ВЫЧИСЛЕНИЕ СОБСТВЕННЫХ ВЕКТОРОВ ГИПЕР-ДУАЛЬНЫХ МАТРИЦ

Олифер В.И.

1. Введение

Гипер-дуальные числа и матрицы (на их основе) представляют собой удобный инструмент для автоматического дифференцирования второго порядка и анализа чувствительности [1, 2]. В отличие от классической линейной алгебры над полем вещественных чисел, операции над гипер-дуальными объектами выполняются в кольце с нильпотентами [3, 4], что приводит к ряду особенностей при вычислении спектральных характеристик. В частности, определение собственных значений и собственных векторов требует модификации стандартных подходов, поскольку оператор первого порядка может не иметь нетривиального ядра даже при нулевом детерминанте. В данной работе рассматривается постановка задачи вычисления гипер-дуальных собственных векторов и предлагается эффективный численный алгоритм, ориентированный на практическое применение.

2. Основные соотношения алгебры гипер-дуальных чисел, векторов и матриц

Алгебра гипер-дуальных чисел 2-го порядка следует соотношениям, полученным в [1]:

$$\begin{aligned} &x = x_0 + x_1\boldsymbol{\varepsilon} + x_2\boldsymbol{\omega} \text{ и } x = y_0 + y_1\boldsymbol{\varepsilon} + y_2\boldsymbol{\omega}, \\ &x \pm y = x_0 \pm y_0 + (x_1 \pm y_1)\boldsymbol{\varepsilon} + (x_2 \pm y_2)\boldsymbol{\omega}, \\ &x{\cdot}y = x_0{\cdot}y_0 + (x_0{\cdot}y_1 + y_0{\cdot}x_1)\boldsymbol{\varepsilon} + (x_0{\cdot}y_2 + 2x_1{\cdot}y_1 + y_0{\cdot}x_2)\boldsymbol{\omega}, \\ &(x_0, x_1, x_2), (y_0, y_1, y_2) \in \mathbb{R}; \qquad \boldsymbol{\varepsilon^2} = 2\boldsymbol{\omega},\ \boldsymbol{\varepsilon\omega} = \boldsymbol{\omega^2} = 0, \end{aligned} \tag{1}$$

алгебра же гипер-дуальных векторов определяется формулами:

$$\begin{aligned} &\vec{V} = \vec{v}_0 + \vec{v}_1\boldsymbol{\varepsilon} + \vec{v}_2\boldsymbol{\omega},\ \ \vec{U} = \vec{u}_0 + \vec{u}_1\boldsymbol{\varepsilon} + \vec{u}_2\boldsymbol{\omega}, \\ &\vec{V} \pm \vec{U} = \vec{v}_0 \pm \vec{u}_0 + (\vec{v}_1 \pm \vec{u}_1)\boldsymbol{\varepsilon} + (\vec{v}_2 \pm \vec{u}_2)\boldsymbol{\omega}, \\ &\vec{V}{\cdot}\,\vec{U} = \vec{v}_0{\cdot}B_0 + (\vec{v}_0{\cdot}\vec{u}_1 + \vec{v}_1{\cdot}\vec{u}_0)\boldsymbol{\varepsilon} + (\vec{v}_0{\cdot}\vec{u}_2 + 2\vec{v}_1{\cdot}\vec{u}_1 + \vec{v}_2{\cdot}\vec{u}_0)\boldsymbol{\omega}, \end{aligned} \tag{2}$$

и, наконец, алгебра гипер-дуальных матриц согласно [2] имеет вид:

$$\begin{aligned} &A = A_0 + A_1\boldsymbol{\varepsilon} + A_2\boldsymbol{\omega},\ \ B = B_0 + B_1\varepsilon + B_2\boldsymbol{\omega}, \\ &A \pm B = A_0 \pm B_0 + (A_1 \pm B_1)\boldsymbol{\varepsilon} + (A_2 \pm B_2)\boldsymbol{\omega}, \\ &A{\cdot}B = A_0{\cdot}B_0 + (A_0{\cdot}B_1 + A_1{\cdot}B_0)\boldsymbol{\varepsilon} + (A_0{\cdot}B_2 + 2A_1{\cdot}B_1 + A_2{\cdot}B_0)\boldsymbol{\omega}, \end{aligned} \tag{3}$$

