

Хохлова Татьяна Наилевна

**Устойчивость моделей нейронных сетей
кольцевой и линейной конфигураций
с запаздывающими взаимодействиями**

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

Содержание

| | |
|--|----|
| Введение | 5 |
| Глава 1 Конус устойчивости для диагностирования устойчивости нейронных сетей. Алгоритмы и программы | 32 |
| 1.1 Нейронные сети и дифференциальные уравнения с запаздываниями | 32 |
| 1.2 Овал устойчивости | 39 |
| 1.3 Конус устойчивости для скалярного уравнения с комплексными коэффициентами | 45 |
| 1.4 Конус устойчивости для матричного уравнения | 49 |
| 1.5 Алгоритм для определения значений запаздывания, гарантирующих устойчивость дифференциального уравнения с запаздыванием | 54 |
| 1.6 Программный продукт «Анализ устойчивости» | 60 |
| 1.7 Сравнение результатов главы 1 с известными результатами . . | 65 |
| Глава 2 Численный и теоретический анализ устойчивости моделей нейронных сетей кольцевой и линейной конфигурации с неограниченным количеством нейронов | 68 |
| 2.1 Постановка задачи и алгоритм диагностирования устойчивости модели кольцевой сети с неограниченным количеством нейронов | 68 |
| 2.2 Программный продукт «Устойчивость нейронных сетей» | 73 |
| 2.3 Результаты исследования устойчивости кольцевой сети нейронов с неограниченным количеством нейронов | 78 |
| 2.4 Модели кольцевых сетей нейронов с неединичным коэффициентом демпфирования | 81 |

| | | |
|-----|---|----|
| 2.5 | Разрыв в кольце: модель сети линейной конфигурации с большим количеством нейронов | 85 |
| 2.6 | Доказательства теорем главы 2 | 88 |
| 2.7 | Сравнение результатов главы 2 с известными результатами . . | 93 |

| | | |
|---|---|------------|
| Глава 3 | Численное и качественное исследование устойчивости моделей нейронных сетей кольцевой и линейной конфигураций с ограниченным количеством нейронов | 95 |
| 3.1 | Алгоритм и программа для построения границ областей устойчивости моделей кольцевых нейронных сетей с ограниченным количеством нейронов | 95 |
| 3.2 | Границы областей устойчивости моделей кольцевых сетей с ограниченным количеством нейронов и односторонним запаздыванием | 103 |
| 3.3 | Границы областей устойчивости моделей кольцевых сетей с ограниченным количеством нейронов и двусторонним запаздыванием | 108 |
| 3.4 | Устойчивость модели нейронной сети линейной конфигурации с ограниченным количеством нейронов | 112 |
| 3.5 | Динамика области устойчивости в процессе разрыва кольцевой сети нейронов | 116 |
| 3.6 | Сравнение результатов главы 3 с известными результатами . . | 118 |
| Заключение | | 121 |
| Приложение А. Исходный код программы «Анализ устойчивости» | | 123 |

| | |
|--|-----|
| Приложение Б. Исходный код программы «Устойчивость нейронных сетей» | 135 |
| Приложение В. Исходный код программы «Построение областей устойчивости круговых нейронных сетей» | 141 |
| Приложение Г. Общие исходные коды программ | 152 |

Введение

Общая характеристика работы

Актуальность работы. Всюду, где в математических моделях имеются узлы и связи между ними, есть основания рассматривать их как нейронные сети. В многочисленных теориях узлы (нейроны) представляют природные объекты [?], блоки компьютерных программ [?], личности [?] и, наконец, собственно нейроны в живых организмах [?] или искусственных нейронных сетях [?]. Взаимодействие узлов в нейронной сети зависит от архитектуры и свойств ее связей. Важной характеристикой сети является запаздывание во взаимодействии нейронов. Первые исследователи нервных систем живых организмов были удивлены, узнав, как мала скорость движения электрохимических импульсов по нервным волокнам. Поэтому учет запаздываний в моделях нейронных сетей требует применения теории дифференциальных уравнений с запаздываниями (функционально-дифференциальных уравнений (ФДУ)). Инструментарий ФДУ создали Н.В. Азбелев, В.П. Максимов и Л.Ф. Рахматуллина [?], Н.В. Азбелев и П.М. Симонов [? ? ?], Р. Беллман и К. Кук [?], А.В. Ким и В.Г. Пименов [?], Н.Н. Красовский [?], В.Б. Колмановский и В.Р. Носов [?], А. Д. Мышкис [?], Дж. Хейл [?], Л.Э. Эльсгольц и С.Б. Норкин [?].

Изучение моделей нейронных сетей посредством дифференциальных уравнений с запаздываниями проведено в монографиях L. O. Chua [?] (1998), L. O. Chua и Т. Roska [?] (2004), К. Gu, V. Kharitonov и J. Chen [?] (2003), J. Wu [?] (2001). Особенно много работ посвящено кольцевым конфигурациям нейронных сетей: S. Guo и L. Huang [? ?] (2007), Y. Horikawa и Н. Kitajima [?] (2009), С. Huang с соавторами [?] (2008), X. Lu и S. Guo [?] (2008), X. Xu [?] (2008). Кольцевые конфигурации нейронов обычны как в искусственных

нейронных сетях [?], так и в биологических. Нейронные кольца обнаружены, например, у нематоды *C. elegans* [?].

Устойчивость нейронных сетей является их важной характеристикой. Глобальная устойчивость изучалась, например, в работах L. Idels и M. Kipnis [?] (2009), Kaslik и Balint [?] (2009), но глобальная устойчивость не всегда желательна в нейронных сетях (например, она неестественна в нейронных сетях, используемых в качестве памяти). В отличие от неё локальная устойчивость, по-видимому, всегда требуется.

Локальная устойчивость моделей нейронных сетей изучалась в работах I. Györi и F. Hartung [?] (2003, нелинейная модель изолированного нейрона), W. Yu и J. Cao [?] (2007, модель системы из двух нейронов), J. Wei и S. Ruan [?] (1999, также из двух нейронов), X. Lu и S. Guo [?] (2008, модель кольцевой сети из четырех нейронов), S.A. Campbell, I. Ncube и J. Wu [?] (2006, модель кольцевой сети из трех нейронов), Y. Yuan и S.A. Campbell [?] (2004, модель кольцевой сети с произвольным количеством нейронов, но с искусственной симметрией в реакции нейронов).

В указанной группе работ нет ответа на естественные вопросы, возникающие при исследовании устойчивости моделей нейронных сетей вообще, а также кольцевых и линейных сетей в частности. Это следующие вопросы. Есть ли значения параметров нейронной сети, при которых сеть остается устойчивой при любом увеличении количества нейронов и сохранении общей архитектуры сети? Каковы эти значения? Каковы значения параметров нейронных сетей, гарантирующих устойчивость сети при любом запаздывании во взаимодействии нейронов (delay-independent stability)? Положительно ли влияет на устойчивость разрыв в кольцевой нейронной сети? Как строить области устойчивости в пространстве параметров? Эти вопросы рассматриваются в настоящей диссертации.

Цель диссертационной работы. Целью работы является изучение проблемы устойчивости математических моделей кольцевых и линейных нейронных сетей. Мы намерены:

- разработать метод построения областей устойчивости в пространстве параметров указанных моделей;
- выявить динамику областей устойчивости при изменении количества нейронов в сети и изменении запаздывания во взаимодействии нейронов;
- найти области устойчивости в пространстве параметров, гарантирующие устойчивость независимо от величины запаздывания;
- выяснить асимптотику поведения областей устойчивости при запаздывании, стремящемся к нулю и бесконечности;
- указать предельные области устойчивости, когда количество нейронов в кольцевой или линейной конфигурации неограничено;
- сравнить области устойчивости моделей кольцевой сети и линейной сети, полученной в результате ее разрыва;
- провести численное моделирование динамики области устойчивости в процессе разрыва нейронного кольца и превращения его в сеть линейной конфигурации.

Методы исследования. Поставленные задачи решаются в диссертации методом конуса устойчивости, разработанным автором совместно с научным руководителем и В.В. Малыгиной. Конус устойчивости это поверхность в \mathbb{R}^3 , построенная для анализа устойчивости систем линейных матричных дифференциальных уравнений произвольного порядка с запаздыванием. На основе метода в диссертации построены алгоритмы для поиска значений запаздываний, гарантирующих устойчивость системы. В свою очередь, алгоритмы реализованы в виде программ для анализа устойчивости как для общих систем, так и для специальных систем, описывающих модели кольцевых и линейных нейронных сетей с запаздываниями во взаимодействии соседних нейронов.

Научная новизна. В диссертации разработан новый метод анализа устойчивости, применимый к классу матричных дифференциальных уравнений, более широкому в сравнении с классами, рассмотренными в работах В. Cahlon и D. Schmidt [?] (2000), а также Н. Matsunaga [?] (2007) и S. Sakata [?] (1998). На основе этого метода разработаны новые алгоритмы и комплексы программ для построения области устойчивости в пространстве параметров указанного класса уравнений. Построены модификации алгоритмов и программ для анализа устойчивости математических моделей кольцевых и линейных нейронных сетей. Впервые указаны области в пространстве параметров указанных моделей, гарантирующие устойчивость независимо от величины запаздывания во взаимодействии нейронов. Получены новые данные об областях устойчивости в пространстве параметров математических моделей кольцевых и линейных нейронных сетей, включая класс сетей с неограниченным количеством нейронов. Поставлен и решен новый вопрос о влиянии разрыва на устойчивость кольцевой нейронной сети. Впервые изучена динамика области устойчивости в процессе разрыва кольцевой нейронной сети.

Практическая значимость. Созданные программные продукты и исследования областей устойчивости позволяют анализировать устойчивость нейронных сетей кольцевой и линейной конфигураций, выявлять диапазоны запаздываний, в которых они приобретают и теряют устойчивость, регулировать коэффициенты моделей нейронных сетей с целью стабилизации их работы.

Апробация работы. Основные результаты, изложенные в диссертации, докладывались на седьмой Всероссийской конференции с международным участием «Математическое моделирование и краевые задачи» (Самара, 2010г.), второй и четвёртой научных конференциях аспирантов и докторантов (Челябинск, 2010г. и 2012г.), Всероссийской конференции «Статистика. Моделирование. Оптимизация» (Челябинск, 2011г.), на международ-

ной конференции в Вене «ICNPAA 2012 World Congress: 9th International Conference on Mathematical Problems in Engineering, Aerospace and Sciences» (Vienna, Austria, 2012), II Международной научно-практической конференции студентов и аспирантов «Математика и её приложения в современной науке и практике» (Курск, 2012г.), Всероссийской научно-практической конференции «Физико-математические науки и образование» (Магнитогорск, 2012 г.), на семинаре профессора М. М. Кипниса в Челябинском государственном педагогическом университете.

Личный вклад автора. Все представленные в диссертации результаты получены лично автором. В совместные работы [? ?] автору принадлежат все конкретные результаты, а научному руководителю и В. В. Малыгиной — общий замысел работы, постановка задачи и общее руководство. В работе [?] алгоритмы и программы принадлежат автору диссертации, соавтор А. Хохлов осуществлял техническую поддержку работы.

Публикации. Материалы диссертации опубликованы в 13 печатных работах, из них 4 статьи в рецензируемых журналах [? ? ? ?], 6 статей в сборниках трудов конференций [? ? ? ? ? ?], и 3 комплекса программ, зарегистрированных в Объединенном фонде электронных ресурсов «Наука и образование» [? ? ?] (программы доступны в интернете [? ? ?]).

Структура и объём диссертации. Диссертация состоит из введения, трёх глав, списка литературы и четырёх приложений. Общий объём работы 167 страниц. Работа содержит 36 рисунков, список литературы содержит 82 наименования.

Содержание диссертации

Во **введении** приводится постановка задачи, история исследуемого вопроса, ставятся цели диссертационного исследования, обосновывается актуальность и научная новизна работы. Кроме того, приводятся публикации по теме диссертации, описывается структура работы и ее краткое содержание.

Первая глава посвящена описанию и обоснованию метода конусов устойчивости для диагностирования устойчивости уравнения

$$\dot{x}(t) + Ax(t) + Bx(t - \tau) = 0, \quad (1)$$

с совместно триангулируемыми матрицами A, B . Метод разработан автором диссертации под руководством научного руководителя по замыслу В. Малыгиной. Как обычно, мы называем линейное уравнение устойчивым, если его нулевое решение устойчиво. В этой же главе излагается алгоритм исследования устойчивости уравнения (1) и описывается программа для его реализации.

