ГИПЕР-ДУАЛЬНЫЕ МАТРИЧНЫЕ УРАВНЕНИЯ И ИХ ЧУВСТВИТЕЛЬНОСТЬ

Гипер-дуальные матрицы – это матрицы компонентами которых служат гипер-дуальные числа. Последние (гипер-дуальные числа) являются расширением классических дуальных чисел. В настоящей работе под гипер-дуальными матрицами будем понимать матрицы с новым типом их элементов, а именно усеченными гипер-дуальными числами.

1. Усечённые гипер-дуальные числа

Введем новый тип гипер-дуальных чисел – *усеченные гипер-дуальные числа* (truncated hyper-dual number) с базисом $\{\mathbf{1},\ \boldsymbol{\varepsilon},\ \boldsymbol{\omega}\}$, определяющиеся выражением:

$$X = x + x_1\boldsymbol{\varepsilon} + x_2\boldsymbol{\omega}, \tag{1.1}$$

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:

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

Табл. 1.1. Правила умножения элементов базиса усечённых гипер-дуальных чисел

Алгебраические операции сложения, умножения, обращения и деления (с учетом табл. 1.1) определяются соотношениями:

$$\begin{aligned}
&A = a + a_1\boldsymbol{\varepsilon} + a_2\boldsymbol{\omega}, \qquad B = b + b_1\boldsymbol{\varepsilon} + b_2\boldsymbol{\omega},\\
&A + B = a + b + (a_1 + b_1)\boldsymbol{\varepsilon} + (a_2 + b_2)\boldsymbol{\omega},\\
&A\cdot B = a\cdot b + (a\cdot b_1 + b\cdot a_1)\boldsymbol{\varepsilon} + (a\cdot b_2 + 2\cdot a_1\cdot b_1 + b\cdot a_2)\boldsymbol{\omega},\\
&A^{-1} = a^{-1} - a_1\cdot a^{-2}\boldsymbol{\varepsilon} + (2\cdot a_1^2\cdot a^{-3} - a_2\cdot a^{-2})\boldsymbol{\omega},\\
&{}^{\mathrm{A}}\!/_{B} = \mathrm{A}\cdot\mathrm{B}^{-1} = a\cdot b^{-1} + (a_1\cdot b^{-1} - a\cdot b_1\cdot b^{-2})\boldsymbol{\varepsilon} + [2\cdot(a\cdot b_1^2\cdot b^{-3} - a_1\cdot b_1\cdot b^{-2}) -\\
&\qquad\qquad a\cdot b_2\cdot b^{-2} + a_2\cdot b^{-1}]\boldsymbol{\omega}
\end{aligned} \tag{1.2}$$

Усеченные гипер-дуальные числа являются усечением гипер-дуальных чисел, рассмотренных в [1, 2] (базисом гипер-дуальных чисел служит $\{\mathbf{1},\ \boldsymbol{\varepsilon_1},\ \boldsymbol{\varepsilon_2},\ \boldsymbol{\varepsilon_{12}}\}$ с правилом умножения $\boldsymbol{\varepsilon_1}^{\mathbf{2}} = \boldsymbol{\varepsilon_2}^{\mathbf{2}} = \boldsymbol{\varepsilon_{12}}^{\mathbf{2}} = \mathbf{0}$, $\boldsymbol{\varepsilon_{12}} = \boldsymbol{\varepsilon_1}\cdot\boldsymbol{\varepsilon_2} = \boldsymbol{\varepsilon_2}\cdot\boldsymbol{\varepsilon_1}$). С другой стороны, усеченные гипер-дуальные числа – это расширение дуальных чисел [3]. Нетрудно видеть, что если в (1.1) и (1.2) аннулировать члены, содержащие $\boldsymbol{\omega}$, то получим алгебру классических дуальных чисел.

