymbol{z} = [\,z_1 \quad z_2 \quad \dots \quad z_n]^T$, $\boldsymbol{z}' = [\,z_1' \quad z_2' \quad \dots \quad z_n']^T$, $\boldsymbol{z}' = [\,z_1'' \quad z_2'' \quad \dots \quad z_n'']^T$ – значения переменной Z, ее первой и второй производных соответственно; весовые параметры w_{0i}, w_{1i} и w_{2i} должны удовлетворять условиям: $\sum w_{0i}\,(x) = 1$, $\sum w_{1i}\,(x) = \sum w_{2i}\,(x) = 0$.
Тогда матрично-блочное разрешающее уравнение с учетом первой $\boldsymbol{z}'$ и второй $\boldsymbol{z}''$ производных представимо следующим образом:

$$\begin{bmatrix} \boldsymbol{R} & \boldsymbol{R}' & \boldsymbol{R}'' & \boldsymbol{I}_{100}^T \\ \boldsymbol{R}' & \boldsymbol{R}'' & \boldsymbol{R}''' & \boldsymbol{I}_{010}^T \\ \boldsymbol{R}'' & \boldsymbol{R}''' & \boldsymbol{R}^{\mathrm{IV}} & \boldsymbol{I}_{001}^T \\ \boldsymbol{I}_{100} & \boldsymbol{I}_{010} & \boldsymbol{I}_{001} & \boldsymbol{O}_3 \end{bmatrix} \cdot \begin{bmatrix} \boldsymbol{w}_0 \\ \boldsymbol{w}_1 \\ \boldsymbol{w}_2 \\ \boldsymbol{\mu}_3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{r} \\ \boldsymbol{r}' \\ \boldsymbol{r}'' \\ \boldsymbol{\nu}_3 \end{bmatrix}, \tag{1.10}$$

где:

$$R_{ij} = R\big(d_{ji}\big), \quad R'_{ij} = -2\cdot\theta\cdot d_{ji}\cdot R_{ij}, \quad R''_{ij} = 2\cdot\theta\cdot\big[2\cdot\theta\cdot d_{ji}^2 - 1\big]\cdot R_{ij},$$

$$R'''_{ji} = -4\cdot\theta^2\cdot\big[3\cdot d_{ji} - 2\cdot\theta\cdot d_{ji}^3\big]\cdot R_{ij}, \quad R^{\mathrm{IV}}_{ji} = [12\cdot\theta^2 - 48\cdot\theta^3\cdot d_{ji}^2 + 16\cdot\theta^4\cdot d_{ji}^4]\cdot R_{ij},$$

$$r_i = R(x - x_i), \quad r_i' = -2\cdot\theta\cdot(x - x_i)\cdot r_i, \quad r_i'' = 2\cdot\theta\cdot[2\cdot\theta\cdot(x - x_i)^2 - 1]\cdot r_i,$$

$$\boldsymbol{I}_{100} = \begin{bmatrix} 1 & \dots & 1 \\ 0 & \dots & 0 \\ 0 & \dots & 0 \end{bmatrix}_{3\times \mathrm{n}}, \quad \boldsymbol{I}_{010} = \begin{bmatrix} 0 & \dots & 0 \\ 1 & \dots & 1 \\ 0 & \dots & 0 \end{bmatrix}_{3\times \mathrm{n}}, \quad \boldsymbol{I}_{001} = \begin{bmatrix} 0 & \dots & 0 \\ 0 & \dots & 0 \\ 1 & \dots & 1 \end{bmatrix}_{3\times \mathrm{n}}, \quad \boldsymbol{O}_3 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}_{3\times 3},$$

$$\boldsymbol{\mu}_3 = [\mu_0 \quad \mu_1 \quad \mu_2]^T, \qquad \boldsymbol{\nu}_3 = [1 \quad 0 \quad 0]^T$$

Элементы матриц $\boldsymbol{R}$, $\boldsymbol{R}''$, $\boldsymbol{R}^{\mathrm{IV}}$ и $\boldsymbol{r}$, $\boldsymbol{r}''$ – четные функции, а элементы матриц $\boldsymbol{R}'$, $\boldsymbol{R}'''$ и $\boldsymbol{r}'$ – нечетные функции.

Таким образом, разрешающие уравнения (1.6), (1.8) и (1.10) соответствуют трем интерполяционным формам: 1) *скалярная* – учет только значений входных данных; 2) *градиентная* – учет значений и первых производных входных данных; 3) *гессиальная* – учет значений, первых и вторых производных входных данных.

Одним из основных условий учета производных является гладкость и дифференцируемость используемой корреляционной функции. Также имеет существенное значение трудоемкость получения значений производных фактической функции и их точность. Последнее обстоятельство может быть преодолено путем использования усечённых гипер-дуальных чисел.