В разделе 1.1 приведены актуальные на сегодняшний день модели, описывающие взаимодействие нейронов в нейронных сетях с помощью матричных дифференциальных уравнений, обосновано введение запаздывания и показано, что во многих случаях исследование устойчивости различных моделей нейронных сетей сводится к изучению поведения решений уравнения (1). Также в этом разделе приведены примеры нейронных сетей, описываемых уравнением (1), отмечены особенности входящих в уравнение матриц A, B , элементами которых служат интенсивности мгновенного взаимодействия нейронов в сети и взаимодействия, происходящего с запаздыванием, соответственно.

В этой главе конус устойчивости вводится постепенно. Вначале (раздел 1.2) дается определение устойчивости и вводятся овалы устойчивости

для скалярного уравнения вида (1)

$$\dot{x}(t) + ax(t) + bx(t - \tau) = 0, \quad \tau > 0. \quad (2)$$

с действительным коэффициентом a и комплексным b . Здесь же формулируется и с помощью метода D -разбиений [?] доказывается теорема об устойчивости такого уравнения в терминах овала устойчивости.

Затем, в разделе 1.3, дается ключевое определение конуса устойчивости для скалярного уравнения (2) с комплексными коэффициентами a, b , которое будет использоваться также и для матричного уравнения (1).

Определение 0.1. *Конусом устойчивости для уравнений (2), (1) назовём множество точек $M = (u_1, u_2, u_3) \in \mathbb{R}^3$, таких, что*

$$\begin{cases} u_1 = -h \cos \omega + \omega \sin \omega, \\ u_2 = h \sin \omega + \omega \cos \omega, \\ u_3 = h, \end{cases} \quad (3)$$

где действительные параметры h, ω подчинены ограничениям

$$\begin{cases} h \geq -\frac{\omega}{\operatorname{tg} \omega}, \\ -\pi < \omega < \pi. \end{cases} \quad (4)$$

Овалами устойчивости в диссертации называются сечения конуса устойчивости плоскостью $u_3 = h$. В терминах конуса устойчивости формулируются и доказываются теоремы об устойчивости уравнения (2) с действительным коэффициентом a и комплексным b , а затем с двумя комплексными коэффициентами a, b .

В основном разделе 1.4 доказывается теорема о конусе устойчивости для матричного уравнения (1), которая будет теоретической основой для алгоритмов диагностирования устойчивости нейронных сетей и их программных реализаций.

Теорема 0.1. 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 λ_{js}, μ_{js} ($1 \leq j, s \leq m$). Построим систему точек $M_j = (u_{1j}, u_{2j}, u_{3j})$, ($1 \leq j \leq m$), так, что

$$\begin{aligned} u_{1j} &= \tau \operatorname{Re}(\mu_{jj} \exp(i\tau \operatorname{Im} \lambda_{jj})), \\ u_{2j} &= \tau \operatorname{Im}(\mu_{jj} \exp(i\tau \operatorname{Im} \lambda_{jj})), \\ u_{3j} &= \tau \operatorname{Re} \lambda_{jj}. \end{aligned} \tag{5}$$

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

Эта теорема является главным результатом первой главы и дает возможность исследования устойчивости широкого класса уравнений вида (1) с совместно триангулируемыми матрицами произвольного порядка.

В этом же разделе получено необходимое и достаточное условие на собственные числа матриц A, B , обеспечивающее устойчивость основного уравнения (1) при всех значениях запаздывания:

Теорема 0.2. 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 λ_{js}, μ_{js} ($1 \leq j, s \leq m$). Для того, чтобы уравнение (1) было асимптотически устойчивым при любом запаздывании $\tau \geq 0$, необходимо и достаточно выполнение условия ($1 \leq j \leq m$)

$$|\mu_{jj}| < \operatorname{Re} \lambda_{jj}. \tag{6}$$

В разделах 1.3, 1.4 также приведены примеры применения полученных критериев для анализа устойчивости уравнений (2), (1). Примеры снабжены

графическим представлением конуса устойчивости и точек M_j , фигурирующих в формулировке основной теоремы 0.1.

Раздел 1.5 посвящен алгоритму определения значений запаздываний, гарантирующих устойчивость уравнения (1) при фиксированных матрицах A и B , и его программной реализации. Алгоритм основан на методе конусов устойчивости (теорема 0.1), но при этом он содержит только аналитические выкладки и позволяет избежать геометрических построений при анализе устойчивости рассматриваемого уравнения. В этом же разделе формулируется и доказывается теорема, обосновывающая данный алгоритм.

В разделе 1.6 описана программа «Анализ устойчивости», разработанная в программном пакете MATLAB 7.11.0 (R2010b). Программа реализует алгоритм, описанный в разделе 1.5. По заданным матрицам A, B программа возвращает объединение интервалов значений запаздывания T , обеспечивающих устойчивость уравнения (1), или пустое множество в случае неустойчивости уравнения при всех значениях запаздывания. Здесь же приводятся скриншоты интерфейса программы (Рис. 1), примеры двух способов введения исходных данных, необходимые пояснения к работе программы.

В разделе 1.7 результаты первой главы сравниваются с известными результатами. Указано, что идейным источником метода конуса устойчивости является давняя работа З. Рехлицкого [?] (1956). Упомянуты работы I. Levitskaya [?] (2006), E. Kaslik [?] (2009), M.M. Kipnis, V.V. Malygina [?] (2011), S.A. Ivanov, M.M.Kipnis, V.V. Malygina [?] (2011) и S.A. Ivanov, M.M.Kipnis [?] (2012), в которых овалы и конусы устойчивости изучены для разностных матричных уравнений. Указано, что А.И. Кирьянен и К.В. Галунова [?] (1989) изучали устойчивость скалярных дифференциальных уравнений с запаздываниями с комплексными коэффициентами, но овалы или конусы устойчивости в их работе не появлялись.

Анализ устойчивости

Анализ устойчивости уравнения

$x'(t) + Ax(t) + Bx(t-\tau) = 0$

Входные данные

Размерность = 3

1 2 3 4 5

Выберите способ задания матриц A и B

☐ Задать матрицы A и B с помощью базовой матрицы D

☒ Задать матрицы A и B с помощью собственных чисел

Собственные числа матрицы A

lambda1 = 0.0508 + -1.6219 * i

lambda2 = 0.0508 + 1.6219 * i

lambda3 = -0.0153 + 0 * i

Собственные числа матрицы B

mu1 = 0.0903 + -0.1592 * i

mu2 = 0.0903 + 0.1592 * i

mu3 = 0.0716 + 0 * i

Результаты вычислений

(0, 0.44345) U (2.3348, 3.9385) U (6.6798, 7.4336)

(0, 0.44345) U (2.3348, 3.9385) U (6.6798, 7.4336)

(0, 19.3784)

Уравнение устойчиво при tau, лежащем в следующем объединении интервалов:

(0, 0.44345) U (2.3348, 3.9385) U (6.6798, 7.4336)

Очистить Заполнить Рассчитать... Закрыть все графики

Рис. 1. Главное окно программы «Анализ устойчивости» при задании собственных чисел матриц вручную.

Далее в разделе 1.7 результаты главы 1 сравниваются с работой В. Sahlon, D. Schmidt [?] (2000), в которой рассматривалась задача об устойчивости класса уравнений вида $\dot{x}(t) = \alpha Ax(t) + (1 - \alpha)Ax(t - \tau b)$, $0 \leq \alpha \leq 1$ с 2×2 матрицей A . Годом позднее те же авторы [?] (2001) усилили свой результат, рассматривая уравнение $\dot{x}(t) = \alpha Ax(t) + \beta Ax(t - \tau)$ с произвольными действительными α, β , и снова для 2×2 матрицы A . Поскольку матрицы αA и βA с действительными α и β , очевидно, приводятся совместно к треугольному виду, делается заключение, что результаты главы 1 диссертации сильнее результатов этих статей как по размерности рассматриваемых задач, так и по охвату изучаемых уравнений, даже если ограничиться 2×2 матрицами.

Обсуждены результаты Н. Matsunaga [?] (2007), давшего критерий устойчивости уравнения $\dot{x}(t) = Ax(t - \tau)$ с 2×2 матрицей A в терминах следа и детерминанта матрицы A . Показано, что эти результаты легко вытекают из результатов диссертации и даже из результатов Рехлицкого [?] 50-летней давности.

Констатируется, что метод конуса устойчивости применим к значительно более широкому классу уравнений, чем рассматриваемые в работе S. Sakata [?] (1998) уравнения вида $\dot{x}(t) = \alpha x(t) + Bx(t - \tau)$ с 2×2 матрицей B и действительным α .

В этом разделе обсуждены условия, данные в работе J. Chen, H. Latchman [?] (1995), гарантирующие устойчивость уравнения (1) независимо от запаздывания (delay-independent stability). Условие Чена-Латчмена в терминах теоремы 0.2 таково: для всех $j = 1, 2 \dots n$ верны неравенства $\operatorname{Re} \lambda_{jj} > 0$ и $\max_{s \in \mathbb{R}} |(is - \lambda_{jj})^{-1} \mu_{jj}| < 1$. Мы заключаем, что наше условие (6) по существу совпадает с условием Чена-Латчмана, но проще проверяется и имеет естественное геометрическое объяснение.

Далее констатируется, что алгоритм и программа разделов 1.5, 1.6 не имеют аналогов в литературе.

Во второй главе решается задача диагностирования устойчивости моделей кольцевых и линейных нейронных сетей с неограниченным количеством нейронов с помощью результатов первой главы диссертации.

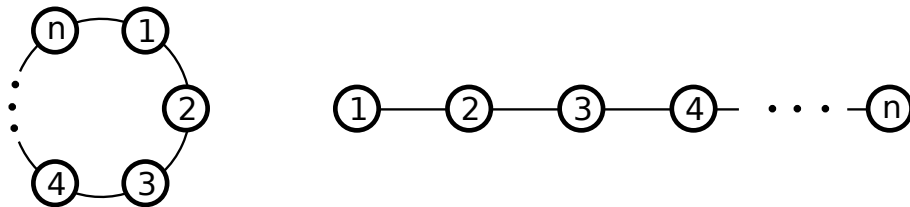


Рис. 2. Кольцевая и линейная система нейронов

В разделе 2.1 вводятся две модели кольцевых нейронных сетей, являющиеся основным объектом изучения второй главы.

$$\dot{x}_j(t) + x_j(t) + a x_{j-1}(t) + b x_{j+1}(t - \tau) = 0 \quad (j \bmod n), \quad (7)$$

$$\dot{x}_j(t) + x_j(t) + a x_{j-1}(t - \tau) + b x_{j+1}(t - \tau) = 0 \quad (j \bmod n). \quad (8)$$

В уравнениях (7), (8) n количество нейронов в кольце, действительные коэффициенты a и b характеризуют интенсивности взаимодействия нейрона с правым и левым соседними нейронами соответственно, τ запаздывание во взаимодействии нейрона с соседними нейронами. Уравнения (7), (8) вместе с вышеуказанными интерпретациями параметров a, b, τ, n мы называем моделями кольцевой сети, имея в виду, что они получаются в результате линеаризации известных нелинейных моделей Хопфилда-Маркуса-Вестервельта [? ?] (1984, 1989) или Коэна-Гроссберга [?] (1983) или Чуа-Янга-Роска [? ?] (1988, 2004) вокруг некоторого решения. Уравнение (7) соответствует малым запаздываниям взаимодействия нейронов с правыми соседними нейронами (сети с односторонним запаздыванием), а (8) — близким запаздываниям взаимодействия нейронов с правыми и левыми соседями (сети с двусторонним запаздыванием). Данные уравнения принадлежат классу систем вида (1) и могут быть исследованы с помощью разработанных автором методов.

