ВОЗМУЩЕНИЯ В ТЕОРИИ КОЛЕБАНИЙ С ГИПЕР-ДУАЛЬНОЙ АЛГЕБРОЙ $\boldsymbol{\varepsilon^2 = 2\omega, \varepsilon\omega = \omega^2 = 0}$

Олифер В.И

В работах [1] и [2] описаны варианты определения собственных значений и векторов соответственно, когда основное уравнение теории колебаний определено над алгеброй гипер-дуальных чисел, что позволяет учитывать возмущения первого и второго рода. Иными словами, собственные векторы, согласно [2], определяются по предварительно найденным собственным числам согласно [1]. При этом, в общем случае получаемые собственные векторы не всегда дают нулевую невязку, а значится требуются корректировка величин некоторых исходных собственных чисел, участвующих в уравнениях возмущения первого и второго рода.

В настоящей работе рассматривается более общий подход предполагающий, что все компоненты основного уравнения теории колебаний являются гипер-дуальными объектами и более того, некоторые из них зависят от гипер-дуального параметра, что позволяет учитывать изменения происходящие в окружающей среде.

1. Гипер-дуальные объекты

Алгебра гипер-дуальных чисел 2-го порядка или тензоров 0-го ранга (HDN) следует соотношениям, полученных в [1, 2]:

$$\begin{aligned}
&X = x_0 + x_1\boldsymbol{\varepsilon} + x_2\boldsymbol{\omega} \text{ и } Y = y_0 + y_1\boldsymbol{\varepsilon} + y_2\boldsymbol{\omega},\\
&X \pm Y = x_0 \pm y_0 + (x_1 \pm y_1)\boldsymbol{\varepsilon} + (x_2 \pm y_2)\boldsymbol{\omega},\\
&X{\cdot}Y = x_0{\cdot}y_0 + (x_0{\cdot}y_1 + y_0{\cdot}x_1)\boldsymbol{\varepsilon} + (x_0{\cdot}y_2 + 2x_1{\cdot}y_1 + y_0{\cdot}x_2)\boldsymbol{\omega},\\
&(x_0, x_1, x_2), (y_0, y_1, y_2) \in \mathbb{R},
\end{aligned} \tag{1.1}$$

а алгебра гипер-дуальных векторов (HDV) задаётся следующими формулами:

$$\begin{aligned}
&\vec{X} = \vec{x}_0 + \vec{x}_1\boldsymbol{\varepsilon} + \vec{x}_2\boldsymbol{\omega} \text{ и } \vec{Y} = \vec{y}_0 + \vec{y}_1\boldsymbol{\varepsilon} + \vec{y}_2\boldsymbol{\omega},\\
&\vec{X} \pm \vec{Y} = \vec{x}_0 \pm \vec{y}_0 + (\vec{x}_1 \pm \vec{y}_1)\boldsymbol{\varepsilon} + (\vec{x}_2 \pm \vec{y}_2)\boldsymbol{\omega},\\
&\vec{X}{\cdot}\vec{Y} = \vec{x}_0{\cdot}\vec{y}_0 + (\vec{x}_0{\cdot}\vec{y}_1 + \vec{y}_0{\cdot}\vec{x}_1)\boldsymbol{\varepsilon} + (\vec{x}_0{\cdot}\vec{y}_2 + 2\vec{x}_1{\cdot}\vec{y}_1 + \vec{y}_0{\cdot}\vec{x}_2)\boldsymbol{\omega},\\
&(\vec{x}, \vec{x}_1, \vec{x}_2), (\vec{y}, \vec{y}_1, \vec{y}_2) \in \mathbb{R}_3,
\end{aligned} \tag{1.2}$$

и наконец, алгебра гипер-дуальных матриц (HDM) определяется как [2]:

$$\begin{aligned}
&\boldsymbol{A} = \boldsymbol{A_0} + \boldsymbol{A_1}\boldsymbol{\varepsilon} + \boldsymbol{A_2}\boldsymbol{\omega}, \quad \boldsymbol{B} = \boldsymbol{B_0} + \boldsymbol{B_1}\boldsymbol{\varepsilon} + \boldsymbol{B_2}\boldsymbol{\omega},\\
&\boldsymbol{A} \pm \boldsymbol{B} = \boldsymbol{A_0} \pm \boldsymbol{B_0} + (\boldsymbol{A_1} \pm \boldsymbol{B_1})\boldsymbol{\varepsilon} + (\boldsymbol{A_2} \pm \boldsymbol{B_2})\boldsymbol{\omega},\\
&\boldsymbol{A}{\cdot}\boldsymbol{B} = \boldsymbol{A_0}{\cdot}\boldsymbol{B_0} + (\boldsymbol{A_0}{\cdot}\boldsymbol{B_1} + \boldsymbol{A_1}{\cdot}\boldsymbol{B_0})\boldsymbol{\varepsilon} + (\boldsymbol{A_0}{\cdot}\boldsymbol{B_2} + 2\boldsymbol{A_1}{\cdot}\boldsymbol{B_1} + \boldsymbol{A_2}{\cdot}\boldsymbol{B_0})\boldsymbol{\omega},
\end{aligned} \tag{1.3}$$