где матриц A_0, A_1, A_2 и B_0, B_1, $B_2 \in \mathbb{R}_{n\times n}$; векторы $\vec{v}_0, \vec{v}_1, \vec{v}_2$ и $\vec{u}_0, \vec{u}_1, u_2 \in \mathbb{R}_{n\times 1}$, а матрицы A_1 и A_2 можно трактовать, как матрицы возмущений 1-го и 2-го рода соответственно. Детерминант гипер-дуальной матрицы раскладывается линейно по нильпотентным компонентам: $\det(A) = \det(A_0) + \operatorname{tr}(\operatorname{adj}(A_0)\,A_1)\boldsymbol{\varepsilon} + \operatorname{tr}(\operatorname{adj}(A_0)\,A_2)\boldsymbol{\omega}$;

где: $\mathrm{adj}(A_0)$ – присоединённая (союзная) матрица; $\mathrm{tr}(\mathrm{C}) = C^T$ – транспонированная матрица [7].

Компоненты гипер-дуальных объектов с индексом 0 называются главной частью и обозначаются через Re, а с индексами 1 и 2 – мнимыми частями (Im_1, Im_2). Гипер-дуальный объект с нулевой главной частью называется нильпотентным или делителем нуля, например, $x = 0 + + x_1\boldsymbol{\varepsilon} + x_2\boldsymbol{\omega}$.

3. Постановка задачи определения собственного вектора при заданном собственном числе

Пусть $\lambda = \lambda_0 + \lambda_1\boldsymbol{\varepsilon} + \lambda_2\boldsymbol{\omega}$ – заранее найденное собственное значение гипер-дуальной матрицы B (см., например, [5]), соответствующее уравнению $\det(B - \lambda I) = 0$, где I – единичная матрица размера $n \times n$ над $\mathbb{R}$.

Требуется найти ненулевой вектор $\vec{V} = \vec{v}_0 + \vec{v}_1\boldsymbol{\varepsilon} + \vec{v}_2\boldsymbol{\omega}$, удовлетворяющий спектральному уравнению $(B - \lambda I)\vec{V} = 0$.

При вычислении вектора $\vec{V}$ для заданного значения λ важно учитывать, что в алгебре гипер-дуальных чисел оператор $(B - \lambda I)$ может содержать нильпотентные компоненты. В этом случае λ, найденное из уравнения $\det(B - \lambda I) = 0$, является корректным алгебраическим собственным значением, однако оператор первого порядка $(B - \lambda I)$ не обязан иметь нетривиальное ядро. Поэтому решение уравнения $(B - \lambda I)\vec{V} = 0$ обычно приводит к ненулевой невязке, а найденный вектор $\vec{V}$ фактически удовлетворяет более слабому условию ($(B - \lambda I)^2\vec{V} \approx 0$ (см. Приложение 1). Такое поведение типично для операторов с нильпотентной частью и означает, что вычисляемое по $\vec{V}$ значение λ следует интерпретировать как *обобщённое собственное значение*, а сам $\vec{V}$ – как *обобщённый собственный вектор*, что естественно для линейных систем над кольцами с нильпотентами.

В классической линейной алгебре условие $\det(B - \lambda I) = 0$ эквивалентно существованию собственного вектора. В гипер-дуальной алгебре это не так, и такое совпадение бывает редко.

4. Метод решения

Подставляя разложение матрицы B, вектора $\vec{V}$ и λ в уравнение $(B - \lambda I)\vec{V} = 0$ и группируя компоненты по степеням $\boldsymbol{\varepsilon}$ и $\boldsymbol{\omega}$, получаем системы уравнений:

$$\begin{aligned}
&(B - \lambda I)\vec{V} = (B_0 - \lambda_0 I)\vec{v}_0 + \\
&+[(B_0 - \lambda_0 I)\vec{v}_1 + (B_1 - \lambda_1 I)\vec{v}_0]\boldsymbol{\varepsilon} + \\
&+ [(B_0 - \lambda_0 I)\vec{v}_2 + (B_2 - \lambda_2 I)\vec{v}_0 + 2(B_1 - \lambda_1 I)\vec{v}_1\ ]\boldsymbol{\omega} = \\
&= A_0\vec{v}_0 + (A_0\vec{v}_1 + A_1\vec{v}_0)\boldsymbol{\varepsilon} + (A_0\vec{v}_2 + A_2\vec{v}_0 + 2A_1\vec{v}_1)\boldsymbol{\omega} \Longrightarrow
\end{aligned} \tag{4}$$