2. Усечённые гипер-дуальные матрицы

Определение и использование матриц с классическими дуальными числами можно найти в [4 – 6]. Матрица же $[\hat{\boldsymbol{A}}]$, составленная из усеченных гипер-дуальных чисел, представима в виде:

$$[\hat{\boldsymbol{A}}] = [A] + [A]_1\boldsymbol{\varepsilon} + [A]_2\boldsymbol{\omega}, \tag{2.1}$$

где: $[A]$, $[A]_1$ и $[A]_2$ – матрицы такого же размера что и $[\hat{\boldsymbol{A}}]$ с вещественными элементами. Матрицы $[A]$ и $[A]_1$, $[A]_2$ являются соответственно главной и мнимыми частями матрицы $[\hat{\boldsymbol{A}}]$. Алгебраические операции сложения, умножения и обращения усеченных гипер-дуальных матриц определяются соотношениями:

$$\begin{aligned}
&[\hat{\boldsymbol{A}}] \pm [\hat{\boldsymbol{B}}] = [A] \pm [B] + ([A]_1 \pm [B]_1)\boldsymbol{\varepsilon} + ([A]_2 \pm [B]_2)\boldsymbol{\omega}, \\
&[\hat{\boldsymbol{A}}]\cdot[\hat{\boldsymbol{B}}] = [A]\cdot[B] + ([A]\cdot[B]_1 + [A]_1\cdot[B])\boldsymbol{\varepsilon} + ([A]\cdot[B]_2 + 2[A]_1\cdot[B]_1 + [A]_2\cdot[B])\boldsymbol{\omega}, \\
&[\hat{\boldsymbol{A}}]^{-1} = [A]^{-1} - [A]^{-1}\cdot[C]\boldsymbol{\varepsilon} + [A]^{-1}\cdot(2[C]\cdot[C] - [A]_2\cdot[A]^{-\mathbf{1}})\boldsymbol{\omega}, \\
&\text{где } [C] = [A]_1\cdot[A]^{-1}.
\end{aligned} \tag{2.2}$$

Нетрудно видеть, что квадратная матрица $[\hat{\boldsymbol{A}}]$ обратима тогда и только тогда, когда ее главная часть $[A]$ обратима и более того, для обращения $[\hat{\boldsymbol{A}}]$ необходимо вычислить только матрицу $[A]^{-1}$ и выполнить ряд операций ее умножения на $[A]_1$ и $[A]_2$. Отметим также справедливость следующего соотношения:

$$[\hat{\boldsymbol{A}}]^{-1}\cdot[\hat{\boldsymbol{A}}] = [\hat{\boldsymbol{I}}] = [I] + [0]\boldsymbol{\varepsilon} + [0]\boldsymbol{\omega}, \tag{2.3}$$

где $[I]$ – единичная диагональная матрица.
Линейное матричное уравнение для гипер-дуальных матриц и его решение (в терминах усеченных гипер-дуальных матриц) имеют вид:

$$[\hat{\boldsymbol{A}}]\cdot\{\hat{\boldsymbol{X}}\} = \{\hat{\boldsymbol{B}}\} \;\longrightarrow\; \{\hat{\boldsymbol{X}}\} = [\hat{\boldsymbol{A}}]^{-1}\cdot\{\hat{\boldsymbol{B}}\}, \tag{2.4}$$

где $\{\hat{\boldsymbol{X}}\} = \{X\} + \{X\}_1\boldsymbol{\varepsilon} + \{X\}_2\boldsymbol{\omega}$ и $\{\hat{\boldsymbol{B}}\} = \{B\} + \{B\}_1\boldsymbol{\varepsilon} + \{B\}_2\boldsymbol{\omega}$ – матрицы-столбцы.
В отличии от усеченных гипер-дуальных чисел множество квадратных матриц с элементами из усеченных гипер-дуальных чисел образуют некоммутативное кольцо т.е., вообще говоря, $[\hat{\boldsymbol{A}}]\cdot[\hat{\boldsymbol{B}}] \neq [\hat{\boldsymbol{B}}]\cdot[\hat{\boldsymbol{A}}]$.
Решение матричного уравнения (2.4) в терминах вещественных матриц представляет собой последовательную трех-шаговую процедуру по нахождению главной и мнимых частей $\{\hat{\boldsymbol{X}}\}$:

$$\{X\} = [A]^{-1}\cdot\{B\}, \tag{2.5}$$

$$\{X\}_1 = [A]^{-1}\cdot(\{B\}_1 - [A]_1\cdot\{X\}),$$
$$\{X\}_2 = [A]^{-1}\cdot(\{B\}_2 - 2[A]_1\cdot\{X\}_1 - [A]_2\cdot\{X\})$$

Если в (2.1) – (2.3) обнулить члены, содержащие $\boldsymbol{\omega}$, то получим алгебру матриц, составленных из классических дуальных чисел [4 – 6].

3. Чувствительность матричных уравнений

Вопрос о чувствительности решения системы алгебраических уравнений к изменениям самой матрицы и столбца правых частей имеет большое значение, поскольку появляющиеся в различных задачах математической физики вспомогательные системы имеют коэффициенты и правую часть, определяемые приближенно. Последнее приводит к погрешности искомого решения. Вообще говоря, чувствительность входит в состав характеристик любой системы и определяет ее реакцию на изменение входной информации, структурные или параметрические вариации, на ошибки самой модели и др. Поэтому чувствительность системы алгебраических уравнений можно определить как влияние возбуждений (изменений) правой и левой частей системы на ее решение. Следует заметить, что указанные возбуждения могут иметь разные источники (род/порядок). Последнее приводит к необходимости дифференцированного учета разных возбуждений и их влияния друг на друга. Если исходить из анализа системы (2.4), то можно считать, что мнимые части матриц $[\widehat{\boldsymbol{A}}]$ и $\{\widehat{\boldsymbol{B}}\}$ описывают возмущения первого и второго порядков, а отклик на них определяют мнимые части решения $\{\widehat{\boldsymbol{X}}\}$.

Возможны три случая возмущений:

1) вызванные правой частью системы ($[A]_1 = [A]_2 = [0]$)

$$\begin{gathered}\{X\} = [A]^{-1}\cdot\{B\},\\ \{X\}_1 = [A]^{-1}\cdot\{B\}_1,\\ \{X\}_2 = [A]^{-1}\cdot\{B\}_2,\\ \|\{X\}_1\| \le \|[A]^{-1}\|\cdot\|\{B\}_1\|,\\ \|\{X\}_2\| \le \|[A]^{-1}\|\cdot\|\{B\}_2\|;\end{gathered} \tag{3.1}$$

2) вызванные левой частью системы ($\{B\}_1 = \{B\}_2 = \{0\}$)

$$\begin{gathered}\{X\} = [A]^{-1}\cdot\{B\},\\ \{X\}_1 = -[A]^{-1}\cdot[A]_1\cdot\{X\},\\ \{X\}_2 = -[A]^{-1}\cdot(2[A]_1\cdot\{X\}_1 + [A]_2\cdot\{X\}),\\ \|\{X\}_1\| \le \|[A]^{-1}\|\cdot\|[A]_1\|\cdot\|\{X\}\|,\\ \|\{X\}_2\| \le \|[A]^{-1}\|\cdot(\|[A]_2\|\cdot\|\{X\}\| + 2\|[A]_1\|\cdot\|\{X\}_1\|);\end{gathered} \tag{3.2}$$