2. Усечённые гипер-дуальные числа

Традиционные способы вычисления производных сложных функций сопряжены с трудностями и имеют существенные недостатки. Реализация вычислений производной функции в виде отдельной программной процедуры является избыточным решением и требует предварительного анализа самой функции. Использование метода конечных разностей нуждается в компромиссе между погрешностями усечения и округления, т. е. нахождения оптимальной величины приращения аргумента функции. Еще большие проблемы возникают при вычислении производных более высоких порядков. Обойти указанные трудности позволяет метод автоматического дифференцирования (AD) [5], связанный с вычислением точных значений производных функции, представленных компьютерным кодом на основе гипер-дуальных чисел [4, 6]. Гипер-дуальные числа являются расширением дуальных чисел [7], так же как кватернионы являются расширением обычных комплексных чисел [8]. Для итерационных процессов, использующих значения функции, ее первых и вторых производных, гипер-дуальные числа обладают некоторой избыточностью (дублирование значений первой производной). Поэтому далее введем новый тип гипер-дуальных чисел – *усеченные гипер-дуальные числа* (truncated hyper-dual number) и функции от них.
Усеченное гипер-дуальное число с базисом $\{\mathbf{1},\ \boldsymbol{\varepsilon},\ \boldsymbol{\omega}\}$ определим выражением:

$$X = x + x_1\boldsymbol{\varepsilon} + x_2\boldsymbol{\omega}, \tag{2.1}$$

где: $\boldsymbol{\varepsilon}$ и $\boldsymbol{\omega}$ – мнимые символы, x, x_1 и x_2 – вещественные числа. Число $x = Re(X)$ называется главной частью X, а $x_1 = Im_1(X)$ и $x_2 = Im_2(X)$ – мнимыми частями X. Пространство усеченных гипер-дуальных чисел отвечает трехмерной алгебре с правилом умножения элементов базиса:

×	$\mathbf{1}$	$\boldsymbol{\varepsilon}$	$\boldsymbol{\omega}$
$\mathbf{1}$	1	$\boldsymbol{\varepsilon}$	$\boldsymbol{\omega}$
$\boldsymbol{\varepsilon}$	$\boldsymbol{\varepsilon}$	$2\boldsymbol{\omega}$	0
$\boldsymbol{\omega}$	$\boldsymbol{\omega}$	0	0

Табл. 2.1. Правила умножения элементов базиса усечённых гипер-дуальных чисел

Алгебраические операции сложения, умножения, обращения и деления (с учетом табл. 2.1) определяются соотношениями:

$$\begin{aligned}
&A = a + a_1\boldsymbol{\varepsilon} + a_2\boldsymbol{\omega}, \qquad B = b + b_1\boldsymbol{\varepsilon} + b_2\boldsymbol{\omega},\\
&A + B = a + b + (a_1 + b_1)\boldsymbol{\varepsilon} + (a_2 + b_2)\boldsymbol{\omega},\\
&A{\cdot}B = a{\cdot}b + (a{\cdot}b_1 + b{\cdot}a_1)\boldsymbol{\varepsilon} + (a{\cdot}b_2 + 2{\cdot}a_1{\cdot}b_1 + b{\cdot}a_2)\boldsymbol{\omega},\\
&A^{-1} = a^{-1} - a_1{\cdot}a^{-2}\boldsymbol{\varepsilon} + (2{\cdot}a_1^2{\cdot}a^{-3} - a_2{\cdot}a^{-2})\boldsymbol{\omega},\\
&{}^{A}/_{B} = A{\cdot}B^{-1} = a{\cdot}b^{-1} + (a_1{\cdot}b^{-1} - a{\cdot}b_1{\cdot}b^{-2})\boldsymbol{\varepsilon} + [2{\cdot}(a{\cdot}b_1^2{\cdot}b^{-3} - a_1{\cdot}b_1{\cdot}b^{-2}) -\\
&\qquad a{\cdot}b_2{\cdot}b^{-2} + a_2{\cdot}b^{-1}]\boldsymbol{\omega}
\end{aligned} \tag{2.2}$$

Раскладывая функцию усеченного гипер-дуального аргумента $F(X)$ в ряд Тейлора (с учетом табл. 2.1 и (2.2)), получим

$$F(X) = f(x) + x_1 \cdot f'(x)\boldsymbol{\varepsilon} + \ (x_2 \cdot f'(x) + x_1^2 \cdot f''(x)\ )\boldsymbol{\omega} \tag{2.3}$$

При $x_1 = 1$ и $x_2 = 0$ выражение (2.3) принимает вид:

$$F(X) = f(x) + f'(x)\boldsymbol{\varepsilon} \ + \ f''(x)\boldsymbol{\omega} \tag{2.4}$$

Соотношение (2.4) свободно от избыточности присущей функциям гипер-дуального аргумента. Воспользовавшись [9], приведем компоненты некоторых элементарных (базовых) функций усечённого гипер-дуального аргумента $X = x + x_1 \cdot \boldsymbol{\varepsilon} + x_2 \cdot \boldsymbol{\omega}$:

$F(\boldsymbol{X})$	$\boldsymbol{f(x)}$	$\boldsymbol{f'(x)}$	$\boldsymbol{f''(x)}$
X^n	x^n	$n \cdot x^{n-1}$	$n \cdot (n-1) \cdot x^{n-2}$
e^X	e^x	e^x	e^x
a^X	a^x	$a^x \ln(a)$	$a^x \cdot \ln(x)^2$
$\ln(X)$	$\ln(x)$	x^{-1}	$-x^{-2}$
$\log_a(X)$	$\log_a(x)$	$(x \ln(a))^{-1}$	$-x^{-2} \cdot \ln(a)^{-1}$
$\sin(X)$	$\sin(x)$	$\cos(x)$	$-\sin(x)$
$\cos(X)$	$\cos(x)$	$-\sin(x)$	$-\cos(x)$
$\mathrm{tg}(X)$	$\mathrm{tg}(x)$	$\sec^2(x) = \cos^{-2}(x)$	$2 \cdot \mathrm{tg}(x) \cdot \cos^{-2}(x)$
$\mathrm{ctg}(X)$	$\mathrm{ctg}(x)$	$-\csc^2(x) = -\sin^{-2}(x)$	$-2 \cdot \mathrm{ctg}(x) \cdot \sin^{-2}(x)$
$\sec(X)$	$\sec(x)$	$\sec(x) \cdot \mathrm{tg}(x)$	$\sec(x) \cdot (2 \cdot \mathrm{tg}^2(x) + 1)$
$\csc(X)$	$\csc(x)$	$-\csc(x) \cdot \mathrm{ctg}(x)$	$\csc(x) \cdot (2 \cdot \mathrm{ctg}^2(x) + 1)$
$\arcsin(X)$	$\arcsin(x)$	$1/\sqrt{1-x^2}$	$x \cdot (1-x^2)^{-3/2}$
$\arccos(X)$	$\arccos(x)$	$-1/\sqrt{1-x^2}$	$-x \cdot (1-x^2)^{-3/2}$
$\mathrm{arctg}(X)$	$\mathrm{arctg}(x)$	$1/(1+x^2)$	$-2 \cdot x/(1+x^2)^2$
$\mathrm{arcctg}(X)$	$\mathrm{arcctg}(x)$	$-1/(1+x^2)$	$2 \cdot x/(1+x^2)^2$
$\mathrm{sh}(X)$	$\mathrm{sh}(x)$	$\mathrm{ch}(x)$	$\mathrm{sh}(x)$
$\mathrm{ch}(X)$	$\mathrm{ch}(x)$	$\mathrm{sh}(x)$	$\mathrm{ch}(x)$
$\mathrm{th}(X)$	$\mathrm{th}(x)$	$\mathrm{sech}^2(x)$	$-2 \cdot \mathrm{sech}^2(x) \cdot \mathrm{th}(x)$
$\mathrm{cth}(X)$	$\mathrm{cth}(x)$	$-\mathrm{csch}^2(x)$	$2 \cdot \mathrm{csch}^2(x) \cdot \mathrm{cth}(x)$
$\mathrm{sech}(X)$	$\mathrm{sech}(x)$	$-\mathrm{sech}(x) \cdot \mathrm{th}(x)$	$\mathrm{sh}^{-1}(x) \cdot (2 \cdot \mathrm{ch}^2(x) \cdot \mathrm{sh}^{-2}(x) - 1)$
$\mathrm{csch}(X)$	$\mathrm{csch}(x)$	$-\mathrm{csch}(x) \cdot \mathrm{cth}(x)$	$\mathrm{ch}^{-1}(x) \cdot (2 \cdot \mathrm{sh}^2(x) \cdot \mathrm{ch}^{-2}(x) - 1)$

Табл. 2.2. Компоненты основных элементарных функций усечённого гипер-дуального аргумента

Вычисление сложной функции (function composition) вида $F = f_1(f_2(\ldots f_n(X)\ \ldots), X), X)$ (где значение f_n используется в качестве аргумента для f_{n-1}) необходимо начать с вычисления $F_n = f_n(X)$, продолжить вычислениями $F_{n-1} = f_{n-1}(F_n, X)$, $F_{n-2} = f_{n-2}(F_{n-1}, X)$, … , $F = f_1(F_2, X)$.

3. Компьютерная реализация

Компьютерная реализация усечённых гипер-дуальных чисел была выполнена на языке SWIFT 5.1 [11] операционной системы macOS 10.15. Был создан новый тип данных `Thdn` (truncated hyper-dual number) и его расширение, которое включает перезагрузку элементарных функции (см. табл. 2.2) и операций (см. Приложение 1). Манипуляции (тест-драйв) с усечёнными гипер-дуальными числами типа `Thdn` даны в Приложение 2. Реализации разрешающих уравнений (1.6), (1.8) и (1.10) с использованием усечённых гипер-дуальных чисел даны в Приложение 3 (`class Kriging`). Как было отмечено в разделе 1, решения во всех трех интерполяционных формах зависят (помимо всего прочего) от расположения узлов интерполяции и параметра θ, связанного с длиной корреляции.

Поэтому в класс `Kriging` добавлена функция `get_best_θ` упрощенного поиска «лучшего» значения θ, на основе метода дихотомии (половинного деления) [10].

Также добавлены процедуры, осуществляющие численные эксперименты (см. Приложение 4). Приведенный программный код не претендует на оптимальную версию. Целью было представить сравнительно легко читаемый, интуитивно понимаемый и по возможности компактный программный код.

4. Численные эксперименты

Для проверки точности, предложенных выше решений, рассчитаны кривые отклика для некоторых относительно простых функций. Продемонстрированы эффекты включения первой и второй производной. Если расчет кривых отклика должен быть основаны на учете первой и второй производных, то точное вычисление последних может быть проблемой. Поэтому первая и вторая производные кривых отклика в опорных точках, а также производные корреляционной функции вычислялись с использованием усеченных гипер-дуальных чисел. Погрешность интерполяционных решений проводилась по формуле $\delta = \sqrt{n^{-1} \sum_i (f(x_i) - z(x_i))^2}$, где i = 1, 2, …, n – количество точек, принимающих участие в оценке точности.