В этом же разделе описана модификация алгоритма диагностирования устойчивости уравнений вида (1) для случая специфических матриц, входящих в модели кольцевых сетей, и неограниченного количества нейронов в сети. Суть изменений заключается в явном нахождении собственных чисел входящих в уравнение матриц и введении вместо дискретной системы точек M_j (5) непрерывной замкнутой кривой $M'(t) = (u'_1(t), u'_2(t), u'_3(t))$ для уравнения

(7):

$$u_1'(t) + iu_2'(t) = \tau b \exp(i(-t + a \tau \sin t)),$$

$$u_3'(t) = \tau(1 + a \cos t), \quad 0 \leq t \leq 2\pi. \quad (9)$$

и $M''(t) = (u_1''(t), u_2''(t), u_3''(t))$ для уравнения (8):

$$u_1''(t) + iu_2''(t) = \tau(a \exp(it) + b \exp(-it)), \quad u_3''(t) = \tau, \quad 0 \leq t \leq 2\pi. \quad (10)$$

Для рассмотренных систем критерий устойчивости принимает следующий вид. Если все точки кривой (9) лежат внутри конуса устойчивости, то уравнение (7) устойчиво при любом n , а если хотя бы одна точка кривой (9) лежит вне конуса устойчивости, то уравнение (7) неустойчиво при всех достаточно больших значениях n . Аналогично изучается поведение системы (8). Идея критерия проиллюстрирована рисунком 3.

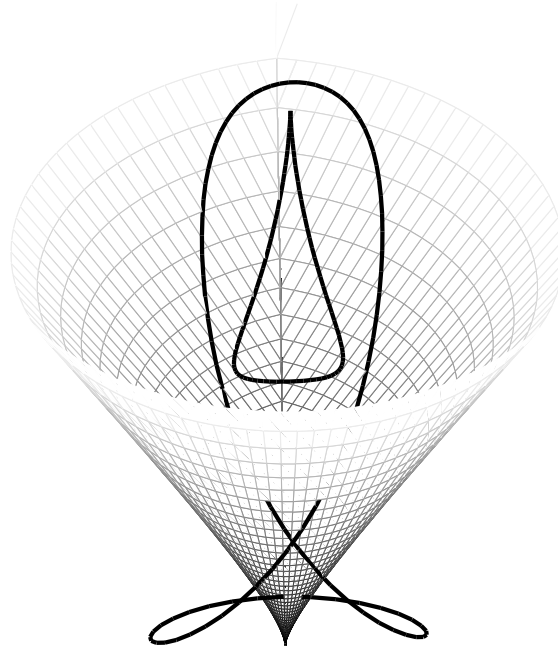


Рис. 3. Конус устойчивости и две кривые (9). Одна кривая для $\tau = 1.5$, $a = -1.4$, $b = 0.7$, находится частично вне конуса устойчивости, следовательно, система (7) неустойчива при достаточно больших n . Вторая кривая для $\tau = 2$, $a = 0.5$, $b = -0.2$, находится полностью внутри конуса устойчивости, следовательно, система (7) устойчива при любом n .

В разделе 2.2 описан программный продукт «Устойчивость нейронных сетей», предназначенный для исследования устойчивости систем (7), (8) с неограниченным количеством нейронов. Продукт разработан автором на основе модификации алгоритма, описанной в разделе 2.1, и позволяет быстро по введенным коэффициентам диагностировать устойчивость конкретной модели нейронной сети и получить геометрическую интерпретацию результатов. В этом разделе представлены скриншоты интерфейса программы с примерами ее работы, необходимые пояснения.

В разделе 2.3 изложены результаты теоретического и численного исследования границ областей устойчивости уравнений (7), (8) в плоскости параметров (a, b) в случае неограниченного количества нейронов.

Теорема 0.3. *Если $|a + b| > 1$, то системы (7) и (8) неустойчивы при любом $\tau > 0$, если n достаточно велико.*

При помощи программы «Устойчивость нейронных сетей» для различных значений запаздывания построены границы областей устойчивости D_τ уравнений (7), (8) в тех частях плоскости (a, b) , которые не охватываются теоремой 0.3 и ранее известным фактом об устойчивости исследуемых моделей при $|a| + |b| < 1$ для любого n и любого $\tau \geq 0$. Полученные результаты для системы (7) отражены на рисунке 4. Аналогичные графические иллюстрации даны для системы (8).

Для обеих систем (7) и (8) важна прямая $a = -b$ в плоскости (a, b) , в окрестности которой сконцентрированы точки устойчивости систем. Поэтому естественно вводятся следующие две системы уравнений:

$$\dot{x}_j(t) + x_j(t) + a(x_{j-1}(t) - x_{j+1}(t - \tau)) = 0 \quad (j \bmod n), \quad (11)$$

$$\dot{x}_j(t) + x_j(t) + a(x_{j-1}(t - \tau) - x_{j+1}(t - \tau)) = 0 \quad (j \bmod n). \quad (12)$$

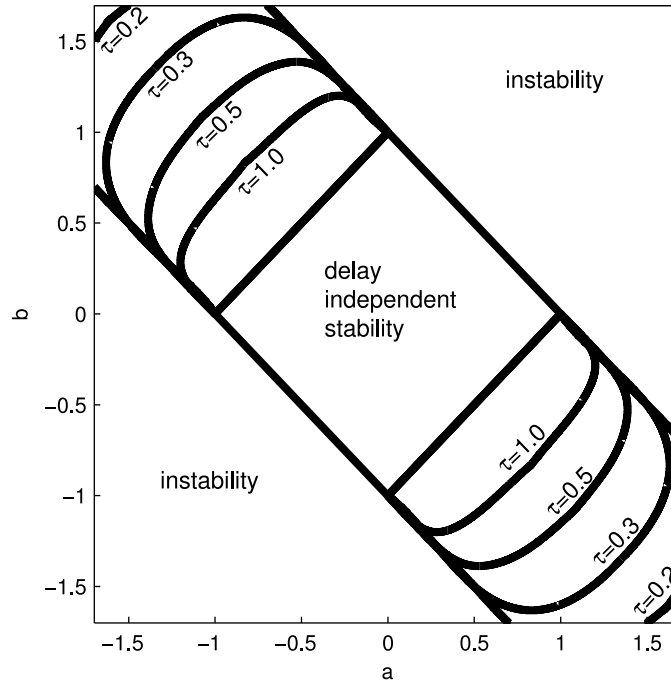


Рис. 4. Области устойчивости для системы (7) с неограниченным числом нейронов.

Определение 0.2. Границей устойчивости системы (2.17) для больших n назовём такое число $a_1(\tau) \in \mathbb{R}$, что если $|a| < a_1(\tau)$, то (11) устойчива при любом n , а если $|a| > a_1(\tau)$, то (11) неустойчива при всех достаточно больших n . Аналогично определим $a_2(\tau)$ как границу устойчивости (12) для больших n .

Очевидно, $\lim_{\tau \rightarrow \infty} a_1(\tau) = \lim_{\tau \rightarrow \infty} a_2(\tau) = 1/2$. Не столь очевидно поведение систем (2.17), (2.18) при $\tau \rightarrow 0$, которое рассматривается в следующей теореме.

Теорема 0.4.

$$\lim_{\tau \rightarrow 0} a_1(\tau) \sqrt{2\tau} = \lim_{\tau \rightarrow 0} a_2(\tau) 2\sqrt{\tau} = 1. \quad (13)$$

В разделе 2.4 рассматривается вопрос устойчивости моделей кольцевых нейронных сетей с неединичным коэффициентом демпфирования на примере аналога системы (7) с $\gamma \in \mathbb{R}$:

$$\dot{x}_j(t) + \gamma x_j(t) + a x_{j-1}(t) + b x_{j+1}(t - \tau) = 0 \quad (j \bmod n). \quad (14)$$

Указаны замена переменных, сводящая это уравнение к уравнению вида (7), и способы диагностирования устойчивости уравнения (14) при ограниченном и неограниченном количестве нейронов. Приведены границы областей устойчивости уравнений (14), (7) в плоскости параметров (a, b) , структура которых не содержит принципиальных различий. Данный раздел призван продемонстрировать целесообразность рассмотрения именно уравнения (7) с единичным коэффициентом при $x_j(t)$ и возможность применения полученных для него результатов к некоторым более общим уравнениям.

Раздел 2.5 посвящен устойчивости моделей нейронных сетей линейной конфигурации, взаимодействие в которых описывается уравнением (1) с матрицами специального вида.

Основным объектом изучения данного раздела является уравнение

$$\dot{x}(t) + I x(t) + D x(t - \tau) = 0, \quad (15)$$

где I есть единичная $n \times n$ матрица, а $n \times n$ матрица D имеет вид

$$D = \begin{pmatrix} 0 & b & 0 & \dots & 0 & 0 \\ a & 0 & b & \dots & 0 & 0 \\ 0 & a & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & b \\ 0 & 0 & 0 & \dots & a & 0 \end{pmatrix}. \quad (16)$$

Уравнение (15),(16) моделирует линейную нейронную сеть с двусторонним запаздыванием, полученную из кольцевой (8) посредством разрыва одной из связей между нейронами. Следующий результат дает возможность отвечать на вопрос об устойчивости системы (15), (16) при любых a, b , если число нейронов n достаточно велико. Пусть функция $F(\tau)$ от запаздывания

$\tau \in (0, \infty)$ определяется следующим образом:

$$F(\tau) = \frac{1}{4 \sin^2 \omega(\tau)}, \quad (17)$$

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

$$\tau = \omega \operatorname{tg} \omega. \quad (18)$$

Тогда справедлива теорема.

- Теорема 0.5.** 1. Если $0 \leq ab < \frac{1}{4}$, то система (15), (16) асимптотически устойчива при любом n и любом $\tau \geq 0$.
2. Если $ab > \frac{1}{4}$, то система (15), (16) неустойчива при любом $\tau \geq 0$, если n достаточно велико.
3. Если $ab < 0$ и $|ab| < F(\tau)$, то система (15), (16) асимптотически устойчива при любом n .
4. Если $ab < 0$ и $|ab| > F(\tau)$, то система (15), (16) неустойчива, если n достаточно велико.

В разделе имеется рисунок, на котором в плоскости (a, b) изображены границы областей устойчивости системы (15), (16), полученные с помощью теоремы (0.5) и численных экспериментов в программе «Устойчивость нейронных сетей». Вычисления потребовались для построения границ, зависящих от значения запаздывания τ . Следующая теорема сравнивает области устойчивости модели кольцевой сети нейронов (8) с достаточно большим количеством нейронов и образованной при ее разрыве нейронной сети линейной конфигурации (15), (16).

Предварительно заметим, что система (8) в матричном виде имеет вид

$$\dot{x}(t) + I x(t) + B x(t - \tau) = 0, \quad t > 0, \quad (19)$$

где циркулянтная матрица B такова:

$$B = \begin{pmatrix} 0 & b & 0 & \dots & 0 & a \\ a & 0 & b & \dots & 0 & 0 \\ 0 & a & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & b \\ b & 0 & 0 & \dots & a & 0 \end{pmatrix}. \quad (20)$$

Теорема 0.6. *Для любых действительных значений параметров $a, b, \tau > 0$ найдется такое n_0 , что для всех $n > n_0$ верно утверждение: либо система (19), (20) для кольца нейронов неустойчива, либо система (15), (16) для сети с линейной конфигурацией нейронов асимптотически устойчива.*

Из Теоремы 0.6 следует, что устойчивость кольца с большим количеством нейронов влечет устойчивость линейной конфигурации, полученной при его разрыве, если параметры систем не меняются и количество нейронов достаточно велико.

Теорема 0.6 дает твердые основания к заключению, что разрыв кольца нейронов расширяет область устойчивости сети при большом количестве нейронов в сети.

Раздел 2.6 содержит доказательства теорем второй главы. В частности, в доказательствах используется тот факт, что при $a = b = 1$ характеристический полином матрицы (16) является полиномом Чебышева второго рода, растянутым на отрезок $[-2, 2]$.

В разделе 2.7 приведено сравнение результатов данной главы с известными результатами. Отмечено, что, насколько известно автору, модели кольцевых нейронных сетей с неопределенно большим количеством нейронов не рассматривались в литературе до работы [?] автора диссертации совместно с научным руководителем. Показано, что из работ Y. Yuan, S. Campbell [?]]

(2004) и S. Campbell с соавторами [?] (2004) невозможно извлечь информацию об устойчивости кольцевых сетей при достаточно большом количестве нейронов в сети. Мы констатируем, что предпринятое во второй главе изучение устойчивости моделей нейронных сетей линейной конфигурации не встречается в литературе (кроме работ автора диссертации), хотя вообще публикации по линейным нейронным сетям имеются (например, [?]). Установлено, что область устойчивости линейной конфигурации нейронов, независимой от запаздывания, очерченная пунктами 1,3 Теоремы 0.5, шире области, гарантированной результатом Мори с соавторами [?] (1981). Указано, что алгоритмы и программы для анализа устойчивости моделей кольцевых и линейных сетей с неограниченным количеством нейронов в известной автору литературе отсутствуют.