3) вызванные левой и правой частями системы ($[A]_1, [A]_2 \neq [0]$, $\{B\}_1, \{B\}_2 \neq \{0\}$)

$$\begin{gathered}\{X\} = [A]^{-1}\cdot\{B\},\\ \{X\}_1 = [A]^{-1}\cdot(\{B\}_1 - [A]_1\cdot\{X\}),\end{gathered} \tag{3.3}$$

$$\{X\}_2 = [A]^{-1}\cdot(\{B\}_2 - 2[A]_1\cdot\{X\}_1 - [A]_2\cdot\{X\}),$$
$$\|\{X\}_1\| \le \|[A]^{-1}\|\cdot(\|\{B\}_1\| + \|[A]_1\|\cdot\|\{X\}\|),$$
$$\|\{X\}_2\| \le \|[A]^{-1}\|\cdot(\|\{B\}_2\| + 2\|[A]_1\|\cdot\|\{X\}_1\| + \|[A]_2\|\cdot\|X\|);$$

где: $\|\{Z\}\| = [\sum_{i=1}^{n} Z_i^2]^{1/2}$, $\|[A]\| = \left[\sum_{i=1}^{n}\sum_{j=1}^{n} A_{ij}^2\right]^{1/2}$ ($n -$ порядок системы) и использованы свойства норм матриц $\|[A]\| \ge 0$, $\|k\cdot[A]\| = |k|\cdot\|[A]\|$, $\|[A] + [D]\| \le \|[A]\| + \|[D]\|$, $\|[A]\cdot[D]\| \le \|[A]\|\cdot\|[D]\|$.

Общая оценка чувствительности исходной системы (при нулевых возмущениях) определяется числом (мерой) обусловленности [7] главной части матрицы $[\widehat{\boldsymbol{A}}]$, а именно $\mu([\widehat{\boldsymbol{A}}]) = cond([A]) = \|[A]\|\cdot\|[A]^{-1}\|$. Чем больше число обусловленности, тем система более чувствительна к уровню возмущений. Вообще говоря, определение границы между «хорошей» и «плохой» обусловленностью в отрыве от конкретно решаемой задачи почти бессмысленно, т.к. имеет значение размерность задачи, требуемая точность решения, тип представления компьютерных чисел и тому подобное. На практике, матрица обычно считается хорошо обусловленной, если $1 \le \mu([\widehat{\boldsymbol{A}}]) \le 10^2$. Используя значения норм $\|\{X\}\|$, $\|\{X\}_1\|$ и $\|\{X\}_2\|$ можно получить величины относительных значений влияния для соответствующих возмущений.

4. Компьютерная реализация

Компьютерная реализация усечённых гипер-дуальных чисел и матриц были выполнены на языке SWIFT 5.1 [11] операционной системы macOS 10.15. Созданы новые типы данных `Thdn` (truncated hyper-dual number) и `ThdnMatrix` (truncated hyper-dual matrix), их расширения, которые включают перезагрузку операций (см. Приложение 1, 2). Была использована системная фреймворк (framework) `Accelerate`, позволяющая оптимизировать матричные (вещественные) операции для высокой производительности и низкого энергопотребления. Манипуляции (тест-драйв) с усечёнными гипер-дуальными матрицами типа `ThdnMatrix` даны в Приложении 3.

Приведенный программный код не претендует на оптимальную версию. Целью было представить сравнительно легко читаемый, интуитивно понимаемый и по возможности компактный программный код.

5. Численные эксперименты

Для обозримости результатов расчета чувствительности в качестве исходной системы рассмотрим матричное уравнение

$$[\widehat{\boldsymbol{A}}]\cdot\{\widehat{\boldsymbol{X}}\} = \{\widehat{\boldsymbol{B}}\} \text{ с } [A] = \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix} \text{ и } [B] = \begin{bmatrix} 4 \\ 7 \end{bmatrix}.$$

Напомним, что мнимые части матриц $[\widehat{\boldsymbol{A}}]$ и $\{\widehat{\boldsymbol{B}}\}$ описывают возмущения первого и второго порядков, а отклик на них определяют мнимые части решения $\{\widehat{\boldsymbol{X}}\}$. Рассмотрим три случая возмущений: 1) вызванные правой частью системы; 2) вызванные левой частью системы; 3) вызванные левой и правой частями системы. Число обусловленности матрицы $[A]$ равно

$$\mu([A]) = \|A\|\cdot\|[A]^{-1}\| = \left\| \begin{matrix} 1 & 2 \\ 2 & 3 \end{matrix} \right\| \cdot \left\| \begin{matrix} -3 & 2 \\ 2 & -1 \end{matrix} \right\| = 18 \ \leq 10^2.$$

Результаты расчета указанных выше трех случаев возмущений приведены в таблице Приложения 4. Там же дана схема влияния возмущений на исходную систему.