В табл. 4.1 приведены результаты расчёта для функции $z(x) = cos(x) - 2 \cdot exp\,(-16 \cdot x^2)$ при $\theta = 5.0$. Серым фоном отмечены: колонки, отвечающие опорным узлам; строка точных значений подлинной функции. Учитывая, что исходная функция $z(x)$ четная, то результаты приведены в положительном интервале ее интерполяции. Графическое представления полученных результатов дано на рис. 5.1 в Приложении 5.

x	0.0	0.25	0.5	1.0	1.25	1.5	1.75	2.0	2.5	3.0	3.5	4.0
z	-1.0000		0.8409		0.3153		-0.1782		-0.8011		-0.9365	
f_0	0.2617	0.2332	0.2239	0.5403	0.3961	0.0707	-0.2354	-0.4161	-0.5326	-0.9899	-0.5850	-0.6536
f_1	-0.6860		1.1724		0.3354		-0.1636		-0.5865		-0.6793	

f_2	-0.9078		0.8338		0.3721		-0.1282		-0.6287		-0.6714	

Табл. 4.1. $z(x) = cos(x) - 2 \cdot exp\,(-16 \cdot x^2)$ *при* $\theta = 5.0$ *(f_0 – без учета производных, f_1 – с учетом первых производных, f_2 – с учетом первых и вторых производных)*

Погрешности решений f_0, f_1 и f_2 равны $\delta_0 \approx 0.426$, $\delta_1 \approx 0.163$ и $\delta_2 \approx 0.098$ соответственно.

Из полученных результатов видно, что интерполяционная кривая $f_2(x)$ отвечает лучшему приближению, особенно в интервале -1 < x < 1. В тоже время, кривая $f_1(x)$ несомненно лучше кривой $f_0(x)$.

Для оценки влияния величины параметра θ использовалась функция $z(x) = 1/(1 + x^2)$. При $\theta = 0.2$ лучшей интерполяцией оказалась та которая учитывает только первую производную (см. табл. 4.2).

Погрешности решений f_0, f_1 и f_2 равны $\delta_0 \approx 0.054$, $\delta_1 \approx 0.009$ и $\delta_2 \approx 0.075$ соответственно. Однако при $\theta = 2.0$ на первое место вышло интерполяционное приближение с учетом первой и второй производных (см. табл. 4.3). Погрешности решений f_0, f_1 и f_2 равны $\delta_0 \approx 0.058$, $\delta_1 \approx 0.017$ и $\delta_2 \approx 0.001$ соответственно. Сопоставление указанных выше лучших решений приведено в табл. 4.4. При этом $\delta_1 \approx 0.009$ и $\delta_2 \approx 0.001$.

x	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0
z	1.00000		0.50000	0.30769		0.13793		0.07547	
f_0	0.88026	0.80000	0.59907	0.37034	0.20000	0.11925	0.10000	0.09068	0.05882
f_1	0.98545		0.47980	0.29437		0.13522		0.07246	
f_2	0.94598	0.79804	0.36393	0.17398	0.19655	0.08345	0.09924	-0.0148	0.05906

Табл. 4.2. $z(x) = 1/(1 + x^2)$ *при* $\theta = 0.2$

x	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0
z	1.00000		0.50000	0.30769		0.13793		0.07547	
f_0	0.83552	0.80000	0.55911	0.30553	0.20000	0.15079	0.10000	0.06951	0.05882
f_1	0.97194		0.46249	0.29018		0.14167		0.08709	
f_2	0.99882		0.50201	0.30797		0.13855		0.07441	

Табл. 4.3. $z(x) = 1/(1 + x^2)$ *при* $\theta = 2.0$

	x	0.0	0.5	1.0	1.5	2.0	2.5	3.0	3.5	4.0
	z	1.00000		0.50000	0.30769		0.13793		0.07547	
$\theta = 0.2$	f_1	0.98545	0.80000	0.47980	0.29437	0.20000	0.13522	0.10000	0.07246	0.05882
$\theta = 2.0$	f_2	0.99882		0.50201	0.30797		0.13855		0.07441	

Табл. 4.4. $z(x) = 1/(1 + x^2)$ (f_1 *–учет первых производных и* $\theta = 0.2$, f_2 *–учет первых и вторых производных и* $\theta = 2.0$)

Наилучшее значение параметра θ зависит от формы интерполяции (см. раздел 1), количества интерполяционных узлов (design points) и их расположения, количества и расположения узлов, принимающих участие в оценке точности интерполяционного решения (tested points). Некоторые результаты определения лучшей величины θ для функции $1/(1 + x^2)$ приведены в следующей таблице (интерполяционные узлы см. табл. 4.2 – 4.4):

Модель	Тестируемые точки	Лучший параметр θ
f_0	$[x]_0$	0.37017822265625

	$[x]$	1.94976806640625
f_1	$[x]_0$	0.88653564453125
	$[x]$	0.92071533203125
f_2	$[x]_0$	2.34588623046875
	$[x]$	2.57110595703125

Табл. 4.5. $z(x) = 1/(1+x^2)$ ($[x]_0$ = [0.0], $[x]$ = [0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0])

Для наглядности эффект использования информации о первой и второй производных был продемонстрирован с использованием относительно простых аналитических функций. Однако применение усеченных гипер-дуальных чисел Thdn позволяет реализовать функцию любой сложности, если она представима посредством элементарных (базовых) функций усечённого гипер-дуального аргумента.

5. Заключение

Получение оптимальных результатов кригинг интерполяции достаточно сложный многошаговый процесс, связанный выбором: критериев интерполяции; интерполяционной формы (см. раздел 1); положения и количества интерполяционных узлов; определения лучшей величины θ. Все это требует использования соответствующего программно-математического обеспечения при участии оценивотеля. Поэтому в настоящей работе была сделана попытка создания прототипа такого программного обеспечения (см. Приложения 1- 4) с применением которого выполнены некоторые численные эксперименты (см. раздел 4).