$$\Rightarrow \begin{cases} A_0\vec{v}_0 = 0; & \text{Уровень } \mathbf{0} \\ A_0\vec{v}_1 = -A_1\vec{v}_0; & \text{Уровень } \boldsymbol{\varepsilon} \\ A_0\vec{v}_2 = -(A_2\vec{v}_0 + 2A_1\vec{v}_1); & \text{Уровень } \boldsymbol{\omega} \end{cases}$$

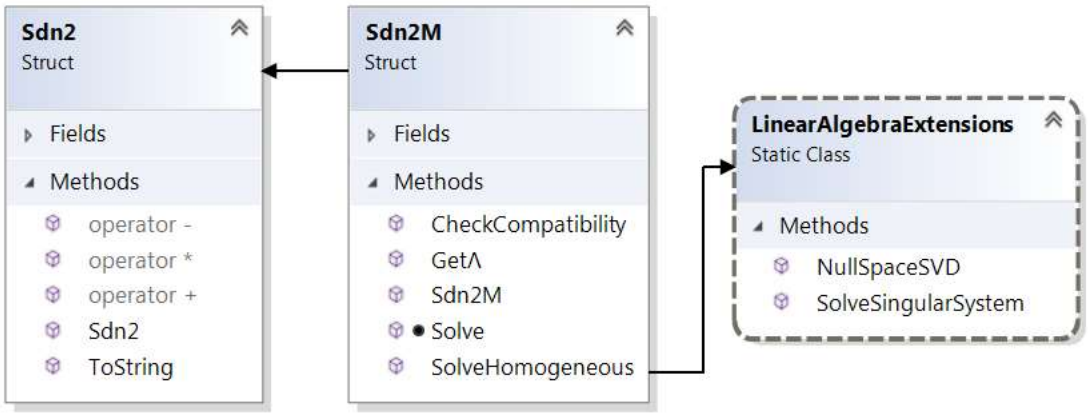
Первая система (Уровень **0**) определяет классический собственный вектор $\vec{v}_0$.
Вторая (Уровень $\boldsymbol{\varepsilon}$) и третья (Уровень $\boldsymbol{\omega}$) – линейные системы для $\vec{v}_1$ и $\vec{v}_2$, которые могут быть несовместны в обычном смысле. Это фундаментальное отличие гипер-дуальной алгебры от обычной.
Поэтому решение ищется в смысле минимизации невязки, что соответствует вычислению обобщённого собственного вектора.

5. Алгоритм решения

1. Определить гипер-дуальное собственное значение λ из $\det(B - \lambda I) = 0$.
2. Решить систему $(B_0 - \lambda_0 I)\vec{v}_0$ и нормировать $\vec{v}_0$.
3. Решить систему $A_0\vec{v}_1 = -A_1\vec{v}_0$ для $\vec{v}_1$ в смысле минимальной невязки.
4. Решить систему $A_0\vec{v}_2 = -(A_2\vec{v}_0 + 2A_1\vec{v}_1)$ для $\vec{v}_2$.
5. Собрать гипер- дуальный вектор $\vec{V} = \vec{v}_0 + \vec{v}_1\boldsymbol{\varepsilon} + \vec{v}_2\boldsymbol{\omega}$.
6. Проверить условие невязки $\vec{r} = \vec{R} = (B - \lambda I)\,\vec{V} \approx 0$ (см. Приложение 2).
7. Пересчитать собственное число λ (см. Приложение 3).

6. Численная реализация

Для численной реализации удобно использовать библиотеку MathNet.Numerics [8].
Исходный код на C# дан в Приложении 4.
Объектная диаграмма исходного кода приведена на следующем рисунке.



