ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

На правах рукописи

УДК 517.929

Хохлова Татьяна Наилевна

# УСТОЙЧИВОСТЬ НЕЙРОННЫХ СЕТЕЙ ХОПФИЛДА С ЗАПАЗДЫВАНИЕМ

05.13.18 – математическое моделирование, численные методы и комплексы программ

ДИССЕРТАЦИЯ

на соискание учёной степени

кандидата физико-математических наук

Научный руководитель

д. ф.-м. н., проф.

М. М. Кипнис

Челябинск – 2012

# Содержание

# Глава 1

# Теоретические основания для диагностирования устойчивости нейронных сетей: конус устойчивости

## 1.1 Нейронные сети и дифференциальные уравнения с запаздываниями

### 1.1.1 Модель Хопфилда-Маркуса-Вестервельта

Одна из самых популярных моделей искусственных нейронных сетей предложена Хопфилдом [1] в 1984 г. и с тех пор стала называться моделью Хопфилда. Она описывается дифференциальными уравнениями вида

$$C_j\dot{x}_j(t) + \frac{1}{R_j}x_j(t) + \sum_{k=1}^{n} T_{jk}g_k(x_k(t)) = 0, \quad j = 1, 2, \ldots, n. \tag{1.1}$$

Здесь $n \geqslant 2$ есть количество нейронов в сети, $x_j$ — сигнал, $C_j$ и $R_j > 0$ — входные ёмкости и сопротивления соответственно $j$-го нейрона. Матрица $T_{jk}$ размера $n \times n$ представляет мощность связи между нейронами. Если $T_{jk} > 0$, то положительный выходной сигнал $k$-го нейрона работает на торможение $j$-го нейрона, если $T_{jk} < 0$, то на возбуждение, если $T_{jk} = 0$, то связь между $k$-м и $j$-м нейронами отсутствует. Гладкие функции $g_j$ являются функциями активации.

Сам Хопфилд заметил, что в переключениях нейронов имеются запаздывания. В модель Хопфилда запаздывания впервые ввели Маркус и Вестервельт в 1989 г. [2]. Они рассмотрели модель вида

$$C_j\dot{x}_j(t) + \frac{1}{R_j}x_j(t) + \sum_{k=1}^{n} T_{jk}g_k(x_k(t-\tau)) = 0, \quad j = 1, 2, \ldots, n, \tag{1.2}$$

где $\tau$ — запаздывание.

Во многих работах рассматривается сети, в которых $C_j = \text{const}$, $R_j = \text{const}$, и в таких случаях, изменяя масштаб времени, мы можем считать, что $C_j = R_j = 1$. Часто рассматриваются функции $g_j(x)$ «сигмоидного вида». В нескольких работах (например, [3–5]) изучается система из двух нейронов вида

$$\begin{aligned} \dot{x}_1(t) + x_1(t) + T_{12}\,\mathrm{th}(x_2(t-\tau_2)) = 0, \\ \dot{x}_2(t) + x_2(t) + T_{21}\,\mathrm{th}(x_1(t-\tau_1)) = 0. \end{aligned} \tag{1.3}$$

В работе К. Гопалсами и И. Леунга [3] рассматривается модель (1.3) с $\tau_1 = \tau_2$. Имеется даже работа И. Дьери и Хартунга [**?** ] (2003 г.), в которой изучается устойчивость нелинейной модели одного нейрона.

В дальнейшем выяснилось, что для большей адекватности модели естественно ввести различные запаздывания в (1.2) и рассматривать модели вида

$$C_j\dot{x}_j(t) + \frac{1}{R_j}u_j(t) + \sum_{k=1}^{n} T_{jk}g_k(x_k(t-\tau_{jk})) = 0, \quad j = 1, 2, \ldots, n, \tag{1.4}$$

Известны трудности изучения устойчивости дифференциальных уравнений с двумя запаздываниями даже в скалярном случае, выявленные М. Кипнисом и И. Левицкой [6, 7] в 2003-2007 гг. По этой причине в большинстве исследований часть запаздываний $\tau_{jk}$, которые либо в точности равны нулю, либо близки к нулю, переводят из последнего слагаемого в левой части (1.4) во второе слагаемое, а остальные запаздывания отождествляют, и мы от (1.4) переходим к уравнению

$$\dot{x}_j(t) + \sum_{k=1}^{n} a_{jk}x_k(t) + \sum_{k=1}^{n} T_{jk}g_k(x_k(t-\tau)) = 0, \quad j = 1, 2, \ldots, n, \tag{1.5}$$

где $a_{jk}$ коэффициенты, характеризующие мощность мгновенного взаимодействия между нейронами. Наконец, линеаризация уравнения (1.5) вокруг стационарного состояния приводит нас к матричному уравнению

$$\dot{x}(t) + Ax(t) + Bx(t-\tau) = 0, \tag{1.6}$$

где $n \times n$ матрица $A$ есть матрица мгновенных взаимодействий, а $n \times n$ матрица $B$ — матрица взаимодействий с запаздыванием.

Уравнение (1.6) будет основным в нашем исследовании. Оно подвергалось изучению также в связи с задачами устойчивости систем автоматического управления, например, В. Харитоновым, К. Гу и Дж. Ченом [8] (2003 г.) и Мори с соавторами [9, 10] (1981, 1989 гг.).

Вопрос о глобальной устойчивости стационарных состояний нейронных сетей иногда ставится и при некоторых условиях решается (например, в работах ван дер Дрише [11] (1998 г.), И. Дьери и Ф. Хартунга [**?** ] (2003 и 2007 гг.), [**?** ], Л. Идельса и М. Кипниса [12](2009 г.)). Но в зависимости от назначения нейронной сети в ней может быть одно или несколько стационарных состояний. Например, если сеть служит в качестве хранилища памяти, то число состояний каждого нейрона больше единицы, а число состояний всей сети, разумеется, растёт экспоненциально в зависимости от числа нейронов.

Поэтому глобальная устойчивость нейронной сети иногда нежелательна. Но локальная устойчивость, по-видимому, является положительным свойством любой нейронной сети. Именно локальная устойчивость стационарных состояний нейронных сетей является предметом диссертационного исследования.

### 1.1.2 Модель нейронных сетей Коэна–Гроссберга

В 1983 году Коэн и Гроссберг [**?** ] предложили модель биологических нейронных сетей, в которых поведение $j$-го нейрона в сети $n$ нейронов $(1 \leqslant n)$ описывается дифференциальными уравнениями

$$\dot{x}_j(t) = c_i(x_j(t)) \left( -d_j(x_j(t)) + \sum_{k=1}^{n} a_{jk} f_k(x_k(t)) + \sum_{k=1}^{n} b_{jk} f_k(x_k(t - \tau_{jk})) - I_j \right). \tag{1.7}$$

Здесь $x_j$ потенциал $j$-го нейрона; $c_j(x_j(t))$ представляет усилитель сигнала $x_j(t)$; положительная функция $d_j(x_j(t))$ обеспечивает затухание собственного сигнала нейрона в отсутствие взаимодействия с другими нейронами и внешнего воздействия $I_j$; функция $f_k(x_k)$ это функция активации $k$-го нейрона, которая предполагается сигмоидной; $a_{jk}$ есть сила мгновенного действия нейрона с номером $k$ на нейрон с номером $j$; наконец, коэффициенты $b_{jk}$ характеризуют силу действия нейрона с номером $k$ на нейрон с номером $j$ с запаздыванием $\tau_{jk}$.

Линеаризация уравнения (1.7) вокруг любого решения приводит к основному объекту анализа диссертации — уравнению (1.6).

### 1.1.3 Модель клеточных нейронных сетей Чуа–Янга–Роски

Клеточные нейронные сети (CNN — Cellular Neural Nets) изобрели Л. Чуа и Л. Янг [**?** ] в 1988 г. Искусственные CNN широко применяются в инженерных приложениях [**?** ], [**?** ] (работы соответственно 1998 и 1992 гг.). Книга [**?** ] (2004 г.) полностью посвящена распознаванию образов с помощью CNN. С самого начала развития клеточных нейронных сетей было замечено, что в CNN имеются запаздывания, вызываемые конечной скоростью переключений. Так появилась теория клеточных нейронных сетей с запаздываниями (DCNN — Delay Cellular Neural Nets) [**?** ], [**?** ] (1992,2004). DCNN описываются дифференциальными уравнениями вида

$$\dot{x}_j(t) = -d_j(x_j(t)) + \sum_{k=1}^{n} a_{jk} f_k(x_k(t)) + \sum_{k=1}^{n} b_{jk} f_k(x_k(t-\tau_{jk})) - I_j). \qquad (1.8)$$

с интерпретациями функций и коэффициентов, описанными в предыдущем подразделе. В роли функции $f$ может выступать функция сигмоидного вида, например, $f(x) = \frac{1}{2}(|x+1|-|x-1|)$, или $f(x) = \tanh(x)$ или $f(x) = \frac{1}{1+e^{-x}}$. Как видим, и в клеточных нейронных сетях с запаздываниями проблема локаль-

ной устойчивости решений сводится к тому же основному уравнению (1.6) нашей диссертации.

### 1.1.4 Особенности линеаризованных дифференциальных уравнений моделей нейронных сетей

Специфика нейронных сетей в том, что они составляются из однотипных нейронов. Поэтому в отличие от общей теории устойчивости систем управления в исследованиях по нейронным сетям изучаются уравнения (1.6) со специфическими матрицами $A$, $B$. Следующие примеры прояснят некоторые особенности описания моделей нейронных сетей.

**Пример 1.1.** Рассмотрим систему из шести нейронов, связи которых указаны на рисунке 1.1. Положим, все связи между различными нейронами

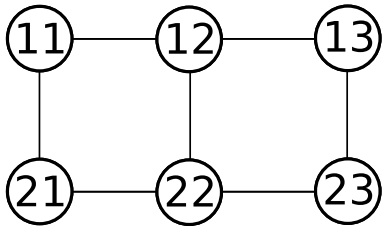



Рис. 1.1. Граф связей в нейронной сети

имеют запаздывание $\tau$. Пусть мощность взаимодействия каждого нейрона с левым соседом равна $a$, с правым — $b$, с верхним — $c$, с нижним — $d$. Положим, реакция каждого нейрона на свой собственный сигнал мгновенна. Тогда сеть на рисунке 1.1 описывается уравнением (1.6) с вектор-функцией $x = (x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23})^T$ с матрицами

$$A = E, \qquad B = \begin{pmatrix} 0 & b & 0 & d & 0 & 0 \\ a & 0 & b & 0 & d & 0 \\ 0 & a & 0 & 0 & 0 & d \\ c & 0 & 0 & 0 & b & 0 \\ 0 & c & 0 & a & 0 & b \\ 0 & 0 & c & 0 & a & 0 \end{pmatrix}. \tag{1.9}$$

**Пример 1.2.** Рассмотрим вариант интерпретации системы связей на рисунке 1.1) как двуслойной сети. Будем считать, что первый слой состоит из нейронов 11, 12, 13, второй слой из 21, 22, 23. Пусть взаимодействие нейронов внутри слоев происходит без запаздывания, а взаимодействие нейронов из разных слоев происходит с запаздыванием $\tau$. Положим, мощности взаимодействия таковы, как они описаны в предыдущем примере.

Усложняя сеть, положим также, что каждый нейрон, кроме мгновенной реакции на собственное состояние, реагирует на собственное состояние ещё и с запаздыванием с некоторой мощностью $e$ (такие реакции нейронов рассматривались С. Гуо и Л. Хуангом [13], а также С. Кемпбелл с соавторами [14, 15]).

Тогда нейронная сеть описывается уравнением (1.6) с матрицами

$$A = \begin{pmatrix} 1 & b & 0 & 0 & 0 & 0 \\ a & 1 & b & 0 & 0 & 0 \\ 0 & a & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & b & 0 \\ 0 & 0 & 0 & a & 1 & b \\ 0 & 0 & 0 & 0 & a & 1 \end{pmatrix}, \qquad B = \begin{pmatrix} e & 0 & 0 & d & 0 & 0 \\ 0 & e & 0 & 0 & d & 0 \\ 0 & 0 & e & 0 & 0 & d \\ c & 0 & 0 & e & 0 & 0 \\ 0 & c & 0 & 0 & e & 0 \\ 0 & 0 & c & 0 & 0 & e \end{pmatrix}. \tag{1.10}$$

Примеры демонстрируют, что в описании нейронных сетей часто фигурируют симметричные матрицы, а также блочные матрицы относительно несложной структуры. Ещё одна особенность матриц, часто встречающаяся в уравнениях нейронных сетей — ненулевыми элементами заполняются только немногие диагонали над и под главной диагональю, то есть матрицы являются ленточными. Это связано с тем, что во многих (но не всех) сетях нейроны соединяются только с ближайшими в некотором смысле соседями, то есть наблюдается близкодействие. Как покажут следующие главы диссертации, существенной особенностью исследуемых в диссертации задач устойчивости нейронных сетей является зависимость устойчивости только от согласованного спектра двух матриц $A$, $B$ в случае их одновременного приведения к треугольной форме.

Дадим обзор содержания остальных разделов главы 1. Достаточные признаки устойчивости матричного уравнения (1.6) даны в [8–10, 16–18]. В [19] исследуется эта задача для матриц $2\times 2$ специального вида. В следующих разделах главы 1 диссертации мы даем метод, позволяющий дать полный геометрический обзор тех значений запаздывания $\tau$, которые обеспечивают устойчивость уравнения (1.6). Метод пригоден для уравнений с матрицами $A, B$, которые могут быть приведены одной трансформирующей матрицей к треугольному виду. Это в точности класс матриц $A, B$, таких что $AB-BA$ нильпотентна [20]. Этот класс включает все коммутирующие пары матриц $A, B$.

Несмотря на указанные ограничения, этот метод даёт возможность изучать устойчивость широких классов нейронных сетей: сетей кольцевой и линейной архитектуры, сетей, описываемых двудольными графами, трёхслойных сетей, так как все вышеуказанные сети описываются уравнениями вида (1.6) с одновременно триангулируемыми матрицами. В примерах 1.1, 1.2 матрицы $A\,, B$ коммутируют.

Этот метод основан на конусе устойчивости — некоторой поверхности в $\mathbb{R}^3$, одной для класса всех уравнений вида (1.6), в которых матрицы $A, B$ одновременно приводимы к треугольному виду. Конус устойчивости впервые был введён в работе автора [21] совместно с научным руководителем и В. В. Малыгиной. Идейно конус устойчивости происходит от давней работы З. Рехлицкого [22] (1956), в которой был построен овал устойчивости для некоторого общего уравнения вида (1.6) с $A = 0$.

В этой главе конус устойчивости будет введён постепенно. Вначале вводятся овалы устойчивости для скалярного уравнения вида (1.6) с действительным коэффициентом $A$ и комплексным $B$ (раздел 1.2), затем конус устойчивости для скалярного уравнения с комплексными коэффициентами (раздел 1.3), который будет работать также и для матричного уравнения (1.6).В основном разделе 1.4 доказывается теорема о конусе устойчивости для матрич-

ного уравнения (1.6), которая будет теоретической основой для алгоритмов диагностирования устойчивости нейронных сетей и их программных реализаций.

Особняком стоит раздел 1.5 о системе неравенств для диагностики устойчивости этого же уравнения. Он не будет использоваться далее в диссертации.

В последнем разделе результаты главы 1 сравниваются с известными результатами.

## 1.2 Овал устойчивости

Рассмотрим уравнение

$$\dot{x}\,(t) + a\,x(t) + b\,x(t-\tau) = 0, \quad \tau > 0. \tag{1.11}$$

Вначале напомним известные результаты об устойчивости уравнения (1.11) при $a, b \in \mathbb{R}$. Уравнение (1.11) считаем (асимптотически) устойчивым, если его нулевое решение является (асимптотически) устойчивым. Будем рассматривать уравнение (1.11) с начальными условиями

$$x(t) = \varphi(t), \qquad t \in [-\tau, 0], \tag{1.12}$$

где $\tau > 0$ — запаздывание, $\varphi(t)$ — начальная функция.

Будем считать, что начальная функция $\varphi(t)$ кусочно-непрерывна на $[-\tau, 0]$. Решением уравнения (1.11) назовём функцию $x(t) : [-\tau, \infty) \to \mathbb{R}$, удовлетворяющую уравнению (1.11) и условию (1.48). Сформулируем известное утверждение об области устойчивости этого уравнения.

**Предложение 1.1.** *Следующие утверждения верны.*

1. *Если* $a \leqslant -\frac{1}{\tau}$, *то уравнение* (1.11) *неустойчиво.*

2. *Если* $a > -\frac{1}{\tau}$, *то уравнение* (1.11) *асимптотически устойчиво тогда и только тогда, когда*

$$-a < b < S(a), \tag{1.13}$$

*где кривая* $b = S(a)$ *задаётся следующим образом:*

$$\begin{cases} a(\omega) = -\dfrac{\omega}{\operatorname{tg}\omega\tau}\,, \\ b(\omega) = \dfrac{\omega}{\sin\omega\tau}\,, \\ 0 < \omega < \dfrac{\pi}{\tau}\,. \end{cases} \tag{1.14}$$

3. *Если* $a > -\frac{1}{\tau}$ *и, кроме того, либо* $b = -a$, *либо* $b = S(a)$, *то уравнение* (1.11) *устойчиво (не асимптотически).*

4. *Если* $a > -\frac{1}{\tau}$ *и, кроме того, либо* $b < -a$, *либо* $b > S(a)$, *то уравнение* (1.11) *неустойчиво.*

На рисунке 1.2 изображена область устойчивости уравнения (1.11) в плоскости параметров уравнения $(b, a)$.

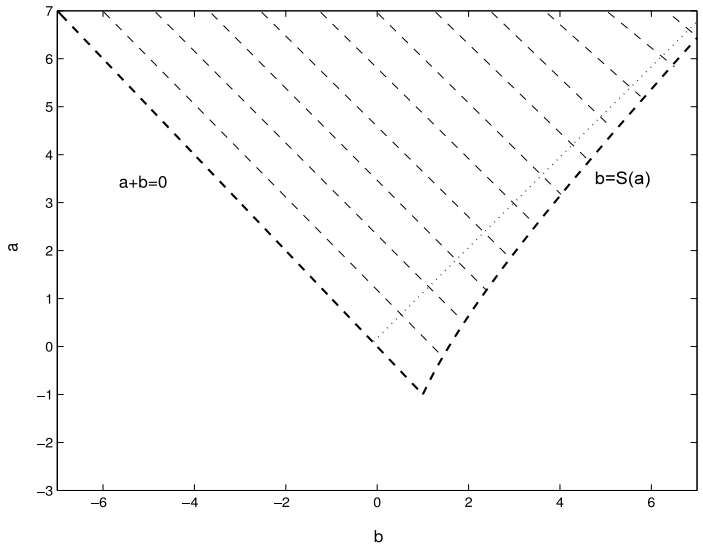



Рис. 1.2. Область устойчивости уравнения (1.11), указанная в предложении (1.1)

Заметим, что кривая $b = S(a)$ имеет наклонную асимптоту $a = b$ (при $\omega \to \to \frac{\pi}{\tau}$ данная кривая неограниченно приближается к прямой $a = b$, отмеченной на графике пунктиром).

Теперь рассмотрим уравнение (1.11) при $a \in \mathbb{R}$, $b \in \mathbb{C}$. Дадим определение овала устойчивости.

**Определение 1.1.** *Овалом устойчивости для уравнения* (1.11) *с данным* $a \in \mathbb{R}$, $a > -\frac{1}{\tau}$ *называется кривая* $b = b(\omega)$ *на комплексной плоскости переменного* $b$, *заданная уравнением*

$$\begin{cases} \operatorname{Re} b(\omega) = -a\cos\omega\tau + \omega\sin\omega\tau, \\ \operatorname{Im} b(\omega) = -a\sin\omega\tau - \omega\cos\omega\tau, \\ -\omega_1 \le \omega \le \omega_1, \end{cases} \tag{1.15}$$

*где* $\omega_1$ *— наименьший положительный корень уравнения* $a\sin\omega\tau + \omega\cos\omega\tau = 0$.

Следующая теорема упомянута в некоторой другой форме в виде недоказанного и небрежно сформулированного утверждения в одной из статей Мори с соавторами (см. [9], Appendix B).

**Теорема 1.1.** *Пусть в уравнении* (1.11) $a \in \mathbb{R}$. *Тогда верны следующие утверждения.*

1. *Если* $a \leqslant -\frac{1}{\tau}$, *то уравнение* (1.11) *неустойчиво при любых комплексных* $b$.

2. *Если* $a > -\frac{1}{\tau}$ *и точка* $(\operatorname{Re} b, \operatorname{Im} b)$ *находится внутри овала устойчивости для данного* $a$ *(см. определение 1.1), то уравнение* (1.11) *асимптотически устойчиво.*

3. *Если* $a > -\frac{1}{\tau}$ *и точка* $(\operatorname{Re} b, \operatorname{Im} b)$ *находится на границе овала устойчивости для данного* $a$*, то уравнение* (1.11) *устойчиво (не асимптотически).*

4. *Если* $a > -\frac{1}{\tau}$ *и точка* $(\operatorname{Re} b, \operatorname{Im} b)$ *находится вне овала устойчивости для данного* $a$*, то уравнение* (1.11) *неустойчиво.*

**Доказательство.** Для определения границ области устойчивости уравнения (1.11) на комплексной плоскости параметра $b$ будем использовать метод $D$-разбиений [23], при этом полагаем, что $a \in \mathbb{R}$ фиксированная величина. Характеристический квазимногочлен уравнения (1.11) имеет вид

$$F(\lambda) = \lambda + a + b\,e^{-\lambda\tau}. \tag{1.16}$$

Согласно методу $D$-разбиений, если точка $(\operatorname{Re} b, \operatorname{Im} b)$ принадлежит границе области устойчивости, то $\exists\omega \quad F(i\omega) = 0, \quad \omega \in (-\infty, \infty)$. Получим

$$F(i\omega) = i\omega + a + b\,e^{-i\omega\tau} = i\omega + a + (\operatorname{Re} b + i \operatorname{Im} b)\,(\cos\omega\tau - i\sin\omega\tau) = 0. \tag{1.17}$$

Выделим действительную и мнимую части:

$$\begin{cases} \operatorname{Re} F(i\omega) = a + \operatorname{Re} b \cdot \cos\omega\tau + \operatorname{Im} b \cdot \sin\omega\tau = 0, \\ \operatorname{Im} F(i\omega) = \omega - \operatorname{Re} b \cdot \sin\omega\tau + \operatorname{Im} b \cdot \cos\omega\tau = 0. \end{cases} \tag{1.18}$$

Отсюда получаем кривую $D$-разбиения на комплексной плоскости переменного $b$:

$$\begin{aligned} \operatorname{Re} b &= -a\cos\omega\tau + \omega\sin\omega\tau, \\ \operatorname{Im} b &= -a\sin\omega\tau - \omega\cos\omega\tau. \end{aligned} \tag{1.19}$$

Кривая (1.19) при $\omega \in (-\infty, \infty)$ разбивает комплексную плоскость параметра $b$ на бесконечное множество областей, соответствующих либо устойчивости, либо неустойчивости уравнения (1.11). Проверить каждую область на устойчивость можно, проверив какую-либо внутреннюю точку (тестовую точку) этой области.

На рисунке 1.3 изображены кривые $D$-разбиения комплексной плоскости параметра $b$ при $\omega \in (-2.7\,\pi,\ 2.7\,\pi),\ a = 1, \tau = 1$.

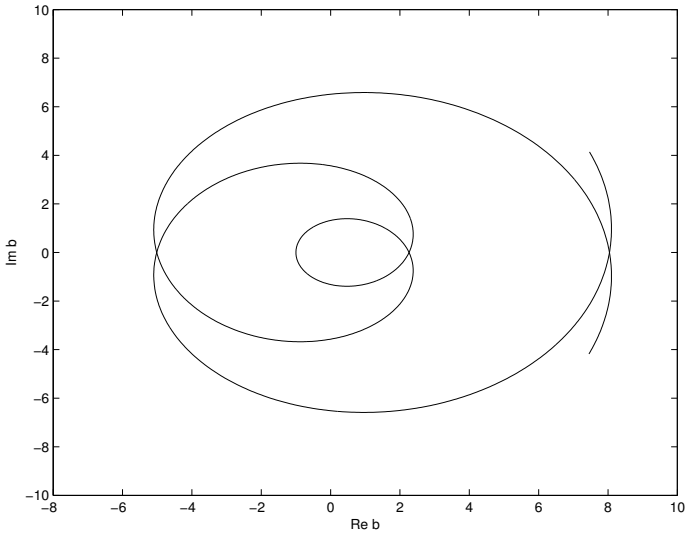



Рис. 1.3. Кривые $D$-разбиения комплексной плоскости параметра $b$

В качестве тестовых будем рассматривать точки на прямой $\operatorname{Im} b = 0$, так как эта прямая проходит через все области.

Перейдём к перечислению случаев различного расположения числа $a$ относительно $-\frac{1}{\tau}$.

1) Пусть $a \leqslant -\frac{1}{\tau}$. Тогда в каждой области $D$-разбиения плоскости комплексного переменного $b$ возьмём в качестве тестовой именно точку с $\operatorname{Im} b = 0$. По части 1 предложения 1.1 в этой точке имеется неустойчивость. Поэтому и во всей рассматриваемой области имеется неустойчивость. Пункт 1 теоремы 1.1 доказан.

2) Пусть $a > -\frac{1}{\tau}$ и точка $(\operatorname{Re} b, \operatorname{Im} b)$ находится внутри овала устойчивости для данного $a$. Рассмотрим в качестве тестовых точки точки внутри овала устойчивости, такие что $b \in \mathbb{R}$, $b \in (b(0), b(\omega_1)$. Овал изображен на рисунке 1.4, а точки $b \in \mathbb{R}$, $b \in (b(0), b(\omega_1))$ отмечены пунктиром.

На овале устойчивости $b(0) = -a$, $b(\omega_1) = \frac{\omega_1}{\sin \omega_1 \tau} = S(a)$, поэтому неравенство $b(0) < b < b(\omega_1)$ равносильно неравенству $-a < b < S(a)$, что в си-

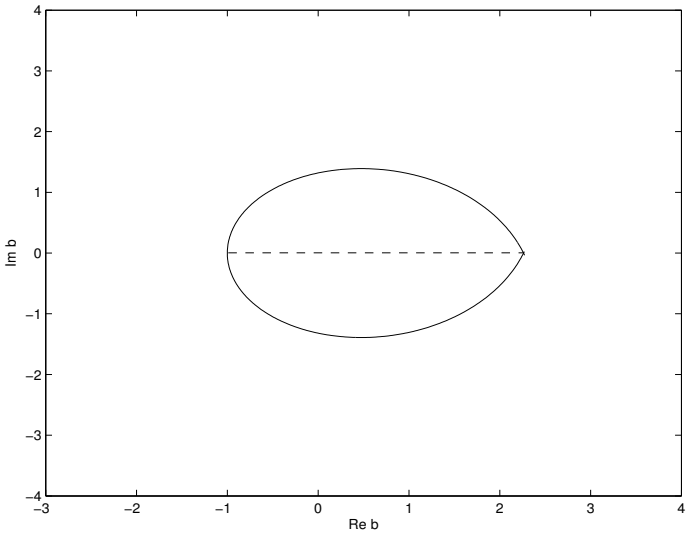



Рис. 1.4. Овал устойчивости уравнения (1.11)

лу части 2 предложения 1.1 влечёт асимптотическую устойчивость уравнения (1.11). Пункт 2 теоремы 1.1 доказан.

3) Пусть $a > -\dfrac{1}{\tau}$ и точка $b$ лежит на границе овала устойчивости для данного $a$. Нам следует доказать, что корень $\lambda$ характеристического уравнения, такой что $\operatorname{Re}\lambda = 0$, является некратным. Действительно, из (1.17) получим

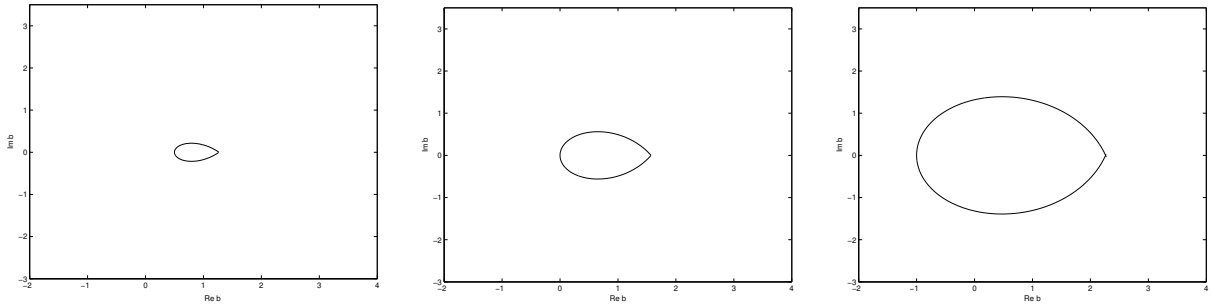
$$\frac{dF}{d\lambda} = 1 - b\tau\, e^{-i\omega\tau}. \tag{1.20}$$

Обращение в нуль функции $F(\lambda)$ и её производной в силу (1.17), (2.15) даёт $\lambda + a + \dfrac{1}{\tau} = 0$, что влечёт $\lambda \in \mathbb{R}$, $\lambda < 0$. Пункт 3 теоремы доказан.

4) Пусть $a > -\frac{1}{\tau}$ и точка $(\operatorname{Re} b,\, \operatorname{Im} b)$ находится вне овала устойчивости. Тогда точка $b$ находится в одной из областей $D$-разбиения, и для проверки устойчивости достаточно рассмотреть действительные значения $b$. Для таких $b$ либо $b < b(0) = -a$, либо $b > b(\omega_1) = S(a)$. В обоих случаях по части 4 предложения 1.1 уравнение (1.11) неустойчиво. Теорема доказана. ■

Примечание 1.1. Положение овала устойчивости на плоскости параметра $b$ зависит от значения $a$. На рисунке (1.5) на комплексной плоскости параметра $b$ изображены овалы устойчивости при различных значениях параметра $a$. При $a \leqslant -\frac{1}{\tau}$ овал устойчивости вообще не существует. При $0 > a > -\frac{1}{\tau}$

овал полностью расположен в полуплоскости $\operatorname{Re} b > 0$. При $a = 0$ крайняя левая точка овала лежит в начале координат, а остальные точки овала находится в полуплоскости $\operatorname{Re} b > 0$. При $a > 0$ граница овала находится в обеих полуплоскостях.




(*а*) (*б*) (*в*)

Рис. 1.5. Овал устойчивости: (а) $-\frac{1}{\tau} < a < 0$, (б) $a = 0$, (в) $a > 0$

## 1.3 Конус устойчивости для скалярного уравнения с комплексными коэффициентами

Поначалу рассмотрим уравнение (1.11) с действительными коэффициентами $a$ и комплексными $b$. Для исследования его устойчивости, а в дальнейшем также и для матричного уравнения (1.6) определим в $\mathbb{R}^3$ некоторую поверхность, которую назовём конусом устойчивости.