Полученные матрично-блочные разрешающие уравнения (1.6), (1.8) и (1.10) с использованием усечённых гипер-дуальных чисел Thdn относительно просто программируются, а представленный программный код легко адоптируется и для иных дифференцируемых корреляционных функций, например, экспоненциальной, функции Матерна [3] и других. Заметим также, что рассмотренные решения можно ассоциировать с реальными физическими процессами. Так, например, интерполяционное решение для $1/(1+x^2)$ весьма близко к описанию процесса (траектории) расхождения (сближения) двух физических объектов с учетом их взаимной скорости и ускорения.

Приложение 1

Описание типа данных Thdn (truncated-hyper-dual number) для Swift 5.1 (macOS 10.14)

```
import Foundation

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

struct Thdn{
```

```
    var re,im1,Im2: Double;
    //--Initializers:
    init() {...}
    init(re:Double) {...}
    init(re:Double,im1:Double,im2:Double) {...}
    //--Methods:
    func norm() -> Double {...}
    func string() -> String {...}
}
extension Thdn {
    //--CONSTANTS:
    static let accuracy:Double = 1e-15, PI:Double = Double.pi;
    //--OPERATORS (overloading):
    static prefix func - ( A:Thdn) -> Thdn {...}
    static prefix func + ( A:Thdn) -> Thdn {...}
    static func +  (A:Thdn, B:Thdn) -> Thdn {...}
    static func += (lhs: inout Thdn, rhs: Thdn) {...}
    static func -  (A:Thdn, B:Thdn) -> Thdn {...}
    static func -= (lhs: inout Thdn, rhs: Thdn) {...}
    static func *  (A: Thdn, B: Thdn) -> Thdn {...}
    static func *  (A: Thdn, B: Double) -> Thdn {...}
    static func *  (A: Double, B: Thdn) -> Thdn {...}
    static func *= (lhs: inout Thdn, rhs: Thdn) {...}
    static func *= (lhs: inout Thdn, rhs: Double) {...}
    static func /  (A: Thdn, B: Thdn) -> Thdn {...}
    static func /  (A: Thdn, B: Double) -> Thdn {...}
    static func ** (left: Thdn, right: Double) -> Thdn {...}
    static func /= (lhs: inout Thdn, rhs: Thdn) {...}
    static func /= (lhs: inout Thdn, rhs: Double) {...}
    static func == (left: Thdn,  right: Thdn) -> Bool {...}
    static func != (left: Thdn,  right: Thdn) -> Bool {...}
    static func <  (left: Thdn,  right: Thdn) -> Bool {...}
    static func >  (left: Thdn,  right: Thdn) -> Bool {...}
    static func <= (left: Thdn, right: Thdn) -> Bool {...}
    static func >= (left: Thdn, right: Thdn) -> Bool {...}
    //--FUNCTIONS:
    static func inverse(A:Thdn) -> Thdn {...}
    static func pow(x:Double, n:Double) -> Thdn {...}
    static func pow(X:Thdn, n:Double) -> Thdn {...}
    static func powX(a:Double, x:Double) -> Thdn {...}
    static func log(x:Double) -> Thdn {...}
    static func log(X:Thdn) -> Thdn {...}
    static func sqrt(x:Double) -> Thdn {...}
    static func sqrt(X:Thdn) -> Thdn {...}
    static func exp(x:Double) -> Thdn {...}
    static func exp(X:Thdn) -> Thdn {...}
    static func sin(x:Double) -> Thdn {...}
    static func sin(X:Thdn) -> Thdn {...}
    static func cos(x:Double) -> Thdn {...}
    static func cos(X:Thdn) ->Thdn {...}
    static func tan(x:Double) -> Thdn {...}
    static func tan(X:Thdn) -> Thdn {...}
    static func asin(x:Double) -> Thdn {...}
    static func asin(X:Thdn) -> Thdn {...}
    static func acos(x:Double) -> Thdn {...}
    static func acos(X:Thdn) -> Thdn {...}
    static func atan(x:Double) -> {...}
    static func atan(X:Thdn) -> Thdn {...}
}
```

Приложение 2

Результаты манипуляций (тест-драйв) с усечёнными гипер-дуальными числами типа `Thdn` (см. *Приложение 1*)

```
//--Operators:
        var A,B,C: Thdn
        A = Thdn (re: 1, im1: 2, im2: 3)     // A = 1.0  + 2.0•ε  + 3.0•ω
        B = Thdn (re: 4, im1: 5, im2: 6)     // B = 4.0  + 5.0•ε  + 6.0•ω
        C = A + B                            // C = 5.0  + 7.0•ε  + 9.0•ω
```

```
        C = A*B                                 // C = 4.0  + 13.0•ε + 38.0•ω
        C = Thdn.inverse(A: A)                  // C = 1.0  - 2.0•ε  + 5.0•ω
        C = A/B                                 // C = 0.25 + 0.1875•ε - 0.09375•ω

//--Basic functions:
        var Y: Thdn = Thdn.exp(x: 0)            // Y = 1 + 1•ε + 1•ω
        Y = Thdn.cos(x: 0)                      // Y = 1 + 0•ε - 1•ω
        Y = Thdn.pow(x: 2, n: 3)                // Y = 8 + 12•ε + 12•ω
        Y = Thdn.sqrt(x: 4)                     // Y = 2 + 0.25•ε - 0.03125•ω
        Y = Thdn.log(x: 1)                      // Y = 0 + 1•ε - 1•ω
```