Третья глава посвящена построению границ областей устойчивости моделей кольцевых и линейных нейронных сетей с ограниченным количеством нейронов в плоскости параметров (a, b) .

Основными объектами изучения данной главы служат уравнения кольцевых нейронных сетей (7) и (8) и линейных нейронных сетей (2.27), (2.28).

Раздел 3.1 посвящен описанию алгоритма и программы построения границ областей устойчивости исследуемых уравнений. Подраздел 3.1.1 содержит общую схему построения замкнутой кривой, состоящей из равномерного массива точек и соответствующей границе области устойчивости круговых нейронных сетей (7) и (8). Для осуществления этой схемы требуется диагностировать большое количество точек плоскости (a, b) на устойчивость при фиксированном значении запаздывания. В подразделе 3.1.2 указан новый алгоритм, предназначенный для этой цели, а также подробно описана работа реализующей его функции *straightStabAnalyzer* (код функции помещен в приложение В). Алгоритм решает частную задачу определения устойчивости системы при данных a, b, τ в отличие от более общей задачи определе-

ния всех значений запаздывания, гарантирующих устойчивость исследуемого уравнения при фиксированных коэффициентах, решенной в разделах 1.5 и 1.6. В связи с этим данный алгоритм выигрывает во времени и хорошо подходит для многократного применения в общей схеме построения границы области устойчивости, изложенной в подразделе 3.1.1. В подразделе 3.1.3 указаны функциональное назначение и область применения программы «Построение областей устойчивости круговых нейронных сетей», реализующей алгоритмы подразделов 3.1.1, 3.1.2. Примеры использования программы, скриншоты ее работы приведены в подразделе 3.1.4.

В разделе 3.2 представлены результаты построения границ областей устойчивости модели кольцевой нейронной сети (7), полученные с помощью данной программы. Приведенные серии графиков соответствуют различным значениям запаздывания и различному количеству нейронов в сети. На графиках отмечена граница области устойчивости, гарантированной для любого n , что позволяет проследить асимптотику найденных кривых с увеличением количества нейронов в сети. Одна из серий графиков представлена на рисунке 5.

В разделе 3.3 аналогичные построения проведены для модели (8).

В разделе 3.4 дается полное описание области устойчивости для модели (15), (16) линейной системы из n нейронов. Эти области, естественно, зависят от n , в то время как область устойчивости модели линейной сети с неограниченным количеством нейронов в разделе 2.5 не зависела от величины n . Определим функцию $F1(\tau, n)$ (ср. с формулой (17)) от запаздывания $\tau \in (0, \infty)$ и количества нейронов в сети $n \in \mathbb{N}$:

$$F1(\tau, n) = \frac{1}{4 \sin^2 \omega(\tau) \cos^2 \frac{\pi}{n+1}}, \quad (21)$$

где $\omega(\tau)$ есть наименьший положительный корень уравнения (18).

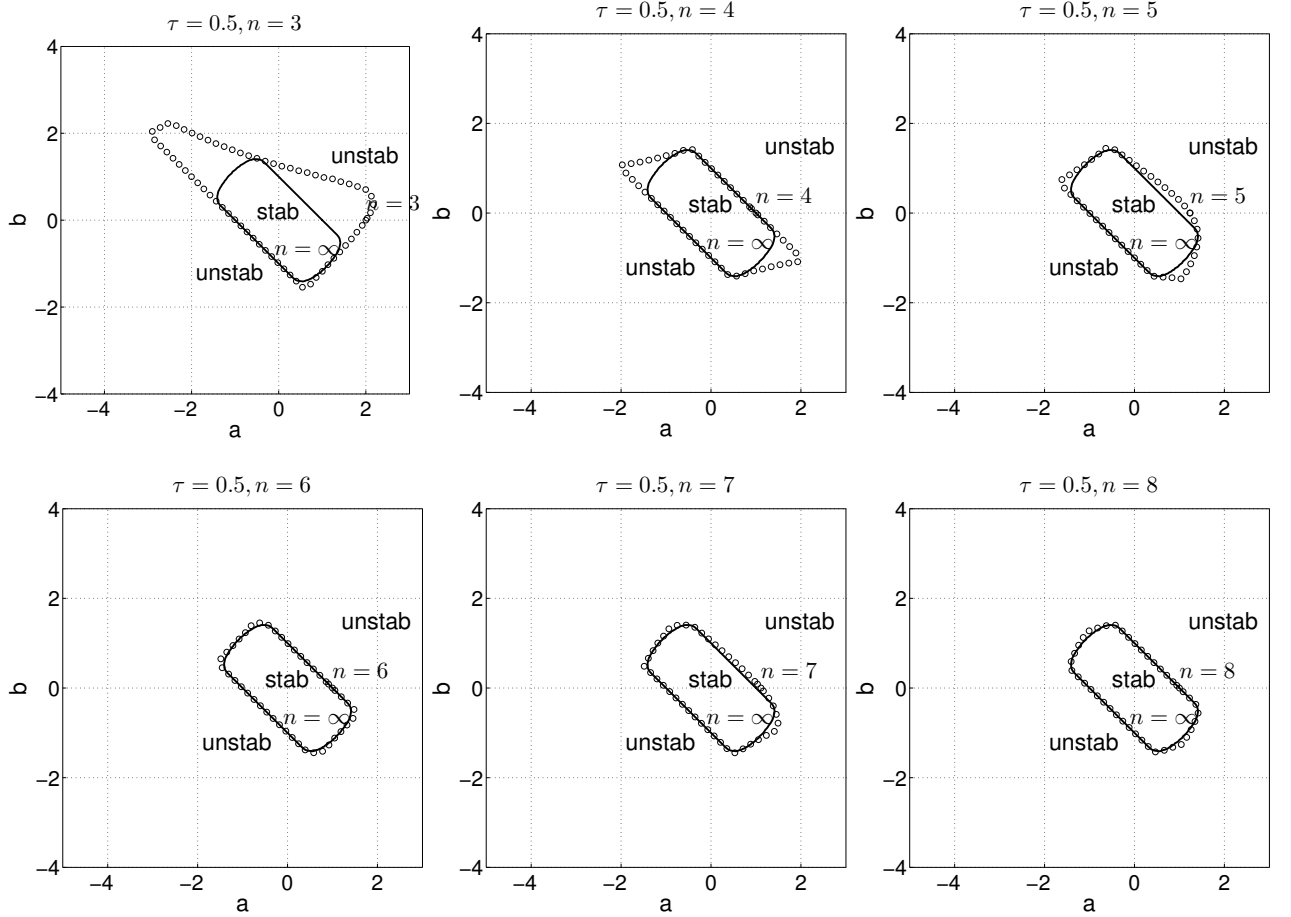


Рис. 5. Границы областей устойчивости для системы (7) для $\tau = 0.5$ и значений n от 3 до 8 показаны кружками. Сплошная линия — граница области устойчивости, гарантированной для любого n .

Следующая теорема (ср. с Теоремой 0.5) дает полное описание области устойчивости модели линейной конфигурации n нейронов.

- Теорема 0.7.** 1. Если $0 \leq ab < \frac{1}{4 \cos^2 \frac{\pi}{n+1}}$, то система (15), (16) асимптотически устойчива при любом $\tau \geq 0$.
2. Если $ab > \frac{1}{4 \cos^2 \frac{\pi}{n+1}}$, то система (15), (16) неустойчива при любом $\tau \geq 0$.
3. Если $ab < 0$ и $|ab| < F1(\tau, n)$, то система (15), (16) асимптотически устойчива.
4. Если $ab < 0$ и $|ab| > F1(\tau, n)$, то система (15), (16) неустойчива.

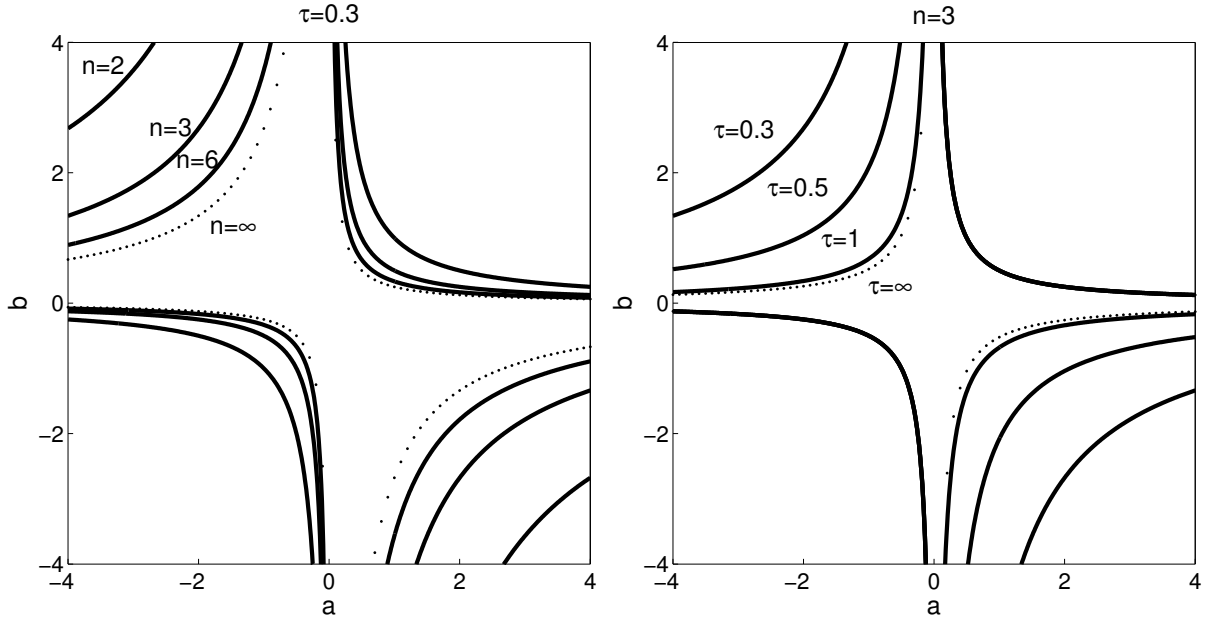


Рис. 6. Границы областей устойчивости для линейной конфигурации нейронов (система (15), (16)). На рисунке слева: запаздывание фиксировано, количество нейронов меняется. Справа: количество нейронов фиксировано, запаздывание меняется.

Теорему 0.7 иллюстрирует рисунок 6. В этом же разделе сравниваются области устойчивости моделей кольцевой и линейной нейронной сети с ограниченным количеством нейронов. В разделе 2.5 было показано, что разрыв кольцевой сети может только улучшить ее устойчивость, если количество нейронов в сети достаточно велико (Теорема 0.5). Наши численные исследования с помощью программ показывают, что это же явление, за некоторыми исключениями, имеет место и при небольшом количестве нейронов (рис. 7).

Область тех значений параметров (a, b) , при которых кольцевая система нейронов устойчива, а линейная с теми же параметрами неустойчива, мы называем парадоксальной областью при данных τ, n . Таким образом, парадоксальная область это область значений параметров (a, b) , в которых нарушается принцип «разрыв кольца благоприятен для устойчивости».