**Определение 1.2.** *Конусом устойчивости для уравнений* (1.11) *и* (1.6) *назовём множество точек* $M = (u_1, u_2, u_3) \in \mathbb{R}^3$, *таких, что*

$$\begin{cases} u_1 = -h\cos\omega + \omega\sin\omega, \\ u_2 = h\sin\omega + \omega\cos\omega, \\ u_3 = h, \end{cases} \tag{1.21}$$

*где действительные параметры $h, \omega$ подчинены ограничениям*

$$\begin{cases} h \geqslant -\dfrac{\omega}{\operatorname{tg}\omega}, \\ -\pi < \omega < \pi. \end{cases} \tag{1.22}$$

Примечание 1.2. Единственный конус устойчивости (1.21), (1.22) строится для класса всех уравнений вида (1.11) и (1.6).

Ключевым свойством конуса устойчивости является следующее. Сечение конуса устойчивости плоскостью $u_3 = a$ таково, что:

1) При $a < -1$ сечение является пустым множеством, при $a = -1$ единственная точка сечения $(1, 0, -1)$ — вершина конуса.

2) При $a > -1$ сечение является овалом устойчивости для данного $a$ для уравнения (1.11) при $\tau = 1$.

Чтобы записать теорему 1.1 в терминах конуса устойчивости, сделаем в уравнении (1.11) замену $t = \theta\,\tau$. Получим

$$\frac{dx\,(\theta\tau)}{d(\theta\tau)} + a\,x\,(\theta\tau) + b\,x\,((\theta-1)\tau) = 0. \tag{1.23}$$

Обозначим $x\,(\theta\tau) = y\,(\theta)$. Отметим, что

$$\frac{dy\,(\theta)}{d\,\theta} = \frac{dx\,(\theta\tau)}{d\,\theta} = \frac{dx\,(\theta\tau)}{d(\theta\tau)}\,\tau, \tag{1.24}$$

откуда

$$\frac{\dot{y}_\theta(\theta)}{\tau} + a\,y\,(\theta) + b\,y\,(\theta-1) = 0. \tag{1.25}$$

Переобозначив $\theta = t$, получим

$$\dot{y}\,(t) + a\tau\,y(t) + b\tau\,y(t-1) = 0. \tag{1.26}$$

При условии $\tau > 0$ уравнение (1.26) устойчиво тогда и только тогда, когда устойчиво уравнение (1.11). Прямое применение теоремы 1.1 к уравнению (1.26) доказывает следующую теорему.

**Теорема 1.2.** *Пусть в уравнении* (1.11) $a \in \mathbb{R}$, $b \in \mathbb{C}$, $\tau > 0$. *Построим точку* $M = (\operatorname{Re} b\tau,\, \operatorname{Im} b\tau,\, a\tau) \in \mathbb{R}^3$. *Тогда имеют место следующие утверждения.*

1. *Уравнение* (1.11) *асимптотически устойчиво тогда и только тогда, когда точка* $M$ *находится внутри конуса устойчивости.*

2. *Если точка* $M$ *находится на поверхности конуса устойчивости (но не в его вершине), то уравнение* (1.11) *устойчиво (не асимптотически).*

3. *Если точка* $M$ *находится вне конуса устойчивости или на его вершине, то уравнение* (1.11) *неустойчиво.*

**Пример 1.3.** Пусть дано уравнение (1.11), для которого

$$a = 0.8,\ b = 1 + 0.5\,i. \tag{1.27}$$

Требуется определить, при каких значениях $\tau$ это уравнение устойчиво, а при каких неустойчиво.

Для асимптотической устойчивости согласно теореме 1.2 необходимо и достаточно, чтобы точка $(\operatorname{Re} b\tau,\, \operatorname{Im} b\tau,\, a\tau)$ находилась внутри конуса устойчивости. Построим конус устойчивости и точки $(\operatorname{Re} b\tau,\, \operatorname{Im} b\tau,\, a\tau)$ при $\tau \in$ $\in [0, +\infty)$, образующие некоторую прямую. Конус устойчивости и данная прямая изображены на рисунке (1.6).

Прямая пересекает конус устойчивости при $\tau = 0$, что соответствует точке $(0, 0, 0)$, и при $\tau = \tau_0 \approx 2.4$. При $\tau \in (0, \tau_0)$ точка $(\operatorname{Re} b\tau,\, \operatorname{Im} b\tau,\, a\tau)$ лежит внутри конуса устойчивости, поэтому уравнение (1.11), (1.27) асимптотически устойчиво. При $\tau > \tau_0$ точка находится вне конуса устойчивости, поэтому уравнение неустойчиво. При $\tau = \tau_0$ уравнение устойчиво (не асимптотически).

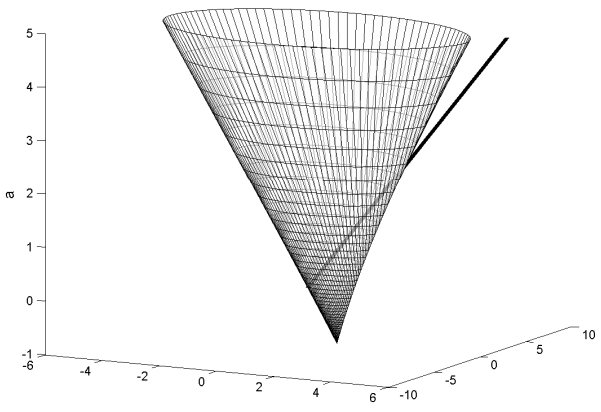



Рис. 1.6. Конус устойчивости и точки ($\operatorname{Re} b\tau$, $\operatorname{Im} b\tau$, $a\tau$) при $\tau \in [0, 5]$

Перейдём к уравнению (1.11) с двумя комплексными коэффициентами. Рассмотрим уравнение

$$\dot{x}\,(t) + \lambda\, x(t) + \mu\, x(t-\tau) = 0, \tag{1.28}$$

где $\lambda, \mu \in \mathbb{C}$.

Для исследования устойчивости уравнения (1.28) сделаем замену

$$x(t) = y(t)\, \exp(-it \operatorname{Im} \lambda). \tag{1.29}$$

В результате замены (1.29) и сокращений получим уравнение

$$\dot{y}\,(t) + (\operatorname{Re} \lambda)\, y(t) + \mu\, \exp(i\tau \operatorname{Im} \lambda)\, y(t-\tau) = 0. \tag{1.30}$$

Так как $|\exp(-it \operatorname{Im} \lambda)| = 1$, уравнение (1.28) (асимптотически) устойчиво тогда и только тогда, когда уравнение (1.30) (асимптотически) устойчиво.

Ввиду того, что коэффициент при $y(t)$ в (1.30) является действительным, задачу устойчивости уравнения (1.30) мы можем решить с помощью теоремы (1.2). Таким образом, мы доказали следующую теорему.

**Теорема 1.3.** *Пусть в уравнении* (1.28) $\lambda, \mu \in \mathbb{C}$. *Построим точку* $M = (u_1, u_2, u_3)$ *так, что*

$$\begin{aligned} u_1 &= \tau \operatorname{Re}(\mu \exp(i\tau \operatorname{Im} \lambda)), \\ u_2 &= \tau \operatorname{Im}(\mu \exp(i\tau \operatorname{Im} \lambda)), \\ u_3 &= \tau \operatorname{Re} \lambda. \end{aligned} \tag{1.31}$$

*Тогда имеют место следующие утверждения.*

1. *Уравнение* (1.28) *асимптотически устойчиво тогда и только тогда, когда точка* $M$ *находится внутри конуса устойчивости.*

2. *Если точка* $M$ *находится на поверхности конуса устойчивости (но не в его вершине), то уравнение* (1.28) *устойчиво (не асимптотически).*

3. *Если точка* $M$ *находится вне конуса устойчивости или на его вершине, то уравнение* (1.28) *неустойчиво.*

## 1.4 Конус устойчивости для матричного уравнения

Возвратимся к рассмотрению матричного уравнения (1.6). Для удобства ссылок выпишем его заново:

$$\dot{x}(t) + Ax(t) + Bx(t - \tau) = 0. \tag{1.32}$$

Следующая теорема является основой для алгоритмов диагностирования устойчивости стационарных состояний нейронных сетей. Теорема пригодна только для случая, когда матрицы $A, B$ могут быть приведены одним преобразованием к треугольному виду. Но, как показывает практика изучения устойчивости нейронных сетей, этого достаточно для анализа многих типичных сетей.

**Теорема 1.4.** *Пусть* $A, B, S \in \mathbb{R}^{m\times m}$ *и* $S^{-1}AS = A_T$ *и* $S^{-1}BS = B_T$, *где* $A_T$ *и* $B_T$ *— нижние треугольные матрицы с элементами соответственно* $\lambda_{j\,s}, \mu_{j\,s}\,(1 \leqslant j,\, s \leqslant m)$. *Построим систему точек* $M_j = (u_{1j}, u_{2j}, u_{3j})$, $(1 \leqslant j \leqslant m)$, *так, что*

$$\begin{aligned} u_{1j} &= \tau \operatorname{Re}(\mu_{jj}\, \exp(i\tau \operatorname{Im} \lambda_{jj})), \\ u_{2j} &= \tau \operatorname{Im}(\mu_{jj}\, \exp(i\tau \operatorname{Im} \lambda_{jj})), \\ u_{3j} &= \tau \operatorname{Re} \lambda_{jj}. \end{aligned} \tag{1.33}$$

*Уравнение* (1.32) *асимптотически устойчиво тогда и только тогда, когда все точки* $M_j$ $(1 \leqslant j \leqslant m)$ *находятся внутри конуса устойчивости (см. определение 1.2). Если хотя бы одна точка* $M_j$ $(1 \leqslant j \leqslant m)$ *лежит вне конуса устойчивости, то уравнение* (1.32) *неустойчиво.*

**Доказательство.** Введём новую функцию $y = (y_1, \ldots, y_m)^T$ равенством $x = Cy$ и умножим уравнение (1.32) на $C^{-1}$ слева. Получим

$$\dot{y}(t) + A_T y(t) + B_T y(t-\tau) = 0. \tag{1.34}$$

Уравнение (1.32) (асимптотически) устойчиво тогда и только тогда, когда (асимптотически) устойчиво (1.34). Рассмотрим диагональную систему

$$\dot{y}_j(t) + \lambda_{jj}\, y_j(t) + \mu_{jj}\, y_j(t-\tau) = 0, \quad 1 \leqslant j \leqslant m. \tag{1.35}$$

Системы (1.34) и (1.35) имеют одинаковые характеристические уравнения

$$\prod_{j=1}^{m} (\lambda + \lambda_{jj} + \mu_{jj} \exp(-\lambda\tau)) = 0. \tag{1.36}$$

Поэтому асимптотическая устойчивость системы (1.34) равносильна асимптотической устойчивости системы (1.35). Неустойчивость (1.34) также равносильна неустойчивости (1.35). Применение теоремы 1.3 к каждому из $n$ скалярных уравнений системы (1.35) завершает доказательство теоремы. ■

Примечание 1.3. Формулы (1.21) в определении конуса устойчивости можно записать компактнее:

$$u_1 + i\,u_2 = (i\,\omega - h)\exp(-i\,\omega), \quad u_3 = h. \tag{1.37}$$

Взамен (1.33) точки $M_j$ можно определить равенствами

$$u_{1j} + i\,u_{2j} = \tau\mu_{jj}\exp(i\tau\,\mathrm{Im}\;\lambda_{jj}), \quad u_{3j} = \tau\,\mathrm{Re}\;\lambda_{jj}. \tag{1.38}$$

**Пример 1.4.** С помощью геометрического метода теоремы 1.4 найдём значения запаздывания $\tau$, обеспечивающие асимптотическую устойчивость уравнения (1.32) с матрицами

$$A = \begin{pmatrix} 0.72 & -0.96 \\ 4.08 & -0.60 \end{pmatrix}, \qquad B = \begin{pmatrix} 0.08 & -0.08 \\ 0.34 & -0.03 \end{pmatrix}. \tag{1.39}$$

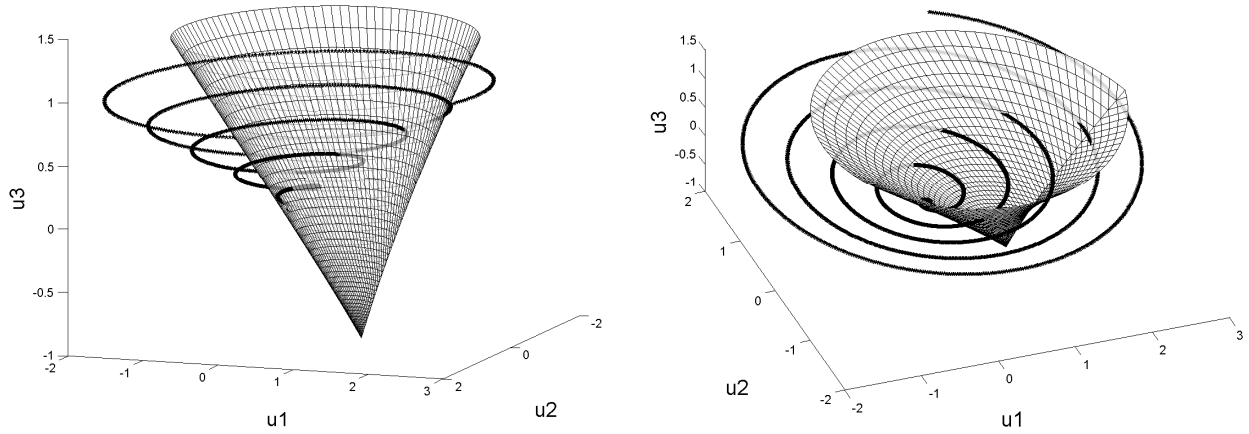
Поскольку $A = 12B - 0.24I$, матрицы $A, B$ коммутируют и, следовательно, могут быть совместно приведены к диагональной форме [20]. Собственные числа матриц $A, B$ суть соответственно

$$\lambda_{11,22} = 0.0600 \pm 1.8658i, \quad \mu_{11,22} = 0.0250 \pm 0.1555i. \tag{1.40}$$

Вследствие симметрии изучим расположение конической винтовой линии (1.33) относительно конуса устойчивости (1.21), (1.22) только при $j = 1$. Кривая имеет вид (см. рис. 1.7)

$$\begin{aligned} u_{11} &= 0.1575\tau\cos(1.4114 + 1.8658\tau), \\ u_{21} &= 0.1575\tau\sin(1.4114 + 1.8658\tau), \\ u_{31} &= 0.06\tau. \end{aligned} \tag{1.41}$$

Точки кривой находятся внутри конуса устойчивости при следующих значениях запаздывания: $\tau \in (0, 0.2729) \cup (1.6893, 3.3978) \cup (5.3439, 6.5218) \cup$ $\cup\,(8.9955, 9.6457) \cup (12.6502, 12.7675)$, поэтому уравнение (1.32) с матрицами (1.39) асимптотически устойчиво только при таких значениях запаздывания.




(*а*) (*б*)

Рис. 1.7. Конус устойчивости и кривая (1.41) в разных проекциях

**Пример 1.5.** Положим в (1.32)

$$A = \begin{pmatrix} 1 & 1 & -3 & 2 \\ -3 & 1 & -2 & 3 \\ -1 & 0.5 & 4 & -2 \\ -3 & 2 & -1 & 1 \end{pmatrix}, B = 0.8I + 0.2A + 2A^2. \tag{1.42}$$

Требуется определить, при каких значениях $\tau$ это уравнение устойчиво, а при каких неустойчиво.

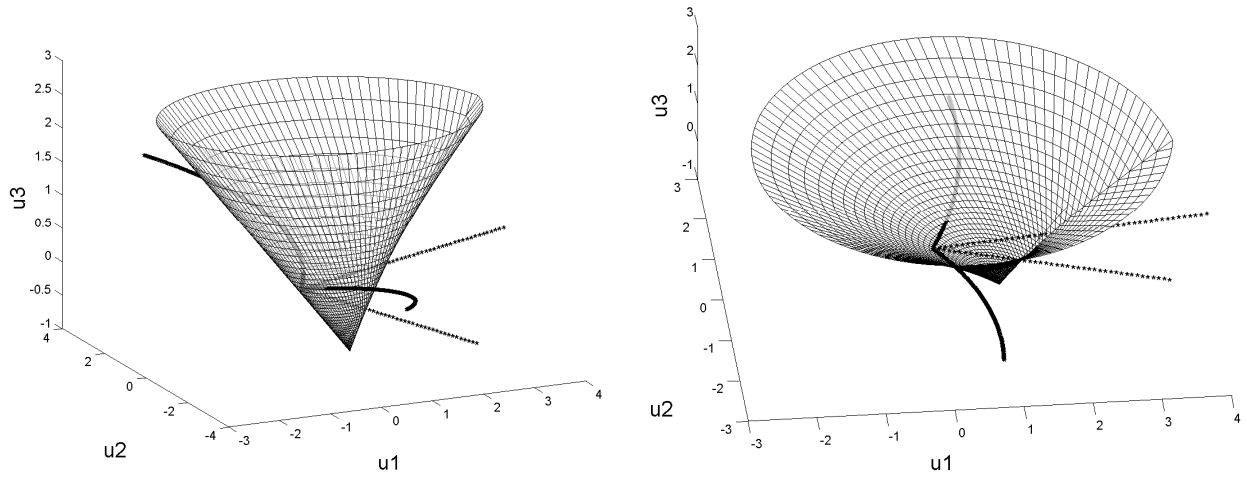
Очевидно, матрицы $A, B$ коммутируют. Собственные числа матриц $A$ и $B$ имеют вид

$$\lambda_1 = 3.562, \qquad \mu_1 = 26.888,$$
$$\lambda_{2,3} = 2.368 \pm 1.514i, \qquad \mu_{2,3} = 7.902 \pm 14.640i,$$
$$\lambda_4 = -1.298, \qquad \mu_4 = 3.907.$$

При этом

$$a_1 = \operatorname{Re}\lambda_1 = 3.562, \qquad a_2 = \operatorname{Re}\lambda_2 = 2.368,$$
$$a_3 = \operatorname{Re}\lambda_3 = 2.368, \qquad a_4 = \operatorname{Re}\lambda_4 = -1.298.$$
$$b_1 = \mu_1 \exp(i\tau \operatorname{Im}\lambda_1) = 26.888,$$
$$b_2 = \mu_2 \exp(i\tau \operatorname{Im}\lambda_2) = 16.637\exp(i(1.076 + 1.514\tau)),$$
$$b_3 = \mu_3 \exp(i\tau \operatorname{Im}\lambda_3) = 16.637\exp(i(-1.076 - 1.514\tau)),$$
$$b_4 = \mu_4 \exp(i(\tau \operatorname{Im}\lambda_4) = 3.9073.$$

Построим кривые $(\operatorname{Re} b_j\tau,\ \operatorname{Im} b_j\tau,\ a_j\tau), j = \overline{1,4}$ и конус устойчивости. Они изображены на рисунке 1.8.




(*а*) (*б*)

Рис. 1.8. Конус устойчивости и кривая $(\operatorname{Re} b_j\tau,\ \operatorname{Im} b_j\tau,\ a_j\tau)$ в разных проекциях

Для асимптотической устойчивости необходимо и достаточно, чтобы все точки кривой $(\operatorname{Re} b_j\tau,\ \operatorname{Im} b_j\tau,\ a_j\tau)$ находились внутри конуса устойчивости. Две системы точек, соответствующие комплексным собственным значениям матрицы $A$, с изменением $\tau$ образуют винтовые линии, пересекающие конус устойчивости при $\tau_{2,3} \approx 0.033$. Две другие системы точек, соответствующие

действительным собственным значениям матрицы $A$, находятся внутри конуса при $\tau \in (0, \tau_1)$, где $\tau_1 \approx 0.063$, и при $\tau \in (0, \tau_4)$, где $\tau_4 \approx 0.321$.

Итак, уравнение (1.32) с матрицами (1.42) асимптотически устойчиво тогда и только тогда, когда $\tau \in (0, \tau_0)$, где $\tau_0 = \min(\tau_1, \tau_{2,3}, \tau_4) = \tau_{2,3} \approx 0.033$. При $\tau > \tau_0$ уравнение (1.32) неустойчиво.

Обсудим следующую задачу: при каких условиях уравнение (1.32) с $n \times n$ матрицами $A, B$ является устойчивым независимо от запаздывания $\tau$?

Рассмотрим $n+2$ конуса в $\mathbb{R}^3$. Первый $K^1$ — конус устойчивости (см. определение 1.2). Второй $K^2$ — конус $u_1^2 + u_2^2 = u_3^2$, $u_3 \geqslant 0$. Конус $K^1$ содержит $K^2$ и оба конуса неограниченно сближаются при $u_3 \to +\infty$. Если $\operatorname{Re} \lambda_{jj} \neq 0$, $\mu_{jj} \neq 0$, то другие $n$ конусов зададим формулами

$$K_j : \quad (\operatorname{Re} \lambda_{jj})^2 (u_{11}^2 + u_{22}^2) = |\mu_{jj}|^2 u_3^2, \quad 1 \leqslant j \leqslant n, \tag{1.43}$$

где $\lambda_{jj}, \mu_{jj}$ диагональные элементы матриц $A, B$ соответственно.

На конусах $K_j$ лежат конические спирали 1.38, указанные в теореме 1.4 (спираль вырождается в прямую, если $\operatorname{Im} \lambda_{jj} = 0$). Если $\operatorname{Re} \lambda_{jj} < 0$, то $u_3 < 0$ в 1.38, и тогда точки $M_j$ выходят из $K^1$ при неограниченном увеличении $\tau$ (при $\mu_{jj} \neq 0$), что влечёт неустойчивость уравнения (1.32) Следовательно, асимптотическая устойчивость (1.32), независимая от запаздывания $\tau$, имеет место тогда и только тогда, когда для любого $j$ ($1 \leqslant j \leqslant n$ ) $\operatorname{Re} \lambda_{jj} > 0$ и конус $K^j$ лежит внутри $K^2$, то есть

$$|\mu_{jj}| < \operatorname{Re} \lambda_{jj}. \tag{1.44}$$

Итак, мы доказали следующую теорему.

**Теорема 1.5.** *Пусть $A, B, S \in \mathbb{R}^{m \times m}$ и $S^{-1}AS = A_T$ и $S^{-1}BS = B_T$, где $A_T$ и $B_T$ — нижние треугольные матрицы с элементами соответственно $\lambda_{j\,s}, \mu_{j\,s}$ $(1 \leqslant j,\, s \leqslant m)$. Для того, чтобы уравнение* (1.32) *было асимпто-*

*тически устойчивым при любом запаздывании* $\tau \geqslant 0$*, необходимо и достаточно выполнение условия* (1.44).

Автору неизвестно, есть ли условие (1.44) в литературе, кроме статьи автора совместно с научным руководителем и В. В. Малыгиной [21]. Другое, не столь удобное условие устойчивости, независимой от запаздывания (delay-independent) дано в [24].

Наш подход даёт геометрическое понимание условия (1.44), гарантирующего устойчивость, независимую от запаздывания.

## 1.5 Проверка устойчивости уравнения с комплексными коэффициентами посредством системы неравенств

Для уравнения (1.28) сформулируем и докажем теорему, позволяющую делать вывод об устойчивости данного уравнения с помощью системы неравенств, не прибегая при этом к геометрическим процедурам. Другое решение этой задачи имеется в работе А. И. Кирьянена и К. В. Галуновой [17]. В дальнейшем в диссертации мы не будем использовать результаты этого раздела. Мы внесли эту тему в диссертацию, чтобы продемонстрировать «от противного» преимущества метода конуса устойчивости.

Рассмотрим уравнение

$$\dot{x}\,(t) + \rho_1 \exp(i\alpha_1)\, x(t) + \rho_2 \exp(i\alpha_2)\, x(t-\tau) = 0, \tag{1.45}$$

где $\rho_1, \rho_2, \alpha_1, \alpha_2 \in \mathbb{R}$.

В соответствии с (1.30) оно эквивалентно следующему уравнению:

$$\dot{y}\,(t) + \rho_1 \cos\alpha_1\; y(t) + \rho_2 \exp(i(\alpha_2 + \rho_1\tau \sin\alpha_1))\, y(t-\tau) = 0. \tag{1.46}$$

Поскольку в (1.46) коэффициент при $y(t)$ действительный, можно применить теорему 1.2 с заменой уравнения (1.11) на (1.45) и при

$$a = \rho_1 \cos\alpha_1, \qquad b = \rho_2 \exp(i(\alpha_2 + \rho_1\tau \sin\alpha_1)). \tag{1.47}$$

Будем использовать функцию $\arg z$ комплексного переменного $z$, считая, что $0 \leqslant \arg z < 2\pi$.

**Теорема 1.6.** *Пусть дано уравнение* (1.45). *Тогда следующие утверждения верны.*

1. *Если* $\rho_2 < \rho_1 \cos\alpha_1$, *то уравнение* (1.45) *асимптотически устойчиво.*

2. *Если* $-\min(1/\tau,\, \rho_2) < \rho_1 \cos\alpha_1 < \rho_2$, *то уравнение* (1.45) *асимптотически устойчиво тогда и только тогда, когда одновременно выполняются условия:*

$$\rho_2 < \sqrt{\omega_1^2 + \rho_1^2 \cos^2\alpha_1}, \tag{1.48}$$

*где* $\omega_1$ *есть наименьший положительный корень уравнения* $a\sin\omega\tau + \omega\cos\omega\tau = 0$, *и*

$$\arg\left(\exp(i(\alpha_2 + \rho_1\tau\sin\alpha_1))\right) \in (0, \beta) \cup (2\pi - \beta, 2\pi), \tag{1.49}$$

*где*

$$\beta = \arg\left(\exp\left(-i\tau\sqrt{\rho_2^2 - \rho_1^2\cos^2\alpha_1}\right)\right) + \gamma, \tag{1.50}$$

$$\gamma = \begin{cases} \operatorname{arctg}\dfrac{\sqrt{\rho_2^2 - \rho_1^2\cos^2\alpha_1}}{\rho_1\cos\alpha_1}, & \cos\alpha_1 < 0, \\ \pi - \operatorname{arctg}\dfrac{\sqrt{\rho_2^2 - \rho_1^2\cos^2\alpha_1}}{\rho_1\cos\alpha_1}, & \cos\alpha_1 > 0. \end{cases}$$

3. *Если* $\rho_1\cos\alpha_1 < -\min(1/\tau,\, \rho_2)$, *то уравнение* (1.45) *неустойчиво.*

**Доказательство.** Уравнение (1.45) асимптотически устойчиво тогда и только тогда, когда асимптотически устойчиво уравнение (1.46). Поэтому будем исследовать устойчивость уравнения (1.46).

1) При $\rho_2 < \rho_1 \cos\alpha_1$ по теореме 1.1 и примечанию 1.1 овал устойчивости существует и начало координат лежит внутри овала устойчивости. При этом окружность радиуса $\rho_2$ полностью лежит внутри овала устойчивости, что обеспечивает асимптотическую устойчивость уравнения (1.46).

2) Пусть $-\min(1/\tau,\, \rho_2) < \rho_1 \cos\alpha_1 < \rho_2$. Рассмотрим 2 случая.

*Случай 2.1.* $-\min(1/\tau,\, \rho_2) < \rho_1 \cos\alpha_1 < 0$. При этом овал устойчивости существует и полностью лежит в правой полуплоскости. Для асимптотической устойчивости необходимо и достаточно выполнение двух условий. Одно из них обеспечивает пересечение окружности радиуса $\rho_2$ с овалом устойчивости, что равносильно условию (1.48). Второе условие связано с тем, что аргумент точки $\exp(i(\alpha_2 + \rho_2\tau\sin\alpha_1))$ должен находится между аргументом точек $M, N$ пересечения окружности радиуса $\rho_2$ с овалом устойчивости. Рассмотрим точку $M$, лежащую в верхней полуплоскости. Отметим, что часть овала устойчивости (1.15), лежащая в верхней полуплоскости, соответствует отрицательным значениям параметра $\omega$.

$$\begin{aligned}\arg M = \arg\left(-\rho_1\cos\alpha_1\exp(i\omega\tau) - i\omega\exp(i\omega\tau)\right) = \\ = \arg(\exp(i\omega\tau)) + \arg(-\rho_1\cos\alpha_1 - i\omega) = \\ = \arg(\exp(i\omega\tau)) + \operatorname{arctg}\frac{|\omega|}{\rho_1\cos\alpha_1}. \quad (1.51)\end{aligned}$$

По определению овала устойчивости (см. определение 1.1) имеем $\omega = \pm\sqrt{\rho_2^2 - \rho_1^2 \cos^2\alpha_1}$. Поэтому

$$\arg M = \arg\left(\exp\left(-i\tau\sqrt{\rho_2^2 - \rho_1^2\cos^2\alpha_1}\right)\right) + \operatorname{arctg}\frac{\sqrt{\rho_2^2 - \rho_1^2\cos^2\alpha_1}}{\rho_1\cos\alpha_1}. \quad (1.52)$$

Из (1.52) следует, что второе условие равносильно (1.49).

*Случай 2.2.* $0 < \rho_1\cos\alpha_1 < \rho_2$. Так как $\cos\alpha_1 > 0$, то овал устойчивости существует и начало координат лежит внутри овала устойчивости. Те же требования приводят к условиям (1.48), (1.49). Отметим, что при $\cos\alpha_1 > 0$

получим

$$\arg(-\rho_1 \cos\alpha_1 - i\omega) = \pi - \operatorname{arctg} \frac{\sqrt{\rho_2^2 - \rho_1^2 \cos^2 \alpha_1}}{\rho_1 \cos\alpha_1}. \tag{1.53}$$

3) Пусть $\rho_1 \cos\alpha_1 < -\min(1/\tau,\, \rho_2)$. Возможны два случая.

*Случай 3.1.* $\rho_1 \cos\alpha_1 < -\frac{1}{\tau}$. Тогда по теореме 1.1 уравнение неустойчиво.

*Случай 3.2.* $-\frac{1}{\tau} < \rho_1 \cos\alpha_1 < -\rho_2$. При этом овал устойчивости существует и начало координат лежит внутри овала устойчивости. Так как $\rho_2 < -\rho_1 \cos\alpha_1$, то окружность радиуса $\rho_2$ не пересекает овал устойчивости, поэтому уравнение неустойчиво. Теорема доказана. ■

## 1.6 Сравнение результатов главы 1 с известными результатами

Идейным источником метода конусов устойчивости явились работы Рехлицкого, в частности, статья [22] (1956), в которой впервые появился овал устойчивости для некоторого линейного уравнения с запаздыванием в банаховом пространстве.

Результаты главы 1 диссертации сильнее результатов статьи Б. Калона и Д. Шмидта [25] (2000), в которой рассматривалась задача об устойчивости узкого класса уравнений вида

$$\dot{x}(t) = \alpha A x(t) + (1-\alpha) A x(t-\tau), \quad 0 \leqslant \alpha \leqslant 1. \tag{1.54}$$

с $n \times n$ матрицей $A$.

Наши результаты также перекрывают результаты Х. Мацунаги [26], который в 2007 г. исследовал устойчивость частного случая нашего основного уравнения (1.6) с некоторыми специальными $2 \times 2$ матрицами $A, B$, коммутируемыми, и поэтому приводимыми к треугольному виду.