где матрицы $\boldsymbol{A_0}$, $\boldsymbol{A_1}$, $\boldsymbol{A_2}$ и $\boldsymbol{B}_0$, $\boldsymbol{B_1}$, $\boldsymbol{B_2}$ $\in \mathbb{R}_{3\times3}$, причем $\boldsymbol{A_1}$, $\boldsymbol{A_2}$ и $\boldsymbol{B_1}$, $\boldsymbol{B_2}$ можно трактовать как матрицы возмущений 1-го и 2-го рода соответственно. Если матрицы с индексом 0 не вырожденные то, например, $\boldsymbol{A^{-1} = A_0^{-1} - A_0^{-1}\cdot C\varepsilon + A_0^{-1}\cdot(2C^2 - A_2\cdot A_0^{-1})\omega}$, $\boldsymbol{C = A_1\cdot A_0^{-1}}$.

И, наконец, в случае когда элементы матриц $\boldsymbol{A}_0$ зависят от некоторого параметра t, то её следует рассматривать как гипер-дуальную матричную функцию, которая после разложение $\boldsymbol{A}(t)$ в ряд Тейлора имеет вид:

$$\boldsymbol{A}_0(t + 1\boldsymbol{\varepsilon}) = \boldsymbol{A}_0(t) + \dot{\boldsymbol{A}}_0(t)\boldsymbol{\varepsilon} + \ddot{\boldsymbol{A}}_0(t)\boldsymbol{\omega} \;\rightarrow \boldsymbol{A}_1(t) = \dot{\boldsymbol{A}}_0(t)\;,\;\; \boldsymbol{A}_2(t) = \ddot{\boldsymbol{A}}_0(t),$$

где: $\dot{\boldsymbol{A}}_0(t) = \frac{d\boldsymbol{A}_0(t)}{dt}$, $\ddot{\boldsymbol{A}}_0(t) = \frac{d^2\boldsymbol{A}_0(t)}{dt^2}$

2. Основное уравнение теории колебания в терминах гипер-дуальной алгебры

Классическое уравнение малых собственных колебаний [3, 4] имеет вид $(K - \omega^2 M)\vec{\Phi} = 0$, где K и M – матрицы жесткости и инерции системы (в евклидовом пространстве), ω – частота колебаний, причем $\omega^2 = \lambda$ – собственное значение, и, наконец, $\vec{\Phi}$ – вектор собственных колебаний. Трансформация это уравнения в гипер-дуальное пространство дает

$$(\boldsymbol{K} - \Lambda\boldsymbol{M})\vec{\boldsymbol{\Phi}} = 0 \qquad (2.1)$$

Здесь уже $\boldsymbol{K}$ и $\boldsymbol{M}$ – гипер-дуальные матрицы жесткости и инерции соответственно, Λ – гипер-дуальное собственное значение и $\vec{\boldsymbol{\Phi}}$ – гипер-дуальный собственный вектор. Далее, если предположить, что матрицы $\boldsymbol{K}$ и $\boldsymbol{M}$ являются гипер-дуальными матричными функциями параметра t, то (см. предыдущий параграф) $\boldsymbol{K}_0(t + 1\boldsymbol{\varepsilon}) = \boldsymbol{K}_0(t) + \dot{\boldsymbol{K}}_0(t)\boldsymbol{\varepsilon} + \ddot{\boldsymbol{K}}_0(t)\boldsymbol{\omega}$ и $\boldsymbol{M}_0(t + 1\boldsymbol{\varepsilon}) = \boldsymbol{M}_0(t) + \dot{\boldsymbol{M}}_0(t)\boldsymbol{\varepsilon} + \ddot{\boldsymbol{M}}_0(t)\boldsymbol{\omega}$.

Более того, т.к. $\omega^2 = \lambda$ трасформируется в $\Omega^2 = \Lambda$, то согласно свойстствам гипердуальных функций получим,

$$\Omega = \sqrt{\Lambda} = \sqrt{\lambda_0 + \lambda_1\boldsymbol{\varepsilon} + \lambda_2\boldsymbol{\omega}} = \sqrt{\lambda_0} + \tfrac{1}{2}\lambda_1/\sqrt{\lambda_0}\boldsymbol{\varepsilon} + \left(\tfrac{1}{2}\lambda_2/\sqrt{\lambda_0} - \tfrac{1}{4}\lambda_1^2/\lambda_0^{3/2}\right)\boldsymbol{\omega}.$$