На рисунке 7 показано, что при $\tau = 0.5$ парадоксальная область заметна в модели сети с количеством нейронов $n = 3$, отсутствует при $n = 4$ и весьма

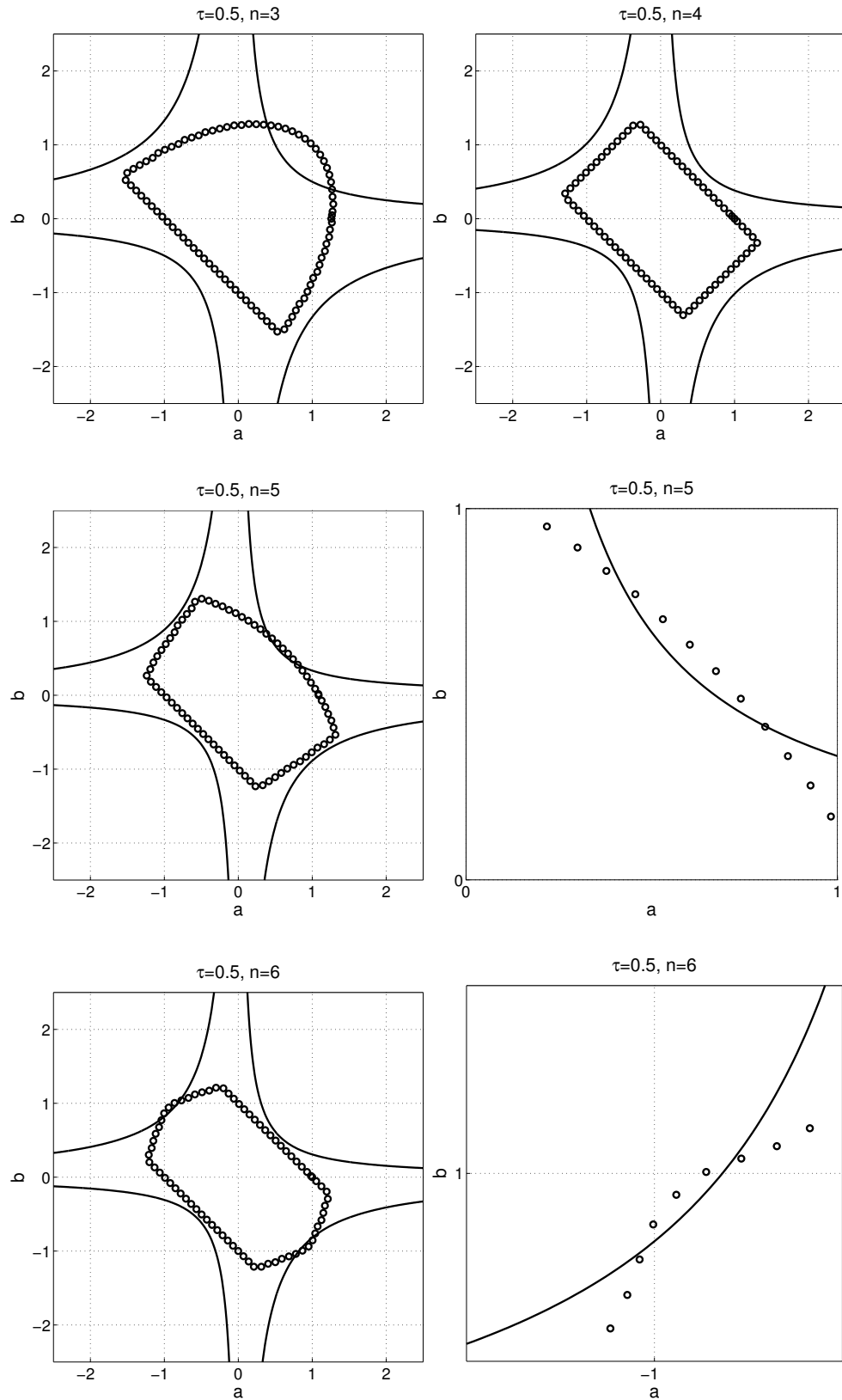


Рис. 7. Границы областей устойчивости для кольцевой (система (8), границы очерчены кружочками) и линейной (система (15), (16), сплошные линии) конфигураций нейронов, $\tau = 0.5$. Для $n = 5$, $n = 6$ дополнительно парадоксальная область показана в увеличении.

мала при $n = 5$, $n = 6$. В результате численных экспериментов выяснилось, что в моделях сетей с $n > 6$ парадоксальная область либо отсутствует, либо пренебрежимо мала.

В разделе 3.5 показана динамика области устойчивости модели кольцевой нейронной сети в процессе постепенного разрыва одной из связей в кольце и превращения сети в линейную.

Рассматривается модель кольцевой сети с двусторонним запаздыванием из шести нейронов. Интенсивности взаимодействия между всеми нейронами, кроме первого и шестого, равны a и b (как в модели (8)), а интенсивности воздействия шестого нейрона на первый и первого нейрона на шестой равны соответственно ac и bc , где c – некоторый неотрицательный параметр. Тогда взаимодействие нейронов в описанной модели задается уравнением (1) с матрицами

$$A = I, \quad B = \begin{pmatrix} 0 & b & 0 & 0 & 0 & ac \\ a & 0 & b & 0 & 0 & 0 \\ 0 & a & 0 & b & 0 & 0 \\ 0 & 0 & a & 0 & b & 0 \\ 0 & 0 & 0 & a & 0 & b \\ bc & 0 & 0 & 0 & a & 0 \end{pmatrix}. \quad (22)$$

При $c = 1$ система (1), (22) описывает кольцевую сеть с двусторонним запаздыванием (8). С изменением c от единицы до нуля интенсивность взаимодействия между первым и шестым нейронами в системе (1), (22) постепенно ослабевает, и при $c = 0$ кольцо нейронов размыкается. На рисунке 8 построены области устойчивости и неустойчивости рассматриваемой сети при различных значениях параметра c . С уменьшением c границы областей устойчивости вытягиваются в четырёх концах и при полном разрыве связи переходят в гиперболы с асимптотами $a = 0$ и $b = 0$ (см. последний график в серии рисунков 8). Как показывает рис. 8, ослабление одной связи между

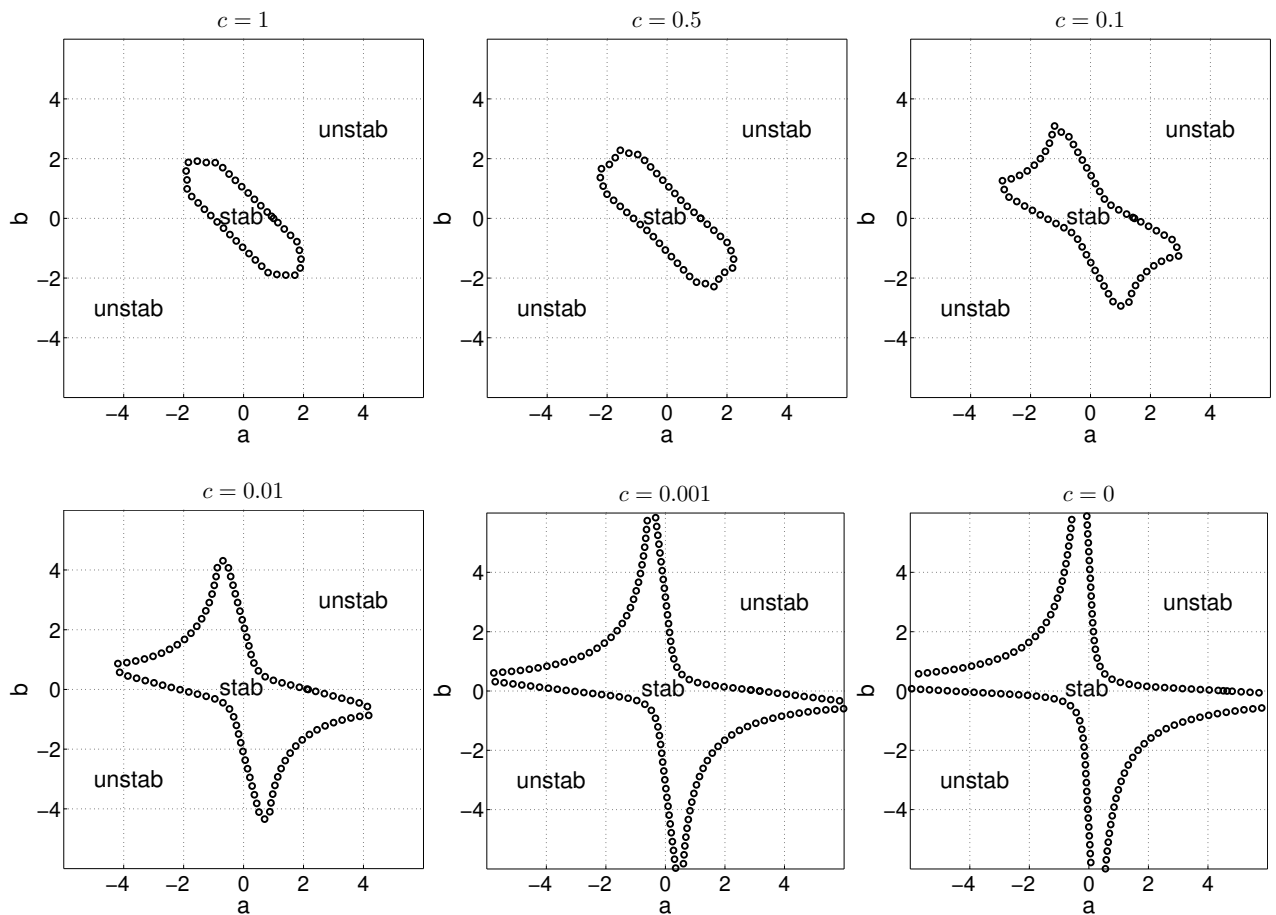


Рис. 8. Динамика области устойчивости в процессе разрыва нейронного кольца. Границы областей устойчивости для системы (1.6), (3.8) для $\tau = 0.1$ и $c = 1$, $c = 0.5$, $c = 0.1$, $c = 0.01$, $c = 0.001$, $c = 0$.

нейронами в кольцевой сети из 6 нейронов расширяет область устойчивости всей системы.

В разделе 3.6 результаты главы 3 сравниваются с известными в литературе. Констатируется, что в большинстве работ по устойчивости кольцевых нейронных сетей рассматривается задача об устойчивости сети из двух [?], трёх [?] или четырёх [?], [?] нейронов. Указаны преимущества рассматриваемых в диссертации моделей кольца нейронов в сравнении с моделью работы [?] (2004), в которой рассматривались кольцевые системы с произвольным количеством нейронов.

Отмечено, что проблема устойчивости линейных конфигураций нейронов не была исследована никем. Предпринятое в главе 3 сравнение областей устойчивости кольцевой и линейной конфигураций с сопоставимыми параметрами в литературе отсутствует.

Указано, что алгоритмы и программы для построения области устойчивости нейронных сетей, описанные в главе 3, не имеют аналогов в известной автору литературе.

В **заключении** суммируются все полученные в диссертации результаты.

В **приложениях** приводятся исходные коды программных продуктов, разработанных в программном пакете MATLAB 7.11.0 (R2010b) и описанных в разделах 1.6, 2.2 и 3.1 диссертации. Код программы «Анализ устойчивости» находится в приложении А, код программы «Устойчивость нейронных сетей» — в приложении Б, код программы «Построение областей устойчивости круговых нейронных сетей» — в приложении В. Во избежание повторений те функции, которые используются в работе всех программ (функция построения конуса устойчивости, функция нахождения значений запаздывания, обеспечивающих устойчивость исследуемого уравнения при фиксированных коэффициентах, и др.), вынесены в отдельное приложение Г.

Основные результаты диссертационной работы

На защиту выносятся следующие новые научные результаты.

1. Разработан метод конуса устойчивости для анализа устойчивости широкого класса матричных дифференциальных уравнений с запаздываниями.
2. Построены алгоритмы и программы для анализа устойчивости математических моделей нейронных сетей кольцевой и линейной конфигурации с неограниченным количеством нейронов.

3. Указаны области устойчивости в пространстве параметров двух моделей кольцевых нейронных сетей с неограниченным количеством нейронов.
4. Найдена асимптотика поведения областей устойчивости в пространстве параметров двух моделей кольцевых конфигураций нейронов, когда количество нейронов неограничено и запаздывание стремится к нулю или к бесконечности.
5. Для моделей кольцевых и линейных нейронных сетей найдены условия устойчивости, независимой от запаздывания (delay-independent stability).
6. Дано полное описание областей устойчивости в пространстве параметров линейных конфигураций нейронных сетей с неограниченным количеством нейронов.
7. Доказано, что область устойчивости кольца с большим количеством нейронов расширяется в случае его разрыва.
8. Построены алгоритмы и программы для анализа устойчивости математических моделей нейронных сетей кольцевой и линейной конфигурации с ограниченным количеством нейронов.
9. Указаны области устойчивости в пространстве параметров двух моделей кольцевых нейронных сетей с ограниченным количеством нейронов.
10. Дано полное описание областей устойчивости в пространстве параметров линейных конфигураций нейронных сетей с ограниченным количеством нейронов.
11. Обнаружены «парадоксальные» области в пространстве параметров некоторых моделей кольцевых сетей с ограниченным количеством нейронов, в которых нарушается принцип «разрыв кольца увеличивает область устойчивости».
12. Численно промоделирована динамика областей устойчивости в процессе постепенного разрыва кольцевой нейронной сети.