Устойчивость круговых нейронных сетей с четырьмя [14] (1999) и с произвольным количеством нейронов [15] (2004) изучены в работах С. Кэмпбелл

с соавторами. Такие сети описываются дифференциальными уравнениями с двумя запаздываниями вида

$$\dot{x}_j(t)+x_j(t)+\alpha x_j(t-\tau_0)+\beta\left(x_{j-1}(t-\tau)+x_{j+1}(t-\tau)\right)=0 \qquad (j \bmod n). \quad (1.55)$$

Наши методы непригодны для работы с системами с двумя и более запаздываниями. В дальнейшем в диссертации мы рассмотрим круговую сеть нейронов, которая, с одной стороны, проще, чем (1.55), поскольку в ней отсутствует запаздывание в реакции нейрона на собственное состояние ($\alpha = 0$), с другой стороны, сложнее, поскольку в ней отсутствует принудительная симметричность мощности взаимодействия нейронов с левым и правым соседями в круге.

# Глава 2

# Исследование устойчивости стандартных нейронных сетей

## 2.1 Алгоритм для определения значений запаздывания, гарантирующих устойчивость дифференциального уравнения с запаздыванием

В этом разделе мы дадим алгоритм для определения всех значений $\tau$, которые гарантируют устойчивость дифференциального уравнения

$$\dot{x}(t) + Ax(t) + Bx(t-\tau) = 0 \tag{2.1}$$

с $n \times n$ матрицами $A\,, B$, допускающими одновременную триангуляцию.

Пусть матрицы $A$ и $B$ приведены к треугольному виду $A_T$, $B_T$ одним преобразованием. Пусть $\lambda_{j\,j}, \mu_{j\,j},\ 1 \leqslant j \leqslant n$ собственные числа матриц $A$ и $B$ в том порядке, в котором они расположены на диагоналях матриц $A_T$, $B_T$. Для каждого $j,\ 1 \leqslant j \leqslant n$ составим множество $T_j$ по следующему **алгоритму.**

Если для некоторого $j$ выполнено $\operatorname{Re}\lambda_{j\,j} > |\mu_{j\,j}|$, то $T_j = (0, +\infty)$. В противном случае находим числа

$$\tau_j^m = \frac{\arg\left(\pm iQ - \operatorname{Re}\lambda_{j\,j}\right) - \arg\mu_{j\,j} + 2\pi m}{\operatorname{Im}\lambda_{j\,j} \pm Q}, \quad m \in \mathbb{Z}, \quad Q = \sqrt{|\mu_{j\,j}|^2 - (\operatorname{Re}\lambda_{j\,j})^2}. \tag{2.2}$$

Из чисел $\tau_j^m$ выбираем такие, для которых

$$0 < \tau_j^m \leq \pi/Q \quad \text{и} \quad \operatorname{Re}\lambda_{j\,j} \geqslant -\frac{Q}{\operatorname{tg}(Q\,\tau_j^m)}, \tag{2.3}$$

и сортируем по возрастанию, исключая повторения. Полученные числа обозначим $\tau_{1j}, \tau_{2j}, \ldots, \tau_{sj}$.

Составим множество $T_j$ следующим образом. Пусть

$$T_j = (0, \tau_{1j}) \cup (\tau_{2j}, \tau_{3j}) \cup \dots \tag{2.4}$$

в следующих случаях:

1) $\operatorname{Re} \lambda_{jj} > 0$ и выполнено хотя бы одно условие

$$\text{а) } \arg \mu_{j\,j} \in (-\pi/2, \pi/2) \quad \text{или} \quad \text{б) } \frac{\operatorname{Re} \lambda_{jj}}{|\operatorname{Re} \mu_{jj}|} > 1. \tag{2.5}$$

$$2) \operatorname{Re} \lambda_{jj} < 0, \quad \left| \frac{\operatorname{Re} \lambda_{jj}}{\operatorname{Re} \mu_{jj}} \right| < 1 \quad \text{и} \quad \arg \mu_{j\,j} \in (-\pi/2, \pi/2). \tag{2.6}$$

В остальных случаях

$$T_j = (\tau_{1j}, \tau_{2j}) \cup (\tau_{3j}, \tau_{4j}) \cup \dots \tag{2.7}$$

Последним интервалом в $T_j$ может быть $(\tau_{s-1\,j}, \tau_{sj})$ или $(\tau_{sj}, +\infty)$. Если для некоторого $j$ множество $(\tau_{1j}, \tau_{2j}, \dots, \tau_{sj})$ пусто и выполнено (2.5) или (2.6), то $T_j = (0, +\infty)$.

Докажем теорему.

**Теорема 2.1.** *Пусть множества $T_j, 1 \leqslant j \leqslant n$ составлены согласно описанному алгоритму. Тогда уравнение* (2.1) *асимптотически устойчиво если и только если $\tau \in \bigcap_j T_j$.*

**Доказательство.** Построим для каждого $j$ $(1 \leqslant j \leqslant n)$ точку $M_j = (u_{1j}, u_{2j}, u_{3j})$ $\mathbb{R}^3$, так что (см. (1.33))

$$\begin{aligned} u_{1j} &= \tau \operatorname{Re}(\mu_{jj} \exp(i\tau \operatorname{Im} \lambda_{jj})), \\ u_{2j} &= \tau \operatorname{Im}(\mu_{jj} \exp(i\tau \operatorname{Im} \lambda_{jj})), \\ u_{3j} &= \tau \operatorname{Re} \lambda_{jj}. \end{aligned} \tag{2.8}$$

Построим также конус устойчивости для уравнения (2.1), а именно множество точек $M = (u_1, u_2, u_3) \in \mathbb{R}^3$, таких, что

$$\begin{cases} u_1 = -h\cos\omega + \omega\sin\omega, \\ u_2 = h\sin\omega + \omega\cos\omega, \\ u_3 = h, \end{cases} \tag{2.9}$$

где действительные параметры $h, \omega$ подчинены ограничениям

$$\begin{cases} h \geqslant -\dfrac{\omega}{\operatorname{tg}\omega}, \\ -\pi < \omega < \pi. \end{cases} \tag{2.10}$$

Согласно Теореме 1.4 уравнение (2.1) асимптотически устойчиво если и только если все точки $M_j$ лежат внутри конуса устойчивости (2.9), (2.10).

С изменением $\tau$ от 0 до $+\infty$ точки (2.8) образуют винтовые линии (прямые в случае $\operatorname{Im}\lambda_{jj} = 0$). Для каждого $j$ найдем значения $\tau$, соответствующие точкам пересечения этих линий с овалом устойчивости $(u_1, u_2)$ на уровне $u_3 = h$. Эти значения $\tau$ являются корнями уравнения

$$\tau\mu_{jj}\exp(i\tau\operatorname{Im}\lambda_{jj}) = (i\omega - h)\,\exp(-i\omega), \quad h = \tau\operatorname{Re}\lambda_{jj}, \tag{2.11}$$

с параметрами $h, \omega$, удовлетворяющими условию (2.10).

Система (2.11) равносильна системе

$$\begin{cases} \tau|\mu_{jj}| = \sqrt{\omega^2 + h^2}, \\ \arg\mu_{jj} + \tau\operatorname{Im}\lambda_{jj} = -\omega + \arg(i\omega - h) + 2\pi m, \quad m \in \mathbb{Z}, \\ h = \tau\operatorname{Re}\lambda_{jj}. \end{cases} \tag{2.12}$$

Из первого и третьего уравнения системы (2.12) найдем $\omega$:

$$\omega = \pm\tau\sqrt{|\mu_{jj}|^2 - (\operatorname{Re}\lambda_{jj})^2} = \pm\tau Q. \tag{2.13}$$

Подставим $\omega$ во второе уравнение системы (2.12), а также воспользуемся свойством $\arg(\tau z) = \arg(z)$ для любого $\tau > 0$. Получим уравнение

$$\tau(\operatorname{Im}\lambda_{j\,j} \pm Q) = \arg(\pm iQ - \operatorname{Re}\lambda_{j\,j}) - \arg\mu_{j\,j} + 2\pi m, \qquad (2.14)$$

корнями которого являются числа $\tau_j^m$, указанные в формулировке теоремы.

Отметим, что $\omega \in [-\pi, \pi]$, поэтому требуется выполнение условия

$$\pm\tau\sqrt{|\mu_{j\,j}|^2 - (\operatorname{Re}\lambda_{j\,j})^2} = \pm\tau Q \in [-\pi, \pi],$$

что означает $0 < \tau \leqslant \pi/Q$.

Кроме того, поскольку $h \geqslant -\dfrac{\omega}{\operatorname{tg}\omega}$, то

$$\operatorname{Re}\lambda_{j\,j} \geqslant -\frac{Q}{\operatorname{tg}(Q\,\tau_j^m)}.$$

Таким образом, для каждого $j$ требуется найти такие $\tau_j^m$, которые удовлетворяют условиям (2.3).

Для правильного определения интервалов устойчивости данной винтовой линии (прямой) необходимо отсортировать числа $\tau_j^m$ по возрастанию, т.е. получить систему чисел $\tau_{1j}, \tau_{2j}, \ldots, \tau_{lj}$, и определить, является ли интервал $(0, \tau_{1j})$ интервалом устойчивости или интервалом неустойчивости. Интервалы устойчивости и неустойчивости чередуются, поэтому в зависимости от характера первого интервала составляем множество $T_j$, которое будет иметь вид $T_j = (0, \tau_{1j}) \cup (\tau_{2j}, \tau_{3j}) \cup \ldots$ или $T_j = (\tau_{1j}, \tau_{2j}) \cup (\tau_{3j}, \tau_{4j}) \cup \ldots$. Это множество тех значений запаздывания, при которых точки винтовой линии $(u_{1j}(\tau), u_{2j}(\tau), u_{3j}(\tau))$ находятся внутри конуса устойчивости.

Для ответа на вопрос, является ли интервал $(0, \tau_{1j})$ интервалом устойчивости, определим, лежат ли точки винтовой линии $(u_{1j}(\tau), u_{2j}(\tau), u_{3j}(\tau))$ при $\tau \to 0+$ внутри овала устойчивости. Заметим, что при $u_3 = 0$ овал устойчивости лежит в правой полуплоскости, а его левая вершина лежит в начале координат.

Сначала рассмотрим случай $\operatorname{Re}\lambda_{jj} > 0$. При этом $u_3 > 0$ и винтовая линия лежит выше плоскости $u_3 = 0$. Чтобы точки винтовой линии при $\tau \to 0+$ лежали внутри конуса устойчивости, достаточно, чтобы вектор производной $\dfrac{du_2}{du_1}$ при $\tau \to 0+$ лежал в полуплоскости $u_1 > 0$ или независимо от этого модуль коэффициента наклона образующих винтовой линии в сечении $u_2 = 0$ был больше единицы.

В первом случае для нахождения угла наклона касательной к винтовой линии в плоскости $(u_1, u_2)$ в точке $\tau = 0$ вычислим производную параметрически заданной кривой $u_{2j}(u_{1j})$ в данной точке.

$$\left.\frac{du_{2j}}{du_{1j}}\right|_{\tau=0} = \left.\frac{(u_{2j})'_\tau}{(u_{1j})'_\tau}\right|_{\tau=0} =$$
$$= \left.\frac{|\mu_{jj}|(\sin(\arg\mu_{jj} + \tau\operatorname{Im}\lambda_{jj}) + \tau\cos(\arg\mu_{jj} + \tau\operatorname{Im}\lambda_{jj})\operatorname{Im}\lambda_{jj})}{|\mu_{jj}|(\cos(\arg\mu_{jj} - \tau\operatorname{Im}\lambda_{jj}) + \tau\sin(\arg\mu_{jj} + \tau\operatorname{Im}\lambda_{jj})\operatorname{Im}\lambda_{jj})}\right|_{\tau=0} =$$
$$= \operatorname{tg}\arg\mu_{jj}. \quad (2.15)$$

Если угол наклона касательной к винтовой линии в точке $\tau = 0$ лежит в интервале $(-\pi/2, \pi/2)$, то точки винтовой линии лежат внутри овала устойчивости при $\tau \to 0+$, тогда интервал $(0, \tau_{1j})$ является интервалом устойчивости.

Отметим, что $\arg z \in [-\pi, \pi]$. Если $\arg\mu_{jj} \in [-\pi/2, \pi/2]$, то интервал $(0, \tau_{1j})$ является интервалом устойчивости. Таким образом, получено условие (2.5) а).

Во втором случае найдем коэффициент наклона образующих винтовой линии в сечении $u_2 = 0$. Он равен

$$\left.\frac{du_{3j}}{du_{1j}}\right|_{\tau=0} = \left.\frac{(u_{3j})'_\tau}{(u_{1j})'_\tau}\right|_{\tau=0} = \frac{\operatorname{Re}\lambda_{jj}}{\operatorname{Re}\mu_{jj}}. \quad (2.16)$$

Таким образом, если $\frac{\operatorname{Re}\lambda_{jj}}{|\operatorname{Re}\mu_{jj}|} > 1$, то точки винтовой линии при $\tau \to 0+$ лежат внутри конуса устойчивости, а значит, интервал $(0, \tau_{1j})$ является интервалом устойчивости. Получено условие (2.5) б).

Теперь рассмотрим случай $\operatorname{Re}\lambda_{jj} < 0$. При этом $u_3 < 0$ и винтовая линия лежит ниже плоскости $u_3 = 0$. В этом случае для нахождения точек винтовой линии внутри конуса устойчивости при $\tau \to 0+$ требуется, чтобы вектор производной $\dfrac{du_2}{du_1}$ при $\tau \to 0+$ лежал в полуплоскости $u_1 > 0$ и модуль коэффициента наклона образующих винтовой линии в сечении $u_2 = 0$ был меньше единицы. Таким образом, должны одновременно выполняться условия $\arg\mu_{jj} \in [-\pi/2, \pi/2]$ и $\left|\frac{\operatorname{Re}\lambda_{jj}}{\operatorname{Re}\mu_{jj}}\right| < 1$, т.е. должно выполняться условие (2.6).

Во всех остальных случаях точки винтовой линии лежат вне конуса устойчивости при $\tau \to 0+$, поэтому множество $T_j$ составляется по формуле (2.7).

Если для некоторого $j$ выполнено $\operatorname{Re}\lambda_{j\,j} > |\mu_{j\,j}|$, то винтовая линия лежит внутри конуса устойчивости при всех значениях запаздывания, поэтому для нее $T_j = (0, +\infty)$.

Таким образом, для каждого $j$ составлено множество $T_j$ тех значений запаздывания, при которых точки винтовой линии $(u_{1j}(\tau), u_{2j}(\tau), u_{3j}(\tau))$ находятся внутри конуса устойчивости.

Для устойчивости уравнения (2.1) требуется, чтобы все точки $M_j = (u_{1j}, u_{2j}, u_{3j})$, $1 \leqslant j \leqslant n$ находились внутри конуса устойчивости при данном $\tau$. Значит, уравнение (2.1) асимптотически устойчиво тогда и только тогда, когда $\tau \in \bigcap_j T_j$.

Утверждение теоремы доказано.

■

Заметим, что если равенство $\operatorname{Re}\lambda_{j\,j} > |\mu_{j\,j}|$ верно для всех $j$, то уравнение (2.1) асимптотически устойчиво при всех $\tau \geqslant 0$ (Теорема 1.5).

## 2.2 Программный комплекс «Анализ устойчивости»

### 2.2.1 Функциональное назначение программы, область применения, её ограничения

Программный комплекс «Анализ устойчивости» предназначен для исследования устойчивости нулевого решения дифференциального уравнения (2.1) с одновременно триангулируемыми матрицами $A, B$. Исследование устойчивости нейронных сетей многих стандартных конфигураций сводится именно к этому уравнению.

Алгоритм нахождения значений запаздывания, гарантирующих устойчивость уравнения (2.1) при данных матрицах $A$ и $B$, указан в разделе 2.1 диссертации. В ходе реализации этого алгоритма строятся система точек (их количество равно порядку матриц) и исследуется взаимное расположение конуса устойчивости и данных точек.

На основе теоретических результатов, полученных автором, в программном пакете MATLAB 7.11.0 (R2010b) посредством matlab APY для создания графического интерфейса пользователя (GUI) была разработана программа для анализа устойчивости уравнения (2.1).

В зависимости от собственных чисел матриц $A$ и $B$ в порядке, определенном их совместным приведением к треугольному виду, программа «Анализ устойчивости» позволяет получить множество значений запаздывания, обеспечивающих устойчивость уравнения (2.1) и таким образом определить, является ли уравнение с данным запаздыванием $\tau$ устойчивым. Пользователь имеет возможность сделать вывод об устойчивости уравнения при конкретном значении запаздывания, а при возможности управления запаздыванием оценить интервалы значений запаздывания, обеспечивающих устойчивость данного уравнения. Теоретически алгоритм позволяет определить устойчи-

вость уравнения (2.1) для матриц любого порядка, но программа реализована для систем размерности не выше пятой.

Программа может применяться при разработке и исследовании моделей, описываемых скалярными и матричными дифференциальными уравнениями с запаздыванием, в частности, моделей нейронных сетей. Продукт может применяться для получения вывода об устойчивости конкретных систем, а также для регулирования коэффициентов модели с целью получения устойчивого решения.

### 2.2.2 Использование

Все входные параметры задаются на панели «Входные данные». Размерность легко задаётся при помощи ползунка. Пользователю предлагается два способа задания матриц $A$ и $B$: с помощью базовой матрицы (по умолчанию выбирается именно этот способ) или с помощью задания собственных чисел матриц.

При выборе первого способа (рис. 2.1) пользователю необходимо выбрать значения базовой матрицы $D$, порядок которой автоматически определяется выбранной размерностью. Далее необходимо ввести коэффициенты в разложении матриц $A$ и $B$ по степеням матрицы $D$.

Такой подход позволяет обеспечить совместную триангулируемость матриц и правильное соответствие между собственными числами матриц. Ввести элементы матрицы $D$ и коэффициенты можно двумя способами: вручную или при помощи кнопки «Заполнить», которая генерирует случайные числа в каждой ячейке. Для очистки матрицы и коэффициентов существует кнопка «Очистить».

При выборе второго способа (рис. 2.2) пользователю требуется ввести собственные значения матриц $A$ и $B$ в порядке, определенном одновременным

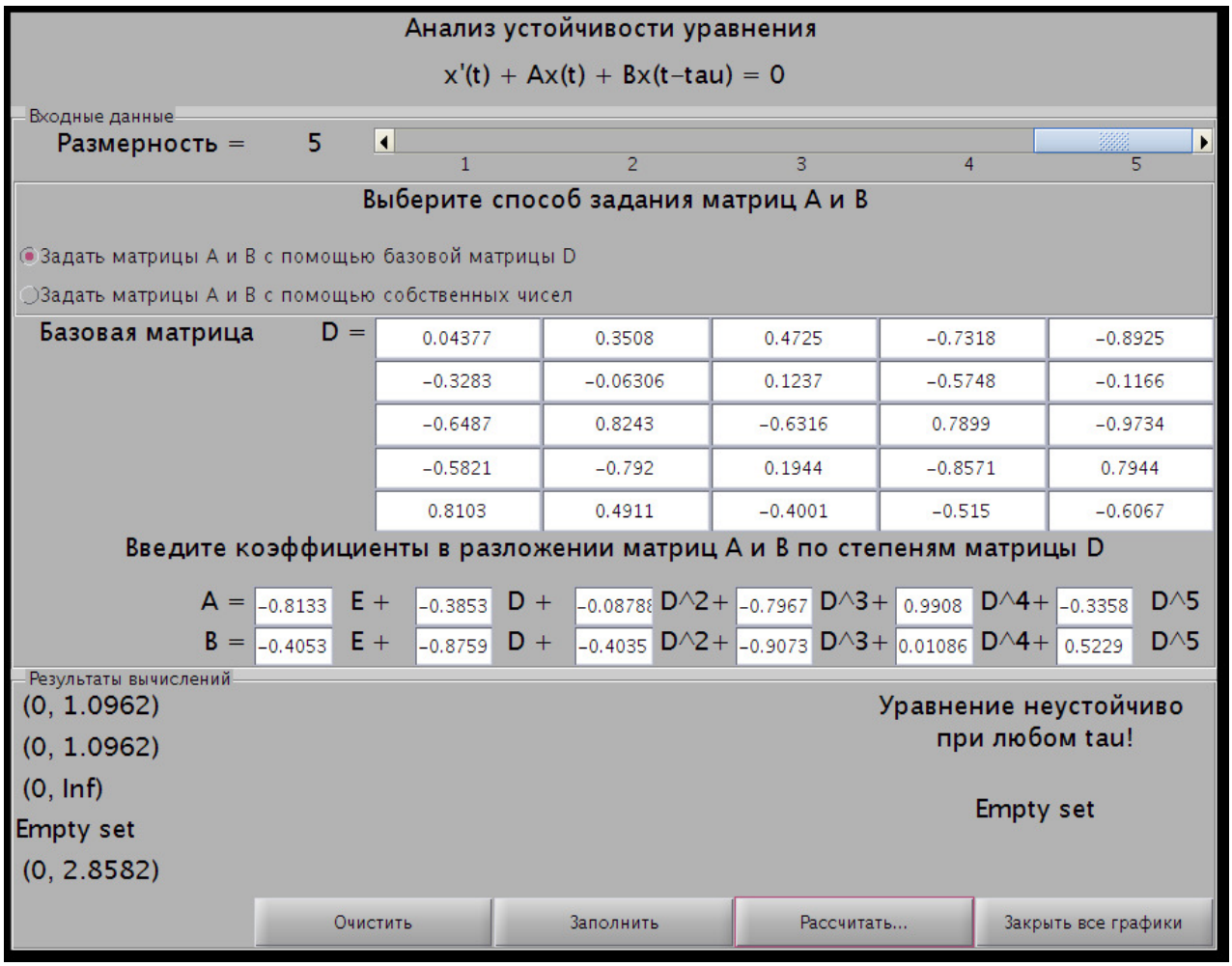



Рис. 2.1. Главное окно программы «Анализ устойчивости» при задании матриц с помощью базовой матрицы

приведением матриц $A, B$ к треугольному виду (количество собственных чисел определяется размерностью). Для этого можно также воспользоваться кнопкой «Заполнить», для ввода новых значений существует кнопка «Очистить».

Все элементы пользовательского интерфейса снабжены всплывающими подсказками.

После ввода данных в обоих случаях для начала расчёта необходимо нажать на кнопку «Рассчитать...». Появляется вспомогательное окно, показанное на рисунке 2.3, представляющее результаты расчётов графически: в трёхмерном пространстве изображаются конус устойчивости и системы то-

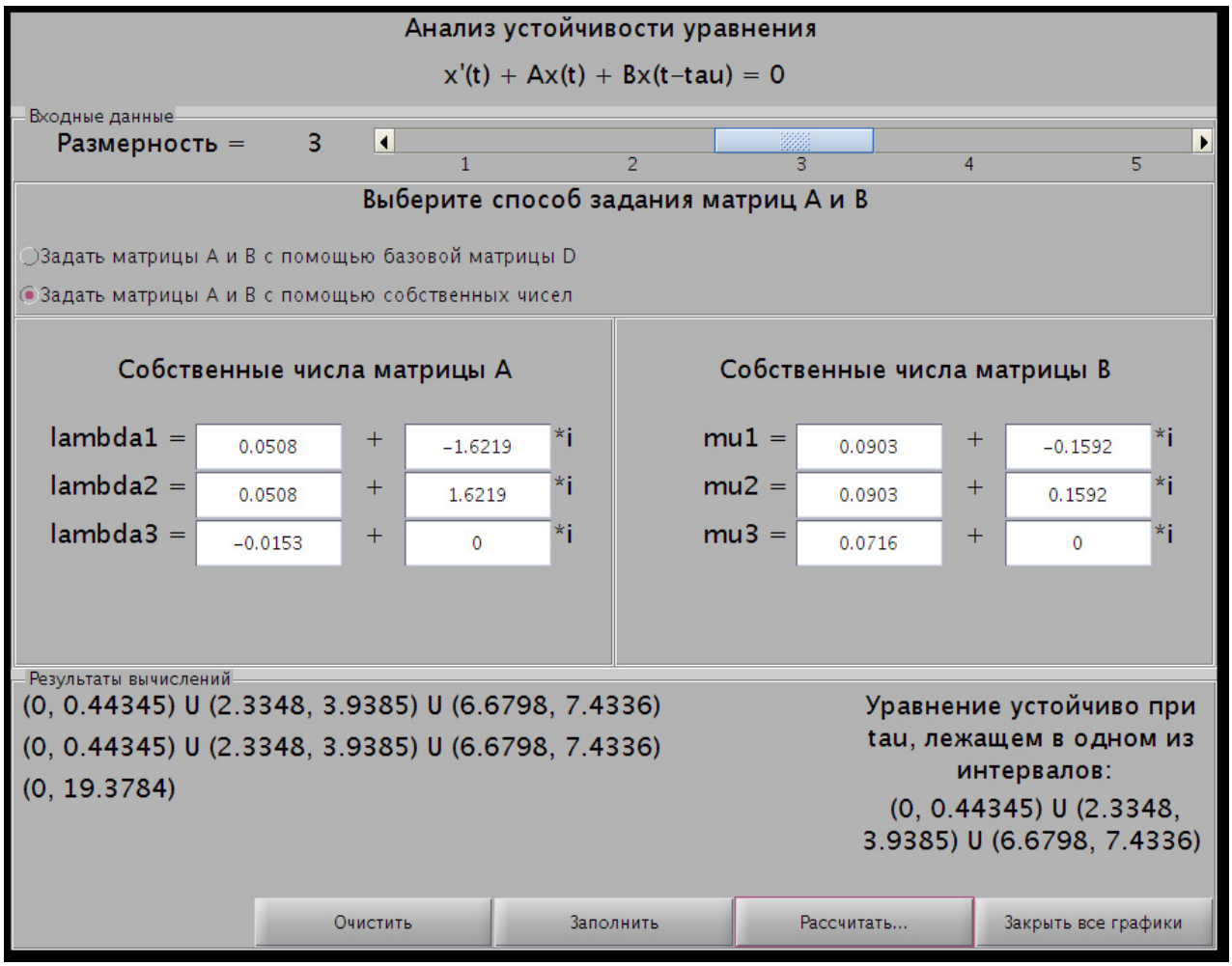



Рис. 2.2. Главное окно программы «Анализ устойчивости» при задании собственных чисел матриц вручную

чек, расположение которых позволяет сделать вывод об устойчивости данного уравнения.

Синим цветом обозначаются точки, находящиеся в конусе устойчивости и соответствующие устойчивости, а красным — находящиеся вне конуса и соответствующие неустойчивости. При этом есть возможность использовать стандартный набор инструментов графического окна пакета MATLAB. На панели «Результаты вычислений» отображается отчёт о результатах вычислений и выводится сообщение об устойчивости уравнения (2.1). Причём, если существует хотя бы один промежуток значений $\tau$, обеспечивающих устойчивость уравнения, то будут показаны промежутки устойчивости (рис. 2.1),

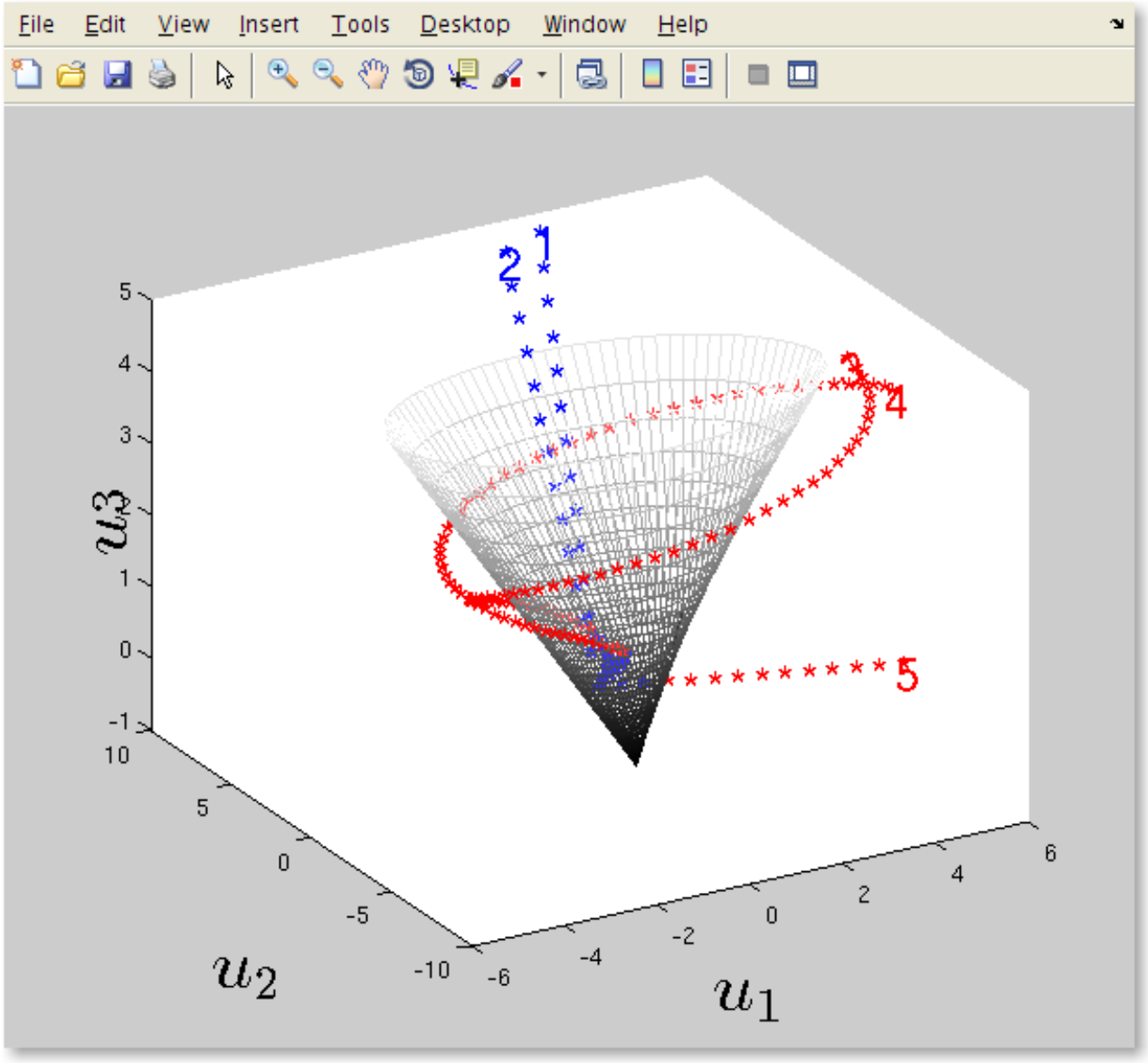



Рис. 2.3. Графический вывод результатов вычисления

если нет — будет выдано сообщение о неустойчивости уравнения при всех значениях запаздывания (рис. 2.2).

После проведения расчётов пользователь может изменить входные параметры и ещё раз выполнить расчёт устойчивости. Появится ещё одно окно с графиком, а отчёт в главном окне будет обновлён. Данную процедуру можно проделывать сколько угодно раз. Для удобства пользователя была создана кнопка «Закрыть все графики», позволяющая закрыть все дополнительные окна.

При вводе некорректных данных программа выдаёт различные сообщения об ошибках. Например, следующее сообщение может быть получено при вводе некорректных значений в матрицу (рис. 2.4).