И, наконец, после раскрытия уравнения (2.1) по правилам гипер-дуальной алгебры запишем развернутую форму уравнения (2.1):

$$\begin{aligned}&(\boldsymbol{K_0} - \lambda_0\boldsymbol{M_0})\vec{\phi}_0 + \left[(\boldsymbol{K_0} - \lambda_0\boldsymbol{M_0})\vec{\phi}_1 + (\boldsymbol{K}_1 - \lambda_0\boldsymbol{M}_1 - \lambda_1\boldsymbol{M_0})\vec{\phi}_0\right]\boldsymbol{\varepsilon} + \\ &+ \left[(\boldsymbol{K_0} - \lambda_0\boldsymbol{M_0})\vec{\phi}_2 + 2(\boldsymbol{K}_1 - \lambda_0\boldsymbol{M}_1 - \lambda_1\boldsymbol{M_0})\vec{\phi}_1 + (\lambda_0\boldsymbol{M}_2 + \lambda_2\boldsymbol{M_0} + 2\lambda_1\boldsymbol{M}_1)\vec{\phi}_0\right]\boldsymbol{\omega} = 0\end{aligned} \qquad (2.2)$$

3. Аглритм решения гипер-дуального уравнение колебаний

Для сокращения письма введем следующие обозначения:

$$(\boldsymbol{K} - \Lambda\boldsymbol{M})\vec{\boldsymbol{\Phi}} = L_0(\lambda_0, \vec{\phi}_0) + L_{\boldsymbol{1}}(\lambda_0, \vec{\phi}_0, \lambda_1, \vec{\phi}_1)\boldsymbol{\varepsilon} + L_{\boldsymbol{2}}(\lambda_0, \vec{\phi}_0, \lambda_1, \vec{\phi}_1, \lambda_2, \vec{\phi}_2)\boldsymbol{\omega} = 0,$$

где (L_0,L_1,L_2 - три уровня):

$L_0(\lambda_0,\vec{\phi}_0) = (\boldsymbol{K_0} - \lambda_0\boldsymbol{M_0})\vec{\phi}_0 = 0,$

$L_1(\lambda_0,\vec{\phi}_0,\lambda_1,\vec{\phi}_1) = (\boldsymbol{K_0} - \lambda_0\boldsymbol{M_0})\vec{\phi}_1 + (\boldsymbol{K}_1 - \lambda_0\boldsymbol{M}_1 - \lambda_1\boldsymbol{M_0})\vec{\phi}_0 = 0,$

$L_2(\lambda_0,\vec{\phi}_0,\lambda_1,\vec{\phi}_1,\lambda_2,\vec{\phi}_2) =$

$= (\boldsymbol{K_0} - \lambda_0\boldsymbol{M_0})\vec{\phi}_2 + 2(\boldsymbol{K}_1 - \lambda_0\boldsymbol{M}_1 - \lambda_1\boldsymbol{M_0})\vec{\phi}_1 + (\lambda_0\boldsymbol{M}_2 + \lambda_2\boldsymbol{M_0} + 2\lambda_1\boldsymbol{M}_1)\vec{\phi}_0 = 0$

Тогда решение полученное на предыдущем уравне используется для получения решений последующих уровней:

$L_0(\lambda_0,\vec{\phi}_0) = 0 \Rightarrow \lambda_0^*,\vec{\phi}_0^*$

$L_1(\lambda_0^*,\vec{\phi}_0^*,\lambda_1,\vec{\phi}_1) = 0 \Rightarrow \lambda_1^*,\vec{\phi}_1^*$

$L_2(\lambda_0^*,\vec{\phi}_0^*,\lambda_1^*,\vec{\phi}_1^*,\lambda_2,\vec{\phi}_2) = 0 \Rightarrow \lambda_2^*,\vec{\phi}_2^*$

УРОВЕНЬ 0:

$(\lambda_0^*,\vec{\phi}_0^*) = Eigenpair\,(\boldsymbol{M_0^{-1}}\cdot\boldsymbol{K_0}\,, modeIndex)$

$\vec{\phi}_0^* = \vec{\phi}_0^*/\sqrt{\vec{\phi}_0^{*\,T}\cdot(\boldsymbol{M_0}\cdot\vec{\phi}_0^*)}$ - *нормировка* $\vec{\phi}_0^*$ *по* $\boldsymbol{M_0}$

УРОВЕНЬ 1:

$\lambda_1^* = \vec{\phi}_0^{*\,T}\cdot\left((\boldsymbol{K}_1 - \lambda_0\boldsymbol{M}_1)\cdot\vec{\phi}_0^*\right)$

$A_1 = (\boldsymbol{K}_1 - \lambda_0^*\boldsymbol{M}_1 - \lambda_1^*\boldsymbol{M_0})$