Приложение 3

Реализация кригинг интерполяции в коде Swift 5.1

```
import Foundation
import Accelerate

class Kriging {
    var x:Double = 0.0, θ:Double = 0.1, xi:[Double] = [];
    var F:(Double)->Hdtn, rank:Rank = .values_only;

    init(F:@escaping (_ x:Double)->Thdn, xi:[Double], θ:Double, rank:Rank){
        self.F = F;
        self.xi = xi;
        self.θ = θ;
        self.rank = rank;
    }

    enum Rank: Int {case values_only = 0, first_derivative, second_derivative}
    private var zz:[Double] = [];

    private func Gauss(x:Double)->Thdn{
        let X = Thdn(re: x, im1: 1.0, im2: 0.0);
        return Thdn(re:1.0)/Thdn.exp(X: self.θ*X*X);
    }

    private func g1234(x: Double)->(y:Double,y1:Double,y2:Double,y3:Double,y4:Double){
       let X  = Thdn(re: x, im1: 1.0, im2: 0.0);
       let Z  = self.Gauss(x: x);
       let Z1 = Thdn(re:2.0*θ)*(Thdn(re:2.0*θ)*X*X - Thdn(re:1.0))*Z;
       return (y:Z.re,y1:Z.im1,y2:Z.im2,y3:Z1.im1,y4:Z1.im2);
    }

    private func invertK(matrix:[Double])->[Double]{
        var inMatrix = matrix;
        var N = __CLPK_integer(sqrt(Double(matrix.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;
    }

    func create_r()->[Double]{
        var  g:Thdn = Thdn(),
             r:[Double] = [], r1:[Double] = [], r2:[Double] = [];
        for j in 0..<xi.count {
            g = Gauss(x: x - xi[j]);       r.append(g.re);
            if rank.rawValue > 0 {r1.append(g.im1)}
            if rank.rawValue > 1 {r2.append(g.im2)}
        }
        r = r + r1 + r2;
        r.append(1.0);
        if rank == .first_derivative {r.append(0.0);}
        else if rank == .second_derivative{r  = r  + [0.0, 0.0]}
```

```
    return r;
}

func createK_1()->[Double]{
    var k:[Double] = [], k1:[Double] = [], k2:[Double] = [], K:[Double] = [];
    var r:[Double] = [], r1:[Double] = [], r2:[Double] = [],
       r3:[Double] = [], r4:[Double] = [],
       gn:(y:Double,y1:Double,y2:Double,y3:Double,y4:Double);
    let n = xi.count;
    for i in 0..<n {
        r = [];   r1 = [];   r2 = [];   r3 = [];   r4 = [];
        for j in 0..<n {
            gn = g1234(x: xi[j] - xi[i]);
            r.append(gn.y);
            if rank == .first_derivative {r1.append(gn.y1);    r2.append(gn.y2)}
            else if rank == .second_derivative{
                r1.append(gn.y1);    r2.append(gn.y2);
                r3.append(gn.y3);    r4.append(gn.y4);
            }
        }
        k  += r;
        if rank.rawValue > 0 {k  += r1;    k1 += r1;    k1 += r2}
        if rank.rawValue > 1 {k  += r2;    k1 += r3;
                              k2 += r2;    k2 += r3;    k2 += r4}
        if rank == .values_only {k += [1.0]}
        else if rank == .first_derivative{k += [1.0, 0.0]; k1 += [0.0, 1.0]}
        else if rank == .second_derivative{
            k  += [1.0, 0.0, 0.0];
            k1 += [0.0, 1.0, 0.0];
            k2 += [0.0, 0.0, 1.0];
        }
    }
    K = k + k1 + k2;
    let I0 =  Array(repeating: Double(0), count: n);
    let I1 =  Array(repeating: Double(1), count: n);
    var O:[Double] = [];
    if rank == .values_only{O = [0.0]}
    else if rank == .first_derivative {O = [0.0, 0.0]}
    else if rank == .second_derivative{O = [0.0, 0.0, 0.0]}

    if rank == .values_only {K += (I1 + O)}
    else if rank == .first_derivative {
        K  += (I1 + I0 + O);
        K  += (I0 + I1 + O);
    }
    else if rank == .second_derivative{
        K  += (I1 + I0 + I0 + O);
        K  += (I0 + I1 + I0 + O);
        K  += (I0 + I0 + I1 + O);
    }
  return   self.invertK(matrix: K)
}

func getWi_μ(K_1:[Double], r:[Double])->[Double]{
    let n = r.count;
    var res:[Double] = [], wi:Double;
    for j in 0..<n {
        wi = 0.0;
        for i in 0..<n {wi += (r[i] * K_1[i + j*n])}
        res.append(wi);
    }
    return res;
}

func getZx(Zi:[Double], Wi_μ:[Double])->Double{
    var Z:Double = 0.0;
     for i in 0..<Zi.count { Z += Zi[i]*Wi_μ[i]}
     return Z;
}

func err(z:[Double], f:[Double])->Double{
    var δ:Double = 0.0;
      let n = z.count;
      for i in 0..<n {δ += pow(f[i] - z[i], 2)}
```

```
            δ /= Double(n);
            return sqrt(δ);
        }

        func get_best_θ(x:[Double], θmax:Double, σ: Double = 0.001)->Double{
            var zn:[Double] = [],  zn':[Double] = [], zn'':[Double] = [], Zn:Thdn = Thdn();
            for i in 0..<self.xi.count {Zn = F(self.xi[i]);  zn.append(Zn.re);
                                        zn'.append(Zn.im1);  zn''.append(Zn.im2)}
            if rank == .first_derivative {zn += zn'}
            else if rank == .second_derivative{zn += (zn' + zn'')}
            var θl = 0.0,   θr = θmax,  θc:Double, θlr:[Double];
            var rn:[Double] = [], wμn:[Double] = [], fi:Double, zi:Double,
                δ:Double, δl:Double, δr:Double,  Kn_1:[Double] = [];
            while abs(θl - θr) >= σ {
                θc = (θl + θr)/2.0;  θlr = [θc - σ, θc + σ];
                δl = 0.0;             δr = 0.0;
                for θ in θlr {
                    self.θ = θ;
                    Kn_1 = self.createK_1();
                    for xi in x {
                       self.x =  xi;
                       rn  = self.create_r();
                       wμn = self.getWi_μ(K_1: Kn_1, r: rn);
                       fi  = self.getZx(Zi: zn, Wi_μ: wμn);
                       zi  = F(xi).re;
                       δ   = abs(fi - zi);
                       if θ < θc {δl += δ} else {δr += δ}
                    }
                }
                if (δl < δr){θr =  θc} else {θl =  θc}
            }
            return (θl + θr)/2.0;
        }
    }
```