Как видно, исходный код состоит из двух структур (`Sdn2M`, `Sdn2`) и одного класса `LinearAlgebraExtensions`. Структура `Sdn2M` является главной, использующей структуру `Sdn2` и класс `LinearAlgebraExtensions`. Главный метод `Solve()` структуры `Sdn2M` осуществляет поиск решения при заданных матрице B и λ, посредством процедуры `SolveHomogeneous()`, которая использует методы `NullSpaceSVD()` и

`SolveSingularSystem()` класса `LinearAlgebraExtensions`. Затем по прцедуре `CheckCompatibility()` осуществляется поиск невязок полученного решения $\vec{V}$, и если невязки превышают допустимое значение происходит корректировка величины λ.
Класс `LinearAlgebraExtensions` использует SVD или QR-разложение [6], что обеспечивает устойчивость при вырожденности матрицы $B_0 - \lambda_0 I$.
Детерминант гипер-дуальной матрицы вычисляется по формуле

$$\mathrm{D} = \det(B) = \mathrm{d}_0 + \mathrm{d}_1\boldsymbol{\varepsilon} + \mathrm{d}_2\boldsymbol{\omega},$$

где: $\mathrm{d}_0 = \det(B_0)$, $\mathrm{d}_1 = \mathrm{tr}(\mathrm{adj}(B_0)\, B_1)$, $\mathrm{d}_2 = \mathrm{tr}(\mathrm{adj}(B_0)\, B_2)$

7. Пример расчёта

Рассмотрим гипер-дуальную матрицу $B = B_0 + B_1\boldsymbol{\varepsilon} + B_2\boldsymbol{\omega}$,
где: B_0, B_1, B_2 – заданные 3×3 матрицы:

$$B_0 = \begin{bmatrix} 5 & 1 & 4 \\ 3 & 3 & 2 \\ 6 & 2 & 10 \end{bmatrix}, \qquad B_1 = \begin{bmatrix} -5 & -1 & -4 \\ 3 & 3 & 2 \\ 6 & 2 & 10 \end{bmatrix}, \qquad B_2 = \begin{bmatrix} -5 & 1 & 4 \\ -3 & 3 & 2 \\ -6 & 2 & 10 \end{bmatrix}$$

А гипер-дуальное собственное значение $\lambda = \lambda_0 + \lambda_1\boldsymbol{\varepsilon} + \lambda_2\boldsymbol{\omega}$ матрицы B, которое является корнем характеристического уравнения $\det(B - \lambda I) = 0$, равным $[2 - 2\boldsymbol{\varepsilon} + 58\boldsymbol{\omega}]$.
В результате расчёта проведенному по программному обеспечению (см. Приложение 4) выяснелось, что при заданном λ решение $\vec{U}$ уравнения $(B - \lambda I)\vec{V} = 0$ имеет невязки превышающие допустимые значения: $r_0 \approx \{0,0,0\}$, $r_1 \approx \{0,0,0\}$, $r_2 \approx \{4.2,\ 0, -4.2\}$. Тогда программа пересчитала λ исходя из условия $(B - \lambda I)\vec{U} = 0$ и получила новое значение λ, которое вообще говоря не является корнем характеристического уравнения $\det(B - \lambda I) = 0$. Результаты расчёта приведены в следующей таблице

	Re	*Im1*	*Im1*
λ	2.0	-2.0	55.11999999…
	x	*y*	*z*
$\vec{v}_0$	-0.31622776	0.94868329	0.00000000
$\vec{v}_1$	-2.27683991	-0.75894663	1.89736659
$\vec{v}_2$	40.76909552	13.58969850	-36.86962769

Таблица результатов расчета

Это как раз демонстрирует тот случай, когда исходный корень уравнения $\det(B - \lambda I) = 0$ не обеспечивает нулевые невязки решения уравнения $(B - \lambda I)\vec{V} = 0$.

8. Выводы

• Спектральная задача для гипер дуальных матриц существенно отличается от классической.

• Собственные значения, найденные из характеристического уравнения, являются алгебраическими, но не гарантируют существования классического собственного вектора.

• Предложенный алгоритм обеспечивает устойчивое вычисление гипер дуальных собственных векторов и может использоваться в задачах автоматического дифференцирования и анализа чувствительности.

В гипер-дуальной алгебре:

• корень характеристического полинома не гарантирует существование собственного вектора;

• корректное гипер-дуальное собственное значение определяется не полиномом, а *условием совместимости уровней* $\boldsymbol{\varepsilon}$ *и* $\boldsymbol{\omega}$;

• совпадение этих двух определений – исключение, а не правило.