$A_{\boldsymbol{reg}} = \boldsymbol{K_0} - \lambda_0^*\boldsymbol{M_0} + reg\cdot\boldsymbol{M_0};\ reg = 10^{-6}$

$\vec{\phi}_1^* = Solve(A_{\boldsymbol{reg}}, -A_1\cdot\vec{\phi}_0^*)$

$\vec{\phi}_1^* = \vec{\phi}_1^* - \vec{\phi}_0^*\cdot\left(\vec{\phi}_0^{*\,T}\boldsymbol{M_0}\cdot\vec{\phi}_1^*\right)$ - *ортогонализация по* $\vec{\phi}_0^{*\,T}\cdot(\boldsymbol{M_0}\cdot\vec{\phi}_1^*)$

УРОВЕНЬ 2:

$\lambda_2^* = \vec{\phi}_0^{*\,T}\cdot\left((\boldsymbol{K_2} - \lambda_0^*\boldsymbol{M}_2 - 2\lambda_1^*\boldsymbol{M}_1)\cdot\vec{\phi}_0^*\right) + 2\,\vec{\phi}_1^{*\,T}\cdot\left((\boldsymbol{K_1} - \lambda_0^*\boldsymbol{M}_1 - \lambda_1^*\boldsymbol{M_0})\,\vec{\phi}_0^*\right)$

$A_2 = \boldsymbol{K_2} - \lambda_0^*\boldsymbol{M}_2 - 2\lambda_1^*\boldsymbol{M}_1 - \lambda_2^*\boldsymbol{M_0}$

$\vec{\phi}_2^* = Solve(A_{\boldsymbol{reg}}, -\left(2(\boldsymbol{K_1} - \lambda_0^*\boldsymbol{M}_1 - \lambda_1^*\boldsymbol{M_0})\vec{\phi}_1^*\right) + A_2\cdot\vec{\phi}_0^*)$

$\vec{\phi}_2^* = \vec{\phi}_2^* - \vec{\phi}_0^*\cdot\left(\vec{\phi}_0^{*\,T}\boldsymbol{M_0}\cdot\vec{\phi}_2^*\right)$ - *ортогонализация по* $\vec{\phi}_0^{*\,T}\cdot(\boldsymbol{M_0}\cdot\vec{\phi}_2^*)$

Реализацию методов $Eigenpair(.)$ и $Solve(.)$ для языка C# целесообразно выполнить используя библиотеки `MathNet.Numerics.LinearAlgebra` и MathNet.Numerics.RootFinding [5].

5. Численная (компьютерная) реализация

Численная реализация выше описанного аглоритма была выполнена на языке C# операционной системы Windows (см. Приложение). Структуры `T0` и `T2` представляют

алгебры гипер-дуальных чисел и гипер-дуальных матриц соответственно с переопределёнными арифметическими операциями $\pm, \times, \div$, что соответствует методу автоматического дифференцирования [6]. Реализацию самого алгоритма осуществляет процедура `SolveModeSecondOrder()`, использующая библиотеки `MathNet.Numerics.LinearAlgebra` и `MathNet.Numerics.RootFinding`. C# код этой процедуры достаточно подробно прокомментирован (см. Приложение).

Представленный программный код был разработан исключительно для проведения численных экспериментов. Для вычисления первых и вторых производных элементов матриц $\boldsymbol{K(t), M(t)}$ необходимо описать эти функции в терминах гипер-дуальных функций, что будут сделано в следующем разделе.

6. Численный эксперимент

Рассматривается вертикально подвешанная гирлянда, состоящая из последовательно соединенных трех пружин одинаковой длины l и жесткости k (см. рис. 1). К концу каждой пружины прикреплен груз весом P.