6. Заключение

Описан новый тип гипер-дуальных чисел – усечённые гипер-дуальные числа (`Thdn`). По сравнению с гипер-дуальными числами, компьютерная реализация этого типа данных (`Thdn`) позволяет сэкономить память. Компьютерная реализация этого типа данных оказалась достаточно компактной. Дана процедура оценки чувствительности матричного уравнения на базе гипер-дуальных матриц, позволяющая учитывать возмущения первого и второго порядков. Проведены численные эксперименты с использованием разработанного программного обеспечения.

Приложение 1

Описание типа данных `Thdn` (truncated-hyper-dual number) для Swift 5 (macOS 10.14.6)

```
import Foundation

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 {
    //--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 {...}
}
```

Приложение 2

```
import Foundation
import Accelerate
struct ThdnMatrix {
    private(set) var size:(rows:Int, cols:Int);
    var array: [Thdn];
    init(array: [Thdn], size:(rows:Int, cols:Int)) {
        assert(size.rows*size.cols == array.count,
              "⚠️ size.rows * size.cols should be equal to array.count");
        self.array = array;
        self.size =  size;
    }
    func array2D()->[[Thdn]]{
        let size:Int = self.size.rows;
        return stride(from: 0, to: array.count, by: size).map {
               Array(array[$0..<min($0 + size, array.count)])}
    }
    func parts() -> (re:[Double], im1:[Double], im2:[Double]) {
        var re:[Double] = [], im1:[Double] = [], im2:[Double] = [];
        for item in array {re.append(item.re);  im1.append(item.im1);
            im2.append(item.im2)}
        return (re:re, im1:im1, im2:im2);
    }
    func parts2D()-> (re:[[Double]], im1:[[Double]], im2:[[Double]]){
        let parts = self.parts();
        let size:Int = self.size.rows;
        let re   = stride(from: 0, to: parts.re.count,  by: size).map {
                   Array(parts.re[$0..<min($0 + size,   parts.re.count)])}
        let im1  = stride(from: 0, to: parts.im1.count, by: size).map {
                   Array(parts.im1[$0..<min($0 + size,  parts.im1.count)])}
        let im2  = stride(from: 0, to: parts.im2.count, by: size).map {
                   Array(parts.im2[$0..<min($0 + size,  parts.im2.count)])}
        return (re:re, im1:im1, im2:im2);
    }
}
extension ThdnMatrix {
//--OPERATORS (overloading):
    static prefix func - (A:ThdnMatrix)->ThdnMatrix{
        return ThdnMatrix(array: A.array.map({$0 * (-1)}), size: A.size);
    }
    static prefix func + (A:ThdnMatrix)->ThdnMatrix{return A}
    static func + (A:ThdnMatrix, B:ThdnMatrix)->ThdnMatrix{
        assert(A.size.rows  == B.size.rows && A.size.cols  == B.size.cols,
               "⚠️ size of A should be equal to size of B.");
        return ThdnMatrix(array: zip(A.array,B.array).map(+), size: A.size);
    }
```

```
    static func +=(lhs: inout ThdnMatrix, rhs: ThdnMatrix) {lhs = lhs + rhs}
    static func - (A:ThdnMatrix, B:ThdnMatrix)->ThdnMatrix{
        assert(A.size.rows  == B.size.rows && A.size.cols  == B.size.cols,
               "⚠️ size of A should be equal to size of B.");