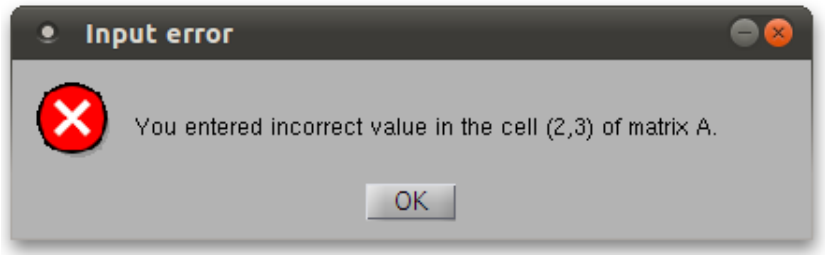



Рис. 2.4. Сообщение об ошибке: введены некорректные данные

### 2.2.3 Используемые технические средства

Программа «Анализ устойчивости» разрабатывалась в высокоуровневой среде для математических вычислений MATLAB 7.11.0 (R2010b) с использованием matlab APY для создания графического интерфейса пользователя (GUI). Пользовательская версия программного продукта поставляется в скомпилированном с помощью компилятора mcc, входящего в дистрибутив. Для запуска программы нужен установленный Matlab соответствующей версии либо библиотеки Matlab, которые можно получить на официальном сайте (http://www.mathworks.com/).

### 2.2.4 Специальные условия и требования организационного, технического и технологического характера

Каких-либо специальных условий применения и требований организационного характера не требуется.

### 2.2.5 Условия передачи документации или её продажи

Программа распространяется на безвозмездной основе.

# Глава 3

# Численный и теоретический анализ устойчивости нейронных сетей с неограниченным количеством нейронов

## 3.1 Постановка задачи и алгоритм диагностирования устойчивости кольцевой сети с неограниченным количеством нейронов

Много работ посвящено проблеме устойчивости нейронных сетей, состоящих из двух нейронов (например, [5, 27]). Проблема устойчивости нейронной сети круговой архитектуры с запаздывающим взаимодействием при количестве нейронов больше двух исследуется, например, в работах [13–15, 28–30] Изучаемая нами в этой главе модель наиболее близка к модели работ [14, 15]. В отличие от работ [14, 15], мы не вводим запаздывание в реакцию нейрона на собственный сигнал (selfconnection), зато рассматриваем несимметричное взаимодействие нейрона с правым и левым соседом. Особенность нашего подхода в этом разделе и вообще в этой главе — рассмотрение конфигураций с неограниченным количеством нейронов. Методы нашего исследования основаны на конусах устойчивости, введенных в главе 1 диссертации (см. также [21]), но требуют изменения в связи с спецификой задачи о неограниченности количестве нейронов в кольце. Аналогичные методы применены в [31] для изучения устойчивости дискретных моделей круговой системы нейронов (см. также [30, 32]).

Переход от нелинейных моделей нейронных сетей с одним запаздыванием к линейным уравнениям посредством линеаризации описан в разделе

1.1 диссертации. Аналогично этому следующее дифференциальное уравнение описывает кольцевую нейронную сеть (рис. 3.1) с запаздыванием $\tau_1$ во взаимодействии нейрона с левым соседним нейроном и $\tau$ с правым:

$$\dot{x}_j(t) + x_j(t) + a\,x_{j-1}(t-\tau_1) + b\,x_{j+1}(t-\tau) = 0 \qquad (j \bmod n). \tag{3.1}$$

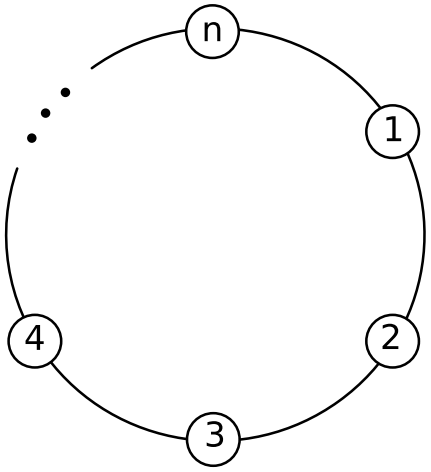



Рис. 3.1. Круговая система нейронов

Здесь $x_j$ $(1 \leqslant j \leqslant n)$ сигнал $j$-го нейрона, действительные коэффициенты $a$ и $b$ характеризуют интенсивности взаимодействия нейрона с правым и левым соседними нейронами соответственно. Общее аналитическое исследование устойчивости системы (3.1) вряд ли возможно, поскольку даже скалярное уравнение $\dot{x}(t) + a\,x(t-\tau_1) + b\,x(t-\tau) = 0$ имеет, как выяснилось (см. [6, 7]), очень сложную структуру области устойчивости в пространстве параметров. Поэтому основными моделями в нашем исследовании будут два упрощенных варианта систем (3.1), в первом из которых $\tau_1 = 0$, во втором $\tau_1 = \tau$:

$$\dot{x}_j(t) + x_j(t) + a\,x_{j-1}(t) + b\,x_{j+1}(t-\tau) = 0 \qquad (j \bmod n), \tag{3.2}$$

$$\dot{x}_j(t) + x_j(t) + a\,x_{j-1}(t-\tau) + b\,x_{j+1}(t-\tau) = 0 \qquad (j \bmod n). \tag{3.3}$$

Уравнение (3.2) соответствует малым запаздываниям взаимодействия нейронов с правыми соседними нейронами, а (3.3) — близким запаздываниям взаимодействия нейронов с правыми и левыми соседями. В обоих уравнениях интенсивность взаимодействия нейронов с правыми и левыми соседями не обязательно одинакова. Последнее свойство отличает нашу модель от модели работ [14, 15]. Еще одно отличие в том, что у нас отсутствует запаздывание в реакции нейрона на собственный сигнал (selfconnection).

Системы (3.2), (3.3) принадлежат классу систем вида

$$\dot{x}(t) + Ax(t) + Bx(t-\tau) = 0, \tag{3.4}$$

где $n \times n$ матрицы $A, B$ одновременно триангулируемы. В главе 1 и статье автора диссертации [21] указан метод диагностики устойчивости таких систем с помощью конусов устойчивости.

Введем специальное обозначение для $n \times n$ матрицы оператора сдвига строк:

$$P = \begin{pmatrix} 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{pmatrix}. \tag{3.5}$$

В матричном виде уравнения (3.2), (3.3) соответственно будут иметь вид

$$\dot{x}(t) + (E + aP)\, x(t) + bP^{n-1}\, x(t-\tau) = 0, \tag{3.6}$$

$$\dot{x}(t) + E\, x(t) + (aP + bP^{n-1})\, x(t-\tau) = 0, \tag{3.7}$$

где $E$ есть единичная $n \times n$ матрица, $x(t) = (x_1(t), \dots, x_n(t))^T$.

Пусть матрицы $A, B$ в уравнении (3.4) приводятся одной трансформирующей матрицей к треугольным формам $A_T, B_T$ с диагональными элементами $\lambda_j,\ \mu_j$ $(1 \leqslant j \leqslant n)$ соответственно. Построим $n$ точек $M_j = (u_{1j}, u_{2j}, u_{3j}) \in$

$\mathbb{R}^3$, так что

$$u_{1j} + iu_{2j} = \tau\mu_j \exp(i\tau \operatorname{Im} \lambda_j), \qquad u_{3j} = \tau \operatorname{Re} \lambda_j, \qquad 1 \leqslant j \leqslant n. \tag{3.8}$$

В силу результатов главы 1 диссертации (см. также [21]), система (3.4) асимптотически устойчива тогда и только тогда, когда все точки $M_j$ $(1 \leqslant j \leqslant n)$ лежат внутри конуса устойчивости. Если хотя бы одна из точек $M_j$ лежит вне конуса, то (3.4) неустойчива.

В разделах 2.1 2.2 и 2.2 диссертации мы описали алгоритм и программный комплекс «Анализ устойчивости» (см. также [33]), предназначенный для диагностирования устойчивости системы (3.4). Программа по согласованному спектру $\lambda_j$, $\mu_j$ $1 \leqslant j \leqslant$ матриц $A, B$ возвращает объединение интервалов

$$T = \cup_{k=1}^{N} (\tau_k,\, \tau_{k+1}), \tag{3.9}$$

такое что если $\tau \in T$, то система (3.4) асимптотически устойчива, а если $\tau \notin [\tau_k,\, \tau_{k+1}]$ при любом $k$ $(1 \leqslant k \leqslant N)$, то (3.4) неустойчива.

Для проверки устойчивости систем (3.2), (3.3) при неограниченном порядке $n$ требуется модификация алгоритма, первоначально предназначенного для общей системы (3.4). В настоящем разделе мы указываем такой алгоритм.

Собственные числа матрицы $P$ суть $\lambda_j = \exp(i\frac{2\pi j}{n})$, $1 \leqslant j \leqslant n$. Поэтому при одновременном приведении четырех циркулянтных матриц $(I + a\,P)$, $b\,P^{n-1}$, $I$, $a\,P + b\,P^{n-1}$ к диагональному виду их диагональные элементы равны соответственно

$$\lambda_j^{'} = 1 + a\, \exp(i\frac{2\pi j}{n}), \quad \mu_j^{'} = b\, \exp(-i\frac{2\pi j}{n}), \quad 1 \leqslant j \leqslant n, \tag{3.10}$$

$$\lambda_j^{''} = 1, \quad \mu_j^{''} = a\, \exp(i\frac{2\pi j}{n}) + b\, \exp(-i\frac{2\pi j}{n}), \quad 1 \leqslant j \leqslant n. \tag{3.11}$$

Для изучения устойчивости (3.5), (3.6) строим систему точек $M_j^{'} = (u_{1j}^{'}, u_{2j}^{'}, u_{3}^{'}$ $\mathbb{R}^3$, определенных равенствами (см. (3.8), (3.10))

$$u_{1j}^{'}+iu_{2j}^{'} = \tau b\, \exp(i(-\frac{2\pi j}{n}+a\tau \sin\frac{2\pi j}{n})), \quad u_{3j}^{'} = \tau(1+a\cos\frac{2\pi j}{n}), \quad 1 \leqslant j \leqslant n. \tag{3.12}$$

Аналогично, для системы (3.5), (3.7) точки $M_j^{''} = (u_{1j}^{''}, u_{2j}^{''}, u_{3j}^{''}) \in \mathbb{R}^3$, определены равенствами (см. (3.8), (3.11))

$$u_{1j}^{''} + iu_{2j}^{''} = \tau a\,(\exp(i\frac{2\pi j}{n}) + b\exp(-i\frac{2\pi j}{n})), \quad u_{3j}^{''} = \tau, \quad 1 \leqslant j \leqslant n. \tag{3.13}$$

Для анализа систем (3.5), (3.6) с большим $n$ вместо дискретной системы точек $M_j$ естественно ввести непрерывную замкнутую кривую $M^{'}(t) = (u_1^{'}(t), u_2^{'}(t), u_3^{'}(t)$ положив $t = \frac{2\pi j}{n}$ в (3.12):

$$u_1^{'}(t) + iu_2^{'}(t) = \tau b\, \exp(i(-t + a\,\tau \sin t)), \quad u_3^{'}(t) = \tau(1 + a\cos t), \quad 0 \leqslant t \leqslant 2\pi. \tag{3.14}$$

Аналогично, для (3.5), (3.7) строим кривую $M^{''}(t) = (u_1^{''}(t), u_2^{''}(t), u_3^{''}(t))$:

$$u_1^{''}(t) + iu_2^{''}(t) = \tau(a\, \exp(it) + b\, \exp(-it)), \quad u_3^{''}(t) = \tau, \quad 0 \leqslant t \leqslant 2\pi. \tag{3.15}$$

При $n \to \infty$ точки $M_j$, определенные by (3.12), образуют плотное на кривой (3.14) множество. Поэтому вопрос об устойчивости системы (3.5), (3.6) сводится к геометрической проблеме. Если все точки кривой (3.14) лежат внутри конуса устойчивости (1.21), (**??**) (см. определение 1.2), то система (3.5), (3.6) устойчива при любом $n$, а если хотя бы одна точка кривой (3.14) лежат вне конуса устойчивости, то система (3.5), (3.6) неустойчива при всех достаточно больших значениях $n$ (Рис.3.2).

В свою очередь, эта геометрическая задача может быть решена численными методами. Для системы (3.5), (3.6) при различных значениях $t \in [0, 2\pi]$ мы вычисляем $\arg(u_1^{'}(t) + iu_2^{'}(t))$ согласно (3.14) и находим на конусе устойчивости на высоте $h = u_3^{'}(t)$ точку или две точки с тем же аргументом. Срав-

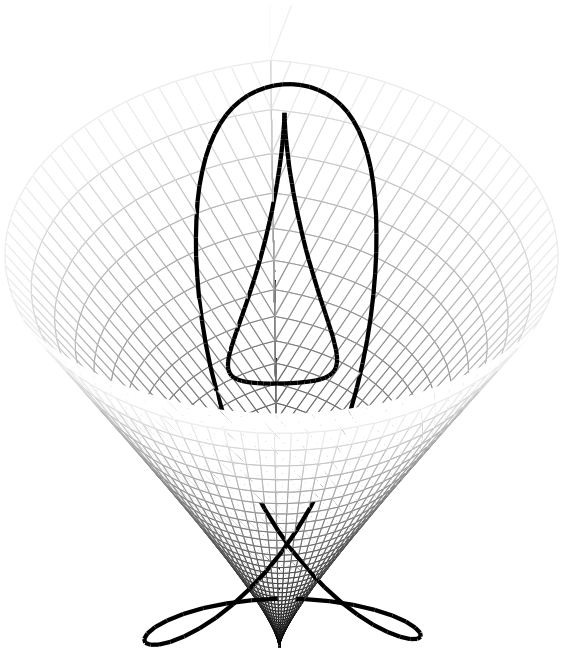


Рис. 3.2. Конус устойчивости и две кривые (3.14). Одна кривая для $\tau = 1.5$, $a = -1.4$, $b = 0.7$, находится частично вне конуса устойчивости, следовательно, система (3.5), (3.6) неустойчива при достаточно больших $n$. Вторая кривая для $\tau = 2$, $a = 0.5$, $b = -0.2$, находится полностью внутри конуса устойчивости, следовательно, система (3.5), (3.6) устойчива при любом $n$.

нение модулей точек на кривой и конусе дает ответ на вопрос, устойчива ли система (3.5), (3.6). Аналогично изучается система (3.5), (3.7).

## 3.2 Программный комплекс «Устойчивость нейронных сетей»

### 3.2.1 Функциональное назначение программы, область применения, её ограничения

Программа «Устойчивость нейронных сетей» по коэффициентам уравнений (3.2), (3.3) и величине запаздывания выдаёт сообщение об устойчивости системы. Программа выполнена в пакете MATLAB 7.11.0 (R2010b) по-

средством matlab APY для создания графического интерфейса пользователя (GUI).

Программа может применяться при разработке и исследовании моделей нейронных сетей, описываемых скалярными и матричными дифференциальными уравнениями с запаздыванием. Продукт может применяться для получения вывода об устойчивости конкретных систем, а также для регулирования коэффициентов модели с целью получения устойчивого решения.

### 3.2.2 Использование

Все входные параметры задаются на одноимённой панели, показанной на рисунке 3.3.

Прежде всего пользователю предлагается выбрать конфигурацию нейронной сети. В программе доступны два варианта кругового соединения нейронов: с малым запаздыванием взаимодействия с правыми соседями (уравнение (3.2)) и с одинаковым запаздыванием взаимодействия с правыми и левыми соседями (уравнение (3.3)). Выбирая первую или вторую радиокнопку, пользователь определяет, для какой именно модели будет проводиться анализ устойчивости. На рисунке 3.3 выбрана первая конфигурация, а на рисунке 4.2 — вторая.

Ниже, на панели ввода представлено название, описание и рисунок выбранной конфигурации. Пользователю необходимо ввести коэффициенты уравнения, соответствующие выбранной конфигурации. Ввести данные коэффициенты можно двумя способами: вручную или при помощи кнопки «Заполнить», которая генерирует случайные числа в каждой ячейке. Для очистки значений коэффициентов используется кнопка «Очистить».

Все элементы пользовательского интерфейса снабжены всплывающими подсказками.

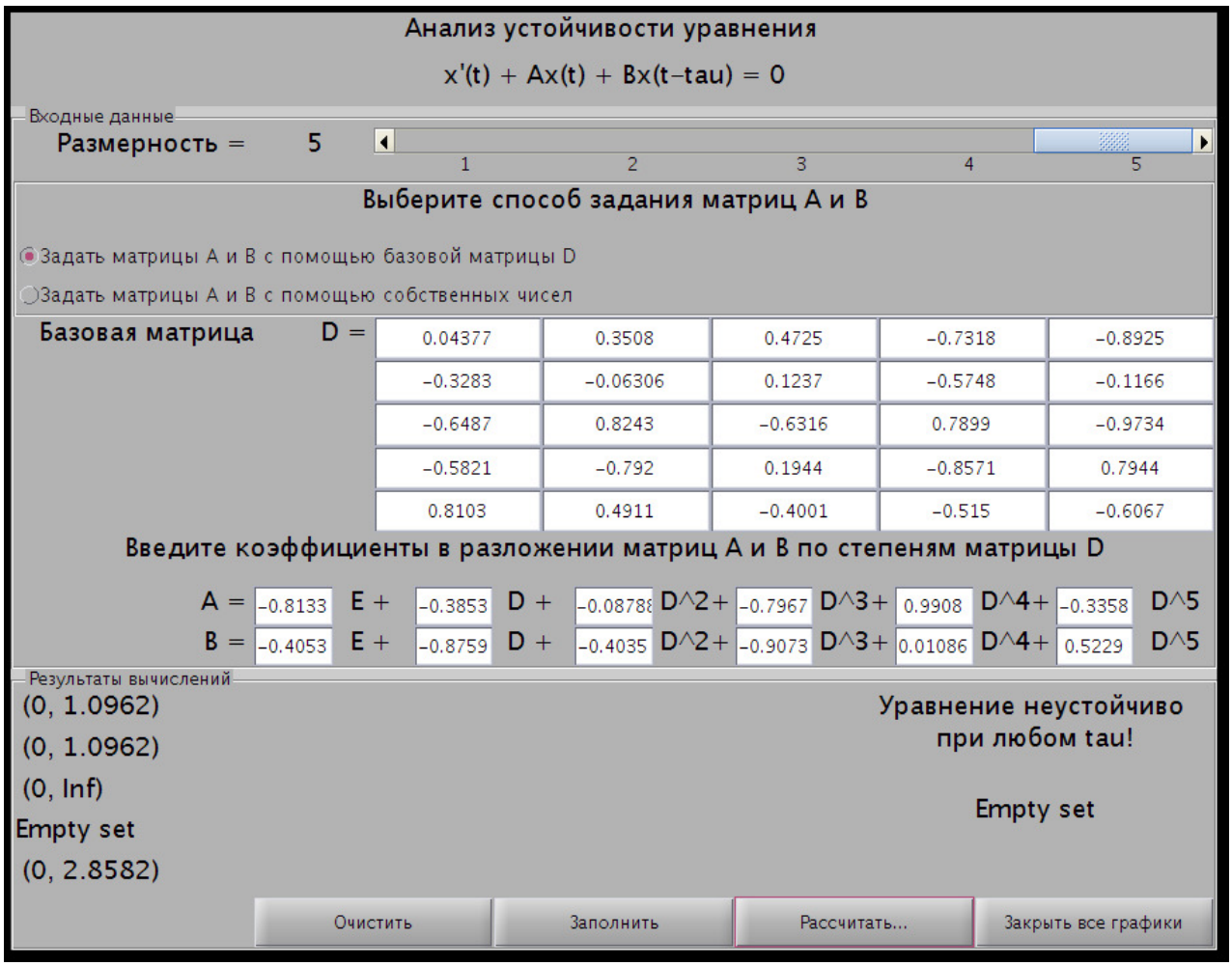



Рис. 3.3. Главное окно программы «Устойчивость нейронных сетей» при выборе первой конфигурации

После ввода данных в обоих случаях для начала расчёта необходимо нажать на кнопку «Рассчитать. . . ». Появляется вспомогательное окно (рис. 3.5), представляющее результаты расчётов графически: в трёхмерном пространстве изображаются конус устойчивости и системы точек, расположение которых позволяет сделать вывод об устойчивости данного уравнения. Чёрным цветом обозначаются точки, находящиеся внутри конуса устойчивости и соответствующие устойчивости, а красным — находящиеся вне конуса и соответствующие неустойчивости. При этом есть возможность использовать стандартный набор инструментов графического окна пакета MATLAB. На панели «Результаты вычислений» отображается отчёт о результатах вычислений и

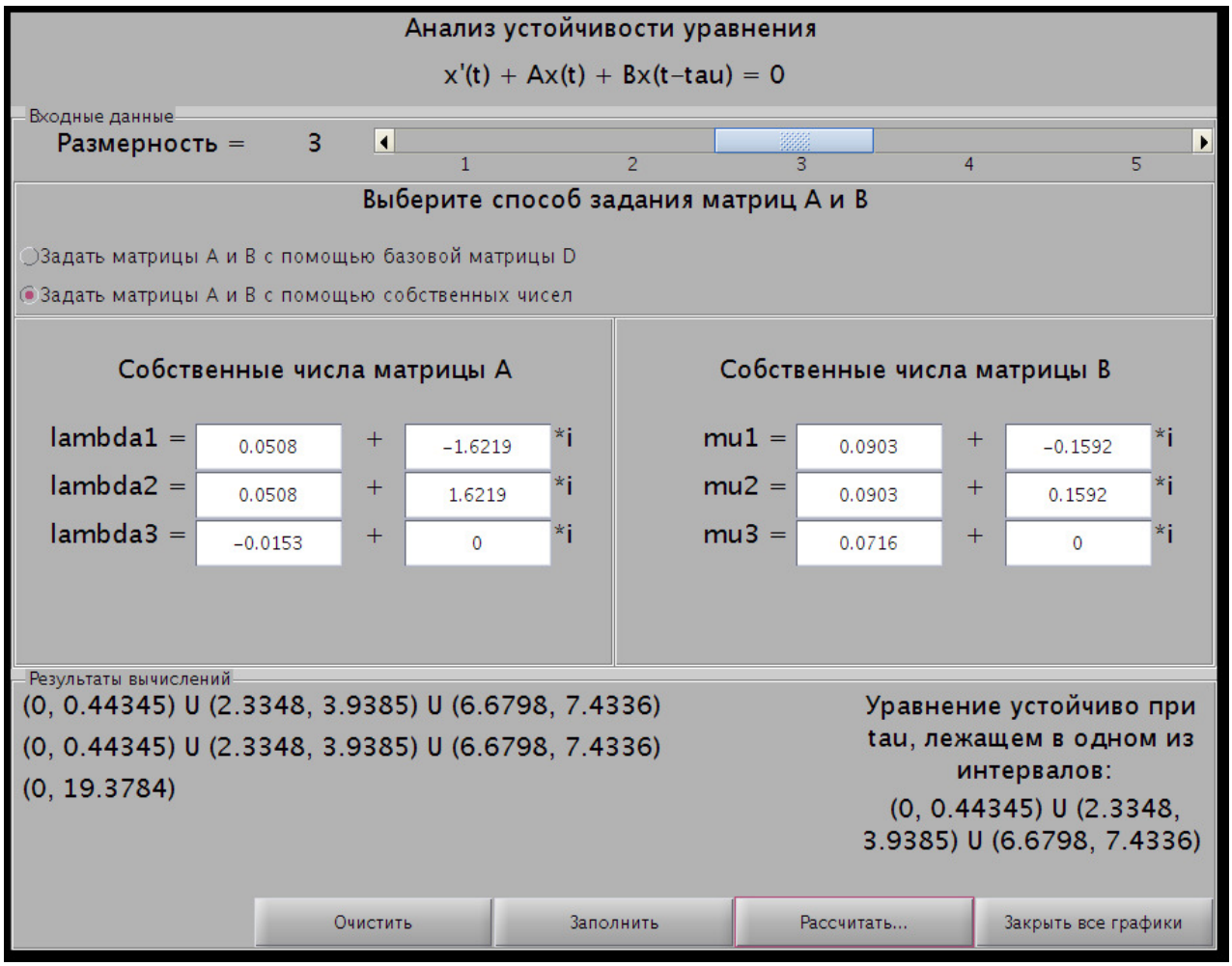



Рис. 3.4. Главное окно программы «Устойчивость нейронных сетей» при выборе второй конфигурации

выводится сообщение об устойчивости уравнения. Причём, если существует хотя бы один промежуток значений $\tau$, обеспечивающих устойчивость уравнения, то будут показаны промежутки устойчивости (см. рис. 3.3), если нет — будет выдано сообщение о неустойчивости уравнения при всех значениях запаздывания (см. рис. 4.2).

После проведения расчётов пользователь может изменить входные параметры и ещё раз выполнить расчёт устойчивости. Появится ещё одно окно с графиком, а отчёт в главном окне будет обновлён. Данную процедуру можно проделывать сколько угодно раз. Для удобства пользователя была создана

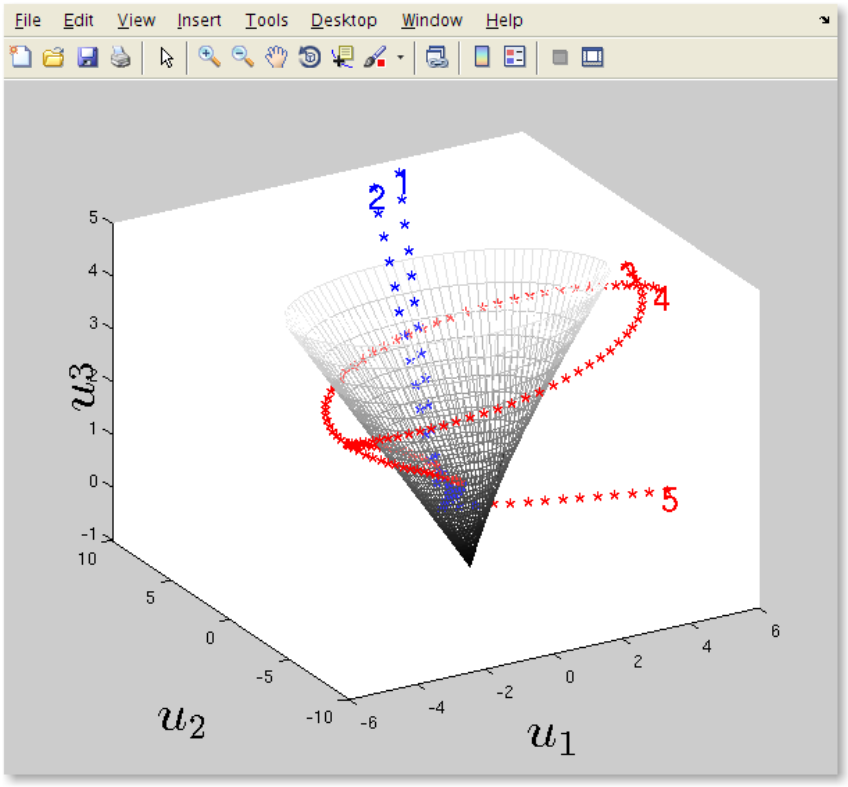



Рис. 3.5. Графический вывод результатов вычисления

кнопка «Закрыть все графики», позволяющая закрыть все дополнительные окна.

При вводе некорректных данных программа выдаёт различные сообщения об ошибках. Например, следующее сообщение может быть получено при вводе некорректных значений в поле ввода параметра $a$ (см. рис. 3.6).

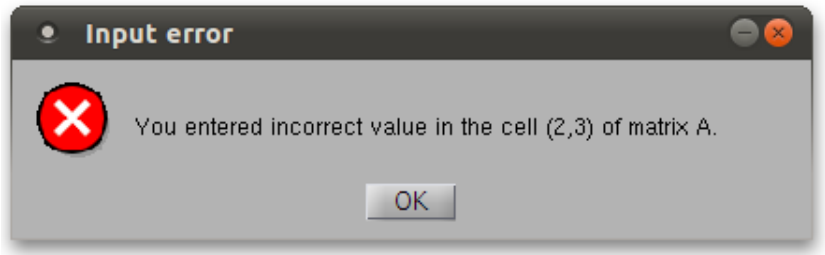



Рис. 3.6. Сообщение об ошибке: введены некорректные данные

### 3.2.3 Используемые технические средства

Программа «Устойчивость нейронных сетей» разрабатывалась в высокоуровневой среде для математических вычислений MATLAB 7.11.0 (R2010b) с использованием matlab APY для создания графического интерфейса пользователя (GUI). Пользовательская версия программного продукта поставляется в скомпилированном с помощью компилятора mcc, входящего в дистрибутив. Для запуска программы нужен установленный Matlab соответствующей версии либо библиотеки Matlab, которые можно получить на официальном сайте (http://www.mathworks.com/).

### 3.2.4 Специальные условия и требования организационного, технического и технологического характера

Каких-либо специальных условий применения и требований организационного характера не требуется.

### 3.2.5 Условия передачи документации или её продажи

Программа распространяется на безвозмездной основе.

## 3.3 Результаты исследования устойчивости кольцевой сети нейронов с неограниченным количеством нейронов

В работе Mori и др. [9] доказано, что устойчивость уравнения (3.4), независимая от запаздывания, гарантирована при условии

$$\min_{1\leqslant j\leqslant n}\{\alpha_{jj}-\sum_{k=1,k\neq j}^{n}|\alpha_{jk}|\}>\max_{1\leqslant j\leqslant n}\sum_{k=1}^{n}|\beta_{jk}|, \tag{3.16}$$

где $\alpha_{jk}, \beta_{jk}$ суть элементы матриц $A, B$ соответственно. Отсюда вытекает следующее Предложение.

**Предложение 3.1.** *Если* $|a| + |b| < 1$, *то системы* (3.5), (3.6) *и* (3.5), (3.7) *асимптотически устойчивы при любом* $n$ *и любом* $\tau \geqslant 0$.

Утверждение 3.1 также несложно вывести из Теоремы 1.5 раздела 1.4 диссертации.

**Теорема 3.1.** *Если* $|a + b| > 1$, *то системы* (3.5), (3.6) *и* (3.5), (3.7) *неустойчивы при любом* $\tau > 0$, *если* $n$ *достаточно велико.*

Доказательство теоремы 3.1 будет дано в разделе 3.5. Предложение 3.1 и теорема 3.1 не дает информации о поведении систем (3.5), (3.6) и (3.5), (3.7) для случая, когда выполнены одновременно неравенства $|a+b| < 1$ и $|a|+|b| > 1$. Для этого случая мы применили программный комплекс «Устойчивость нейронных сетей».

Здесь мы представляем результаты, полученные применения программного комплекса, Предложения 3.1 и Теоремы 3.1 .