9. Заключение

Разработанный метод вычисления собственных векторов гипер дуальных матриц позволяет корректно учитывать нильпотентную структуру алгебры и обеспечивает устойчивость численных процедур. Представленный подход может быть расширен на матрицы большего размера, а также использован в задачах оптимизации, моделирования и вычислительной физики, где требуется анализ вариаций второго порядка.

Хочется также отметить следующее. В строительной механике есть фундаментальный принцип:«Оптимальное решение под статические нагрузки не является оптимальным под динамические».

Тогда имеет место следующая структурная аналогия (физическая интерпретация)

Строительная механика	**Гипер-дуальная алгебр**а
Статика → одно условие	Уровень $\mathbf{0}$ → одно условие
Динамика → доп. условия совместности	Уровни $\boldsymbol{\varepsilon}$ и $\boldsymbol{\omega}$ → доп. условия совместности
Оптимум по статике ≠ оптимум по динамике	Корень $\det(B-\lambda I)=0$ не обеспечивает нулевые невязки решения $(B-\lambda I)\vec{V}=0$
Система может быть «вырожденной» под динамикой	A_0 может быть вырожденной, A_1 и A_2 несовместны

ПРИЛОЖЕНИЕ 1.

Обоснование условия второго порядка $\boldsymbol{(B-\lambda I)^2\vec{V}\approx 0}$.

Уравнение первого порядка может быть несовместно, даже если λ – корректное собственное значение. Матрица $(B-\lambda I)\overrightarrow{V}$ сингулярна, её ядро одномерно, но правые части уравнений для $\vec{v}_1$ и $\vec{v}_2$ могут не лежать в её образе.

Пусть невязка: $\vec{R} = (B-\lambda I)\overrightarrow{V} = 0 + \vec{r}_1\boldsymbol{\varepsilon} + \vec{r}_2\boldsymbol{\omega}$.

Она всегда нильпотентна. Применим оператор ещё раз: $(B-\lambda I)\vec{R} = (B-\lambda I)^2\vec{V}$.

При раскрытии выражения появляются члены с $\boldsymbol{\varepsilon^2}$, $\boldsymbol{\varepsilon\omega}$, $\boldsymbol{\omega^2}$, но $\boldsymbol{\varepsilon^2} = \boldsymbol{\varepsilon\omega} = \boldsymbol{\omega^2} = 0$.

Поэтому все члены второго порядка исчезают или переходят в элементы порядка $\boldsymbol{\omega}$, которые считаются малыми. Таким образом: $(B-\lambda I)^2\vec{V} \approx 0$.

Это аналог обобщённого собственного вектора в теории жордановых цепочек [4].

ПРИЛОЖЕНИЕ 2.

Определение невязок решения

Если решение системы $A\vec{V} = 0$ равно $\vec{U} = \vec{u}_0 + \vec{u}_1\boldsymbol{\varepsilon} + \vec{u}_2\boldsymbol{\omega}$, то невязка решения первого порядка имеет вид $r = r_0 + r_1\boldsymbol{\varepsilon} + r_2\boldsymbol{\omega} \le 10^{-12}$, где:

$$\begin{cases} r_0 = A_0\vec{u}_0 \le 10^{-12} \\ r_1 = (A_0\vec{u}_1 + A_1\vec{u}_0) \le 10^{-12} \\ r_2 = (A_0\vec{u}_2 + B_2\vec{u}_0 + 2A_1\vec{u}_1) \le 10^{-12} \end{cases}$$

Для невязки второго порядка

$$A^2\vec{U} = (A_0^2 + 2A_0A_1\boldsymbol{\varepsilon} + 2(A_0A_2 + A_1^2)\boldsymbol{\omega})(\vec{u}_0 + \vec{u}_1\boldsymbol{\varepsilon} + \vec{u}_2\boldsymbol{\omega}) =$$
$$= A_0^2\vec{u}_0 + (A_0^2\vec{u}_1 + 2A_0A_1\vec{u}_0)\boldsymbol{\varepsilon} + (A_0^2\vec{u}_2 + 4A_0A_1\vec{u}_1 + 2(A_0A_2 + A_1^2)\vec{u}_0)\boldsymbol{\omega},$$

и тогда

$$\begin{cases} R_0 = A_0^2\vec{u}_0 \le 10^{-12} \\ R_1 = (A_0^2\vec{u}_1 + 2A_0A_1\vec{u}_0) \le 10^{-12} \\ R_2 = (A_0^2\vec{u}_2 + 4A_0A_1\vec{u}_1 + 2(A_0A_2 + A_1^2)\vec{u}_0) \le 10^{-12} \end{cases}$$

ПРИЛОЖЕНИЕ 3.