        return ThdnMatrix(array: zip(A.array,B.array).map(-),size: A.size);
    }
    static func -=(lhs: inout ThdnMatrix, rhs: ThdnMatrix) {lhs = lhs - rhs}
    static func *(A: ThdnMatrix, B: ThdnMatrix) -> ThdnMatrix {
        let size A = A.size, sizeB = B.size;
        assert(sizeA.cols == sizeB.rows,
               "⚠️ A.size.cols should be equal to B.size.rows.");
        var C:[Thdn] = [Thdn](repeatElement(Thdn(re: 0.0),
                                count:sizeA.rows*sizeB.cols));
        var p:Int = 0;
        for i in 0..<sizeA.rows {
            for j in 0..<sizeB.cols { p = i*sizeB.cols + j;
                for n in 0..<sizeB.rows {C[p] +=
                    A.array[i*sizeA.cols + n]*B.array[n*sizeB.cols + j]}
            }
        }
        return ThdnMatrix(array: C, size: (rows: sizeA.rows, cols: sizeB.cols));
    }
    static func *=(lhs: inout ThdnMatrix, rhs: ThdnMatrix) {lhs = lhs * rhs}

//--METHODS:
    static func trans(A: ThdnMatrix)->ThdnMatrix{
       let size = A.size;
       var C:[Thdn] = [Thdn](repeatElement(Thdn (re: 0.0), count:A.array.count));
       for i in 0..<size.rows {
            for j in 0..<size.cols {C[j*size.rows + i] = A.array[i*size.cols + j]}
       }
       return ThdnMatrix(array: C, size: (rows: size.cols, cols: size.rows));
    }

    static func inverse(A: ThdnMatrix)->ThdnMatrix{
        assert(A.size.rows  == A.size.cols, "⚠️ matrix A must be square.");
        let parts = A.parts();
        let size = (rows:A.size.rows, cols:A.size.cols);
        let a1 = parts.im1, a2 = parts.im2;
        let a_1 = self.invertRe(matrix: parts.re);
        let α = multiplyRe(a: a_1, b: a1, sizeA: size, sizeB: size);
        var a1_1 =  ThdnMatrix.multiplyRe(a: α, b: a_1, sizeA: size, sizeB: size);
        a1_1 = a1_1.map {$0 * -1.0}
        var a2_1 = ThdnMatrix.multiplyRe(a: α,  b: α, sizeA: size, sizeB: size);
        a2_1 = ThdnMatrix.multiplyRe(a: a_1,  b: a2_1, sizeA: size, sizeB: size);
        a2_1 = a2_1.map {$0 * 2.0}
        var b = ThdnMatrix.multiplyRe(a: a2, b: a_1, sizeA: size, sizeB: size);
        b = ThdnMatrix.multiplyRe(a: a_1, b: b, sizeA: size, sizeB: size);
        a2_1 = a2_1 - b;
        var array:[Hdtn] = [];
        for i in 0..<A.array.count{array.append(Thdn(re: a_1[i],
            im1: a1_1[i], im2: a2_1[i]))}
      return ThdnMatrix(array: array, size: A.size)
    }
    static public func invertRe(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;
    }
    static public func multiplyRe (a:[Double], b:[Double],
                   sizeA:(rows:Int, cols:Int), sizeB:(rows:Int, cols:Int))->[Double]{
        var ab = [Double](repeating: 0.0, count: sizeA.rows*sizeB.cols);
        vDSP_mmulD(a, 1, b, 1, &ab, 1, vDSP_Length(sizeA.rows),
                                       vDSP_Length(sizeB.cols),
                                       vDSP_Length(sizeA.cols));
        return ab;
    }
    static private func transRe (a:[Double], size:(rows:Int, cols:Int))->[Double]{
        var a_t = [Double](repeating: 0.0, count: a.count)
        vDSP_mtransD(a, 1, &a_t, 1, vDSP_Length(size.cols), vDSP_Length(size.rows))
        return a_t;
    }
```