На рисунках 3.7, 3.8 показаны области $D_\tau$ в плоскости параметров $(a, b)$ для некоторых значений $\tau$. Область $D_\tau$ является расширением области, заданной неравенством $|a| + |b| < 1$, на северо-запад и юго-восток до границ, зависящих от $\tau$. Если точка $(a, b)$ лежит внутри $D_\tau$ на Рис. 3.7, то система (3.5), (3.6) асимптотически устойчива. Если $(a, b)$ лежит вне $D_\tau$ на рисунке 3.7, то система (3.5), (3.6) неустойчива при достаточно больших $n$. Аналогичные утверждения верны для рисунка 3.8 и системы (3.5), (3.7). Область $D_\tau$ центрально-симметрична: если $(a, b) \in D_\tau$, то $(-a, -b) \in D_\tau$. При одинаковых $\tau$ область устойчивости для системы (3.5), (3.6) шире области для (3.5), (3.7).

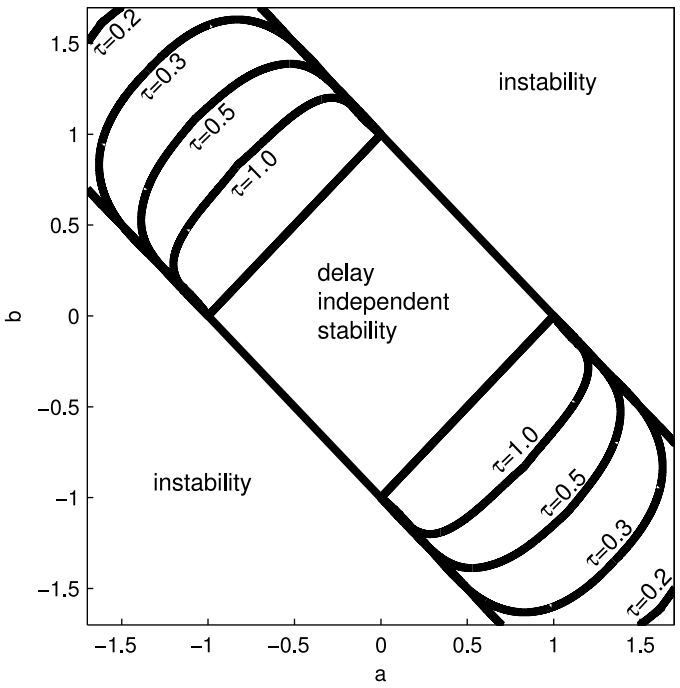



Рис. 3.7. Области устойчивости для системы (3.5), (3.6)

Для обеих систем важна прямая $a = -b$ в плоскости $(a, b)$, в окрестности которой сконцентрированы точки устойчивости систем. Поэтому естественно рассмотреть следующие две системы уравнений:

$$\dot{x}_j(t) + x_j(t) + a\,(x_{j-1}(t) - x_{j+1}(t-\tau)) = 0 \qquad (j \bmod n), \tag{3.17}$$

$$\dot{x}_j(t) + x_j(t) + a\,(x_{j-1}(t-\tau) - x_{j+1}(t-\tau)) = 0 \qquad (j \bmod n). \tag{3.18}$$

**Определение 3.1.** *Границей устойчивости системы* (3.17) *для больших* $n$ *назовем такое число* $a_1(\tau) \in \mathbb{R}$, *что если* $|a| < a_1(\tau)$, *то* (3.17) *устойчива при любом* $n$, *а если* $|a| > a_1(\tau)$, *то* (3.17) *неустойчива при всех достаточно больших* $n$. *Аналогично определим* $a_2(\tau)$ *как границу устойчивости* (3.18) *для больших* $n$.

В таблице 3.1 представлены результаты анализа устойчивости систем (3.17), (3.18) с помощью программного комплекса «Устойчивость нейронных сетей».

Очевидно, $\lim_{\tau\to\infty} a_1(\tau) = \lim_{\tau\to\infty} a_2(\tau) = 1/2$.

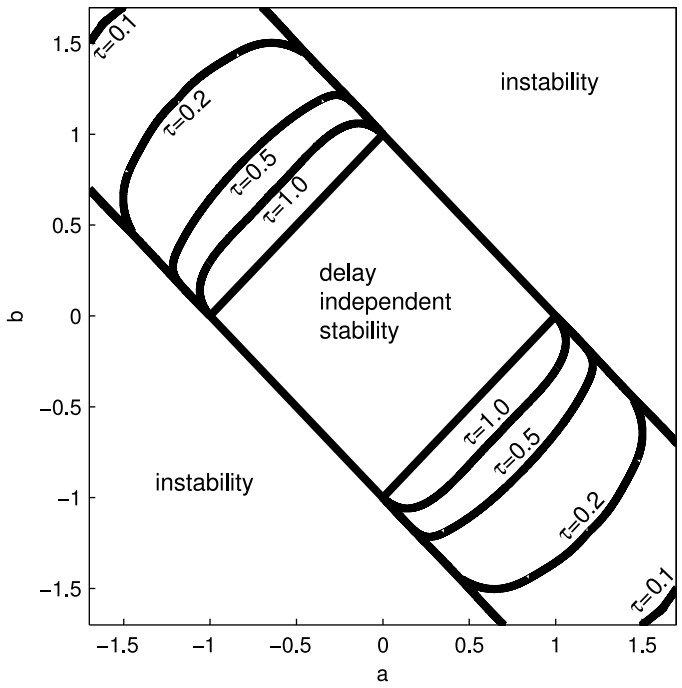



Рис. 3.8. Области устойчивости для системы (3.5), (3.7)

Не столь очевидно поведение систем (3.17), (3.18) при $\tau \to 0$, которое рассматривается в следующей теореме.

**Теорема 3.2.**

$$\lim_{\tau\to 0} a_1(\tau)\sqrt{2\tau} = \lim_{\tau\to 0} a_2(\tau)2\sqrt{\tau} = 1. \tag{3.19}$$

Доказательство теоремы 3.2 дано в разделе 3.5. Таблица 3.1 подтверждает оценки теоремы 3.2. Действительно, согласно таблице 3.1, при $\tau = 0.01$ имеем $a_1(\tau)\sqrt{2\tau} \simeq 1.0017$, $a_2(\tau)2\sqrt{\tau} \simeq 1.0033$.

Таблица 3.1. Границы областей устойчивости для систем (3.17), (3.18)

| $\tau$ | 0.01 | 0.05 | 0.1 | 0.2 | 0.5 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|---|---|---|---|
| $a_1(\tau)$ | 7.0830 | 3.1901 | 2.2742 | 1.6337 | 1.0829 | 0.8229 | 0.6597 | 0.5988 | 0.5678 |
| $a_2(\tau)$ | 5.0166 | 2.2733 | 1.6337 | 1.1921 | 0.8226 | 0.6595 | 0.5678 | 0.5380 | 0.5244 |

## 3.4 Разрыв в кольце: нейронная сеть линейной конфигурации

Выясним, что происходит с устойчивостью, когда разорвана связь между, скажем, первым и последним нейроном в кольцевой сети нейронов (Рисунок 3.9).

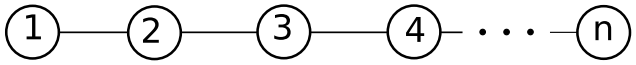



Рис. 3.9. Линейная система нейронов.

В этом случае система (3.5), (3.6) переходит в некоторое уравнение вида (3.4) с матрицами $A, B$, которые, вообще говоря, не приводятся одновременно к треугольному виду. Наш метод конусов устойчивости непригоден для анализа устойчивости таких систем. Поэтому мы вынуждены ограничиться анализом системы (3.5), (3.7) с разорванной связью. Итак, рассмотрим следующий аналог системы (3.5), (3.7):

$$\dot{x}_1(t) + x_1(t) + b\,x_2(t-\tau) = 0,$$

$$\dot{x}_j(t) + x_j(t) + a\,x_{j-1}(t-\tau) + b\,x_{j+1}(t-\tau) = 0, \qquad 2 \leqslant j \leqslant (n-1), \quad (3.20)$$

$$\dot{x}_n(t) + x_n(t) + a\,x_{n-1}(t-\tau) = 0, \qquad n > 2.$$

Матричная форма уравнения (3.20) такова:

$$\dot{x}(t) + I\,x(t) + D\,x(t-\tau) = 0, \quad (3.21)$$

где $n \times n$ матрица $D$ имеет вид

$$D = \begin{pmatrix} 0 & b & 0 & \dots & 0 & 0 \\ a & 0 & b & \dots & 0 & 0 \\ 0 & a & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & b \\ 0 & 0 & 0 & \dots & a & 0 \end{pmatrix}. \tag{3.22}$$

Для формулировки следующей теоремы определим функцию $F(\tau)$ от запаздывания $\tau \in (0, \infty)$:

$$F(\tau) = \frac{1}{4 \sin^2 \omega(\tau)}, \tag{3.23}$$

где $\omega(\tau)$ есть наименьший положительный корень уравнения

$$\tau = \omega \operatorname{tg} \omega. \tag{3.24}$$

**Теорема 3.3.** *1. Если* $|ab| < \frac{1}{4}$, *то система* (3.21), (3.22) *асимптотически устойчива при любом* $n$ *и любом* $\tau \geqslant 0$.

*2. Если* $ab > \frac{1}{4}$, *то система* (3.21), (3.22) *неустойчива при любом* $\tau \geqslant 0$, *если* $n$ *достаточно велико.*

*3. Если* $ab < 0$ *и* $|ab| < F(\tau)$, *то система* (3.21), (3.22) *асимптотически устойчива при любом* $n$.

*4. Если* $ab < 0$ *и* $|ab| > F(\tau)$, *то система* (3.21), (3.22) *неустойчива, если* $n$ *достаточно велико.*

Доказательство Теоремы 3.3 дано в разделе 3.5. Область устойчивости, независимой от запаздывания, очерченная пунктом 1 Теоремы 3.3, шире области $|a| + |b| < 1$, гарантированной достаточным условием (3.16). Результаты Теоремы 3.3 отражены на Рисунке 3.10. Сравнивая Рисунки 3.8 и 3.10, мы видим, что при разрыве кольца нейронов область устойчивости в пространстве параметров увеличивается. Еще одно наблюдение: в системе (3.21), (3.22)

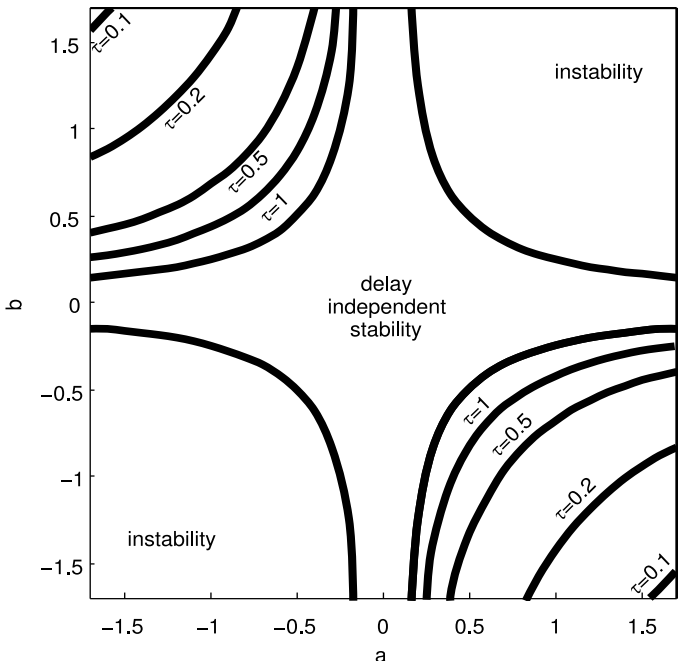



Рис. 3.10. Области устойчивости системы (3.21), (3.22).

при $a = 0$ (или $b = 0$) Теорема 3.3 гарантирует устойчивость независимо от запаздывания. В интерпретации это означает, что при разрыве кольца нейронов блокировка взаимодействия нейронов с правым (или левым) соседом гарантирует устойчивость при любом запаздывании, что отнюдь неверно для неповрежденного кольца нейронов (см. Теорему 3.1 и Рисунки 2,3).

Рассмотрим вариант $a = -b$ в (3.21), (3.22). Положим $H(\tau) = \sqrt{F(\tau)}$ (см. (3.23), (3.24)). Очевидно, $\lim_{\tau\to\infty} H(\tau) = 1/2$ и $\lim_{\tau\to 0} H(\tau)2\sqrt{\tau} = 1$. В системе (3.21), (3.22) с $a = -b$ функция $H(\tau)$ играет роль границы устойчивости, аналогичную функции $a_2(\tau)$ для системы (3.18) (см. Определение 3.1). Сравнение поведения $H(\tau)$ с $a_2(\tau)$ (см. (3.19)) обнаруживает весьма малые различия в условиях устойчивости для замкнутой и разорванной цепи нейронов, если соединение каждого нейрона с соседними антисимметрично, то есть при $a = -b$ в (3.21), (3.22).

## 3.5 Доказательства теорем главы 3

### 3.5.1 Доказательство Теоремы 3.1

Начнем доказательство с замечания, что неравенство

$$u_1 + u_3 \geqslant 0 \tag{3.25}$$

верно на поверхности конуса устойчивости (1.21), (1.22) (см. определение 1.2) и внутри него. Из (3.14) и $a + b > 1$ получим $u_1^{'}(\pi) + u_3^{'}(\pi) = -\tau(b + a - 1) < 0$. Последнее неравенство показывает, что точка $M^{'}(\pi)$ находится вне конуса устойчивости (см. (3.25)), поэтому при $a + b > 1$ система (3.5), (3.6) неустойчива, если $n$ достаточно велико.

Преобразование $t \to t + \pi, a \to -a, b \to -b$ переводит кривую (3.14) в себя, поэтому область устойчивости системы (3.5), (3.6) центрально-симметрична. Следовательно, и неравенство $a+b < -1$ обеспечивает неустойчивость системы (3.5), (3.6) при достаточно больших $n$. Аналогично, положив в (3.15) $t = \pi/2$, докажем неустойчивость (3.5), (3.7) при достаточно больших $n$ и $|a + b| > 1$. Теорема 3.1 доказана.

### 3.5.2 Доказательство Теоремы 3.2

1. Для выяснения асимптотики $a_1(\tau)$ требуется вычислить точки касания в $\mathbb{R}^3$ конуса устойчивости с кривой (3.14) при $a = a_1(\tau)$, $b = -a_1(\tau)$, $t = \pi/2$. На кривой (3.14)

$$\arg(u_1^{'}(\frac{\pi}{2}) + iu_2^{'}(\frac{\pi}{2})) - \frac{\pi}{2} = \tau a_1(\tau). \tag{3.26}$$

Когда $\tau \to 0$ и $h = u_3(\pi/2) = \tau$, на конусе (1.21), (1.22)

$$\arg(u_1 + iu_2) - \frac{\pi}{2} = -\omega + \arg(i\omega - \tau) - \frac{\pi}{2} \sim \frac{\tau}{\omega} - \omega. \tag{3.27}$$

Здесь $\alpha \sim \beta$ означает, что $(\alpha/\beta) \to 1$. Касание кривой и конуса влечет равенство аргументов (3.26), (3.27), а также равенство $|u_1 + iu_2| = |u_1^{'}(\pi/2) +$

$iu_2'(\pi/2)|$. Поэтому

$$\tau a_1(\tau) \sim \frac{\tau}{\omega} - \omega, \quad \tau a_1(\tau) = \sqrt{\omega^2 + \tau^2} \sim \omega. \tag{3.28}$$

Из (3.28) получим $\omega \sim \sqrt{\tau/2}$, что дает требуемую оценку $a_1(\tau) \sim 1/\sqrt{2\tau}$. 2. Для выяснения асимптотики $a_2(\tau)$ положим $a = a_2(\tau)$, $b = -a_2(\tau)$ в (3.15). Получим сегмент

$$u_1''(t) = 0, \quad u_2''(t) = 2\tau a_2(\tau) \sin t, \quad u_3''(t) = 0, \quad 0 \leqslant t \leqslant 2\pi. \tag{3.29}$$

На конусе (1.21), (1.22) положим $u_3 = \tau$, $u_1 = 0$. Получим $\omega^2 \sim \tau$, поэтому

$$u_2 = \omega \cos \omega + \tau \sin \omega \sim \sqrt{\tau}. \tag{3.30}$$

Из (3.30) и (3.29) при $t = \pi/2$ получим $a_2(\tau) \sim 1/(2\sqrt{\tau})$. Теорема 3.2 доказана.

### 3.5.3 Доказательство Теоремы 3.3

Начнем доказательство Теоремы 3.3 с леммы.

**Лемма 3.1.** Для любого натурального $n$ собственные числа $n \times n$ матрицы $D$ (см. (3.22)) суть

$$\mu_{jn} = 2\sqrt{ab} \cos \frac{\pi j}{n+1}, \quad 1 \leqslant j \leqslant n. \tag{3.31}$$

**Доказательство.** Характеристический многочлен матрицы $D$ (см. (3.22)) $D_n(\mu) = |\mu\, I - D|$ удовлетворяет соотношениям ($n = 1, 2, \ldots$)

$$D_1(\mu) = \mu, \quad D_2(\mu) = \mu^2 - ab, \quad D_{n+2}(\mu) = \mu D_{n+1}(\mu) - ab\, D_n(\mu). \tag{3.32}$$

При $ab = 0$ заключение леммы очевидно. Пусть $ab \neq 0$. Сделаем замену переменной и построим новую последовательность функций $U_n(y)$:

$$\mu = 2y\sqrt{ab}, \quad D_n(2y\sqrt{ab}) = (\sqrt{ab})^n U_n(y). \tag{3.33}$$

Из (3.32) и (3.33) выведем

$$U_1(y) = 2y, \quad U_2(y) = 4y^2 - 1, \quad U_{n+2}(y) = 2yU_{n+1}(y) - U_n(y), \quad n = 1, 2, \ldots \tag{3.34}$$

Равенства (3.34) описывают многочлены Чебышева второго рода $U_n(y)$, нули которых таковы:

$$y_{jn} = \cos\frac{\pi j}{n+1}, \quad 1 \leqslant j \leqslant n. \tag{3.35}$$

Из (3.35) и (3.33) получим требуемое. Лемма 3.1 доказана. ■

Продолжим доказательство Теоремы 3.3. Собственные числа $n \times n$ матрицы $E$ равны $\lambda_{jn} = 1$. Поэтому из (3.31) получим при $|ab| < 1/4$

$$\frac{|\mu_{jn}|}{\operatorname{Re}\lambda_{jn}} = 2\sqrt{ab}\,|\cos\frac{\pi j}{n+1}| < 1, \tag{3.36}$$

что достаточно для асимптотической устойчивости, независимой от запаздывания (см. Теорему 1.5). Пункт 1 Теоремы 3.3 доказан.

Для доказательства пунктов 2-4 Теоремы 3.3 построим точки $M_{jn} = (u_{1jn}, u_{2jn}, u_{3jn}) \in \mathbb{R}^3$ $(1 \leqslant j \leqslant n)$, такие что (см. (3.31))

$$u_{1jn} + iu_{2jn} = \tau\mu_{jn}\exp(i\tau\operatorname{Im}\,\lambda_{jn}) = 2\tau\sqrt{ab}\cos\frac{\pi j}{n+1}, \quad u_{3jn} = \tau\operatorname{Re}\lambda_{jn} = \tau. \tag{3.37}$$

Пусть $ab > 1/4$. Тогда по (3.37) найдется такое $n_0$, что для любого $n > n_0$

$$u_{1nn} = 2\tau\sqrt{ab}\cos\frac{\pi n}{n+1} < -\tau. \tag{3.38}$$

При этом $u_{2nn} = 0$, $u_{3nn} = \tau$. Но на поверхности и внутри конуса устойчивости (1.21), (1.22) на высоте $u_3 = \tau$ имеем

$$u_1 \geqslant -u_3 = -\tau. \tag{3.39}$$

Сравнивая (3.39) с (3.38), выясняем, что точка $M_{nn}$ при $n > n_0$ лежит вне конуса устойчивости. Поэтому система (3.21), (3.22) неустойчива при достаточно больших $n$. Пункт 2 Теоремы 3.3 доказан.

Из (3.37) и Леммы 3.1 при $ab < 0$ получим

$$u_{1jn} = 0, \quad u_{2jn} = 2\tau\sqrt{|ab|}\cos\frac{\pi j}{n+1}, \quad u_{3jn} = \tau. \tag{3.40}$$

В то же время на поверхности конуса устойчивости на высоте $u_3 = \tau$ при $u_1 = 0$

$$|u_2| = \frac{\omega}{\cos\omega}, \tag{3.41}$$

где параметр $\omega$ ввиду $u_1 = 0$ удовлетворяет условию (3.24). Теперь из (3.40), (3.23) и (3.24) получим

$$u_{2jn} = \tau\cos\frac{\pi j}{n+1}\sqrt{\frac{|ab|}{F(\tau)}}\frac{1}{\sin\omega}, \tag{3.42}$$

Если $|ab| < F(\tau)$, то из (3.42) и неравенства $\mu_{jn} < 2$ (см. Лемму 3.1) вытекает $|u_{2jn}| < |u_2|$ при любых $j, n$, поэтому все точки $M_{jn}$ лежат внутри конуса устойчивости, следовательно, система (3.21), (3.22) асимптотически устойчива при любых $n$. Пункт 3 Теоремы 3.3 доказан.

Пусть $|ab| > F(\tau)$. Тогда найдется такое значение $n_0$, что $\cos\frac{\pi}{n+1} > \sqrt{\frac{F(\tau)}{|ab|}}$ при любом $n > n_0$. Ввиду (3.40) отсюда следует, что

$$u_{21n} > 2\tau\sqrt{F(\tau)}. \tag{3.43}$$

Из (3.43), (3.23) и (3.24) получим $u_{21n} > \frac{\omega}{\cos\omega}$, что ввиду (3.41) дает $u_{21n} > u_2$. Поэтому точка $M_{nn}$ находится вне конуса устойчивости, что влечет неустойчивость системы (3.21), (3.22) при всех $n > n_0$. Теорема 3.3 доказана.

## 3.6 Сравнение результатов главы 3 с известными результатами

В большинстве работ по устойчивости кольцевых нейронных сетей рассматривается задача об устойчивости сети из двух [4], трех [28] или четырех

[14], [29] нейронов. Задача об устойчивости кольцевой системы произвольного количества нейронов рассматривали Ю. Юан и С. Кэмпбелл [15]. В [15] изучена следующая система, описывающая кольцевую нейронную сеть:

$$\dot{x}_j(t) + x_j(t) + \alpha\, x_j(t-\tau_s) + \beta\,(x_{j-1}(t) + x_{j+1}(t-\tau)) = 0 \qquad (j \bmod n), \quad (3.44)$$

В уравнении (3.44) реакция нейрона на собственный сигнал (selfconnection) разделена на две части: мгновенная реакция с интенсивностью 1 и запаздывающая на время $\tau_s$ с интенсивностью $\alpha$. Реакции нейрона на сигнал правого и левого соседа имеет одинаковые запаздывания $\tau$ и одинаковые интенсивности $\beta$. Если положить $\alpha = 0$, то система (3.44) совпадет с частным видом нашей системы (3.3) с $a = b$. Как показывают Предложение 3.1 и Теорема 3.1, а также рисунок 3.8, поведение таких систем несложно: при $|\beta| < 1/2$ система устойчива независимо от запаздывания, при $|\beta| > 1/2$ система неустойчива независимо от запаздывания. Сложность динамики системы нейронов в системе Юан-Кэмпбелл (3.44) происходит именно из-за запаздывания в реакции на собственный сигнал.

Это свидетельствует о недостаточной адекватности модели (3.44). В нашей модели источником сложности поведения моделей (3.2), (3.3) является несимметричность (когда $a \neq b$) и даже антисимметричность (когда $a = -b$) реакции нейрона на сигналы правого и левого соседа, что соответствует традиционному разделению сигналов на возбуждающие и тормозящие.

После публикации [28] 2006 года работы С. Кэмпбелл по нейронным сетям были посвящены только сетям с распределенным запаздыванием.

В работе [29] изучена модель кольцевой сети (3.44) с $\tau_s = \tau$, в которой $n = 4$. В этой статье указаны только области устойчивости, независимой от запаздывания, статья посвящена в основном проблемам симметрия в поведении нелинейной системы.

Насколько известно автору, кольцевые нейронные сети с неопределенно большим количеством нейронов до сих пор не рассматривались в литературе. Благодаря рассмтрению большого количества нейронов изложение решения задачи об устойчивости кольцевой сети в диссертации является достаточно прозрачным. Вот для сравнения цитата из работы [15] (2004) Ю. Юан и С. Кэмпбелл:

«Области устойчивости при нечетных $n$ сходны с областями при $n = 3$, которые были исследованы в [28] поэтому мы сосредоточимся на области устойчивости для четных $n$.»

Изучение устойчивости линейных конфигураций нейронов не встречается в литературе, хотя в публикациях по устойчивости кольцевых сетей такая тема вполне естественна, так как линейная конфигурация в некотором смысле является предельной для кольцевой.

# Глава 4

# Численное и качественное исследование устойчивости сетей кольцевой и линейной конфигураций с ограниченным количеством нейронов

## 4.1 Алгоритм и программа для построения границ областей устойчивости кольцевых нейронных сетей с ограниченным количеством нейронов

В этом разделе мы опишем программу для построения границ области устойчивости кольцевых нейронных сетей с ограниченным количеством нейронов. Программа использует фрагменты предыдущих программ «Анализ устойчивости» (раздел 2.2) и «Устойчивость нейронных сетей» (раздел **??**).

Мы рассматриваем задачу устойчивости системы (см. формулу (3.2))

$$\dot{x}_j(t) + x_j(t) + a\,x_{j-1}(t) + b\,x_{j+1}(t-\tau) = 0 \qquad (j \bmod n). \tag{4.1}$$

В матричном виде система имеет вид (см. (3.4))

$$\dot{x}(t) + (E + aP)\,x(t) + bP^{n-1}\,x(t-\tau) = 0, \tag{4.2}$$

где (см. (3.5))

$$P = \begin{pmatrix} 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{pmatrix}. \tag{4.3}$$

### 4.1.1 Функциональное назначение программы, область применения, её ограничения

Программа «Построение областей устойчивости круговых нейронных сетей» строит области устойчивости для круговых нейронных сетей с заданным количеством нейронов в плоскости параметров $a$ и $b$ в сравнении с областью устойчивости таких же нейронных сетей, но с бесконечным числом нейронов, и сохраняет результаты в графических файлах определённого пользователем формата. Программа выполнена в высокоуровневой среде для математических вычислений MATLAB 7.11.0 (R2010b) посредством matlab APY для создания графического интерфейса пользователя (GUI).

### 4.1.2 Использование

Все входные параметры задаются на одноимённой панели, показанной на рисунке 4.1.

Прежде всего пользователю предлагается выбрать конфигурацию нейронной сети. В программе доступны две варианта кругового соединения нейронов: с малым запаздыванием взаимодействия с правыми соседями (уравнение (1)) и с одинаковым запаздыванием взаимодействия с правыми и левыми соседями (уравнение (2)). Выбирая первую или вторую радиокнопку, пользователь определяет, для какой именно модели будет проводиться анализ устойчивости. На рисунке 1 выбрана первая конфигурация, а на рисунке 2 – вторая. В дальнейшем кольцевые сети, описываемые уравнением (1), будем называть сетями с односторонним запаздыванием, а сети, описываемые уравнением (2), - сетями с двусторонним запаздыванием. Ниже, на панели ввода представлено название, описание и рисунок выбранной конфигурации. Пользователю необходимо ввести параметр $\tau$, для которого будут строиться области устойчивости, определить количество нейронов в сети (причём графики

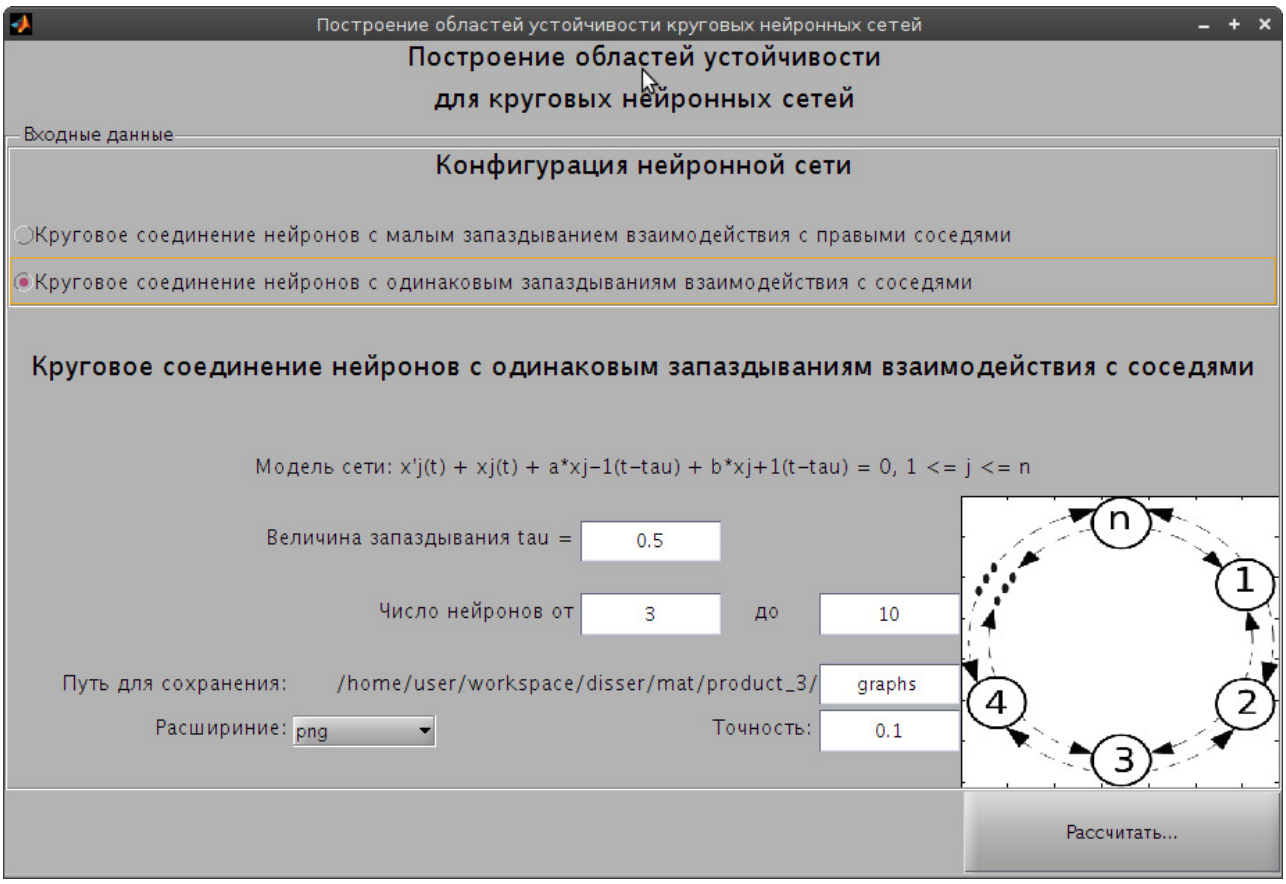



Рис. 4.1. Главное окно программы «Построение областей устойчивости круговых нейронных сетей» при выборе первой конфигурации

будут построены для всех $n$, определённых пользователем), задать директорию для сохранения, выбрать расширение, с которым будут сохраняться графики, и определить погрешность построения.