Глава 1

Конус устойчивости для диагностирования устойчивости нейронных сетей. Алгоритмы и программы

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 [?] в 1984 г. и с тех пор стала называться моделью Хопфилда. Она описывается дифференциальными уравнениями вида

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

Здесь $n \geq 2$ 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. [?]. Они рассмотрели модель вида

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

где τ запаздывание.

Во многих работах рассматриваются 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, [? ? ?]) изучается система из двух нейронов вида

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

В работе К. Гопалсами и И. Леунга [?] 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

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

Известны трудности изучения устойчивости дифференциальных уравнений с двумя запаздываниями даже в скалярном случае, выявленные М. Кипнисом и И. Левицкой [?] в 2003-2007 гг. По этой причине в большинстве исследований часть запаздываний τ_{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

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

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 [?] (2003 г.) и Мори с соавторами [?] (1981, 1983, 1989 гг.).

Вопрос о глобальной устойчивости стационарных состояний нейронных сетей иногда ставится и при некоторых условиях решается (например, в работах ван дер Дрише [?] (1998 г.), И. Дьери и Ф. Хартунга [?] (2003 и 2007 гг.), [?], Л. Идельса и М. Кипниса [?](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 ($1 \leq n$) описывается дифференциальными уравнениями

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

Здесь x_j потенциал j -го нейрона; $c_j(x_j(t))$ представляет результат усиления сигнала $x_j(t)$; положительный коэффициент d_j 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 τ_{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

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

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 часто состояются 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

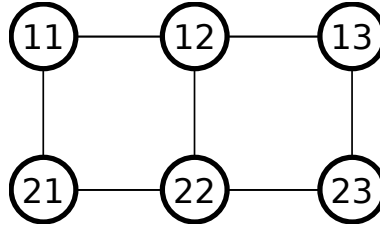


Рис. 1.1. Граф связей в нейронной сети

имеют запаздывание τ . 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

$$A = I, \quad B = \begin{pmatrix} 0 & b & 0 & d & 0 & 0 \\ a & 0 & b & 0 & d & 0 \\ 0 & a & 0 & 0 & 0 & d \\ c & 0 & 0 & 0 & b & 0 \\ 0 & c & 0 & a & 0 & b \\ 0 & 0 & c & 0 & a & 0 \end{pmatrix}. \quad (1.9)$$

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 τ . 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 [?], а также С. Кемпбелл с соавторами [? ?]).

Тогда нейронная сеть описывается уравнением (1.6) с матрицами

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

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 [? ? ? ? ?]. В [?] исследуется эта задача для матриц 2×2 специального вида. В следующих разделах главы 1 диссертации мы даем метод, позволяющий дать полный геометрический обзор тех значений запаздывания τ , которые обеспечивают устойчивость уравнения (1.6). Метод пригоден для уравнений с матрицами A, B , которые могут быть приведены одной трансформирующей матрицей к треугольному виду. Критерий возможности одновременного приведения двух матриц к треугольному виду известен [?]. Этот критерий неэффективен. Но для наших целей существенно, что любые коммутирующие пары матриц A, B могут быть приведены к треугольному виду одним преобразованием.

Несмотря на указанные ограничения, этот метод даёт возможность изучать устойчивость широких классов моделей 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 [?] совместно с научным руководителем и В. В. Малыгиной. Идейно конус устойчивости происходит от давней работы З. Рехлицкого [?] (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 действи-

тельным 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 разделе 1.4 доказывается теорема о конусе устойчивости для матричного уравнения (1.6), которая будет теоретической основой для алгоритмов диагностирования устойчивости нейронных сетей и их программных реализаций.

В разделах 1.5, 1.6 вводится алгоритм и программа для анализа устойчивости основного уравнения (1.6).

В последнем разделе результаты главы 1 сравниваются с известными результатами.

1.2 Овал устойчивости

Рассмотрим уравнение

$$\dot{x}(t) + ax(t) + bx(t - \tau) = 0, \quad t > 0, \quad (1.11)$$

где $a, b \in \mathbb{R}$, $\tau > 0$. Будем рассматривать уравнение (1.11) с начальными условиями

$$x(t) = \varphi(t), \quad t \in [-\tau, 0], \quad (1.12)$$

где $\tau > 0$ — запаздывание, $\varphi(t)$ — начальная функция. Будем считать, что начальная функция $\varphi(t)$ кусочно-непрерывна на промежутке $[-\tau, 0]$. Решением уравнения (1.11) назовём функцию $x(t) : [-\tau, \infty) \rightarrow \mathbb{R}$, удовлетворяющую уравнению (1.11) и условию (1.12).

Уравнение (1.11) считаем (асимптотически) устойчивым, если его нулевое решение является (асимптотически) устойчивым.

Сформулируем известное утверждение об области устойчивости этого уравнения.

Предложение 1.1. *Следующие утверждения верны.*

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

$$-a < b < S(a), \quad (1.13)$$

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

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

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) .

Заметим, что кривая $b = S(a)$ имеет наклонную асимптоту $a = b$ (при $\omega \rightarrow \frac{\pi}{\tau}$ 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

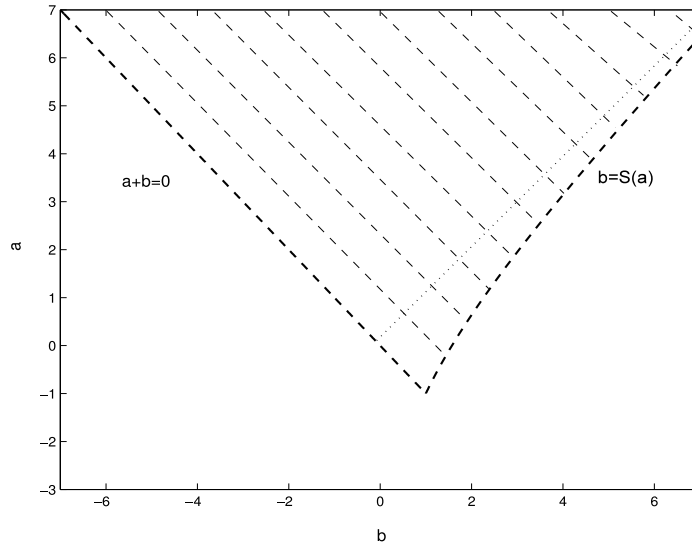


Рис. 1.2. Область устойчивости уравнения (1.11), указанная в предложении (1.1)

переменного b , заданная уравнением

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

где ω_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. [?], Appendix B).

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

1. Если $a \leq -\frac{1}{\tau}$, 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 [?], при этом полагаем, что $a \in \mathbb{R}$ фиксированная величина. Характеристический квазимногочлен уравнения (1.11) имеет вид

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

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

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

Выделим действительную и мнимую части:

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

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

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

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 $\omega \in (-2.7\pi, 2.7\pi)$, $a = 1, \tau = 1$.

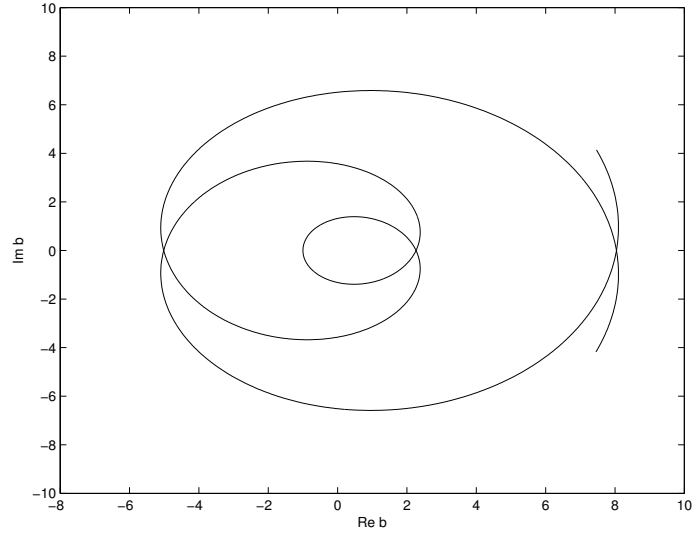


Рис. 1.3. Кривые D -разбиения комплексной плоскости параметра b

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

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

1) Пусть $a \leq -\frac{1}{\tau}$. Тогда в каждой области D -разбиения плоскости комплексного переменного b возьмём в качестве тестовой именно точку с $\text{Im } b = 0$. 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 $(\text{Re } b, \text{Im } b)$ находится внутри овала устойчивости для данного a . Рассмотрим в качестве тестовых 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-

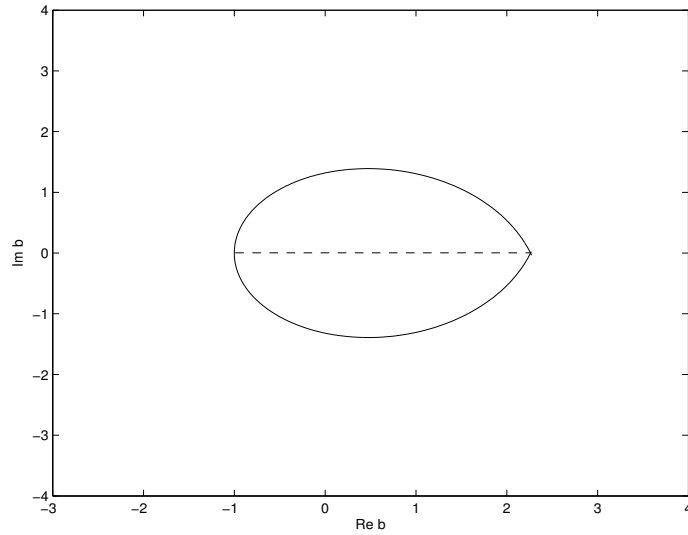


Рис. 1.4. Овал устойчивости уравнения (1.11)

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

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

$$\frac{dF}{d\lambda} = 1 - b\tau e^{-i\omega\tau}. \quad (1.20)$$

Обращение в нуль функции $F(\lambda)$ и её производной в силу (1.17), (1.20) даёт $\lambda + a + \frac{1}{\tau} = 0$, 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 $a \leq -\frac{1}{\tau}$ 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 находятся в полуплоскости $\operatorname{Re} b > 0$. При $a > 0$ граница овала находится в обеих полуплоскостях.

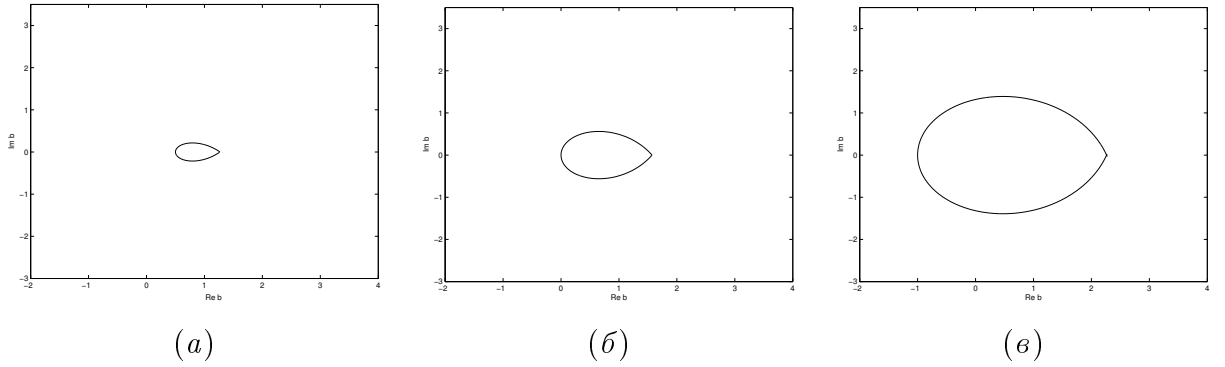


Рис. 1.5. Овал устойчивости: (а) $-\frac{1}{\tau} < a < 0$, (б) $a = 0$, (в) $a > 0$

1.3 Конус устойчивости для скалярного уравнения с комплексными коэффициентами

Вначале рассмотрим, как и в предыдущем разделе, 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), (1.6) назовём множество точек $M = (u_1, u_2, u_3) \in \mathbb{R}^3$, таких, что

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

где действительные параметры h, ω подчинены ограничениям

$$\begin{cases} h \geq -\frac{\omega}{\operatorname{tg} \omega}, \\ -\pi < \omega < \pi. \end{cases} \quad (1.22)$$

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

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

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

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

откуда

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

Переобозначив $\theta = t$, получим

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

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.