Приложение 3

```
   let A = ThdnMatrix (array:
            [Thdn (re: 1, im1: 0.1, im2: 0.5), Thdn (re: 2, im1: 0.2, im2: 0.6),
             Thdn (re: 3, im1: 0.3, im2: 0.7), Thdn (re: 4, im1: 0.4, im2: 0.8)],
             size: (rows:2, cols: 2));
  let A_1 = ThdnMatrix.inverse(A: AA);
  let A_1A = A_1*A;
//-- source matrix A:
//  1.0 + 0.1ε + 0.5ω;           2.0 + 0.2ε  + 0.6ω;
//  3.0 + 0.3ε + 0.7ω;           4.0 + 0.4ε  + 0.8ω;
//-- inverse matrix A_1:
// -2.0 + 0.2ε  - 0.04ω;        1.0 - 0.1ε  + 0.12ω;
//  1.5 - 0.15ε + 0.08ω;      -0.5 + 0.05ε - 0.16ω;
//-- product of matrices A_1*A:
// 1.0 + 0.0ε + 0.0ω;           0.0 + 0.0ε + 0.0ω;
// 0.0 + 0.0ε + 0.0ω;           1.0 + 0.0ε + 0.0ω;
let B = ThdnMatrix (array: [Thdn (re: 1.0, im1: 0.1, im2: 0.3),
                            Thdn (re: 2.0, im1: 0.5, im2: 0.6)],
                            size: (rows: 2, cols: 1));
let X = A_1*B;  // equation solution of A*X = B
// X = [0.0 + 0.3ε + 0.14ω;  0.5 - 0.15ε - 0.07ω;]
```

Приложение 4

Результаты анализа чувствительности гипер-дуального матричного уравнения.

Возмущение правой части		
$[A] = \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}$	$[A]_1 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$	$[A]_2 = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$
$\{B\} = \begin{bmatrix} 4 \\ 7 \end{bmatrix}$	$\{B\}_1 = \begin{bmatrix} 0.001 \\ 0.001 \end{bmatrix}$	$\{B\}_2 = \begin{bmatrix} 0.0005 \\ 0.0005 \end{bmatrix}$
$\{X\} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$	$\{X\}_1 = \begin{bmatrix} -0.001 \\ 0.001 \end{bmatrix}$	$\{X\}_2 = \begin{bmatrix} -0.0005 \\ 0.0005 \end{bmatrix}$

$\lVert\{X\}\rVert = 2.2361$	$\lVert\{X\}_1\rVert = 0.0447$	$\lVert\{X\}_2\rVert = 0.0007$
	$\dfrac{\lVert\{X\}_1\rVert}{\lVert\{X\}\rVert} = 0.01999$	$\dfrac{\lVert\{X\}_2\rVert}{\lVert\{X\}\rVert} = 0.00031$
Возмущение левой части		
$[A] = \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}$	$[A]_1 = \begin{bmatrix} 0.01 & 0.01 \\ 0.01 & 0.01 \end{bmatrix}$	$[A]_2 = \begin{bmatrix} 0.005 & 0.005 \\ 0.005 & 0.005 \end{bmatrix}$
$\{B\} = \begin{bmatrix} 4 \\ 7 \end{bmatrix}$	$\{B\}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\{B\}_2 = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$
$\{X\} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$	$\{X\}_1 = \begin{bmatrix} 0.03 \\ -0.03 \end{bmatrix}$	$\{X\}_2 = \begin{bmatrix} 0.015 \\ -0.015 \end{bmatrix}$
$\lVert\{X\}\rVert = 2.2361$	$\lVert\{X\}_1\rVert = 0.0424$	$\lVert\{X\}_2\rVert = 0.0212$
	$\dfrac{\lVert\{X\}_1\rVert}{\lVert\{X\}\rVert} = 0.01896$	$\dfrac{\lVert\{X\}_2\rVert}{\lVert\{X\}\rVert} = 0.00949$
Возмущение обеих частей		
$[A] = \begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}$	$[A]_1 = \begin{bmatrix} 0.01 & 0.01 \\ 0.01 & 0.01 \end{bmatrix}$	$[A]_2 = \begin{bmatrix} 0.005 & 0.005 \\ 0.005 & 0.005 \end{bmatrix}$
$\{B\} = \begin{bmatrix} 4 \\ 7 \end{bmatrix}$	$\{B\}_1 = \begin{bmatrix} 0.001 \\ 0.001 \end{bmatrix}$	$\{B\}_2 = \begin{bmatrix} 0.0005 \\ 0.0005 \end{bmatrix}$
$\{X\} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}$	$\{X\}_1 = \begin{bmatrix} 0.029 \\ -0.029 \end{bmatrix}$	$\{X\}_2 = \begin{bmatrix} 0.0145 \\ -0.0145 \end{bmatrix}$
$\lVert\{X\}\rVert = 2.2361$	$\lVert\{X\}_1\rVert = 0.0410$	$\lVert\{X\}_2\rVert = 0.0205$
	$\dfrac{\lVert\{X\}_1\rVert}{\lVert\{X\}\rVert} = 0.01835$	$\dfrac{\lVert\{X\}_2\rVert}{\lVert\{X\}\rVert} = 0.00917$