Корректировка характеристического числа

Обозначим новую величину характеристического числа через $\lambda^* = \lambda_0^* + \lambda_1^*\boldsymbol{\varepsilon} + \lambda_2^*\boldsymbol{\omega}$.

Тогда $A\vec{U} = 0 \longrightarrow \vec{u}_0A\vec{U} = 0 \longrightarrow$

$$\longrightarrow \vec{u}_0A\vec{U} = \vec{u}_0A_0\vec{u}_0 + \vec{u}_0(A_0\vec{u}_1 + A_1\vec{u}_0)\boldsymbol{\varepsilon} + \vec{u}_0(A_0u_2 + A_2\vec{u}_0 + 2A_1\vec{u}_1)\boldsymbol{\omega} =$$
$$= \vec{u}_0(B_0 - \lambda_0 I)\vec{u}_0 + \vec{u}_0(A_0\vec{u}_1 + (B_1 - \lambda_1 I)\vec{u}_0)\boldsymbol{\varepsilon} + \vec{u}_0(A_0\vec{u}_2 + (B_2 - \lambda_2 I)\vec{u}_0 + 2A_1\vec{u}_1)\boldsymbol{\omega} = 0$$

Откуда следует:

$$\begin{cases} \lambda_0^* = \vec{u}_0(B_0\vec{u}_0)/\|\vec{u}_0\| \\ \lambda_1^* = \vec{u}_0(A_0\vec{u}_1 + B_1\vec{u}_0)/\|\vec{u}_0\| \\ \lambda_2^* = \vec{u}_0(A_0\vec{u}_2 + B_2\vec{u}_0 + 2A_1\vec{u}_1)/\|\vec{u}_0\| \end{cases}$$

ПРИЛОЖЕНИЕ 4.