Приложение 4

Примеры использования класса **Kriging**

```
func F (x:Double) -> Thdn{let X = Thdn(re: x, im1: 1.0, im2: 0.0);
                          let 1 = Thdn(re:1); return 1/(1 + X*X)}  //-- 1/(1+x²)
//-- function interpolation taking into account the first and second derivatives
let θ = 2.0, rank = Kriging.Rank.second_derivative;
let xn:[Double] = [-4, -3, -2, -0.5, 0.5, 2, 3, 4];
var zn:[Double] = [],  zn':[Double] = [], zn'':[Double] = [], Zn:Thdn = Thdn();
for i in 0..<xn.count {Zn = F(x:xn[i]);  zn.append(Zn.re);  zn'.append(Zn.im1);
zn''.append(Zn.im2)}
if rank == .first_derivative {zn += zn'}
else if rank == .second_derivative{zn += (zn' + zn'')}

var myKriging = Kriging(F: F, xi: xn, θ: θ, rank: rank);
let Kn_1 = myKriging.createK_1();

for x in stride(from: 0.0, to: 4.5, by: 0.5) {
    myKriging.x =  x;
    let rn  = myKriging.create_r();
    let wμn = myKriging.getWi_μ(K_1: Kn_1, r: rn);
    let fi  = myKriging.getZx(Zi: zn, Wi_μ: wμn);
    print("x=\(x);   fi=\(fi)");
}

//-- determination of the best parameter θ
let x:[Double] = [0.0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0]; // tested points
let θ_opt = myKriging.get_best_θ(x: x, θmax: 10.0, σ: 0.001);
```

Приложение 5

Графическое представление некоторых численных результатов расчета.

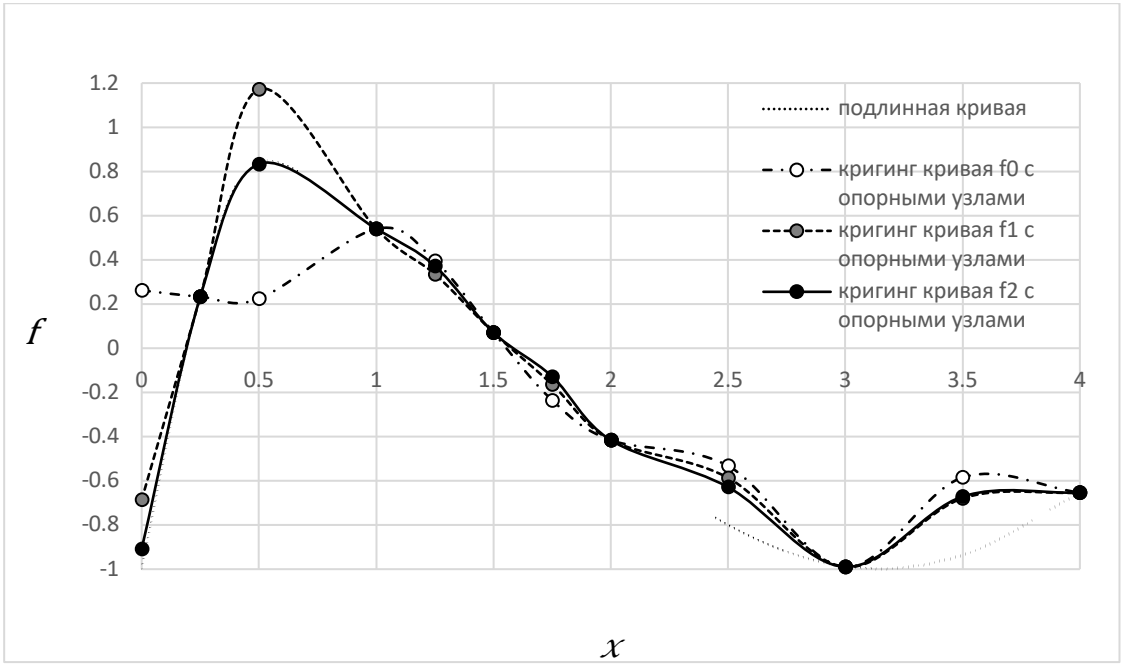

Рис. 5.1. Кривые отклика для подлинной кривой $z(x) = cos(x) - 2 \cdot exp\,(-16 \cdot x^2)$ *при* $: \theta = 5.0$ (f_0 *– без учета производных*, f_1 *– с учетом первых производных*, f_2 *– с учетом первых и вторых производных*)