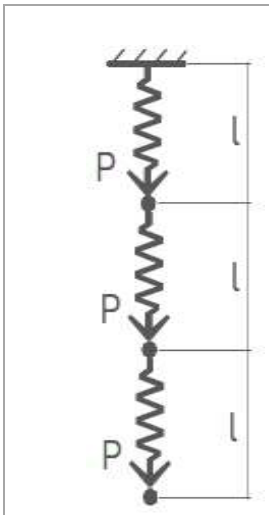

Рис. 1. Расчетная схема

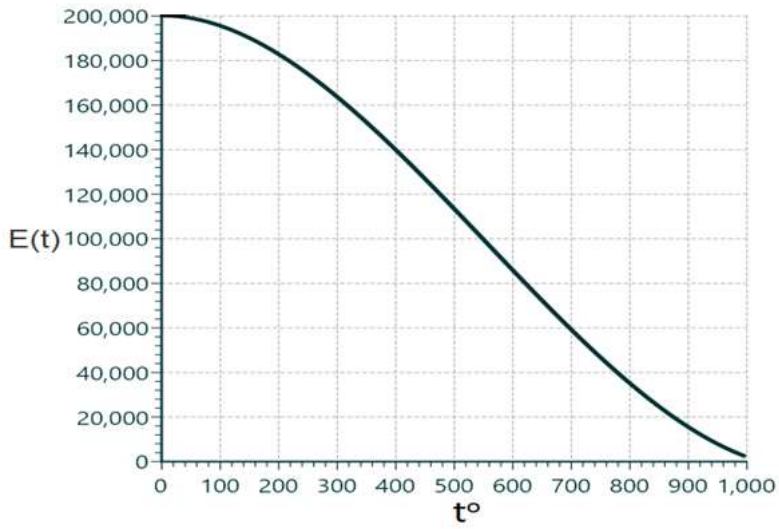

Рис. 2. График зависимости модуля Юнга от температуры

Система имеет три степени свободы и подвергается равномерному нагреву температурой t, которая меняет жесткости пружин k посредством изменения модуля Юнга. Зависимость модуля Юнга от температуры t принимается в виде $E(t) = 2{\cdot}10^5{\cdot}(1 + 10^{-5}t - 2.5{\cdot}10^{-6}t^2 + 1.5{\cdot}10^{-9}t^3)$, которая качественно неплохо описывает обычную сталь (см. рис. 2). Для решения уравнения (2.1) исходными данными служат:

$$K = k\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix},\quad M_0 = m\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix},\ M_1 = M_2 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix},\quad m = \frac{P}{g},$$

где: $P = 100$ Н, $g =$ 9,80665 м/с, $k = A(E(t) + \dot{E}(t)\boldsymbol{\varepsilon} + \ddot{E}(t))/l$, $l = 1$ м, А $= 10^{-4}$ м^2

Для вычисления элементов матриц K_1 и K_1 используется процедура `E(t)` (см. Приложение). Расчеты производились для t = 50, 300, 600 C° и модах 0, 1, 2, где мода 0 – низшая частота, 1 – средняя частота и 2 – высшая частота.

Результаты расчёта представлены в следующих таблицах.

t = 50, *mode* = 0											
Λ	λ_0	0.0378835422	Ω	ω_0	0.1946369496	$\vec{\phi}_0$	-0.0321643682	$\vec{\phi}_1$	≈ 0	$\vec{\phi}_2$	≈ 0
	λ_1	-0.0000087143		ω_1	-0.0000223861		-0.0579581889		≈ 0		≈ 0
	λ_2	-0.0000001733		ω_2	-0.0000004479		-0.0722726794		≈-0		≈-0
t = 300, *mode* = 0											
Λ	λ_0	0.0311811243	Ω	ω_0	0.1765817778	$\vec{\phi}_0$	-0.0321643682	$\vec{\phi}_1$	≈ -0	$\vec{\phi}_2$	≈ 0
	λ_1	-0.0000413336		ω_1	-0.0001170380		-0.0579581889		≈ -0		≈- 0
	λ_2	-0.0000000876		ω_2	-0.0000003257		-0.0722726794		≈ 0		≈ 0
t = 600, *mode* = 0											
Λ	λ_0	0.0163810427	Ω	ω_0	0.1279884475	$\vec{\phi}$	-0.0321643682	$\vec{\phi}_1$	≈ 0	$\vec{\phi}_2$	≈ 0
	λ_1	-0.0000521908		ω_1	-0.0002038886		-0.0579581889		≈ -0		≈ -0
	λ_2	0.0000000152		ω_2	-0.0000002653		-0.0722726794		≈ -0		≈ -0

Табл. 1. Результаты расчёта для mode = 0

t = 50, *mode* = 1											
Λ	λ_0	0.2974181993	Ω	ω_0	0.5453606140	$\vec{\phi}_0$	0.0722726794	$\vec{\phi}_1$	≈ 0	$\vec{\phi}_2$	≈ 0
	λ_1	-0.0000684150		ω_1	-0.0000627245		0.0321643682		≈ 0		≈ 0
	λ_2	-0.0000013608		ω_2	-0.0000012548		-0.0579581889		≈ 0		≈ 0
t = 300, *mode* = 1											
Λ	λ_0	0.2447984877	Ω	ω_0	0.4947711468	$\vec{\phi}_0$	0.0722726794	$\vec{\phi}_1$	≈ 0	$\vec{\phi}_2$	≈ 0
	λ_1	-0.0003245038		ω_1	-0.0003279332		0.0321643682		≈ 0		≈ 0
	λ_2	-0.0000006879		ω_2	-0.0000009125		-0.0579581889		≈ 0		≈ 0
t = 600, *mode* = 1											
Λ	λ_0	0.1286051921	Ω	ω_0	0.3586156607	$\vec{\phi}_0$	0.0722726794	$\vec{\phi}_1$	≈ 0	$\vec{\phi}_2$	≈ -0
	λ_1	-0.0004097421		ω_1	-0.0005712831		0.0321643682		≈ 0		≈ 0
	λ_2	0.0000001196		ω_2	-0.0000007433		-0.0579581889		≈ 0		≈ 0