После ввода данных в обоих случаях для начала расчета необходимо нажать на кнопку «Рассчитать...». В процессе построения будут появляться вспомогательные окна, представляющее результаты расчетов. Все графики сохраняются в выбранную пользователем директорию в заданном формате и именуются определённым образом: «type_tau_n.ext»,

где type — тип соединения нейронов: «asym» для первого соединения и «sym» для второго;

tau — величина запаздывания;

n — количество нейронов;

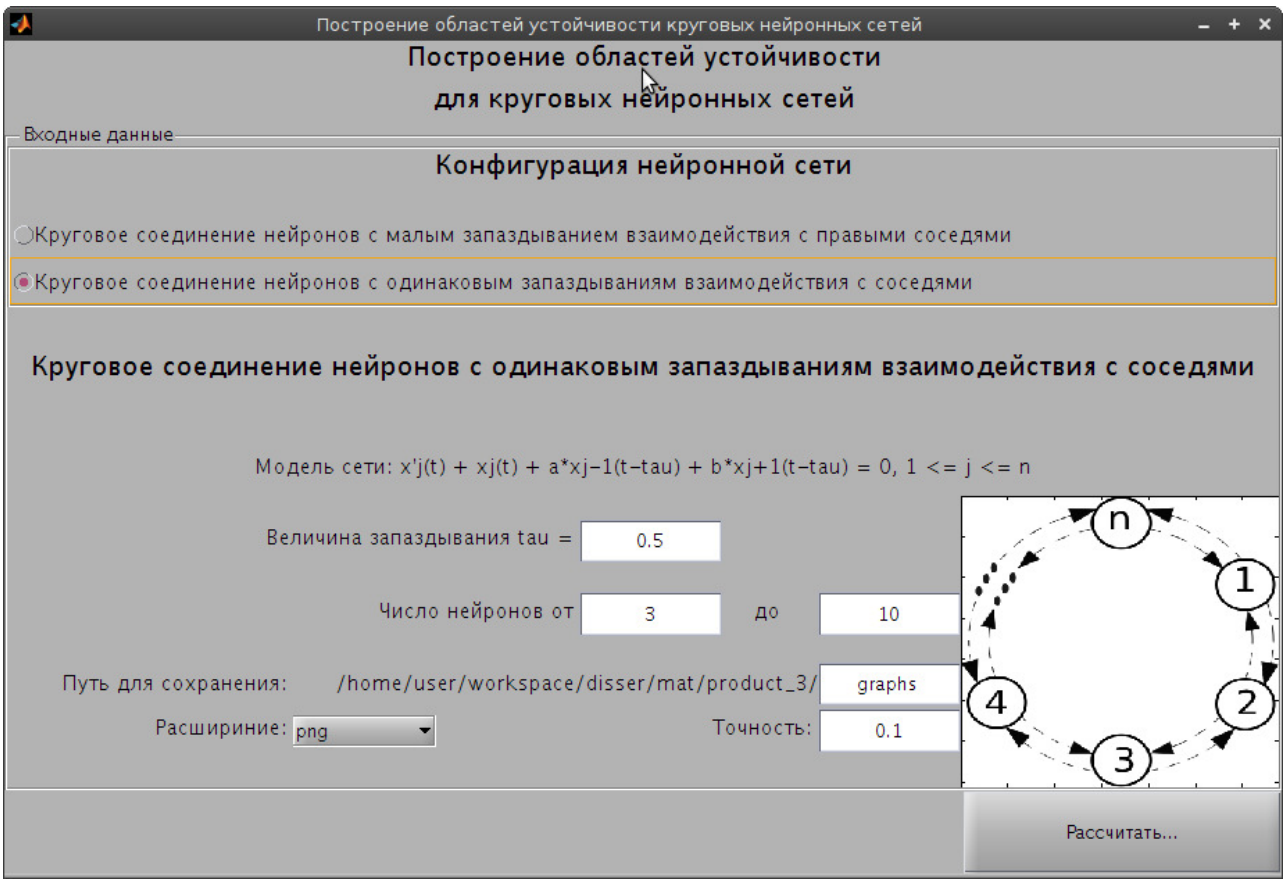



Рис. 4.2. Главное окно программы «Построение областей устойчивости круговых нейронных сетей» при выборе второй конфигурации

ext — расширение файла.

Программа также строит область устойчивости для бесконечного числа нейронов.

После проведения расчетов пользователь может изменить входные параметры и еще раз выполнить расчет областей устойчивости. Графики будут сохранятся во вновь назначенную пользователем директорию или в ту же, если он не поменял её. Во втором случае она будут дополняться к уже существующим, а если найдутся графики с одинаковыми параметрами, то файлы будут перезаписаны. Данную процедуру можно проделывать сколько угодно раз.

При вводе некорректных данных программа выдает различные сообщения об ошибках. Например, следующее сообщение может быть получено при вводе некорректных значений в поле (рисунок 4.3).

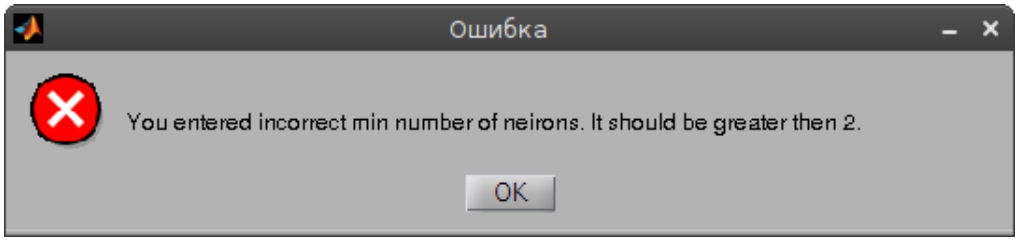



Рис. 4.3. Сообщение об ошибке: введены некорректные данные

Также программа анализирует потенциальную возможность затратить большое количество времени на вычисления, когда пользователь назначает слишком высокую точность, либо большое количество графиков. В этом случае программа просит пользователя подтвердить, что он действительно хочет выполнить расчёты при заданных параметрах. Пример такого сообщения приведён на рисунке 4.4.

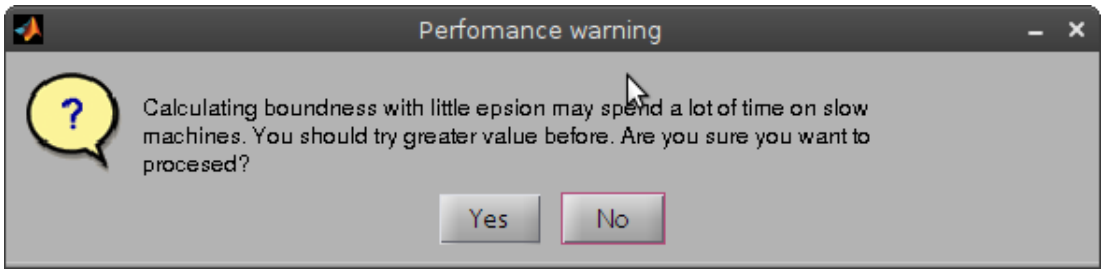



Рис. 4.4. Сообщение об ошибке: введены некорректные данные

## 4.2 Границы областей устойчивости кольцевых сетей с ограниченным количеством нейронов и односторонним запаздыванием

В этом разделе мы укажем результаты поиска границ устойчивости для кольцевой системы нейронов с односторонним запаздыванием. Система опи-

сывается уравнениями (4.2), (4.3). В отличие от предыдущей главы, здесь мы будем рассматривать сети с ограниченным количеством нейронов. Разделим результаты на три группы: сети со средним запаздыванием, малым и большим запаздываниями.

Во всех трех группах количество нейронов не меньше трех, так как для двух нейронов описание кольцевой архитектуры противоречиво.

### 4.2.1 Сети со средними значениями запаздываний

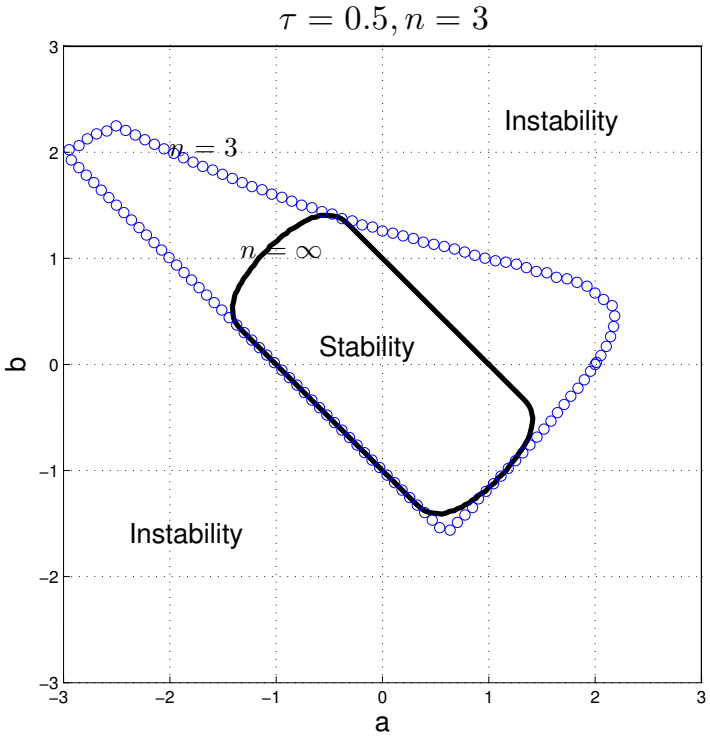



Рис. 4.5. Область устойчивости для сети (4.2), (4.3) с односторонним запаздыванием при $\tau = 0.5$, $n = 3$.

текст

текст

текст

текст

текст

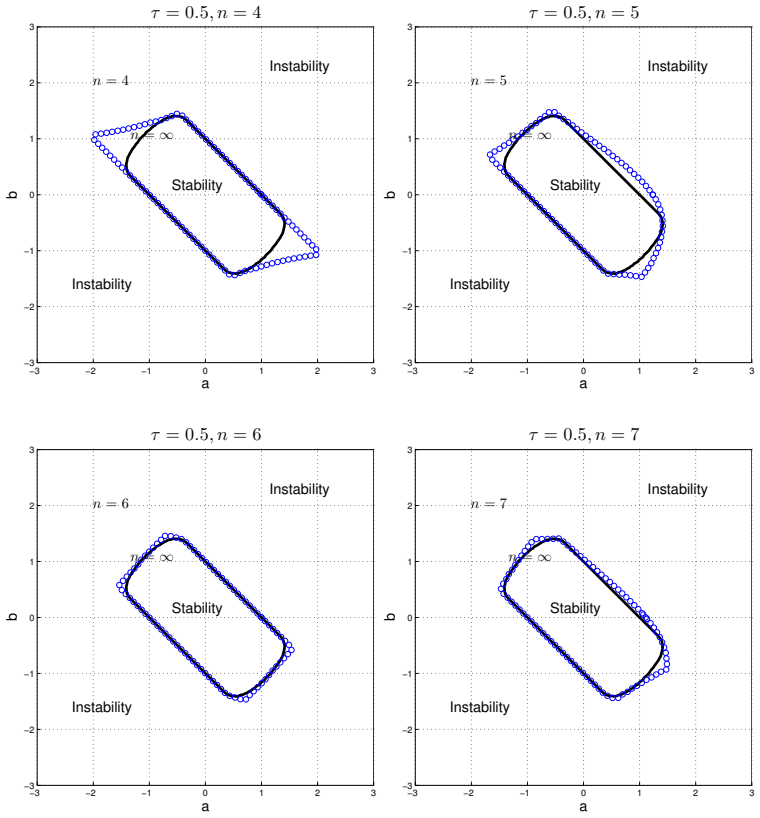



Рис. 4.6. Границы областей устойчивости для системы (4.2), (4.3) с односторонним запаздыванием для $\tau = 0.5$ и значений $n$ от 4 до 7 показана кружками. Сплошная линия — граница области устойчивости, гарантированной для любого $n$.

### 4.2.2 Сети с малыми и большими значениями запаздываний

текст

текст

текст

текст

текст

текст

текст

текст

текст

текст

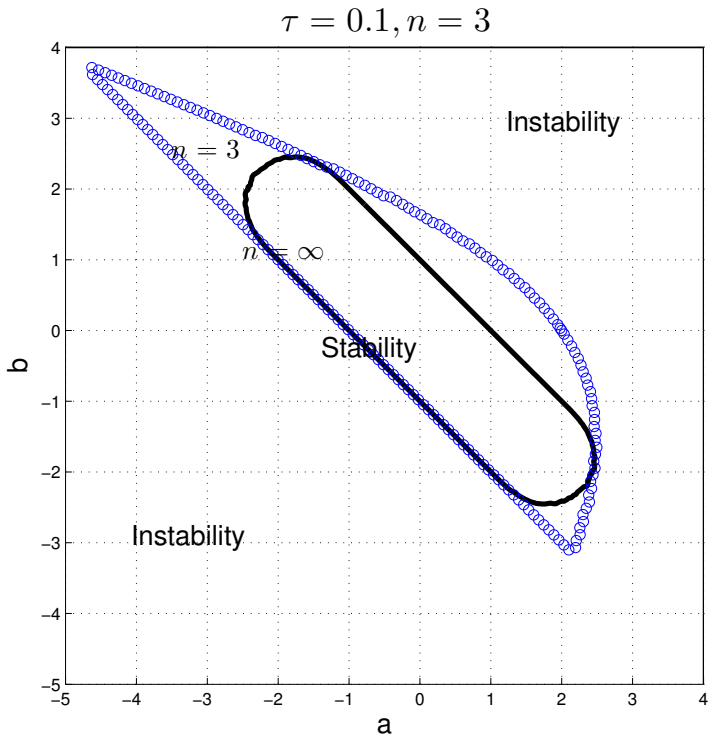



Рис. 4.7. Границы областей устойчивости для системы (4.2), (4.3) с односторонним запаздыванием для $\tau = 0.1$. Сплошная линия — граница области устойчивости, гарантированной для любого $n$.

текст

текст

текст

текст

текст

текст

текст

текст

текст

текст

текст

текст

текст

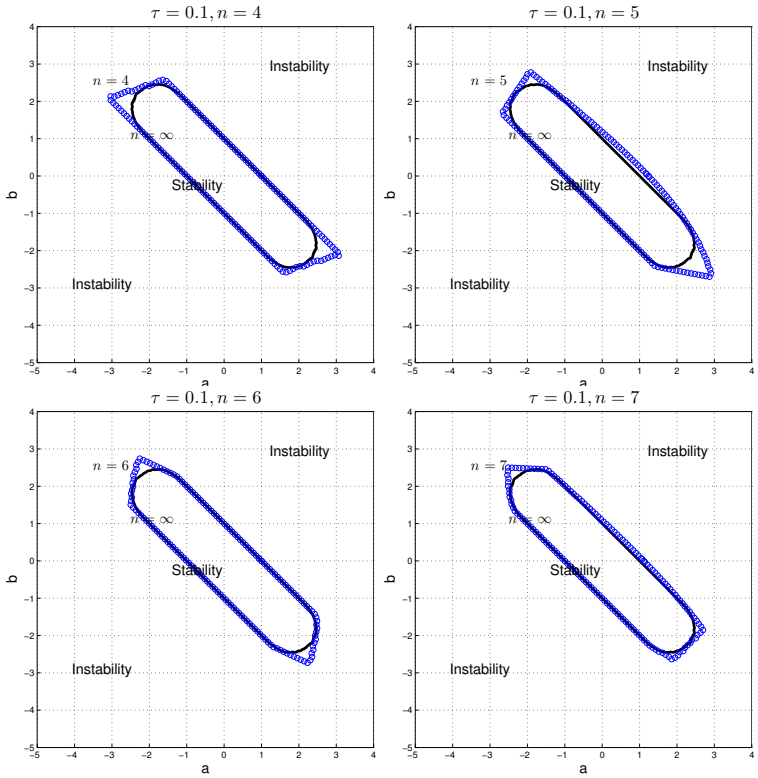



Рис. 4.8. Границы областей устойчивости для системы (4.2), (4.3) с односторонним запаздыванием для $\tau = 0.1$. Сплошная линия — граница области устойчивости, гарантированной для любого $n$.

текст

текст

## 4.3 Границы областей устойчивости кольцевых сетей с ограниченным количеством нейронов и двусторонним запаздыванием

Мы рассматриваем задачу устойчивости системы (см. формулу (3.3))

$$\dot{x}_j(t) + x_j(t) + a\, x_{j-1}(t-\tau) + b\, x_{j+1}(t-\tau) = 0 \qquad (j \bmod n). \tag{4.4}$$

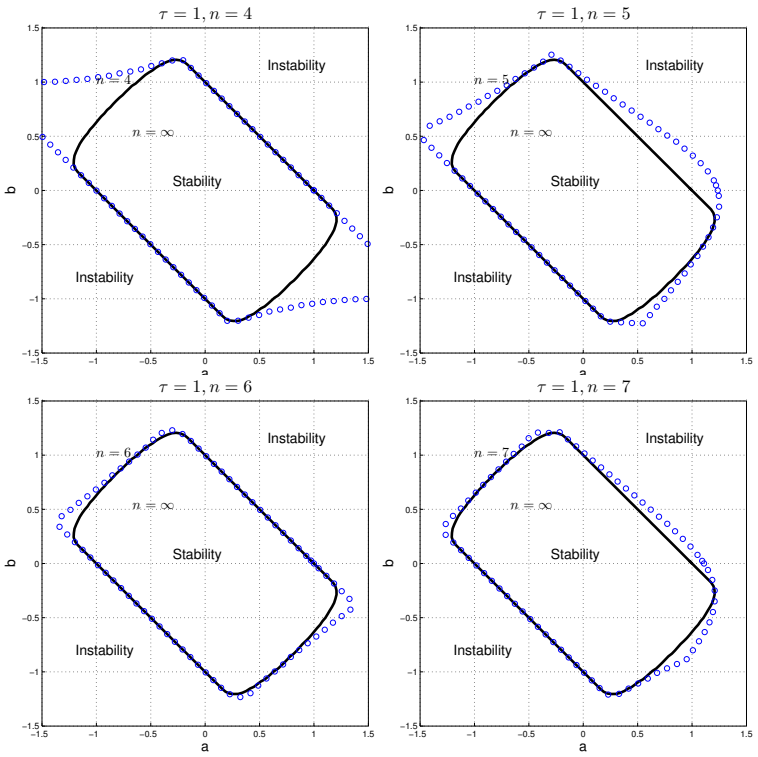



Рис. 4.9. Границы областей устойчивости для системы (4.2), (4.3) с односторонним запаздыванием для $\tau = 1.0$. Сплошная линия — граница области устойчивости, гарантированной для любого $n$.

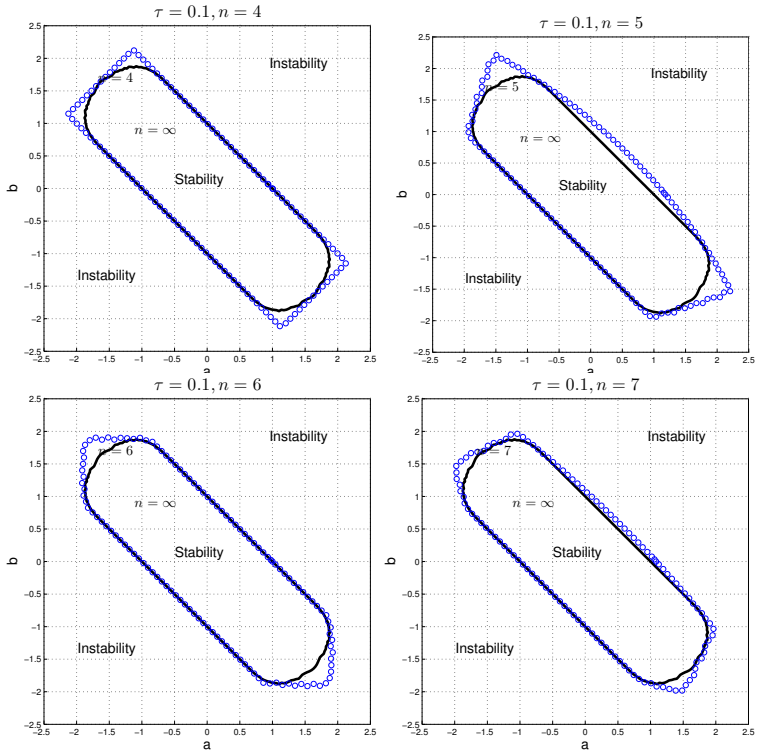



Рис. 4.10. Границы областей устойчивости для системы (**??**), (4.3) с двусторонним запаздыванием для $\tau = 0.1$. Сплошная линия — граница области устойчивости, гарантированной для любого $n$.

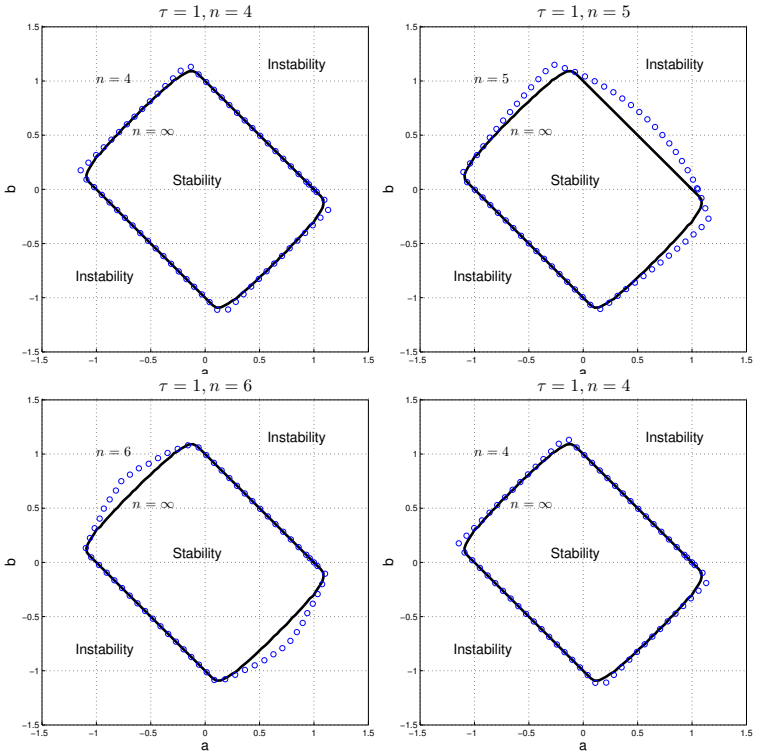



Рис. 4.11. Границы областей устойчивости для системы (**??**), (4.3) с двусторонним запаздыванием для $\tau = 1.0$. Сплошная линия — граница области устойчивости, гарантированной для любого $n$.

# Литература


1.

2.

3.

4.

5.

6.

7.

8.

9.

10.

11.

12.

13.

14.

15.

16.

17.

18.

19.

20.

21.

22.

23.

24.

25.

26.

27.

28.

29.

30.

31.

32.

33.