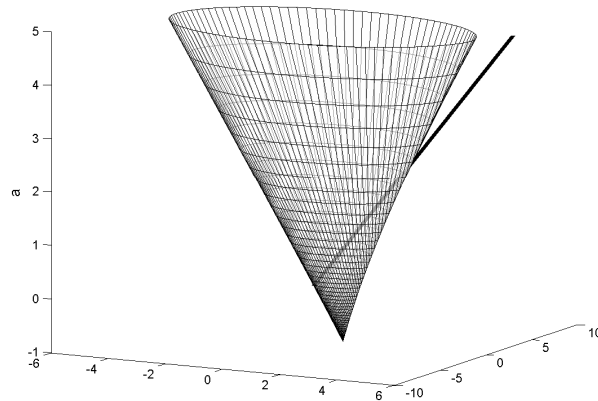
Пример 1.3. Пусть дано уравнение (1.11), для которого

$$a = 0.8, b = 1 + 0.5i. \quad (1.27)$$

Требуется определить, при каких значениях τ 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 $\tau \in [0, +\infty)$, 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

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

где $\lambda, \mu \in \mathbb{C}$.

Для исследования устойчивости уравнения (1.28) сделаем замену

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

В результате замены (1.29) и сокращений получим уравнение

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

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

$$u_1 + iu_2 = \tau(\mu \exp(i\tau \operatorname{Im} \lambda)), \quad u_3 = \tau \operatorname{Re} \lambda. \quad (1.31)$$

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

1.4.1 Критерий асимптотической устойчивости

Для удобства ссылок выпишем матричное уравнение (1.6) заново:

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

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 типовых 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 λ_{js}, μ_{js} ($1 \leq j, s \leq m$). Построим систему точек $M_j = (u_{1j}, u_{2j}, u_{3j}) \in \mathbb{R}^3$, ($1 \leq j \leq m$), так, что

$$u_{1j} + iu_{2j} = \tau\mu_{jj} \exp(i\tau \operatorname{Im} \lambda_{jj}), \quad u_{3j} = \tau \operatorname{Re} \lambda_{jj}. \quad (1.33)$$

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

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

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

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

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

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

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

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. ■

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

$$A = \begin{pmatrix} 0.72 & -0.96 \\ 4.08 & -0.60 \end{pmatrix}, \quad B = \begin{pmatrix} 0.08 & -0.08 \\ 0.34 & -0.03 \end{pmatrix}. \quad (1.37)$$

Поскольку $A = 12B - 0.24I$, матрицы A, B коммутируют и, следовательно, могут быть совместно приведены к треугольной (в данном случае диагональной) форме [?]. Собственные числа матриц A, B суть соответственно

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

$$\begin{aligned} u_{11} &= 0.1575\tau \cos(1.4114 + 1.8658\tau), \\ u_{21} &= 0.1575\tau \sin(1.4114 + 1.8658\tau), \\ u_{31} &= 0.06\tau. \end{aligned} \quad (1.38)$$

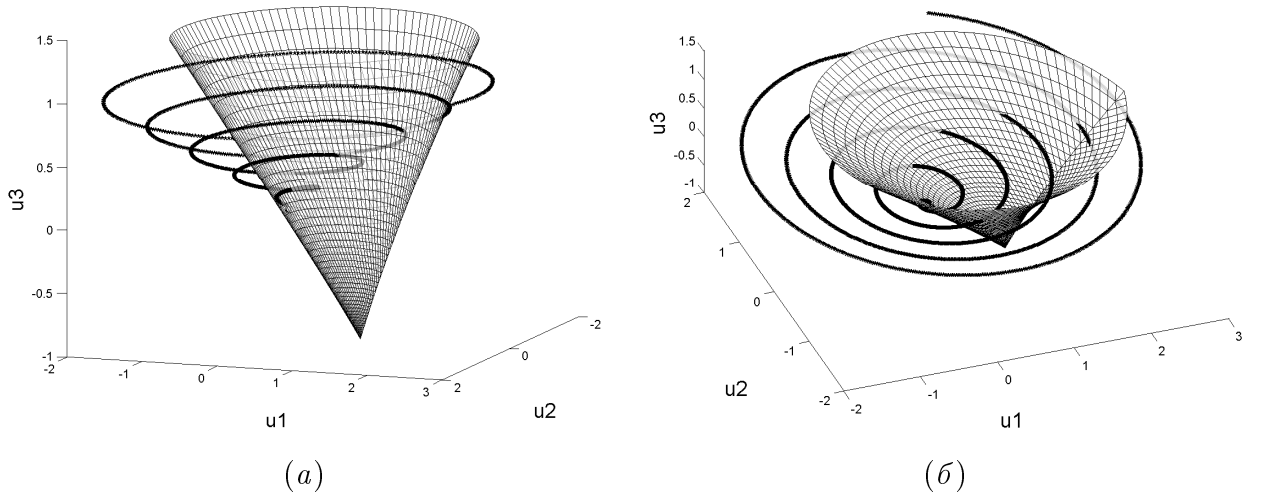


Рис. 1.7. Конус устойчивости и кривая (1.38) в разных проекциях

Точки кривой находятся внутри конуса устойчивости при следующих значениях запаздывания: $\tau \in (0, 0.2729) \cup (1.6893, 3.3978) \cup (5.3439, 6.5218) \cup (8.9955, 9.6457) \cup (12.6502, 12.7675)$, поэтому уравнение (1.32) с матрицами (1.37) асимптотически устойчиво только при таких значениях запаздывания.

Пример 1.5. Положим в (1.32)

$$A = \begin{pmatrix} 1 & 1 & -3 & 2 \\ -3 & 1 & -2 & 3 \\ -1 & 0.5 & 4 & -2 \\ -3 & 2 & -1 & 1 \end{pmatrix}, \quad B = 0.8I + 0.2A + 2A^2. \quad (1.39)$$

Требуется определить, при каких значениях τ это уравнение устойчиво, а при каких неустойчиво.

Очевидно, матрицы A, B коммутируют. Согласованный спектр матриц A и B таков: $\lambda_1 = 3.562$, $\mu_1 = 26.888$, $\lambda_{2,3} = 2.368 \pm 1.514i$, $\mu_{2,3} = 7.902 \pm 14.640i$, $\lambda_4 = -1.298$, $\mu_4 = 3.907$. При этом

$$a_1 = \operatorname{Re} \lambda_1 = 3.562, a_2 = \operatorname{Re} \lambda_2 = 2.368, a_3 = \operatorname{Re} \lambda_3 = 2.368,$$

$$a_4 = \operatorname{Re} \lambda_4 = -1.298, b_1 = \mu_1 \exp(i\tau \operatorname{Im} \lambda_1) = 26.888,$$

$$b_2 = \mu_2 \exp(i\tau \operatorname{Im} \lambda_2) = 16.637 \exp(i(1.076 + 1.514\tau)),$$

$$b_3 = \mu_3 \exp(i\tau \operatorname{Im} \lambda_3) = 16.637 \exp(i(-1.076 - 1.514\tau)),$$

$$b_4 = \mu_4 \exp(i\tau \operatorname{Im} \lambda_4) = 3.9073. \quad (1.40)$$

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.

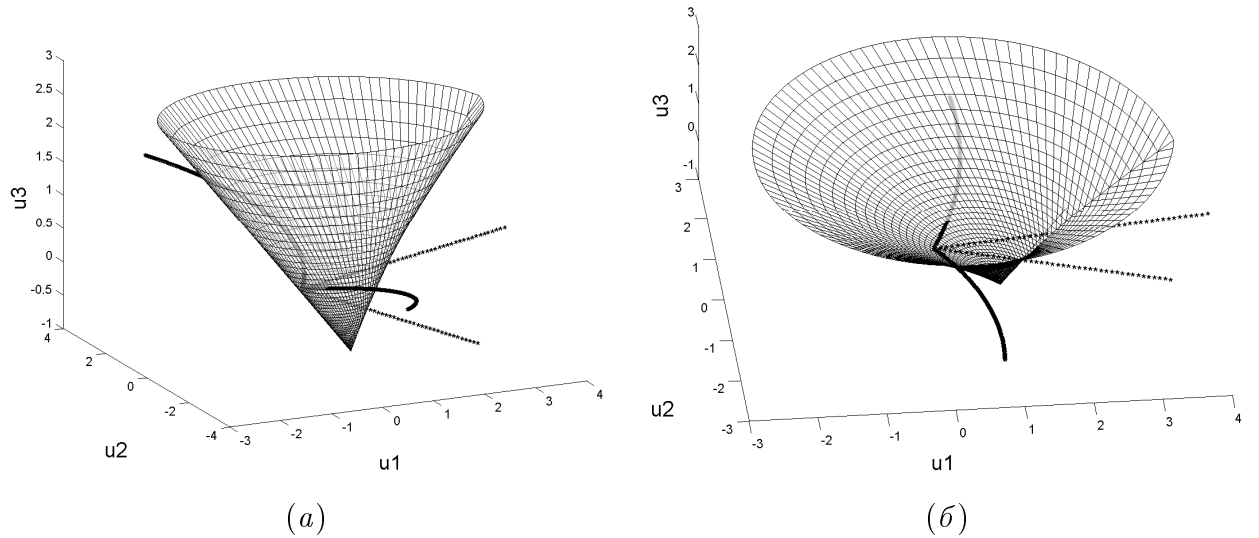


Рис. 1.8. Конус устойчивости и кривая $(\operatorname{Re} b_j \tau, \operatorname{Im} b_j \tau, a_j \tau)$ в разных проекциях.

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 τ 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 (1.39) 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.

Таким образом, метод конусов устойчивости, описанный в данном разделе, позволяет исследовать устойчивость систем (1.32) с совместно триангулируемыми матрицами (см. также [?], [?]) и наглядно интерпретировать результаты. Нахождение интервалов значений запаздываний, обеспечивающих устойчивость рассматриваемого уравнения и указанных в примерах 1.4, 1.5, требует конструирования специального алгоритма, основанного на методе конусов устойчивости, и будет подробно описано в следующих пунктах настоящей главы.

1.4.2 Критерий устойчивости, независимой от запаздывания

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

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

$$K_j : (\operatorname{Re} \lambda_{jj})^2 (u_{11}^2 + u_{22}^2) = |\mu_{jj}|^2 u_3^2, \quad 1 \leq j \leq n, \quad (1.41)$$

где λ_{jj}, μ_{jj} диагональные элементы матриц A, B соответственно.

На конусах K_j лежат конические спирали 1.31, 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 1.33, и тогда точки M_j выходят из K^1 при неограниченном увеличении τ (при $\mu_{jj} \neq 0$), что влечёт неустойчивость уравнения (1.32). Следовательно, асимптотическая устойчивость (1.32), независимая от запаздывания τ , имеет место тогда и только тогда, когда для любого j ($1 \leq j \leq n$) $\operatorname{Re} \lambda_{jj} > 0$ и конус K^j лежит внутри K^2 , то есть

$$|\mu_{jj}| < \operatorname{Re} \lambda_{jj}. \quad (1.42)$$

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 λ_{js}, μ_{js} ($1 \leq j, s \leq m$). Для того, чтобы уравнение (1.32) было асимптотически устойчивым при любом запаздывании $\tau \geq 0$, необходимо и достаточно выполнение условия (1.42).

В следующих главах будут приведены примеры применения этой теоремы для анализа устойчивости кольцевых и линейных нейронных сетей, независимой от запаздывания во взаимодействии нейронов.

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

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

$$\dot{x}(t) + Ax(t) + Bx(t - \tau) = 0 \quad (1.43)$$

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 λ_{jj}, μ_{jj} , $1 \leq j \leq n$ собственные числа матриц A и B в том порядке, в котором они расположены на диагоналях матриц A_T, B_T . Для каждого j , $1 \leq j \leq n$ составим множество T_j по следующему **алгоритму**.