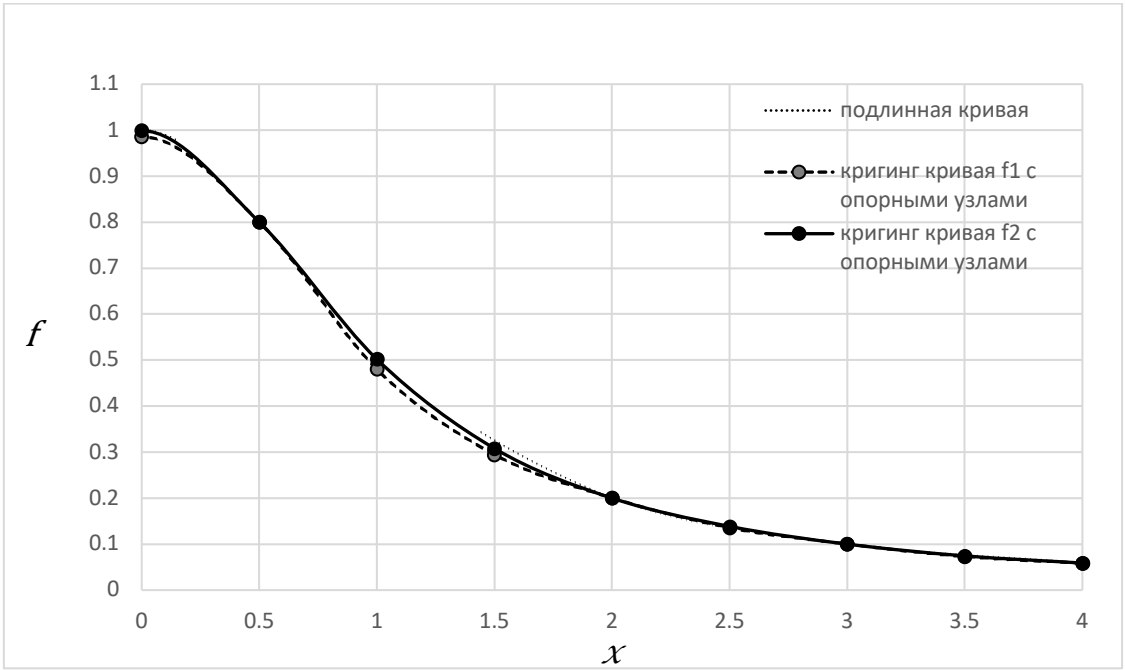

Рис. 5.2. Кривые отклика для подлинной кривой $z(x) = 1/(1 + x^2)$ (f_1 *– с учетом первых производных при* $\theta = 0.2$, f_2 *– с учетом первых и вторых производных при* $\theta = 2.0$)

Литература

1. *В. В. Демьянов, Е. А. Савельева*. ГЕОСТАТИСТИКА: теория и практика. – М.: Наука, 2010. – 327 с.
2. *E. H. Isaaks and R. M. Srivastava*. An Introduction to Applied Geostatistics. Oxford University Press, 1989.
3. *J. R. Koehler and A. B. Owen*. Computer experiments, 1996.
4 *J. A. Fike.* Multi-Objective Optimization Using Hyper-Dual Numbers. Ph.D. Dissertation, Stanford University, 2013.
5. *U. Naumann*. The Art of Differentiating Computer Programs. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2012.
6. *J. A. Fike and J. J. Alonso.* The Development of Hyper-Dual Numbers for Exact Second Derivative Calculations. AIAA paper 2011-886, 49th AIAA Aerospace Sciences Meeting, January 4-7, 201.
7. *Ф. М. Диментберг*. Теория винтов и ее приложения. – М.: Наука, 1978. – 328с.
8. *В. В. Сильвестров*. Системы чисел. – Соросовский образовательный журнал, № 8, 1998, 121-127с.
9. *Г. Б. Двайт*. Таблицы интегралов и другие математические формулы. – М.: Наука, 1973. – 228с.
10. *С. П. Шарый.* Курс вычислительных методов. – Новосибирск: Новосиб. гос. ун-т., 2014. - 507 с.
11. The Swift programming language 5.1 – *https://docs.swift.org/swift-book/#*

Абстракт

В данной статье рассматриваются вопросы, связанные с применением специальных чисел (гипер-дуальных чисел) и компьютерного автоматического дифференцирования в кригинг интерполяции. Дается матрично-блочные представления разрешающих уравнений кригинг интерполяции с учетом первых и вторых производных. Вводится новый тип чисел (усечённые гипер-дуальные числа), которые свободны от избыточности (redundancy), присущей гипер-дуальным числам. Приводятся основные операции и базовые функции пространства усечённых гипер-дуальных чисел. Представлена компьютерная реализация Кригинг интерполяции с учетом усечённых гипер-дуальных чисел для языка SWIFT операционной системы macOS.

Ключевые слова*: кригинг интерполяция, гипер-дуальные числа, усеченные гипер-дуальные числа, автоматическое дифференцирование.*

20 ноября 2019 г.