Код программы на C#

```
using System;
using System.Collections.Generic;
using System.Diagnostics;
using System.Linq;
using MathNet.Numerics.LinearAlgebra;

public struct Sdn2
{ public double Re, Im1, Im2;
  public Sdn2(double x, double x1, double x2) { Re = x; Im1 = x1; Im2 = x2;}

  public static Sdn2 operator *(Sdn2 X, Sdn2 Y)
  { return new Sdn2(X.Re * Y.Re,
                    X.Re * Y.Im1 + Y.Re * X.Im1,
                    X.Re * Y.Im2 + 2.0 * X.Im1 * Y.Im1 + Y.Re * X.Im2);}

 public static Sdn2 operator +(Sdn2 X, Sdn2 Y)
 { return new Sdn2(X.Re + Y.Re, X.Im1 + Y.Im1, X.Im2 + Y.Im2);}

 public static Sdn2 operator -(Sdn2 X, Sdn2 Y)
 { return new Sdn2(X.Re - Y.Re, X.Im1 - Y.Im1, X.Im2 - Y.Im2);}

 public override string ToString() => $"{Re} + {Im1}ε + {Im2}ω";
}

public struct Sdn2M
{ public Matrix<double> A0, A1, A2;
  public Matrix<double> B0, B1, B2;
  private double[] λ;
  public Sdn2M(double[,] M, double[,] M1, double[,] M2, double[] λ )
  { this.λ = λ;
    B0 = Matrix<double>.Build.DenseOfArray(M);
    B1 = Matrix<double>.Build.DenseOfArray(M1);
    B2 = Matrix<double>.Build.DenseOfArray(M2);
    for (int i=0; i < B0.RowCount; i++){M[i,i]-=λ[0]; M1[i,i]-=λ[1]; M2[i,i]-=λ[2];}
    A0 = Matrix<double>.Build.DenseOfArray(M);
    A1 = Matrix<double>.Build.DenseOfArray(M1);
    A2 = Matrix<double>.Build.DenseOfArray(M2);
   }

   public (double[], List<double[]>) Solve()
   { double[,] b1 = B0.ToArray(), b2 = B1.ToArray(), b3 = B2.ToArray();
     var V = new List<Sdn2[]>();  var  b = new Sdn2M(b1, b2, b3, λ);
     V = b.SolveHomogeneous();
     if (b.CheckCompatibility(V).Max() >= 1e-12) λ = b.GetΛ(V);
     var Q = new List<double[]>{V[0].Select(x => x.Re).ToArray(),
                                V[0].Select(x => x.Im1).ToArray(),
                                V[0].Select(x => x.Im2).ToArray()};
     return (λ, Q);
   }

   public List<Sdn2[]> SolveHomogeneous()
   {  int n = A0.ColumnCount;
      var basisX0 = LinearAlgebraExtensions.NullSpaceSVD(A0);
```

```
      var solutions = new List<Sdn2[]>();
      foreach (var x0 in basisX0)
      {  // ε-компонента: A0 x1 = -A1 x0
         var rhs1 = -(A1 * x0);
         var x1 = LinearAlgebraExtensions.SolveSingularSystem(A0, rhs1);
         // ω-компонента: A0 x2 = -A2 x0 - 2 A1 x1
         var rhs2 = -(A2 * x0 + 2.0 * (A1 * x1));
         var x2 = LinearAlgebraExtensions.SolveSingularSystem(A0, rhs2);
         // Собираем гипердуальный вектор
         var vec = new Sdn2[n];
         for (int i = 0; i < n; i++) vec[i] = new Sdn2(x0[i], x1[i], x2[i]);
          solutions.Add(vec);
       }
      return solutions;
   }

   public double [] CheckCompatibility( List<Sdn2[]> V)
   {   var V0 = Vector<double>.Build.Dense(3, i => V[0][i].Re);
       var V1 = Vector<double>.Build.Dense(3, i => V[0][i].Im1);
       var V2 = Vector<double>.Build.Dense(3, i => V[0][i].Im2);
       Vector<double> r0 = (A0 * V0);
       Vector<double> r1 = (A0 * V1 + A1 * V0);
       Vector<double> r2 = A0 * V2 + 2.0 * (A1 * V1) + A2 * V0;
     return new double[] { r0.L1Norm(), r1.L1Norm(), r2.L1Norm()};
    }

    public  double [] GetΛ(List<Sdn2[]> U)
    {   var U0 = Vector<double>.Build.Dense(3, i => U[0][i].Re);
        var U1 = Vector<double>.Build.Dense(3, i => U[0][i].Im1);
        var U2 = Vector<double>.Build.Dense(3, i => U[0][i].Im2);
        var U02 = U0 * U0;
        double λ0 = U0 * (B0 * U0)/ U02;
        double λ1 = U0 * (A0*U1 + B1*U0) / U02;
        double λ2 = U0 * (A0*U2 + 2.0*(A1*U1)+ B2*U0) / U02;
      return  new double[] { λ0, λ1, λ2 };
    }
}

public static class LinearAlgebraExtensions
 {   // Ядро матрицы через SVD: A·x = 0
   public static List<Vector<double>> NullSpaceSVD(Matrix<double> A, double tol = 1e-12)
   {   var svd = A.Svd(true);
       var S = svd.S;
       var V = svd.VT.Transpose();
       var basis = new List<Vector<double>>();
       // Нулевые сингулярные значения → ядро
       for (int i = 0; i < S.Count; i++) { if (S[i] < tol) basis.Add(V.Column(i)); }
        // Если ядро пустое — возвращаем нулевой вектор
       if (basis.Count == 0)
           basis.Add(Vector<double>.Build.Dense(A.ColumnCount, 0.0));
      return basis;
    }
  // Решение вырожденной системы A·x = b через псевдообратную
  // 1) Находим ненулевые сингулярные значения
  // 2) Проецируем b на образ A
```

```
  // 3) Строим псевдообратную A⁺
  // 4) x = A⁺·b_proj — решение минимальной нормы
   public static Vector<double> SolveSingularSystem(Matrix<double> A, Vector<double> b,
                                double tol = 1e-12)
    {  var svd = A.Svd(true);
       var U = svd.U; var S = svd.S; var Vt = svd.VT;
       int m = A.RowCount,  n = A.ColumnCount;
       // Индексы ненулевых сингулярных значений
       var nonZero = new List<int>();
       for (int i = 0; i < S.Count; i++) if (S[i] > tol) nonZero.Add(i);
       int r = nonZero.Count;
       // U_r — столбцы U, соответствующие ненулевым сингулярным значениям
       var U_r = Matrix<double>.Build.Dense(m, r);
       for (int k = 0; k < r; k++)
          { int idx = nonZero[k];  U_r.SetColumn(k, U.Column(idx));}
       // Проекция b на образ A
       var b_proj = U_r * (U_r.Transpose() * b);
       // Псевдообратная S⁺
       var Splus = Matrix<double>.Build.Dense(n, m, 0.0);
       for (int i = 0; i < r; i++)
            { int idx = nonZero[i];  Splus[idx, idx] = 1.0 / S[idx];}
            var Aplus = Vt.Transpose() * Splus * U.Transpose();
     // Решение минимальной нормы
     return Aplus * b_proj; ;
   }
}

//ПРИМЕР РАСЧЁТА
double[,] b1 = {{  5,  1,  4 }, {  3, 3, 2 }, {  6, 2, 10 } },
          b2 = {{ -5, -1, -4 }, {  3, 3, 2 }, {  6, 2, 10 } },
          b3 = {{ -5,  1,  4 }, { -3, 3, 2 }, { -6, 2, 10 } };
double[] λ = { 2, -2, 58.0 };
var B = new Sdn2M(b1, b2, b3, λ);
var Q  = B.Solve();
var Λ  = Q.Item1;     // [2.0, -2.0, 55.119999999999962]               Λ
var V0 = Q.Item2[0];  // [-0.31622776,  0.94868329, 0]           Re
var V1 = Q.Item2[1];  // [-2.27683991, -0.75894663, 1.89736659]  Im1
var V2 = Q.Item2[2];  // [42.46306442, 14.15435480,-38.32680524] Im2
```