Табл. 2. Результаты расчёта для mode = 1

t = 50, *mode* = 2											
Λ	λ_0	0.6210526232	Ω	ω_0	0.7880689203	$\vec{\phi}_0$	-0.0579581889	$\vec{\phi}_1$	≈ -0	$\vec{\phi}_2$	≈ 0
	λ_1	-0.0001428604		ω_1	-0.0000906396		0.0722726794		≈ 0		≈ 0
	λ_2	-0.0000028416		ω_2	-0.0000018133		-0.0321643682		≈-0		≈ -0
t = 300, *mode* = 2											
Λ	λ_0	0.5111749829	Ω	ω_0	0.7149650221	$\vec{\phi}_0$	-0.0579581889	$\vec{\phi}_1$	≈ 0	$\vec{\phi}_2$	≈ 0

	λ_1	-0.0006776113		ω_1	-0.0004738772		0.0722726794		≈ 0		≈ 0
	λ_2	-0.0000014364		ω_2	-0.0000013186		-0.0321643682		≈ -0		≈ -0
t = 600, *mode* = 2											
	λ_0	0.2685464174		ω_0	0.5182146441		-0.0579581889		≈ 0		≈ 0
Λ	λ_1	-0.0008556014	Ω	ω_1	-0.0008255280	$\vec{\phi}_0$	0.0722726794	$\vec{\phi}_1$	≈ 0	$\vec{\phi}_2$	≈ 0
	λ_2	0.0000002498		ω_2	-0.0000010741		-0.0321643682		≈ -0		≈ -0

Табл. 3. Результаты расчёта для mode = 2

Из приведённых результатов расчета видно, что с ростом t жесткость системы падает и λ_0, ω_0, λ_1, ω_1, λ_2, ω_2 уменьшаются во всех модах. λ_1, λ_2, ω_1 и ω_2 малы, но ненулевые. Гипер-дуальные поправки $\vec{\phi}_1$ и $\vec{\phi}_2$ почти нулевые, т.е. их направление не меняется и они просто масштабируются. Ω в высшей моде убывает быстрее всего, а в нисшей моде - медленее.

Таким образом, модуль Юнга убывает с ростом температуры, что приводит к монотонному уменьшению жёсткости пружин. Собственные значения $\Lambda(t)$ и частоты $\Omega(t)$ всех трёх мод также убывают при увеличении температуры. Собственные формы $\vec{\phi}_0$ для всех трёх мод не зависят от температуры, а зависит только от моды. Гипер-дуальные поправки $\vec{\phi}_1$ и $\vec{\phi}_2$ весьма незначительны, что подтверждает: формы колебаний инвариантны относительно t; изменяются только собственные значения.

Температурная чувствительность (коэффициенты при $\boldsymbol{\varepsilon}$ и $\boldsymbol{\omega}$) возрастает от низшей моды к высшей: Это означает, что высокочастотные режимы сильнее реагируют на изменение жёсткости материала.

7. Заключение

Использование гипер-дуальных чисел второго порядка является актуальным и перспективным инструментом для задач спектрального анализа и механики деформируемого твёрдого тела. В отличие от традиционных методов численного дифференцирования, гипер-дуальная алгебра обеспечивает точное вычисление первых и вторых производных без потерь точности и без необходимости подбора шагов конечных разностей. Это делает метод особенно эффективным при исследовании температурной чувствительности собственных значений, анализе устойчивости и построении параметрических моделей. Полученные результаты демонстрируют, что гипер-дуальные объекты позволяют естественным образом разделять вклад нулевого, первого и второго порядков, обеспечивая высокую точность и устойчивость вычислений при минимальных изменениях исходного кода. Такой подход открывает возможности для дальнейшего применения в задачах оптимизации, инверсного анализа и автоматического построения моделей с параметрической зависимостью.