Табл. Результаты расчеты влияния возмущений на исходную систему

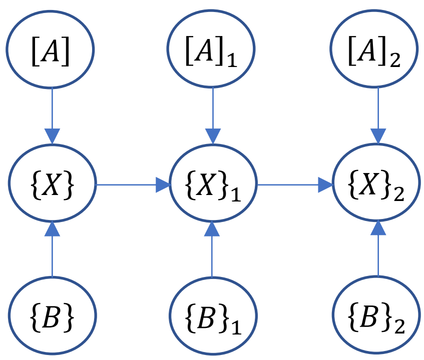

Рис. Схема влияния возмущений на исходную систему

Литература

1. *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, 2011.
2. *J. A. Fike.* Multi-Objective Optimization Using Hyper-Dual Numbers. Ph.D. Dissertation, Stanford University, 2013.
3. *Ф. М. Диментберг*. Теория винтов и ее приложения. – М.: Наука, 1978. – 328с.
4. *Gu You-Liang, Luh JYS*. Dual-number transformations and its applications to robotics. IEEE Journal of Robotics and Automation, RA-3(615-623), 1987.
5. *J. Angeles*. The application of dual algebra to kinematic analysis, eds J. Angeles and E. Zakhariev, editors, Computational Methods in Mechanical Systems, pp 3–32. Springer Verlag, New York, 1991.
6. *E. Pennestri, R. Stefanelli (2009).* Linear algebra and numerical algorithms using dual numbers. Multibody System Dynamics, 18:323–344, 2009.
7. *Р. Цей, М. М. Шумафов*. Число обусловленности матрицы как показатель устойчивости при решении прикладных задач. – Труды ФОРА, №16, 2011. 61–67с.
8. The Swift programming language 5.1 – *https://docs.swift.org/swift-book/#*

Абстракт

В данной статье представлен специальный тип чисел (усечённые гипер-дуальные числа). Даны основные алгебраические операции над этими числами. Рассматриваются матрицы и матричные уравнения компонентами которых служат усечённые гипер-дуальные числа. В приложениях 1, 2 и 3 приведена компьютерная реализация усечённых гипер-дуальных чисел и матриц для языка SWIFT операционной системы macOS. В приложении 4 даны результаты расчета чувствительности матричного уравнения с усечёнными гипер-дуальными компонентами.

Ключевые слова: *дуальные числа и матрицы, гипер-дуальные числа и матрицы, dual numbers and matrices, hyper-dual numbers and matrices*

---

18 января 2020 г.