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, – 2020.

2. *Олифер В.И.* Гипер-дуальные матричные уравнения и их чувствительность. – URL: http://viosolutions.amerihomesrealty.com/pdf/гипер-дуальные матрицы.pdf, – 2020.

3. *McDonald, Bernard R.* Linear Algebra over Commutative Rings. –Monographs and Textbooks in Pure and Applied Mathematics, Vol. 87. – Marcel Dekker, New York, 1984.

4. *Можей Н. П.* Линейные алгебры Ли, состоящие из нильпотентных эндоморфизмов. – Труды БГТУ, 2020, серия 3, № 1, с. 20–25.

5. *Олифер В.И.* Характеристические многочлены матриц над гипер-дуальной алгеброй с соотношениями $\boldsymbol{\varepsilon^2 = 2\omega,\ \varepsilon\omega = \omega^2 = 0}$. – URL: https://viosolutions.amerihomesrealty.com/pdf/Характеристические многочлены матриц.pdf, – 2026.

6. *Зубов Н.Е., Микрин Е.А., Рябченко В.Н.* О вычислении псевдообратной квадратной матрицы на основе обращения // Вестник МГТУ им. Н.Э. Баумана. Сер. Естественные науки. 2018. № 3. С. 24–31. DOI: 10.18698/1812-3368-2018-3-24-31

7. *Курош А.Г.* Курс высшей алгебры. – Москва : Наука, 1968. – 431 с. –1968.

8. Math.NET Numerics – https://www.nuget.org/packages/MathNet.Numerics

Абстракт

В статье рассматривается метод определения собственных векторов, соответствующих спектральному уравнению над усечённой гипер-дуальной алгеброй, определяемой отношениями $\boldsymbol{\varepsilon^2 = 2\omega,\ \varepsilon\omega = \omega^2 = 0}$. *Приведен код программного обеспечения и рассмотрены численные примеры, полученные на его основе.*

This article examines a method for determining eigenvectors corresponding to a spectral equation over a truncated hyper-dual algebra defined by the relations $\boldsymbol{\varepsilon^2 = 2\omega,\ \varepsilon\omega = \omega^2 = 0}$. *Software code is provided, along with numerical examples generated using it.*

Ключевые слова: *спектральное уравнение, собственные векторы, гипре-дуальные числа и матрицы, spectral equation, eigenvectors, hyper-dual numbers and matrices.*

20 апреля 2026 г