ПРИЛОЖЕНИЕ

C# коде реализующий алгебры гипер-дуальных чисел 2-го порядка (`T0`) и гипердуалных матриц (`T2`), а также дана процедура `SolveModeSecondOrder(…)`, выполняющая алгоритм метода, рассмотренного в параграфах 3 и 4. Приведен код для выполнения численного эксперимента.

```
using MathNet.Numerics.LinearAlgebra;
using MathNet.Numerics.LinearAlgebra.Factorization;
using MathNet.Numerics.LinearAlgebra.Double;
using MathNet.Numerics.RootFinding;
using System.Collections.Generic;
using System.Linq;
using System;

public struct T0 {
        public double Re, Im1, Im2;
        public T0(double x, double x1, double x2) { Re = x; Im1 = x1; Im2 = x2;}
        public static T0 operator *(double x, T0 X)
        { return new T0(x * X.Re, x * X.Im1, x * X.Im2);}
        public static T0 operator *( T0 X, double x){ return x * X;}
        public static T0 operator /(T0 X, double x)
        { return new T0(X.Re / x, X.Im1 / x, X.Im2 / x);}
        public static T0 operator *(T0 X, T0 Y)
        { return new T0(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 T0 operator +(T0 X, T0 Y)
        { return new T0(X.Re + Y.Re, X.Im1 + Y.Im1, X.Im2 + Y.Im2);}
        public static T0 operator -(T0 X, T0 Y)
        { return new T0(X.Re - Y.Re, X.Im1 - Y.Im1, X.Im2 - Y.Im2); }
        public static T0 operator -(T0 X) { return new T0(-X.Re, -X.Im1, -X.Im2);}
}

public struct T2 {
        public Matrix<double> Re, Im1, Im2;
        public T2(double[,] M, double[,] M1, double[,] M2)
        {   Re = Matrix<double>.Build.DenseOfArray(M);
            Im1 = Matrix<double>.Build.DenseOfArray(M1);
            Im2 = Matrix<double>.Build.DenseOfArray(M2);
        }
        public T2(Matrix<double> M, Matrix<double> M1, Matrix<double> M2)
        { Re = M; Im1 = M1; Im2 = M2;}
        public static T2 operator *(double x, T2 X)
        { return new T2(x * X.Re, x * X.Im1, x * X.Im2);}
        public static T2 operator *(T0 x, T2 X)
        { return new T2(x.Re * X.Re, x.Im1 * X.Re, x.Im2 * X.Re);}
        public static T2 operator *(T2 X, T2 Y)
        { return new T2(X.Re * Y.Re, X.Re * Y.Im1 + X.Im1 * Y.Re, X.Re * Y.Im2 + 2.0 *
          X.Im1 * Y.Im1 + X.Im2 * Y.Re);}
        public static T2 operator +(T2 X, T2 Y)
        { return new T2(X.Re + Y.Re, X.Im1 + Y.Im1, X.Im2 + Y.Im2);}
        public static T2 operator -(T2 X, T2 Y)
```

```
        { return new T2(X.Re - Y.Re, X.Im1 - Y.Im1, X.Im2 - Y.Im2);}
}

public struct ModeResult
        {   public double Lambda0, Lambda1, Lambda2;
            public Vector<double> Phi0, Phi1, Phi2;
        }

public static ModeResult SolveModeSecondOrder(
            Matrix<double> K0, Matrix<double> K1, Matrix<double> K2,
            Matrix<double> M0, Matrix<double> M1, Matrix<double> M2,
            int modeIndex, double regularization = 1e-6)
 {  // ********************************************************
    //  LEVEL 0 — базовая собственная задача
    //  (K0 - λ0*M0)*Φ0 = 0
    // ********************************************************
    // Приведение обобщённой задачи к стандартной: A0 = M0^{-1}*K0  →  A0*Φ0 = λ0*Φ0
    // Это эквивалентно (K0 - λ0 M0) Φ0 = 0.
            var A0 = M0.Inverse() * K0;
    // Собственные значения и векторы A0
            var evd = A0.Evd();
    // λ0 — собственное значение нулевого порядка
            double lambda0 = evd.EigenValues[modeIndex].Real;
    // Φ0 — собственный вектор нулевого порядка
            var phi0 = evd.EigenVectors.Column(modeIndex);
    // Нормировка по M0: Φ0^T*M0*Φ0 = 1
            phi0 = phi0 / Math.Sqrt(phi0.DotProduct(M0 * phi0)); // нормировка
     // ********************************************************
     //  LEVEL 1 — первый порядок
     //  λ1 и Φ1
     //  λ1 = Φ0^T*K1*Φ0 – λ0*Φ0^T*M1*Φ0
     //  (K0 – λ0*M0) Φ1 = –(K1 – λ1*M0 – λ0*M1)*Φ0
     //  В коде используется регуляризация и ортогонализация.
     // ********************************************************
     // Формула λ1 = Φ0^T*(K1 – λ0*M1)*Φ0
            double lambda1 = phi0.DotProduct((K1 - lambda0 * M1) * phi0);
    // Оператор первого порядка A1 = K1 – λ0*M1 – λ1*M0