# Приложение А

# Исходный код программы «Анализ устойчивости»

## application.m

```
function application
debug = false;
%format short;

    MAX_DIM = 5;
    fontsize0 = 12;
    fontsize1 = 15;

    % Error strings
    label_error = 'Ошибка';
    err_incorrect_cell = 'Вы ввели некорректное значение в ячейке ';



    label_graph = 'График';
    appTitle = 'Анализ устойчивости';
    header = 'Анализ устойчивости уравнения ';
    header_eq = 'x''(t) + Ax(t) + Bx(t-tau) = 0';
    inputPanelTitle = 'Входные данные';
    inputPanelDesc = 'Базовая матрица';
    resultPanelTitle = 'Результаты вычислений';
    label_dim = 'Размерность = ';
    label_method = 'Выберите способ задания матриц A и B';
    label_method1 = 'Задать матрицы A и B с помощью базовой матрицы D';
    label_method2 = 'Задать матрицы A и B с помощью собственных чисел';
    label_eig = 'Собственные числа матрицы ';
    label_A = 'A = ';
    label_B = 'B = ';
    label_D = 'D = ';
    label_rand = 'Заполнить';
    tip_rand = 'Заполняет матрицу случайными значениями';
    label_clear = 'Очистить';
    tip_clear = 'Очищает матрицу';
    label_calc = 'Рассчитать...';
    label_calc_tip = 'Определяет интервалы устойчивости системы, заданной матрицами A и B';
    label_close = 'Закрыть все графики';
    tooltip_close = 'Закрывает все ранее открытые графики';
    label_result_good = 'Уравнение устойчиво при tau, лежащем в следующем объединении интервалов:';
    label_result_bad = 'Уравнение неустойчиво при любом tau!';
    label_vector = 'Введите коэффициенты в разложении матриц A и B по степеням матрицы D';
```

```
    % Create a figure that will have a uitable, axes and checkboxes
f = figure('Position', [50, 100, 900, 700],...
        'Name', appTitle,...  % Title figure
        'NumberTitle', 'off',... % Do not show figure number
        'MenuBar', 'none');      % Hide standard menu bar menus

    uicontrol('style', 'text', ...
        'string', header, ...
        'fontsize', fontsize1, ...
        'units', 'normalized', ...
        'position', [0, .95, 1, .05]);
    uicontrol('style', 'text', ...
        'string', header_eq, ...
        'fontsize', fontsize1, ...
        'units', 'normalized', ...
        'position', [0, .9, 1, .05]);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Input panel.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
inputPanel = uipanel('Parent',f,'Title',inputPanelTitle,...
         'Position',[0 .3 1 .6]);
inputPanel1 = uipanel('Parent',f,'Title',inputPanelTitle,...
         'Position',[1 .3 1 .6]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sliderPanel = uipanel('Parent',inputPanel, ...
    'bordertype', 'none', 'Position',[0 .9 1 .1]);
if debug == true
set (sliderPanel, 'bordertype', 'etchedout') ;
end

uicontrol(sliderPanel, 'style', 'text', 'fontsize', fontsize1, 'string', label_dim, ...
        'horizontalalignment', 'right',...
        'units', 'normalized', 'position', [0, 0, .2, 1]);
dimText = uicontrol(sliderPanel, 'style', 'text', ...
        'units', 'normalized', ...
        'fontsize', fontsize1, ...
        'position', [0.2, 0, .1, 1]);
    % scale
scale = uipanel('parent', sliderPanel, 'bordertype', 'none', 'Position',[0.3, 0, .7, 1]);
    for n=1:5
        uicontrol(scale, 'style', 'text', ...
            'string', n, ...
            'horizontalalignment', 'left', ...
            'units', 'normalized', ...
            'position', [n/5-.1, 0, .1, .5]);
```

```
    end
dimSlider =    uicontrol(sliderPanel, 'style', 'slider', ...
        'min', 0, 'max', 4, 'value', 4, 'sliderstep', [.25,.25], ...
        'units', 'normalized', 'position', [0.3, .5, .7, .5], ...
        'callback', @dimSlider_callback);
        set(dimText, 'string', get(dimSlider, 'value')+1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% input method
group = uibuttongroup('parent', inputPanel, 'Position',[0 .65 1 .25]);
uicontrol(group, 'style', 'text', 'fontsize', fontsize1, 'string', label_method, ...
        'units', 'normalized', 'position', [0, 0.7, 1, .3]);
set(group,'SelectionChangeFcn',@sel_callback);
uicontrol('parent',group, 'Style','Radio',...
    'fontsize', fontsize0, 'String',label_method1,...
    'units', 'normalized', 'pos',[0, .3, 1,.3]);
uicontrol('parent',group, 'Style','Radio',...
    'fontsize', fontsize0, 'String',label_method2,...
    'units', 'normalized', 'pos',[0, 0, 1,.3]);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% matrix input
matrixPanel = uipanel('Parent',inputPanel, 'bordertype', 'none',...
                    'Position',[0 .25 1 .4]);
if debug == true
   set (matrixPanel, 'bordertype', 'etchedout') ;
end

dim = str2double(get(dimText, 'string'));
    uicontrol(matrixPanel, 'style', 'text', 'fontsize', fontsize1, 'string', inputPanelDesc, ...
        'horizontalalignment', 'right',...
        'units', 'normalized', 'position', [0, 0.6, .2, .4]);
    uicontrol(matrixPanel, 'style', 'text', 'fontsize', fontsize1, 'string', label_D, ...
        'horizontalalignment', 'right',...
        'units', 'normalized', 'position', [0.2, 0.6, .1, .4]);
clearAButton =    uicontrol(matrixPanel, 'style', 'pushbutton', ...
    'string', label_clear, 'tooltipString', tip_clear,...
        'units', 'normalized', 'position', [0.2, 0.4, .1, .2], ...
        'visible', 'off', 'callback', @clear_callback);
randAButton =     uicontrol(matrixPanel, 'style', 'pushbutton', ...
    'string', label_rand, 'tooltipString', tip_rand,...
        'units', 'normalized', 'position', [0.2, 0.2, .1, .2], ...
        'visible', 'off', 'callback', @random_callback);
if debug == true
   set (clearAButton, 'visible', 'on') ;
   set (randAButton, 'visible', 'on') ;
end

    dim = MAX_DIM;
```

```
    matrixD = createTable(dim, dim);
        set(matrixD.root, 'Parent', matrixPanel);
        set(matrixD.root, 'Position',[.3 0 .7 1]);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Vector Panel

vectorPanel = uipanel('Parent',inputPanel, 'bordertype', 'none',...
                    'Position',[0 0 1 .25]);
if debug == true
   set (vectorPanel, 'bordertype', 'etchedout') ;
end
uicontrol(vectorPanel, 'style', 'text', 'fontsize', fontsize1, 'string', label_vector, ...
    'units', 'normalized', 'position', [0, 0.7, 1, .3]);

uicontrol(vectorPanel, 'style', 'text', 'fontsize', fontsize1, 'string', label_A, ...
    'horizontalalignment', 'right',...
    'units', 'normalized', 'position', [0.1 .3 .1 .3]);
vectorA = createVector(dim);
    set(vectorA.root, 'Parent', vectorPanel);
    set(vectorA.root, 'Position',[0.2 .3 .8 .3]);

uicontrol(vectorPanel, 'style', 'text', 'fontsize', fontsize1, 'string', label_B, ...
    'horizontalalignment', 'right',...
    'units', 'normalized', 'position', [0.1 0 .1 .3]);
vectorB = createVector(dim);
    set(vectorB.root, 'Parent', vectorPanel);
    set(vectorB.root, 'Position',[0.2 0 .8 .3]);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Eig Panel
eigPanel = uipanel('Parent',inputPanel, 'bordertype', 'none',...
                    'Position',[1 0 1 .65]);
if debug == true
   set (eigPanel, 'bordertype', 'etchedout') ;
end
eigInput1 = createEigInput('A', 'lambda') ;
set (eigInput1.root, 'parent', eigPanel) ;
set (eigInput1.root, 'position', [0, 0, .5, 1]) ;

eigInput2 = createEigInput('B', 'mu') ;
set (eigInput2.root, 'parent', eigPanel) ;
set (eigInput2.root, 'position', [0.5, 0, .5, 1]) ;
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Result panel.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
resultPanel = uipanel('Parent',f,'Title',resultPanelTitle,...
         'Position',[0 0 1 .3]);

%     result_list = uicontrol(resultPanel, 'style', 'listbox', ...
%          'string', ['ddd'; 'ads'], ...
%          'fontsize', fontsize1, ...
%          'units', 'normalized', 'position', [0, .3, 1, .7]);

    resultChildPanel = uipanel('Parent',resultPanel,'bordertype', 'none',...
         'Position',[0 0.2 .7 .8]);
    if debug == true
       set (resultChildPanel, 'bordertype', 'etchedout') ;
    end
    resultLabels = zeros (1, MAX_DIM) ;
    for n=1:MAX_DIM
        resultLabels(n) = uicontrol(resultChildPanel, 'style', 'text', ...
            'fontsize', fontsize1, 'horizontalalignment', 'left',...
            'units', 'normalized', 'position', [0, 1-.2*n, 1, .2]);
    end


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    resultMainPanel = uipanel('Parent',resultPanel,'bordertype', 'none',...
         'Position',[0.7 0.2 .3 .8]);
    if debug == true
       set (resultMainPanel, 'bordertype', 'etchedout') ;
    end
    result_label = uicontrol(resultMainPanel, 'style', 'text', ...
        'fontsize', fontsize1, ...
        'units', 'normalized', 'position', [0, .5, 1, .5]);
    intervals_label = uicontrol(resultMainPanel, 'style', 'text', ...
        'units', 'normalized', ...
        'fontsize', fontsize1, ...
        'position', [0, 0, 1, .5]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    bottomPanel = uipanel('Parent',resultPanel,'bordertype', 'none',...
         'Position',[0 0 1 .2]);
    if debug == true
       set (bottomPanel, 'bordertype', 'etchedout') ;
    end
close_btn = uicontrol(bottomPanel, 'style', 'pushbutton', ...
        'string', label_close, 'tooltipString', tooltip_close, ...
        'units', 'normalized', 'position', [0.8, 0, .2, 1], ...
        'enable', 'off', 'callback', @close_callback);
uicontrol(bottomPanel, 'style', 'pushbutton', 'string', label_calc, ...
```

```
        'tooltipString', label_calc_tip,...
        'units', 'normalized', 'position', [0.6, 0, .2, 1], ...
        'callback', @calc_callback);
uicontrol(bottomPanel, 'style', 'pushbutton', ...
    'string', label_rand, 'tooltipString', tip_rand,...
        'units', 'normalized', 'position', [0.4, 0, .2, 1], ...
        'callback', @random_callback);
uicontrol(bottomPanel, 'style', 'pushbutton', ...
    'string', label_clear, 'tooltipString', tip_clear,...
        'units', 'normalized', 'position', [0.2, 0, .2, 1], ...
        'callback', @clear_callback);

%------------------------ Handlers --------------------------------------%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    function sel_callback(~,~)
        pos = get (matrixPanel, 'position');
        if  pos(1) == 0
            % hide panels
            set (matrixPanel, 'position', [1 .25 1 .4]) ;
            set (vectorPanel, 'position', [1 0 1 .25]) ;

            % show panel
            set (eigPanel, 'position', [0 0 1 .65]) ;
        else
            % remove panel
            set (eigPanel, 'position', [1 0 1 .65]) ;

            % show panels
            set (matrixPanel, 'position', [0 .25 1 .4]) ;
            set (vectorPanel, 'position', [0 0 1 .25]) ;
        end
    end

    function test_callback(~,~)
        random_callback(0, 0) ;
    end


    app_x = 100;
    app_y = 200;
    graph_count = 0;
    function calc_callback(~, ~)
        % getting eig values.
        dim = getDim () ;
        eigA = zeros (1, dim) ;
        eigB = zeros (1, dim) ;

        % Choosing the input method
```

```
pos = get (matrixPanel, 'position');
if  pos(1) == 0
    D = getValues(matrixD.children, dim);
    eigD = eig(D) ;
    rowA = getVectorValues(vectorA.children, dim) ;
    rowB = getVectorValues(vectorB.children, dim) ;

    for j=1:length(eigD)
        e = [1 eigD(j) eigD(j)^2 eigD(j)^3 eigD(j)^4 eigD(j)^4]';
        eigA (j) = rowA*e(1:dim+1) ;
        eigB (j) = rowB*e(1:dim+1) ;
    end
else
    eigA = getEigValues(eigInput1);
    eigB = getEigValues(eigInput2);
end


if isempty(eigA) || isempty(eigB)
    return;
end

app_x = app_x + 80;
app_y = app_y - 80;
graph_count = graph_count + 1;
figure('Position', [app_x app_y, 600, 500],...
'Name', [label_graph ' ' num2str(graph_count)],...  % Title figure
'NumberTitle', 'off'... % Do not show figure number
);
[intervals, intervalsCell]= cone(eigA, eigB);
set (close_btn, 'enable', 'on') ;

dim = str2double(get(dimText, 'string'));
for n=1:dim
    set(resultLabels(n), 'string', buildResult(intervalsCell{n}));
end
for n=dim+1:MAX_DIM
    set(resultLabels(n), 'string', '');
end


% Set main result label.
if isempty(intervals)
    set(result_label, 'string', label_result_bad);
else
    set(result_label, 'string', label_result_good);
end

resultString = buildResult(intervals);
```

```
        set(intervals_label, 'string', resultString);
    end

% Closes all graph windows.
    function close_callback(~, ~)
        for n=1:graph_count
            try
                close([label_graph ' ' num2str(n)]);
            catch
                % Do nothing: the window was already closed.
            end
        end
        app_x = 100;
        app_y = 200;
        graph_count = 0;
        set (close_btn, 'enable', 'off') ;
    end

    % DimSlider was changed.
    function dimSlider_callback(~, ~)
        val = int32(get(dimSlider, 'value'))+1;
        if str2double(get(dimText, 'string')) == val
            return;
        end
        set(dimText, 'string', val);
        setDim(matrixD.children, val);
        setVectorDim(vectorA.children, vectorA.children1, val) ;
        setVectorDim(vectorB.children, vectorB.children1, val) ;
        setEigInputDim(eigInput1, val);
        setEigInputDim(eigInput2, val);
    end

    % Regenerate matrix.
    function random_callback(~, ~)
        setValues(matrixD.children, rand (getDim())*2-1);
        generateRowValues(vectorA.children, MAX_DIM+1) ;
        generateRowValues(vectorB.children, MAX_DIM+1) ;
        generateRowValues(eigInput1.children, MAX_DIM) ;
        generateRowValues(eigInput1.children1, MAX_DIM) ;
        generateRowValues(eigInput2.children, MAX_DIM) ;
        generateRowValues(eigInput2.children1, MAX_DIM) ;
    end

    % Clear matrix.
    function clear_callback(~, ~)
        %if obj == clearAButton
        clearMatrix(matrixD.children);
        clearRowValues(vectorA.children, MAX_DIM+1) ;
        clearRowValues(vectorB.children, MAX_DIM+1) ;
```

```
        clearRowValues(eigInput1.children, MAX_DIM) ;
        clearRowValues(eigInput1.children1, MAX_DIM) ;
        clearRowValues(eigInput2.children, MAX_DIM) ;
        clearRowValues(eigInput2.children1, MAX_DIM) ;
    end

%------------------------ Functions -----------------------------------%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

    function d = getDim ()
        d = str2double(get(dimText, 'string'));
    end

    function value = getRandomValAsString ()
        value = num2str(rand (1)*2-1,4);
    end

    % Build result string.
    function [resultString] = buildResult(intervals)
        resultString = '';
        for k = 1:2:length(intervals)
            from = num2str(intervals(k));
            to = num2str(intervals(k+1));
            resultString = [resultString ' (' from ', ' to ') U'];
        end
        resultString = resultString(1:end-1);
        if isempty(resultString)
            resultString = 'Empty set';
        end
    end

    function generateRowValues(children, dim)
        for n=1:dim
            set(children(n), 'string', getRandomValAsString());
        end
    end

    function clearRowValues(children, dim)
        for n=1:dim
            set(children(n), 'string', '');
        end
    end

%------------------------ Eig input -----------------------------------%
    function eigInput = createEigInput (matrixName, eigName)
        root = uipanel();
        uicontrol(root, 'style', 'text', 'fontsize', fontsize1,...
            'string', [label_eig matrixName],...
            'units', 'normalized', 'position', [0, .7, 1,.2]);
```

```
    panel = uipanel('parent', root, 'bordertype', 'none',...
        'position', [0, 0, 1, .7]);
    childrenLabels = zeros (1, MAX_DIM) ;
    children = zeros (1, MAX_DIM) ;
    childrenLabels1 = zeros (1, MAX_DIM) ;
    children1 = zeros (1, MAX_DIM) ;
    childrenLabels2 = zeros (1, MAX_DIM) ;
    for n=1:MAX_DIM
        strnum = num2str (n) ;
        childrenLabels(n) = uicontrol(panel, 'style', 'text', 'fontsize', fontsize1,...
            'horizontalAlignment', 'right', 'string', [eigName strnum ' = '],...
            'units', 'normalized', 'position', [0, 1-.2*n, .3, .2]);
        children(n) = uicontrol(panel, 'style', 'edit', ...
            'string', num2str (rand (1)*2-1, 4), 'backgroundcolor', 'w',...
            'units', 'normalized', 'position', [.3, 1-.2*n, .25, .2]);
        childrenLabels1(n) = uicontrol(panel, 'style', 'text', 'fontsize', fontsize1,...
            'horizontalAlignment', 'center', 'string', ' + ',...
            'units', 'normalized', 'position', [0.55, 1-.2*n, .1, .2]);
        children1(n) = uicontrol(panel, 'style', 'edit', ...
            'string', num2str (rand (1)*2-1, 4), 'backgroundcolor', 'w',...
            'units', 'normalized', 'position', [.65, 1-.2*n, .25, .2]);
        childrenLabels2(n) = uicontrol(panel, 'style', 'text', 'fontsize', fontsize1,...
            'horizontalAlignment', 'left', 'string', '*i',...
            'units', 'normalized', 'position', [0.9, 1-.2*n, .1, .2]);
    end
    eigInput.root = root;
    eigInput.children = children;
    eigInput.children1 = children1;
    eigInput.childrenLabels = childrenLabels;
    eigInput.childrenLabels1 = childrenLabels1;
    eigInput.childrenLabels2 = childrenLabels2;
end

function setEigInputDim(eigInput, dim)
    for n = 1:dim
        set(eigInput.children(n), 'visible', 'on');
        set(eigInput.children1(n), 'visible', 'on');
        set(eigInput.childrenLabels(n), 'visible', 'on');
        set(eigInput.childrenLabels1(n), 'visible', 'on');
        set(eigInput.childrenLabels2(n), 'visible', 'on');
    end
    for n = dim+1:MAX_DIM
        set(eigInput.children(n), 'visible', 'off');
        set(eigInput.children1(n), 'visible', 'off');
        set(eigInput.childrenLabels(n), 'visible', 'off');
        set(eigInput.childrenLabels1(n), 'visible', 'off');
        set(eigInput.childrenLabels2(n), 'visible', 'off');
    end
end
```

```
    function vector = getEigValues(eigInput)
        vector = zeros(1, getDim());
        for j=1:getDim()
            valRe = get(eigInput.children(j), 'string');
            valIm = get(eigInput.children1(j), 'string');
            parseValueRe = str2double(valRe);
            parseValueIm = str2double(valIm);
            if length(parseValueRe) ~= 1 || length(parseValueIm) ~= 1
                str = [err_incorrect_cell '(' num2str(j) ').'];
                errordlg(str, label_error, 'on');
                vector = [];
                return;
            end
            vector(j)=parseValueRe + parseValueIm*1i;
        end
    end


%------------------------ Vector input ----------------------------------%
    function vector = createVector(dim)
        dim = dim+1;
        root = uipanel('bordertype', 'none');
        children = zeros(1, dim); % row
        children1 = zeros(1, dim); % row
        strings = {'E + ', 'D + ', 'D^2+', 'D^3+', 'D^4+', 'D^5'};
        for j=1:dim
            children(j) = uicontrol(root, 'style', 'edit', ...
                'string', num2str (rand (1)*2-1, 4), 'backgroundcolor', 'w',...
                'units', 'normalized', 'position', [(j-1)/dim, 0, 1/dim/2, 1]);
            children1(j) = uicontrol(root, 'style', 'text', 'string', strings (j),...
                'fontsize', fontsize1, 'units', 'normalized', ...
                'position', [(j-1)/dim+1/dim/2, 0, 1/dim/2, 1]);
        end
        vector.root = root;
        vector.children = children;
        vector.children1 = children1;
    end

    function setVectorDim(children, children1, dim)
        for j= 1:MAX_DIM+1
            if j>dim+1
                set(children(j), 'visible', 'off');
                set(children1(j), 'visible', 'off');
            else
                set(children(j), 'visible', 'on');
                set(children1(j), 'visible', 'on');
            end
        end
```

```
    end

    function vector = getVectorValues(children, dim)
        vector = zeros(1, dim+1);
        for j=1:dim+1
            val = get(children(j), 'string');
            parseValue = str2double(val);
            if length(parseValue) ~= 1
                str = [err_incorrect_cell '(' num2str(j) ').'];
                errordlg(str, label_error, 'on');
                vector = [];
                return;
            end
            vector(j)=str2double(val);
        end
    end



%------------------------ Matrix input ----------------------------------%
    function matrix = createTable(rows, cols)
        root = uipanel('bordertype', 'none');
        children = zeros(rows, cols);
        for i=1:rows
            for j=1:cols
                children(i,j) = uicontrol(root, 'style', 'edit', ...
                    'string', num2str(rand (1)*2-1,4), 'backgroundcolor', 'w',...
                    'units', 'normalized', 'position', [(i-1)/rows, 1-j/cols, 1/cols, 1/rows]);
            end
        end
        matrix.root = root;
        matrix.children = children';
    end

    function [matrix] = getValues(children, dim)
        matrix = zeros(dim, dim);
        for i=1:dim
            for j=1:dim
                val = get(children(i,j), 'string');
                parseValue = str2double(val);
                if length(parseValue) ~= 1
                    str = [err_incorrect_cell '(' num2str(i) ', ' num2str(j) ').'];
                    errordlg(str, label_error, 'on');
                    matrix = [];
                    return;
                end
                matrix(i,j)=str2double(val);
            end
        end
```

```
    end

    function setValues(children, matrix)
        for i=1:length(matrix)
            for j=1:length(matrix)
                if i <= length(children) && j <= length(children)
                    set(children(i,j), 'string', num2str (matrix(i,j), 4));
                end
            end
        end
    end

    function clearMatrix(children)
        for i=1:length(children)
            for j=1:length(children)
                set(children(i,j), 'string', '');
            end
        end
    end

    function setDim(children, dim)
        for i = 1:MAX_DIM
            for j= 1:MAX_DIM
                if i>dim || j>dim
                    set(children(i,j), 'visible', 'off');
                else
                    set(children(i,j), 'visible', 'on');
                end
            end
        end
    end

end
```

## cone.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Displays cone and point set according different tau.
% Returns stability interval set.
% Details - stability intervals for each pair of lambda, mu.
%
% Usage: cone(lambda, mu),
%        where lambda, mu are the eigenvalues of matrixes A, B.
% Example:
% cone ([0.06+1.8658i 0.06-1.8658i],[0.025+0.1555i 0.025-0.1555i], 7, 0.5)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [result, intervalsCell] = cone(lambda, mu, maxdistanse, resolution)
```

```
if exist('n', 'var') == false
    n=100;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Построение конуса.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

tau_cone = 1;
W = pi/tau_cone-0.5; %pi/6;

w=-W:2*W/100:W;         %вектор для реализации основания конуса(круг)
%z=-1/tau_cone:0.1:1.5;            %накопление вектора z
z=-w./tan(w*tau_cone);
[W,Z]=meshgrid(w,z);        %задание 3-х мерной матрицы(пространства)

X=W.*sin(tau_cone*W) - Z.*cos(tau_cone*W); %задание вектора фигуры по оси x
Y=-W.*cos(tau_cone*W) - Z.*sin(tau_cone*W);%задание вектора фигуры по оси y


for ii=n/2+1 : n+1
for j=1 : n+1
Z(ii,j)=1;%tau_cone/(tau_cone);
X(ii,j)=1-z(50);
Y(ii,j)=0;
end
for j=1:1:ii-51
X(ii-50,j)=X(ii-50,ii-50);
X(ii-50,102-j)=X(ii-50,ii-50);
Y(ii-50,j)=0;
Y(ii-50,102-j)=0;
end
end


%отрисовка конуса по 3-м заданным координатам
hold on;
hold on;
shading interp;
colormap(gray);

mesh(X, Y, Z)
xlabel('$u_{1}$','Interpreter','latex','FontSize',30);
ylabel('$u_{2}$','Interpreter','latex','FontSize',30);
zlabel('$u_{3}$','Interpreter','latex','FontSize',30);

view([-30 30]);
alpha(.2);
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Построение точек.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Contains all stability intervals.
all = [0 Inf];

intervalsCell = cell(size (lambda) ) ;
for j=1:length(lambda)
    la = lambda(j);
    m = mu(j);

    % Calculation stability intervals.
    intervals = getintervals(lambda(j), mu(j));
    intervalsCell{j} = intervals;

    all = intersection(all, intervals);


if exist('maxdistanse', 'var') == false
    maxdistanse = 7;
end
if exist('resolution', 'var') == false
    resolution =0.5;
end

    dist_real = distance(m, la, 0, 1) ;

    max_tau = maxdistanse/dist_real;

    dist_real = distance(m, la, max_tau, max_tau-max_tau/10) ;

    quantity = dist_real/resolution*10;

%    max_tau =7;
%    quantity = 20;
    tau = 0.001 : max_tau/quantity : max_tau;


    for jj=1:length(tau)

        x = tau*abs(m).*cos(arg(m) + tau*imag(la));
        y = tau*abs(m).*sin(arg(m) + tau*imag(la));
        z = tau*real (la);

        if isInside(intervals, tau(jj))
            color = 'b';
```

```
        else
            color = 'r';
        end
        text(x(jj), y(jj), z(jj), '*', 'color', color, 'fontsize', 20);
    end
    text(x(jj), y(jj), z(jj), num2str(j), 'color', color, 'fontsize', 20);
end

result = all;
```

## point.m

```
function [x,y,z] = point(m, la, tau)
    x = tau*abs(m).*cos(arg(m) + tau*imag(la));
    y = tau*abs(m).*sin(arg(m) + tau*imag(la));
    z = tau*real (la);
end
```

## distance.m

```
function [distance] = distance(m, la, tau0, tau1)
    [x0, y0, z0] = point(m, la, tau0) ;
    [x1, y1, z1] = point(m, la, tau1) ;
    distance = sqrt ((x1-x0)^2 + (y1-y0)^2 + (z1-z0)^2) ;
end
```

## arg.m

```
function [result] = arg(x)
% Defines arg [-pi, pi]

% Preallocating variable result.
result = zeros(1, length(x));

if length(x)>1
    for k=1:length(x)
        result(k)=arg(x(k));
    end
    result = result';
    return;
end

a=real(x);
b=imag(x);

if a>0 && b==0
    result=0;

elseif a>0 && b>0
```

```
    result=atan(b/a);

elseif a==0 && b>0
    result=pi/2;

elseif a<0 && b>0
    result=pi+atan(b/a);

elseif a<0 && b==0
    result=pi;

elseif a<0 && b<0
    result=-pi+atan(b/a);

elseif a==0 && b<0
    result=-pi/2;

elseif a>0 && b<0
    result=atan(b/a);
end
```

## coalition.m

```
% Returns intersection set of 'set1' and 'set2'.
% 'set1' and 'set2' need to be even.
function result = coalition1(set1, set2)

% Validating
validate(set1);
validate(set2);



% Simple checking for empty set.
if isempty(set1)
    result = set2;
    return;
elseif isempty (set2)
    result = set1;
    return;
end

% Now dimention of set1, set2 >= 2.
% Determine complex set.
complexSet = [];
set = [];
if length (set1) == 2
    complexSet = set2;
    set = set1;
elseif length (set2) == 2;
```

```
    complexSet = set1;
    set = set2;
else
    result = set1;
    for n=1:2:length(set2)
        result = coalition1(result, set2(n:n+1));
    end
    return;
end

% Perform coalition complex and simple set.
% First and last indexed of simple set in the complexSet.
maxIndex = length(complexSet)+1;
first=maxIndex;
last=maxIndex;
for n=1:length(complexSet)
    if first == maxIndex && set(1)<complexSet(n)
        first = n;
    end
    if last == maxIndex && set(2)<complexSet(n)
        last = n;
    end
end

% Insert simple set into complex set.
if mod (first,2) == 1  % odd index
    if first == last
        result = [complexSet(1:first-1), set, complexSet(last:end)];
        return;
    elseif last>length (complexSet)
        result = [complexSet(1:first-1), set];
        return;
    end
    if mod (last, 2) == 1 % odd index
        result = [complexSet(1:first-1), set, complexSet(last:end)];
        return;
    else
        result = [complexSet(1:first-1), set(1), complexSet(last:end)];
        return;
    end
else % first index even.
    if first == last
        result = complexSet;
        return;
    end
    if mod (last, 2) == 1 % odd index
        result = [complexSet(1:first-1), set(2), complexSet(last:end)];
        return;
    else
```

```
        result = [complexSet(1:first-1), complexSet(last:end)];
        return;
    end
end



    function validate(set)
        % Check for the even lenth.
        if mod(length(set), 2) == 1
            error(['Input array [' num2str(set) '] should has the even length!']);
        end
        for n=1:length(set)-1
            if set(n)>set(n+1)
                error(['Input array [' num2str(set) '] should consist of increasing values!']);
            end
        end
    end

end
```

## distance.m

```
function [distance] = distance(m, la, tau0, tau1)
    [x0, y0, z0] = point(m, la, tau0) ;
    [x1, y1, z1] = point(m, la, tau1) ;
    distance = sqrt ((x1-x0)^2 + (y1-y0)^2 + (z1-z0)^2) ;
end
```

## getIntervals.m

```
% Gets the Stability intervales of the equation.
function result = getIntervals(lambda, mu)

% Simple cheking for stability.
if real(lambda) > abs(mu)
    disp('INFO: stability anyway!');
    result = [0 Inf];
    return;
end

% Perform necessary calculations.
Q = sqrt((abs(mu))^2 - (real(lambda))^2);
M = 15;
m = -M:M;

tau_plus = (arg(1i*Q - real(lambda)) + 2*pi*m - arg(mu)) / ...
            (imag(lambda) + Q);
tau_minus = (arg(-1i*Q - real(lambda)) + 2*pi*m - arg(mu)) / ...
```

```
            (imag(lambda) - Q);


% Check if atau in [0, pi/Q].
baund = pi/Q;
all = [tau_plus, tau_minus];    % concatenate arrays.
k=1;
for j = 1 : length(all)
    atau = all(j);
    if atau > 0 && atau <= baund && real(lambda) >= -Q/tan(Q*atau)
        result_set(k) = atau;
        k = k+1;
    end
end

if exist('result_set', 'var') == 0
    disp('INFO: empty set!');
    result = [];
    return;
end

% Check if [0, tau1] is included in the result set.
if (real(lambda) > 0 && mu > -pi/2 && mu < pi/2)    % spiral goes up.
    result_set = [0 result_set]; % add zero to result set.
elseif (real(lambda) > 0 && (real(lambda)) > abs(real(mu)))    % spiral goes up.
    result_set = [0 result_set]; % add zero to result set.
elseif (real(lambda) < 0 && abs(real(lambda)/real(mu)) < 1)
    result_set = [0 result_set]; % add zero to result set.
end

% Sort the result set.
disp('INFO: Sorted set: ');
result_set = sort(result_set)

% Remove the same values.
if length(result_set) > 1
    cleanSet = result_set;
    if result_set(1) == result_set(2)
        cleanSet = cleanSet(2:end);
    end
    for k = 2:length(result_set)-1
        if result_set(k) == result_set(k+1)
            cleanSet = [cleanSet(1:k-1), cleanSet(k+1:end)];
        end
    end
    result_set = cleanSet;
end

% Check for even.
```

```
if mod(length(result_set), 2) == 1
    disp('WARNING: stability interval has the odd lenth!!!');
    result_set = [0 result_set];
end

result = result_set;
```

## intersection.m

```
% Returns intersection set of 'set1' and 'set2'.
% 'set1' and 'set2' need to be even.
function result = intersection(set1, set2)

% Simple checking for empty set.
if isempty(set1) || isempty(set2)
    result = [];
    return;
end

% Check for the even lenth.
if mod(length(set1), 2) == 1 || mod(length(set2), 2) == 1
    error('Input arrays ''set1'' and ''set2'' must has the even length!');
end

% Check order.

% Perform simple checking of two intervals.
if length(set1)==2 && length(set2)==2
    result = [max(set1(1), set2(1)), min(set1(end), set2(end))];
    if result(1) > result(2)
        result = [];
    end
    return;
end



% Decomposition.
if length (set1) > 2 || length (set2) > 2
    result = [] ;
    for n = 1:2:length (set1)
        for k = 1:2:length (set2)
            set = intersection(set1(n:n+1) , set2(k:k+1)) ;
            result = coalition(set, result) ;
        end
    end
end
```

## isInside.m

```
% Determine if 'tau' belongs the 'evenSet'.
```

```
function result = isInside(evenSet, tau)

% Supporting 'tau' as array.
result = zeros(1, length(tau));
if length(tau)>1
    for k=1:length(tau)
        result(k) = isInside(tau(k));
    end
    result = result';
    return;
end

% Check for even.
if mod(length(evenSet), 2) == 1
    error('Input array ''evenSet'' must has the even length!');
end

% Check if tau is included in one of the interval.
for j = 1:2:length(evenSet)
    if evenSet(j) < tau && tau < evenSet(j+1)
        result = 1;
        return;
    end
end

% tau is excluded from all intervals.
result = 0;
```

# Приложение Б

# Исходный код программы «Устойчивость нейронных сетей»

## application2.m

```
function application2
debug = false;
%format short;
%whitebg(ax1, 'blue');
%colordef none
%text(.1,.5,'\tau', 'color', [0 0 0]);



fontsize0 = 12;
fontsize1 = 15;

    figure_width = 600;
    figure_height = 500;
    screen_size = get(0,'ScreenSize');
    screen_width = screen_size(1,3);
    screen_height = screen_size(1,4);
    figure_x = 0;
    figure_y = 0;
    graph_count = 0;
    resetFigureVars();
    image1 = imread('circles_right.png');
    image2 = imread('circles.png');



% Error strings
label_error = 'Ошибка';
err_incorrect_cell = 'Вы ввели некорректное значение в ячейке ';



label_graph = 'График';
appTitle = 'Устойчивость нейронных сетей';
header = 'Анализ устойчивости нейронных сетей';
header_eq = 'с большим количеством нейронов';
inputPanelTitle = 'Входные данные';
    resultPanelTitle = 'Результаты вычислений';
label_method = 'Конфигурация нейронной сети';
label_method1 = 'Круговое соединение нейронов с малым запаздыванием взаимодействия с правыми соседями';
label_method2 = 'Круговое соединение нейронов с одинаковым запаздываниям взаимодействия с соседями';
```

```
label_model1 = 'Модель сети: x''j(t) + xj(t) + a*xj-1(t) + b*xj+1(t-tau) = 0, 1 <= j <= n';
    label_model2 = 'Модель сети: x''j(t) + xj(t) + a*xj-1(t-tau) + b*xj+1(t-tau) = 0, 1 <= j <= n';
    label_intens = 'Интенсивность взаимодействия нейрона c';
    label_right = 'правым соседним нейроном a =';
    label_left = 'левым соседним нейроном b =';
    label_set_tau = 'Величина запаздывания tau = ';

label_rand = 'Заполнить';
tip_rand = 'Заполняет входные данные случайными значениями';
label_clear = 'Очистить';
tip_clear = 'Очищает входные данные ';
label_calc = 'Рассчитать...';
    label_calc_tip = 'Определяет интервалы устойчивости системы, заданной матрицами A и B';
    label_close = 'Закрыть все графики';
    tooltip_close = 'Закрывает все ранее открытые графики';
label_result_good = 'Уравнение устойчиво при tau, находящемся в интервале:';
label_result_bad = 'Уравнение неустойчиво.';

% Create a figure that will have a uitable, axes and checkboxes
f = figure('Position', [50, 100, 900, 700],...
'Name', appTitle,...  % Title figure
'NumberTitle', 'off',... % Do not show figure number
'MenuBar', 'none');      % Hide standard menu bar menus

uicontrol('style', 'text', ...
'string', header, ...
'fontsize', fontsize1, ...
'units', 'normalized', ...
'position', [0, .95, 1, .05]);
uicontrol('style', 'text', ...
'string', header_eq, ...
'fontsize', fontsize1, ...
'units', 'normalized', ...
'position', [0, .9, 1, .05]);

rootPanel = uipanel('Parent',f, 'bordertype', 'none', 'Position',[0 0 1 1]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Input panel.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
inputPanel = uipanel('Parent',rootPanel,'Title',inputPanelTitle,...
 'Position',[0 .3 1 .6]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% input method
group = uibuttongroup('parent', inputPanel, 'Position',[0 .75 1 .25]);
uicontrol(group, 'style', 'text', 'fontsize', fontsize1, 'string', label_method, ...
'units', 'normalized', 'position', [0, 0.7, 1, .3]);
set(group,'SelectionChangeFcn',@sel_callback);
```

```
rb1 = uicontrol('parent',group, 'Style','Radio',...
    'fontsize', fontsize0, 'String',label_method1,...
    'units', 'normalized', 'pos',[0, .3, 1,.3]);
rb2 = uicontrol('parent',group, 'Style','Radio',...
    'fontsize', fontsize0, 'String',label_method2,...
    'units', 'normalized', 'pos',[0, 0, 1,.3]);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% first method
inputPanel1 = uipanel('Parent',inputPanel, 'bordertype', 'none',...
                    'Position',[0 0 1 .65]);
                                                                        if debug == true
                                                                            set (inputPanel1, 'bordertype', 'etchedout
                                                                        end
method_text = uicontrol(inputPanel1, 'style', 'text', 'fontsize', fontsize1, 'string', label_method1, ...
'units', 'normalized', 'position', [0, 0.85, 1, .2]);
model_text = uicontrol(inputPanel1, 'style', 'text', 'fontsize', fontsize0, 'string', label_model1, ...
'units', 'normalized', 'position', [0, 0.7, 1, .1]);

leftPanel1 = uipanel('Parent',inputPanel1, 'bordertype', 'none',...
                    'Position',[0 0 .6 .7]);
                                                                        if debug == true
                                                                            set (leftPanel1, 'bordertype', 'etchedout'
                                                                        end

uicontrol(leftPanel1, 'style', 'text', 'fontsize', fontsize0, 'string', label_intens, ...
'units', 'normalized', 'position', [0, 0.7, 1, .2]);

uicontrol(leftPanel1, 'style', 'text', 'fontsize', fontsize0, 'string', label_right, ...
'units', 'normalized', 'position', [0, 0.5, .6, .15]);
input_a = uicontrol(leftPanel1, 'style', 'edit', ...
    'string', num2str (rand (1)*2-1, 4), 'backgroundcolor', 'w',...
    'units', 'normalized', 'position', [0.6 .5 .4 .15]);

uicontrol(leftPanel1, 'style', 'text', 'fontsize', fontsize0, 'string', label_left, ...
'units', 'normalized', 'position', [0, 0.3, .6, .15]);
input_b = uicontrol(leftPanel1, 'style', 'edit', ...
    'string', num2str (rand (1)*2-1, 4), 'backgroundcolor', 'w',...
    'units', 'normalized', 'position', [0.6 .3 .4 .15]);

uicontrol(leftPanel1, 'style', 'text', 'fontsize', fontsize0, 'string', label_set_tau, ...
'units', 'normalized', 'position', [0, 0, .6, .15]);
input_tau = uicontrol(leftPanel1, 'style', 'edit', ...
    'string', num2str (rand (1), 4), 'backgroundcolor', 'w',...
    'units', 'normalized', 'position', [0.6 .05 .4 .15]);

rightPanel1 = uipanel('Parent',inputPanel1, 'bordertype', 'none',...
                    'Position',[.75 0 .25 .7]);
```

```
                                                                    if debug == true
                                                                        set (rightPanel1, 'bordertype', 'etchedout
                                                                    end
ax = axes('parent', rightPanel1, 'position', [0 0 1 1]);
image( image1, 'parent', ax);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Result panel.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
resultPanel = uipanel('Parent',rootPanel,'Title',resultPanelTitle,...
 'Position',[0 0 1 .3]);
result_text = uicontrol(resultPanel, 'style', 'text', ...
'fontsize', fontsize1, ...
'units', 'normalized', 'position', [0, .5, .7, .5]);
interval_text = uicontrol(resultPanel, 'style', 'text', ...
'fontsize', fontsize1, ...
'units', 'normalized', 'position', [0.7, .5, .3, .5]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Bottom panel.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
bottomPanel = uipanel('Parent',resultPanel,'bordertype', 'none',...
 'Position',[0 0 1 .2]);
                                                                    if debug == true
                                                                        set (bottomPanel, 'bordertype', 'etchedout
                                                                    end
close_btn = uicontrol(bottomPanel, 'style', 'pushbutton', ...
        'string', label_close, 'tooltipString', tooltip_close, ...
'units', 'normalized', 'position', [0.8, 0, .2, 1], ...
'enable', 'off', 'callback', @close_callback);
uicontrol(bottomPanel, 'style', 'pushbutton', 'string', label_calc, ...
'tooltipString', label_calc_tip,...
'units', 'normalized', 'position', [0.6, 0, .2, 1], ...
'callback', @calc_callback);
uicontrol(bottomPanel, 'style', 'pushbutton', ...
    'string', label_rand, 'tooltipString', tip_rand,...
'units', 'normalized', 'position', [0.4, 0, .2, 1], ...
'callback', @random_callback);
uicontrol(bottomPanel, 'style', 'pushbutton', ...
    'string', label_clear, 'tooltipString', tip_clear,...
'units', 'normalized', 'position', [0.2, 0, .2, 1], ...
'callback', @clear_callback);


%------------------------ Functions ------------------------------------%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    function [a, b, tau] = getValues()
```

```
            a = str2double(get(input_a, 'string'));
            b = str2double(get(input_b, 'string'));
            tau = str2double(get(input_tau, 'string'));

    end


    function sel_callback(source,~)
        if  get(source,'SelectedObject') == rb1
            % show method 1
            set (method_text, 'string', label_method1);
            set (model_text, 'string', label_model1);
            image( image1, 'parent', ax);
        else
            % show method 2
            set (method_text, 'string', label_method2);
            set (model_text, 'string', label_model2);
            image( image2, 'parent', ax);
        end
    end


function calc_callback(~, ~)
        [a, b, tau] = getValues();
figure_y = figure_y - 30;
graph_count = graph_count + 1;
figure('Position', [figure_x figure_y, figure_width, figure_height],...
'Name', [label_graph ' ' num2str(graph_count)],...  % Title figure
'NumberTitle', 'off'... % Do not show figure number
);

        % determine the algorithm to calculate the result.
        if  get(group,'SelectedObject') == rb1
            result = cone_solver2 (a, b, tau, 1, 7, 0.5);
        elseif get(group,'SelectedObject') == rb2
            result = cone_solver2 (a, b, tau, 2, 7, 0.5);
        end

        % check for result and set label.
        if isempty(result)
            set (result_text, 'string', label_result_bad);
            set (interval_text, 'string', '');
        else
            set (result_text, 'string', label_result_good);
            set (interval_text, 'string', ...
                ['(' num2str(result(1)) ', ' num2str(result(2)) ').' ]);
        end
        set (close_btn, 'enable', 'on') ;
    end
```

```
    function resetFigureVars()
        figure_x = screen_width - figure_width;
        figure_y = screen_height - figure_height - 25;
        graph_count = 0;
    end

% Closes all graphic windows.
function close_callback(~, ~)
        for n=1:graph_count
            try
                close([label_graph ' ' num2str(n)]);
            catch
                % Do nothing: the window was already closed.
            end
        end
        resetFigureVars();
        set (close_btn, 'enable', 'off') ;
end

% Regenerate input values.
function random_callback(~, ~)
        set(input_a, 'string', num2str(rand (1)*2-1,4));
        set(input_b, 'string', num2str(rand (1)*2-1,4));
        set(input_tau, 'string', num2str(rand (1)*5,4));
end

% Clear input values.
function clear_callback(~, ~)
        set(input_a, 'string', '');
        set(input_b, 'string', '');
        set(input_tau, 'string', '');
    end


end
```

## coneSolver.m

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Displays cone and point set according different tau.
% Returns stability interval set.
% Details - stability intervals for each pair of lambda, mu.
%
% Usage: cone(lambda, mu),
%        where lambda, mu are the eigenvalues of matrixes A, B.
% Example:
% cone ([0.06+1.8658i 0.06-1.8658i],[0.025+0.1555i 0.025-0.1555i], 7, 0.5)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
function [result, intervalsCell] = cone(lambda, mu, maxdistanse, resolution)

if exist('n', 'var') == false
    n=100;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Построение конуса.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

tau_cone = 1;
W = pi/tau_cone-0.5; %pi/6;

w=-W:2*W/100:W;         %вектор для реализации основания конуса(круг)
%z=-1/tau_cone:0.1:1.5;            %накопление вектора z
z=-w./tan(w*tau_cone);
[W,Z]=meshgrid(w,z);        %задание 3-х мерной матрицы(пространства)

X=W.*sin(tau_cone*W) - Z.*cos(tau_cone*W); %задание вектора фигуры по оси x
Y=-W.*cos(tau_cone*W) - Z.*sin(tau_cone*W);%задание вектора фигуры по оси y


for ii=n/2+1 : n+1
for j=1 : n+1
Z(ii,j)=1;%tau_cone/(tau_cone);
X(ii,j)=1-z(50);
Y(ii,j)=0;
end
for j=1:1:ii-51
X(ii-50,j)=X(ii-50,ii-50);
X(ii-50,102-j)=X(ii-50,ii-50);
Y(ii-50,j)=0;
Y(ii-50,102-j)=0;
end
end


%отрисовка конуса по 3-м заданным координатам
hold on;
hold on;
shading interp;
colormap(gray);

mesh(X, Y, Z)
xlabel('$u_{1}$','Interpreter','latex','FontSize',30);
ylabel('$u_{2}$','Interpreter','latex','FontSize',30);
zlabel('$u_{3}$','Interpreter','latex','FontSize',30);
```

```
view([-30 30]);
alpha(.2);



%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Построение точек.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Contains all stability intervals.
all = [0 Inf];

intervalsCell = cell(size (lambda) ) ;
for j=1:length(lambda)
    la = lambda(j);
    m = mu(j);

    % Calculation stability intervals.
    intervals = getintervals(lambda(j), mu(j));
intervalsCell{j} = intervals;

all = intersection(all, intervals);


if exist('maxdistanse', 'var') == false
    maxdistanse = 7;
end
if exist('resolution', 'var') == false
    resolution =0.5;
end

    dist_real = distance(m, la, 0, 1) ;

    max_tau = maxdistanse/dist_real;

    dist_real = distance(m, la, max_tau, max_tau-max_tau/10) ;

    quantity = dist_real/resolution*10;

%    max_tau =7;
%    quantity = 20;
    tau = 0.001 : max_tau/quantity : max_tau;


    for jj=1:length(tau)

        x = tau*abs(m).*cos(arg(m) + tau*imag(la));
        y = tau*abs(m).*sin(arg(m) + tau*imag(la));
        z = tau*real (la);
```

```
        if isInside(intervals, tau(jj))
            color = 'b';
        else
            color = 'r';
        end
        text(x(jj), y(jj), z(jj), '*', 'color', color, 'fontsize', 20);
    end
text(x(jj), y(jj), z(jj), num2str(j), 'color', color, 'fontsize', 20);
end

result = all;
```

## distance.m

```
function [distance] = distance(m, la, tau0, tau1)
    [x0, y0, z0] = point(m, la, tau0) ;
    [x1, y1, z1] = point(m, la, tau1) ;
    distance = sqrt ((x1-x0)^2 + (y1-y0)^2 + (z1-z0)^2) ;
end
```

## getIntervals.m

```
% Gets the Stability intervales of the equation.
function result = getIntervals(lambda, mu)

% Simple cheking for stability.
if real(lambda) > abs(mu)
disp('INFO: stability anyway!');
result = [0 Inf];
return;
end

% Perform necessary calculations.
Q = sqrt((abs(mu))^2 - (real(lambda))^2);
M = 15;
m = -M:M;

tau_plus = (arg(1i*Q - real(lambda)) + 2*pi*m - arg(mu)) / ...
(imag(lambda) + Q);
tau_minus = (arg(-1i*Q - real(lambda)) + 2*pi*m - arg(mu)) / ...
(imag(lambda) - Q);


% Check if atau in [0, pi/Q].
baund = pi/Q;
all = [tau_plus, tau_minus]; % concatenate arrays.
k=1;
```

```
for j = 1 : length(all)
atau = all(j);
if atau > 0 && atau <= baund && real(lambda) >= -Q/tan(Q*atau)
result_set(k) = atau;
k = k+1;
end
end

if exist('result_set', 'var') == 0
disp('INFO: empty set!');
result = [];
return;
end

% Check if [0, tau1] is included in the result set.
if (real(lambda) > 0 && mu > -pi/2 && mu < pi/2) % spiral goes up.
result_set = [0 result_set]; % add zero to result set.
elseif (real(lambda) > 0 && (real(lambda)) > abs(real(mu))) % spiral goes up.
result_set = [0 result_set]; % add zero to result set.
elseif (real(lambda) < 0 && abs(real(lambda)/real(mu)) < 1)
result_set = [0 result_set]; % add zero to result set.
end

% Sort the result set.
disp('INFO: Sorted set: ');
result_set = sort(result_set)

% Remove the same values.
if length(result_set) > 1
    cleanSet = result_set;
    if result_set(1) == result_set(2)
        cleanSet = cleanSet(2:end);
    end
    for k = 2:length(result_set)-1
        if result_set(k) == result_set(k+1)
            cleanSet = [cleanSet(1:k-1), cleanSet(k+1:end)];
        end
    end
    result_set = cleanSet;
end

% Check for even.
if mod(length(result_set), 2) == 1
    disp('WARNING: stability interval has the odd lenth!!!');
result_set = [0 result_set];
end

result = result_set;
```

# Приложение В

# Исходный код программы «Построение областей устойчивости круговых нейронных сетей»

## application3.m

```
function application3
debug = 0;



fontsize0 = 12;
fontsize1 = 15;

    figure_width = 600;
    figure_height = 500;
    screen_size = get(0,'ScreenSize');
    screen_width = screen_size(1,3);
    screen_height = screen_size(1,4);
    figure_x = 0;
    figure_y = 0;
    graph_count = 0;
    resetFigureVars();
    image1 = imread('circles_right.png');
    image2 = imread('circles.png');



% Error strings
label_error = 'Ошибка';
err_incorrect_tau = 'Вы ввели некорректное значение tau.';
%err_incorrect_number = 'Вы ввели некорректное максимальное число нейронов. Оно должно быть больше 2.';
%err_incorrect_number = 'You entered incorrect max number of neirons. It should be greater then 2.';
err_incorrect_epsilon = 'You entered incorrect epsilon.';

label_graph = 'График';
appTitle = 'Построение областей устойчивости круговых нейронных сетей';
header = 'Построение областей устойчивости';
header_eq = 'для круговых нейронных сетей';
inputPanelTitle = 'Входные данные';
label_method = 'Конфигурация нейронной сети';
label_method1 = 'Круговое соединение нейронов с малым запаздыванием взаимодействия с правыми соседями';
label_method2 = 'Круговое соединение нейронов с одинаковым запаздываниям взаимодействия с соседями';
```

```
label_model1 = 'Модель сети: x''j(t) + xj(t) + a*xj-1(t) + b*xj+1(t-tau) = 0, 1 <= j <= n';
    label_model2 = 'Модель сети: x''j(t) + xj(t) + a*xj-1(t-tau) + b*xj+1(t-tau) = 0, 1 <= j <= n';
    label_set_tau = 'Величина запаздывания tau = ';
label_number = 'Число нейронов от ';
    label_number1 = 'до ';
    label_path = 'Путь для сохранения: ';
    label_ext = 'Расширение: ';
    label_epsilon = 'Точность: ';



label_calc = 'Рассчитать...';
    label_calc_tip = 'Определяет интервалы устойчивости системы, заданной матрицами A и B';
    label_close = 'Закрыть все графики';
    tooltip_close = 'Закрывает все ранее открытые графики';

% Create a figure that will have a uitable, axes and checkboxes
f = figure('Position', [50, 100, 920, 600],...
'Name', appTitle,...  % Title figure
'NumberTitle', 'off',... % Do not show figure number
'MenuBar', 'none');      % Hide standard menu bar menus

uicontrol('style', 'text', ...
'string', header, ...
'fontsize', fontsize1, ...
'units', 'normalized', ...
'position', [0, .95, 1, .05]);
uicontrol('style', 'text', ...
'string', header_eq, ...
'fontsize', fontsize1, ...
'units', 'normalized', ...
'position', [0, .9, 1, .05]);

rootPanel = uipanel('Parent',f, 'bordertype', 'none', 'Position',[0 0 1 1]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Input panel.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
inputPanel = uipanel('Parent',rootPanel,'Title',inputPanelTitle,...
 'Position',[0 .1 1 .8]);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% input method
group = uibuttongroup('parent', inputPanel, 'Position',[0 .75 1 .25]);
uicontrol(group, 'style', 'text', 'fontsize', fontsize1, 'string', label_method, ...
'units', 'normalized', 'position', [0, 0.7, 1, .3]);
set(group,'SelectionChangeFcn',@sel_callback);
rb1 = uicontrol('parent',group, 'Style','Radio',...
    'fontsize', fontsize0, 'String',label_method1,...
    'units', 'normalized', 'pos',[0, .3, 1,.3]);
```

```
rb2 = uicontrol('parent',group, 'Style','Radio',...
    'fontsize', fontsize0, 'String',label_method2,...
    'units', 'normalized', 'pos',[0, 0, 1,.3]);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% first method
inputPanel1 = uipanel('Parent',inputPanel, 'bordertype', 'none',...
                    'Position',[0 0 1 .65]);
                                                                        if debug == true
                                                                            set (inputPanel1, 'bordertype', 'etchedout
                                                                        end
method_text = uicontrol(inputPanel1, 'style', 'text', 'fontsize', fontsize1, 'string', label_method1, ...
'units', 'normalized', 'position', [0, 0.85, 1, .2]);
model_text = uicontrol(inputPanel1, 'style', 'text', 'fontsize', fontsize0, 'string', label_model1, ...
'units', 'normalized', 'position', [0, 0.7, 1, .1]);

leftPanel1 = uipanel('Parent',inputPanel1, 'bordertype', 'none',...
                    'Position',[0 0 .75 .7]);
                                                                        if debug == true
                                                                            set (leftPanel1, 'bordertype', 'etchedout'
                                                                        end



% tau
uicontrol(leftPanel1, 'style', 'text', 'fontsize', fontsize0, 'string', label_set_tau, ...
    'horizontalAlignment', 'right', 'units', 'normalized', 'position', [0, 0.75, .6, .15]);
input_tau = uicontrol(leftPanel1, 'style', 'edit', ...
    'string', '0.5', 'backgroundcolor', 'w',...
    'units', 'normalized', 'position', [0.6 .77 .15 .15]);


% min max number
uicontrol(leftPanel1, 'style', 'text', 'fontsize', fontsize0, 'string', label_number, ...
    'horizontalAlignment', 'right', 'units', 'normalized', 'position', [0, 0.5, .6, .15]);
input_minnumber = uicontrol(leftPanel1, 'style', 'edit', ...
    'string', '3', 'backgroundcolor', 'w',...
    'units', 'normalized', 'position', [0.6 .52 .15 .15]);
uicontrol(leftPanel1, 'style', 'text', 'fontsize', fontsize0, 'string', label_number1, ...
    'horizontalAlignment', 'center', 'units', 'normalized', 'position', [0.75, 0.5, .1, .15]);

input_number = uicontrol(leftPanel1, 'style', 'edit', ...
    'string', '10', 'backgroundcolor', 'w',...
    'units', 'normalized', 'position', [0.85 .52 .15 .15]);


% path
uicontrol(leftPanel1, 'style', 'text', 'fontsize', fontsize0, 'string', label_path, ...
```

```
    'horizontalAlignment', 'right', 'units', 'normalized', 'position', [0, 0.25, .3, .15]);
uicontrol(leftPanel1, 'style', 'text', 'fontsize', fontsize0, 'string', [pwd '/'], ...
    'horizontalAlignment', 'right', 'units', 'normalized', 'position', [0.3, 0.25, .55, .15]);
input_dir = uicontrol(leftPanel1, 'style', 'edit', ...
    'string', 'graphs', 'backgroundcolor', 'w',...
    'units', 'normalized', 'position', [0.85 .28 .15 .15]);


% extension
uicontrol(leftPanel1, 'style', 'text', 'fontsize', fontsize0, 'string', label_ext, ...
    'horizontalAlignment', 'right', 'units', 'normalized', 'position', [0, 0.1, .3, .15]);
extCombo = uicontrol(leftPanel1, 'style', 'popupmenu', 'string', {'png', 'eps'}, ...
'tooltipString', label_calc_tip,...
'units', 'normalized', 'position', [0.3 .1 .15 .15]);

% epsilon
uicontrol(leftPanel1, 'style', 'text', 'fontsize', fontsize0, 'string', label_epsilon, ...
    'horizontalAlignment', 'right', 'units', 'normalized', 'position', [0.65, 0.1, .2, .15]);
input_epsilon = uicontrol(leftPanel1, 'style', 'edit', ...
    'string', '0.05', 'backgroundcolor', 'w',...
    'units', 'normalized', 'position', [0.85 .12 .15 .15]);




rightPanel1 = uipanel('Parent',inputPanel1, 'bordertype', 'none',...
                    'Position',[.75 0 .25 .7]);

                                                                    if debug == true
                                                                        set (rightPanel1, 'bordertype', 'etchedout
                                                                    end
ax = axes('parent', rightPanel1, 'position', [0 0 1 1]);
image( image1, 'parent', ax);


%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Bottom panel.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
bottomPanel = uipanel('Parent',rootPanel,'bordertype', 'none',...
 'Position',[0 0 1 .1]);
                                                                    if debug == true
                                                                        set (bottomPanel, 'bordertype', 'etchedout
                                                                    end



uicontrol(bottomPanel, 'style', 'pushbutton', 'string', label_calc, ...
'tooltipString', label_calc_tip,...
'units', 'normalized', 'position', [0.75, 0, .25, 1], ...
```

```
'callback', @calc_callback);


%------------------------ Functions ------------------------------------%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
    function [tau, minNumber, maxNumber, epsilon, dir, ext] = getValues()
            tau = str2double(get(input_tau, 'string'));
            minNumber = str2double(get(input_minnumber, 'string'));
            maxNumber = str2double(get(input_number, 'string'));
            dir = get(input_dir, 'string');
            epsilon = str2double(get(input_epsilon, 'string'));

            val = get(extCombo, 'value');
            values = get(extCombo, 'string');
            ext = values(val);

    end


    function sel_callback(source,arg2)
        if  get(source,'SelectedObject') == rb1
            % show method 1
            set (method_text, 'string', label_method1);
            set (model_text, 'string', label_model1);
            image( image1, 'parent', ax);
        else
            % show method 2
            set (method_text, 'string', label_method2);
            set (model_text, 'string', label_model2);
            image( image2, 'parent', ax);
        end
    end


function calc_callback(arg1, arg2)
        [tau, minNumber, maxNumber, epsilon, dir, ext] = getValues();
        if isnan(tau)
            str = err_incorrect_tau;
            errordlg(str, label_error, 'on');
            return;
        end
        if isnan(minNumber) || maxNumber < 3
            str = 'You entered incorrect min number of neirons. It should be greater then 2.';
            errordlg(str, label_error, 'on');
            return;
        end
        if isnan(maxNumber) || maxNumber < 3 || maxNumber < minNumber
            str = 'You entered incorrect max number of neirons. It should be greater then 2 and not less then min number.
            errordlg(str, label_error, 'on');
```

```
        return;
    end
    if isnan(epsilon)
        str = err_incorrect_epsilon;
        errordlg(str, label_error, 'on');
        return;
    end

    if abs(epsilon) < 0.05
        warn_eps = questdlg(['Calculating boundness with little epsion may spend a lot of time on slow machines. ' ..
            'You should try greater value before. Are you sure you want to procesed?'], ...
                    'Perfomance warning', ...
                    'Yes', 'No', 'No');
        switch warn_eps,
            case 'No', return;
        end % switch
    end

    if abs(minNumber - maxNumber) > 7
        warn_eps = questdlg(['Calculating boundness for many numbers of neironous may spend a lot of time on slow mac
            'You should try greater value before. Are you sure you want to procesed?'], ...
                    'Perfomance warning', ...
                    'Yes', 'No', 'No');
        switch warn_eps,
            case 'No', return;
        end % switch
    end



    if  get(group,'SelectedObject') == rb1
        type = 1;
    elseif get(group,'SelectedObject') == rb2
        type = 2;
    end

    mkdir(dir);
    for number=minNumber:maxNumber
        plotter_boundness(type, tau, number, epsilon, dir, ext);
    end

end

function resetFigureVars()
    figure_x = screen_width - figure_width;
    figure_y = screen_height - figure_height - 25;
    graph_count = 0;
end
```

```
end
```

## straightStabAnalizer.m

```
function [stab] = straightStabAnalizer(type, a, b, number, tau)

j = 1:number;

if type == 1
    % первый тип соединения нейронов.
    lambda = 1 + a*exp(1i*2*pi.*j/number);
    mu = b*exp(-1i*2*pi.*j/number);
else
    % второй тип соединения нейронов.
    lambda = ones (size (j));
    mu = a*exp(1i*2*pi.*j/number) + b*exp(-1i*2*pi.*j/number);
end

% вычисление границы w.
PI=2*pi;
w=0:PI/100:PI;

maxW = zeros (1, number) ;
for k=j
    z = tau*real (lambda (k));
    if z < -1
        stab = false;
        return;
    end
    ycone = -z.*sin (w)- w.*cos (w) ;

    jj =1;
    while ycone (jj) <= 0
        jj = jj+1;
    end
    maxW (k) = w(jj) ;
%     if maxW (k) == 0
%         maxW (k) = 3/2*pi;
%     end
end

for k=j
    PI = maxW (k) ;
    w=-PI:PI/100:PI;

    m = mu (k) ;
    la = lambda (k) ;
    x = tau*abs(m).*cos(arg(m) + tau*imag(la));
    y = tau*abs(m).*sin(arg(m) + tau*imag(la));
```

```
    z = tau*real (la);

% ORIGIN
    xcone =  w.*sin (w) - z.*cos (w);
    ycone = -w.*cos (w) - z.*sin (w) ;



    distArr = (x-xcone).^2 + (y-ycone).^2 ;

    minInd1 = 1;
    minInd2 = 2;
    for jj= 2:length (distArr)
        if distArr (jj) < distArr (minInd1)
            minInd2 = minInd1;
            minInd1 = jj ;
        end
    end



    distCone = (xcone(minInd1)-1)^2 + ycone(minInd1)^2;
    distPoint = (x-1)^2 + y^2;

    if distPoint > distCone
        stab = 0;
        return;
    end

end

stab = 1;
end
```

## searchPointInRowGeom.m

```
% Алгоритм:
% 1. Ищем r(2) такое, чтобы отрезок устойчивости по tau был либо пустым, либо
% второе значение было меньше tau.
% 2. Методом половинного деление отрезка [r(1), r(2)] добиваемся заданной
% точности по tau.
function [half, res] = searchPointInRowGeom(type, number, tau, phi, epsilon)
r = [0 2];
% step 1.
res = straightStabAnalizer(type, r(2)*cos(phi), r(2)*sin(phi), number, tau);
while res
    r(2) = r(2) + 1;
    res = straightStabAnalizer(type, r(2)*cos(phi), r(2)*sin(phi), number, tau);
end
% step 2.
```

```
for iter = 1:20
    %disp(' ');
    %disp(['Step ' num2str(iter) ' ======================>'])
    half = sum(r)/2;
    res = straightStabAnalizer(type, half*cos(phi), half*sin(phi), number, tau);

%     if length(res) >= 2 && abs(res(2) - tau) < epsilon
%         break;
%     end
    if res
        r(1) = half;
    else
        r(2) = half;
    end

    if abs(r(1) - r(2)) < epsilon
        %disp (['Stopped on ' num2str(iter)]);
        %return;
    end
end
end
```

## solverBoundnessSmart.m

```
function [phi r] = solver_boundness_smart(type, tau, number, epsilon)

radius = 2*epsilon;
%engl=acos(epsilon^2/2/step^2-1);
%radiusIteration = pi/2/engl;

r = zeros(1, 1000);
phi = zeros(1, 1000);

phi(1) = 0;
[r(1)] = searchPointInRowGeom(type, number, tau, 0, radius);

n=1;
while n<1000
    [r(n+1), phi(n+1)] = searchNeighbourhoodPoint(r(n), phi(n), radius);
    n = n+1;
    if phi(n-1) - phi(n) > 1
        break
    end
end

phi = phi(1:n);
r = r(1:n);
```

```
function [x y] = translate (r, phi, r1, phi1)
    x = r*cos (phi) + r1*cos (phi + phi1);
    y = r*sin(phi) + r1*sin(phi + phi1);
end

    function [r, phi] = toPolar(x, y)
        r = sqrt(x^2 + y^2);
        if x>0
            if y>=0
                phi = atan(y/x);
            else
                phi = atan(y/x) + 2*pi;
            end
        elseif x < 0
            phi = atan(y/x) + pi;
        else %if x == 0
            if y > 0
                phi = pi/2;
            elseif y<0
                phi = 3*pi/2;
            else
                phi = 0;
            end
        end
    end

    function [rEnd phiEnd] = searchNeighbourhoodPoint(r, phi, radius)
        phi0 = 0;
        phi1 = pi;
        for jj=1:10
            phiHalf = (phi1+phi0)/2;
            [xHalf yHalf] = translate(r, phi, radius, phiHalf);

            stabHalf = straightStabAnalizer(type, xHalf, yHalf, number, tau);
            if stabHalf == 1
                phi1 = phiHalf;
            else
                phi0 = phiHalf;
            end
        end
        [rEnd phiEnd] = toPolar(xHalf, yHalf);
    end

end
```

## plotterBoundness.m

```
function plotter_boundness(type, tau, number, epsilon, dir, ext)
    addpath ../common
```

```
    %close all;

    % Create figure
    figure1 = figure('InvertHardcopy','off','Color',[1 1 1], ...
        'position', [680 0 600 570]);

    hold on;
    grid on;


    datafile = [num2str(type) '_' num2str(tau) '.mat'];
    try
        load(datafile);
    catch me
        display(me);
        % supposing that file does not exist.
    end
    if exist('infiniteR', 'var') == 0 || exist('infinitePhi', 'var') == 0
        infiniteNumber = 100;
        [infinitePhi, infiniteR] = solver_boundness_smart(type, tau, infiniteNumber, .02);
        save(datafile, 'infinitePhi', 'infiniteR');
    end
    h = polar(infinitePhi, infiniteR, '-');
    set(h, 'LineWidth', 3, 'color', 'k') ;

    [phi, r] = solver_boundness_smart(type, tau, number, epsilon);
    polar(phi, r, 'o');

    title(['$\tau = ' num2str(tau) ', n = ' num2str(number) '$' ],'FontSize',20, 'interpreter', 'latex');


% Autogenerated code.
% Create xlabel
xlabel({'a'}, 'FontSize',16);

% Create ylabel
ylabel({'b'}, 'FontSize',16);

% Create textbox
annotation(figure1,'textbox',...
    [0.429930957050318 0.48027939504748 0.133224346076459 0.0847826086956548],...
    'String','Stability',...
    'FontSize',16,...
    'FitBoxToText','off',...
    'LineStyle','none',...
    'EdgeColor',[1 1 1]);
```

```
% Create textbox
annotation(figure1,'textbox',...
    [0.19904017857143 0.254566473988439 0.133459821428571 0.0708202587393341],...
    'String',{'Instability'},...
    'FontSize',16,...
    'FitBoxToText','off',...
    'LineStyle','none',...
    'EdgeColor',[1 1 1]);

% Create textbox
annotation(figure1,'textbox',...
    [0.654821428571429 0.783236994219653 0.133459821428571 0.0708202587393341],...
    'String',{'Instability'},...
    'FontSize',16,...
    'FitBoxToText','off',...
    'LineStyle','none',...
    'EdgeColor',[1 1 1]);

% Create textbox
annotation(figure1,'textbox',...
    [0.24802956628661 0.739096207470175 0.133459821428571 0.0708202587393342],...
    'Interpreter','latex',...
    'String',{['$n = ' num2str(number) '$']},...
    'FontSize',16,...
    'FitBoxToText','off',...
    'LineStyle','none',...
    'EdgeColor',[1 1 1]);


% Create textbox
annotation(figure1,'textbox',...
    [0.334445841716969 0.60174631098984 0.133459821428571 0.0708202587393342],...
    'Interpreter','latex',...
    'String',{'$n = \infty$'},...
    'FontSize',16,...
    'FitBoxToText','off',...
    'LineStyle','none',...
    'EdgeColor',[1 1 1]);



    % Scaling
    ax=axis;
    x1=min(ax(1), ax(3));
    x2=max(ax(2), ax(4));
    x1=-1.5;
    x2=1.5;
    axis([x1 x2 x1 x2]);
```

```
    % Boundness
    ax=axis;
    plot(ax(1:2), [ax(4) ax(4)], 'k', 'LineWidth', 1);
    plot([ax(2) ax(2)], ax(3:4), 'k', 'LineWidth', 1);
    axis([x1 x2 x1 x2+.01]);

    if type == 1
        typeName = 'asym';
    else
        typeName = 'sym';
    end

    %mkdir(dir);
    figureName = [typeName '_' num2str(tau) '_' num2str(number)];

    % replace decimal point
    figureName = strrep(figureName, '.', '');

    set(figure1, 'PaperPositionMode', 'auto')   % Use screen size

    if strcmp(ext, 'png')
        saveas(figure1, [dir filesep figureName '.png'], 'png');
    else
        saveas(figure1, [dir filesep figureName '.eps'], 'psc2');
    end


    close;



end
```