            var A1 = K1 - lambda0 * M1 - lambda1 * M0;
    // Правая часть уравнения первого порядка: RHS1 = –A1*Φ0
            var RHS1 = -A1 * phi0;
    // Areg = K0 – λ0*M0 + M0 ε
    // Это регуляризованный оператор, чтобы решить(K0 – λ0*M0)*Φ1 = RHS1,
    // который в точности сингулярен.
            var Areg = K0 - lambda0 * M0 + regularization * M0;
    // Решение для Φ1 через регуляризованный оператор
            var phi1 = Areg.Solve(RHS1);
    // Ортогонализация:
    // навязываем Φ0^T*M0*Φ1 = 0 (упрощённая нормировка первого порядка)
            phi1 = phi1 - phi0 * phi0.DotProduct(M0 * phi1); // ортогонализация
    // ********************************************************
    //  LEVEL 2 — второй порядок
    //  λ2 и Φ2
    //  λ2 = Φ0^T*(K2 – λ0*M2 – 2*λ1*M1)*Φ0 + 2*Φ1^T*(K1 – λ0*M1 – λ1*M0)*Φ0
    //  (K0 – λ0*M0) Φ2 = –(K2 – λ2*M0 – λ1*M1 – λ0*M2)*Φ0 –(K1 – λ1*M0 – λ0*M1) *Φ1
    //   В коде снова используется регуляризация и ортогонализация.
    // ********************************************************
    // Формула λ2 — полностью совпадает с классической теорией
        double lambda2 = phi0.DotProduct((K2 - lambda0 * M2 - 2 * lambda1 * M1) * phi0)
                       + 2 * phi1.DotProduct((K1 - lambda0 * M1 - lambda1 * M0) * phi0);
    // Оператор второго порядка A2 = K2 – λ0*M2 – 2*λ1*M1 – λ2*M0
        var A2 = K2 - lambda0 * M2 - 2 * lambda1 * M1 - lambda2 * M0;
    // Правая часть второго порядка:
   // RHS2 = –2*(K1 – λ0*M1 – λ1*M0)*Φ1– A2*Φ0
```

```
        var RHS2 = -2 * (K1 - lambda0 * M1 - lambda1 * M0) * phi1 - A2 * phi0;
    // Решение Φ2 через тот же регуляризованный оператор
        var phi2 = Areg.Solve(RHS2);
     // Ортогонализация второго порядка:навязываем Φ0^T*M0*Φ2 = 0
        phi2 = phi2 - phi0 * phi0.DotProduct(M0 * phi2);
        return new ModeResult
            {   Lambda0 = lambda0,
                Lambda1 = lambda1,
                Lambda2 = lambda2,
                Phi0 = phi0,
                Phi1 = phi1,
                Phi2 = phi2
            };
 }

public static T0 get_Ω(double λ0, double λ1, double λ2)
      { double ω0 = Math.Sqrt(λ0),  ω1 = 0.5 * λ1 / ω0,
               ω2 = 0.5 * λ2 / ω0 - 0.25 * λ1 * λ1 / (ω0 * ω0 * ω0);
        return new T0(ω0, ω1, ω2);
}

//  ПРИМЕР РАСЧЕТА
public static T0 E(double t)
  { T0 T = new T0(t, 1, 0), One = new T0(1, 0, 0);
       return 2.0E5*(One + 1E-5*T - 2.5E-6*T*T + 1.5E-9*T*T*T);
   }

double P = 100.0, g = 9.80665, m = P / g,   l=1.0, A=1.0E-4, t=300.0;
var M0 = m * Matrix<double>.Build.DenseOfArray(new double[,] {
            { m, 0, 0 }, { 0, m, 0 }, { 0, 0, m }});
var M1= Matrix<double>.Build.DenseOfArray(new double[,] {
            { 0, 0, 0 }, { 0, 0, 0 }, { 0, 0, 0 }});
var M2 = M1;
var K = Matrix<double>.Build.DenseOfArray(new double[,] {
            { 2.0, -1.0, 0.0 }, { -1.0, 2.0, -1.0 }, { 0.0, -1.0, 1.0 }});
int mode = 0;  var Et = E(t);  var k = Et * A;
var K0 = k.Re * K;  var K1 = k.Im1 * K;  var K2 = k.Im2 * K;
var result = SolveModeSecondOrder(K0, K1, K2, M0, M1, M2, mode);
var Ω = get_Ω(result.Lambda0, result.Lambda1, result.Lambda2);
var Phi0 = result.Phi0;
var Phi1 = result.Phi1;
var Phi2 = result.Phi2;
```

Литература

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

2. *Олифер В.И.* Вычисление собственных векторов гипер-дуальных матриц. – URL: http://viosolutions.amerihomesrealty.com/pdf/Вычисление собственных гипер-дульных

векторов.pdf, – 2026.

3. *Макченко В.А., Славин В.В.* Обратная задача теории малых колебаний. - Харьков-2014, - 227 с.

4. *Глэдвелл Г. М. Л.* Обратные задачи теории колебаний. - М.-Ижевск: НИЦ "Регулярная и хаотическая динамика", Институт компьютерных исследований, 2008. - 608 с.

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

6. *Naumann U.* The Art of Differentiating Computer Programs. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2012.

Абстракт

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

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

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

20 июня 2026