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

$$\tau_j^m = \frac{\arg(\pm iQ - \operatorname{Re} \lambda_{jj}) - \arg \mu_{jj} + 2\pi m}{\operatorname{Im} \lambda_{jj} \pm Q}, \quad m \in \mathbb{Z},$$

$$Q = \sqrt{|\mu_{jj}|^2 - (\operatorname{Re} \lambda_{jj})^2}. \quad (1.44)$$

Из чисел τ_j^m выбираем такие, для которых

$$0 < \tau_j^m \leq \pi/Q \quad \text{и} \quad \operatorname{Re} \lambda_{jj} \geq -\frac{Q}{\operatorname{tg}(Q \tau_j^m)}, \quad (1.45)$$

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

Для конструирования множества T_j сформулируем условия $H1, H2$.

$$H1: \quad \operatorname{Re} \lambda_{jj} > 0 \text{ и выполнено хотя бы одно условие}$$

$$\text{а) } \arg \mu_{jj} \in (-\pi/2, \pi/2) \quad \text{или} \quad \text{б) } \frac{\operatorname{Re} \lambda_{jj}}{|\operatorname{Re} \mu_{jj}|} > 1. \quad (1.46)$$

$$H2: \quad \operatorname{Re} \lambda_{jj} < 0, \quad \left| \frac{\operatorname{Re} \lambda_{jj}}{\operatorname{Re} \mu_{jj}} \right| < 1 \quad \text{и} \quad \arg \mu_{jj} \in (-\pi/2, \pi/2). \quad (1.47)$$

Если выполняется одно из условий $H1$ или $H2$, то

$$\begin{aligned} \text{при чётных } s: \quad T_j &= (0, \tau_{1j}) \cup (\tau_{2j}, \tau_{3j}) \cup \dots \cup (\tau_{sj}, \infty), \\ \text{при нечётных } s: \quad T_j &= (0, \tau_{1j}) \cup (\tau_{2j}, \tau_{3j}) \cup \dots \cup (\tau_{s-1j}, \tau_{sj}). \end{aligned} \quad (1.48)$$

Если не выполнено ни условие $H1$, ни условие $H2$, то

$$\begin{aligned} \text{при чётных } s: \quad T_j &= (\tau_{1j}, \tau_{2j}) \cup (\tau_{3j}, \tau_{4j}) \cup \dots \cup (\tau_{s-1j}, \tau_{sj}), \\ \text{при нечётных } s: \quad T_j &= (\tau_{1j}, \tau_{2j}) \cup (\tau_{3j}, \tau_{4j}) \cup \dots \cup (\tau_{sj}, \infty). \end{aligned} \quad (1.49)$$

Если для некоторого j множество $(\tau_{1j}, \tau_{2j}, \dots, \tau_{sj})$ пусто и выполнено (1.46) или (1.47), то $T_j = (0, +\infty)$.

Докажем теорему.

Теорема 1.6. Пусть множества $T_j, 1 \leq j \leq n$ составлены согласно описанному алгоритму. Тогда уравнение (1.43) асимптотически устойчиво если и только если $\tau \in \bigcap_j T_j$.

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

$$\begin{aligned} u_{1j} &= \tau \operatorname{Re}(\mu_{jj} \exp(i\tau \operatorname{Im} \lambda_{jj})), \\ u_{2j} &= \tau \operatorname{Im}(\mu_{jj} \exp(i\tau \operatorname{Im} \lambda_{jj})), \\ u_{3j} &= \tau \operatorname{Re} \lambda_{jj}. \end{aligned} \quad (1.50)$$

Построим также конус устойчивости для уравнения (1.43), а именно множество точек $M = (u_1, u_2, u_3) \in \mathbb{R}^3$, таких, что

$$\begin{cases} u_1 = -h \cos \omega + \omega \sin \omega, \\ u_2 = h \sin \omega + \omega \cos \omega, \\ u_3 = h, \end{cases} \quad (1.51)$$

где действительные параметры h, ω подчинены ограничениям

$$\begin{cases} h \geq -\frac{\omega}{\operatorname{tg} \omega}, \\ -\pi < \omega < \pi. \end{cases} \quad (1.52)$$

Согласно Теореме 1.4 уравнение (1.43) асимптотически устойчиво если и только если все точки M_j лежат внутри конуса устойчивости (1.51), (1.52).

С изменением τ от 0 до $+\infty$ точки (1.50) образуют винтовые линии (прямые в случае $\operatorname{Im} \lambda_{jj} = 0$). Для каждого j найдем значения τ , соответствующие точкам пересечения этих линий с овалом устойчивости (u_1, u_2) на уровне $u_3 = h$. Эти значения τ являются корнями уравнения

$$\tau \mu_{jj} \exp(i\tau \operatorname{Im} \lambda_{jj}) = (i\omega - h) \exp(-i\omega), \quad h = \tau \operatorname{Re} \lambda_{jj}, \quad (1.53)$$

с параметрами h, ω , удовлетворяющими условию (1.52).

Система (1.53) равносильна системе

$$\begin{cases} \tau |\mu_{jj}| = \sqrt{\omega^2 + h^2}, \\ \arg \mu_{jj} + \tau \operatorname{Im} \lambda_{jj} = -\omega + \arg(i\omega - h) + 2\pi m, \quad m \in \mathbb{Z}, \\ h = \tau \operatorname{Re} \lambda_{jj}. \end{cases} \quad (1.54)$$

Из первого и третьего уравнения системы (1.54) найдем ω :

$$\omega = \pm \tau \sqrt{|\mu_{jj}|^2 - (\operatorname{Re} \lambda_{jj})^2} = \pm \tau Q. \quad (1.55)$$

Подставим ω во второе уравнение системы (1.54), а также воспользуемся свойством $\arg(\tau z) = \arg(z)$ для любого $\tau > 0$. Получим уравнение

$$\tau(\operatorname{Im} \lambda_{jj} \pm Q) = \arg(\pm iQ - \operatorname{Re} \lambda_{jj}) - \arg \mu_{jj} + 2\pi m, \quad (1.56)$$

корнями которого являются числа τ_j^m , указанные в формулировке теоремы.

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

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

что означает $0 < \tau \leq \pi/Q$.

Кроме того, поскольку $h \geq -\frac{\omega}{\operatorname{tg} \omega}$, то

$$\operatorname{Re} \lambda_{jj} \geq -\frac{Q}{\operatorname{tg}(Q \tau_j^m)}.$$

Таким образом, для каждого j требуется найти такие τ_j^m , которые удовлетворяют условиям (1.45).

Для правильного определения интервалов устойчивости данной винтовой линии (прямой) необходимо отсортировать числа τ_j^m по возрастанию и исключить повторения, т.е. получить систему чисел $\tau_{1j}, \tau_{2j}, \dots, \tau_{sj}$, 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 и чётности s составленным множеством T_j по формулам (1.48) или (1.49). 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 $\tau \rightarrow 0+$ 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 $\tau \rightarrow 0+$ лежали внутри конуса устойчивости, достаточно, чтобы вектор производной $\frac{du_2}{du_1}$ при $\tau \rightarrow 0+$ 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.

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

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 $\tau \rightarrow 0+$, 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 (1.46) а).

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

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

Таким образом, если $\frac{\operatorname{Re} \lambda_{jj}}{|\operatorname{Re} \mu_{jj}|} > 1$, то точки винтовой линии при $\tau \rightarrow 0+$ лежат внутри конуса устойчивости, а значит, интервал $(0, \tau_{1j})$ является интервалом устойчивости. Получено условие (1.46) 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 $\tau \rightarrow 0+$ требуется, чтобы вектор производной $\frac{du_2}{du_1}$ при $\tau \rightarrow 0+$ 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 (1.47).

Во всех остальных случаях точки винтовой линии лежат вне конуса устойчивости при $\tau \rightarrow 0+$, поэтому множество T_j составляется по формулам (1.49).

Если для некоторого j выполнено $\operatorname{Re} \lambda_{jj} > |\mu_{jj}|$, 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 (1.43) требуется, чтобы все точки $M_j = (u_{1j}, u_{2j}, u_{3j})$, $1 \leq j \leq n$ находились внутри конуса устойчивости при данном τ . Значит, уравнение (1.43) асимптотически устойчиво тогда и только тогда, когда $\tau \in \bigcap_j T_j$. Утверждение теоремы доказано. ■

Заметим, что если равенство $\operatorname{Re} \lambda_{jj} > |\mu_{jj}|$ верно для всех j , то уравнение (1.43) асимптотически устойчиво при всех $\tau \geq 0$ (Теорема 1.5).

1.6 Программный продукт «Анализ устойчивости»

1.6.1 Функциональное назначение и область применения программы

Программный продукт «Анализ устойчивости» предназначен для исследования устойчивости нулевого решения дифференциального уравнения (1.43) с одновременно триангулируемыми матрицами A, B . Исследование устойчивости моделей нейронных сетей многих стандартных конфигураций сводится именно к этому уравнению.

Алгоритм нахождения значений запаздывания, гарантирующих устойчивость уравнения (1.43) при данных матрицах A и B , указан в разделе 1.5 диссертации. В ходе реализации этого алгоритма строится 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) была разработана программа для анализа устойчивости уравнения (1.43). Исходный код программы доступен в Приложении А.

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 (1.43), и таким образом определить, является ли уравнение с данным запаздыванием τ 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 устойчивость уравнения (1.43) для матриц любого порядка, но программа реализована для систем размерности не выше пятой.

1.6.2 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. 1.9) пользователю необходимо заполнить элементы 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».

Анализ устойчивости уравнения
 $x'(t) + Ax(t) + Bx(t-\tau) = 0$

Входные данные
Размерность = 5

Выберите способ задания матриц A и B

☒ Задать матрицы A и B с помощью базовой матрицы D
☐ Задать матрицы A и B с помощью собственных чисел

Базовая матрица D =

| | | | | |
|---------|----------|---------|---------|---------|
| 0.04377 | 0.3508 | 0.4725 | -0.7318 | -0.8925 |
| -0.3283 | -0.06306 | 0.1237 | -0.5748 | -0.1166 |
| -0.6487 | 0.8243 | -0.6316 | 0.7899 | -0.9734 |
| -0.5821 | -0.792 | 0.1944 | -0.8571 | 0.7944 |
| 0.8103 | 0.4911 | -0.4001 | -0.515 | -0.6067 |

Введите коэффициенты в разложении матриц A и B по степеням матрицы D

A = E + D + D² + D³ + D⁴ + D⁵
B = E + D + D² + D³ + D⁴ + D⁵

Результаты вычислений

(0, 1.0962)
(0, 1.0962)
(0, Inf)
Empty set
(0, 2.8582)

Уравнение неустойчиво при любом tau!

Empty set

Очистить Заполнить Рассчитать... Закрыть все графики

Рис. 1.9. Главное окно программы «Анализ устойчивости» при задании матриц с помощью базовой матрицы.

При выборе второго способа (рис. 1.10) пользователю требуется ввести собственные значения матриц A и B в порядке, определенном одновременным 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...». Показанное на рисунке 1.11 вспомогатель-

Анализ устойчивости уравнения
 $x'(t) + Ax(t) + Bx(t-\tau) = 0$

Входные данные
Размерность = 3

Выберите способ задания матриц A и B

☐ Задать матрицы A и B с помощью базовой матрицы D
☒ Задать матрицы A и B с помощью собственных чисел

| Собственные числа матрицы A | | | | Собственные числа матрицы B | | | |
|-----------------------------|---------|---|------------|-----------------------------|--------|---|------------|
| lambda1 = | 0.0508 | + | -1.6219 *i | mu1 = | 0.0903 | + | -0.1592 *i |
| lambda2 = | 0.0508 | + | 1.6219 *i | mu2 = | 0.0903 | + | 0.1592 *i |
| lambda3 = | -0.0153 | + | 0 *i | mu3 = | 0.0716 | + | 0 *i |

Результаты вычислений

(0, 0.44345) U (2.3348, 3.9385) U (6.6798, 7.4336)

(0, 0.44345) U (2.3348, 3.9385) U (6.6798, 7.4336)

(0, 19.3784)

Уравнение устойчиво при tau, лежащем в следующем объединении интервалов:

(0, 0.44345) U (2.3348, 3.9385) U (6.6798, 7.4336)

Рис. 1.10. Главное окно программы «Анализ устойчивости» при задании собственных чисел матриц вручную.

ное окно представляет результаты расчётов графически: в трёхмерном пространстве изображаются конус устойчивости и системы точек, расположение которых позволяет сделать вывод об устойчивости исследуемого 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 (1.43). Причём, если существует хотя бы один промежуток значений τ , обеспечивающих устойчи-

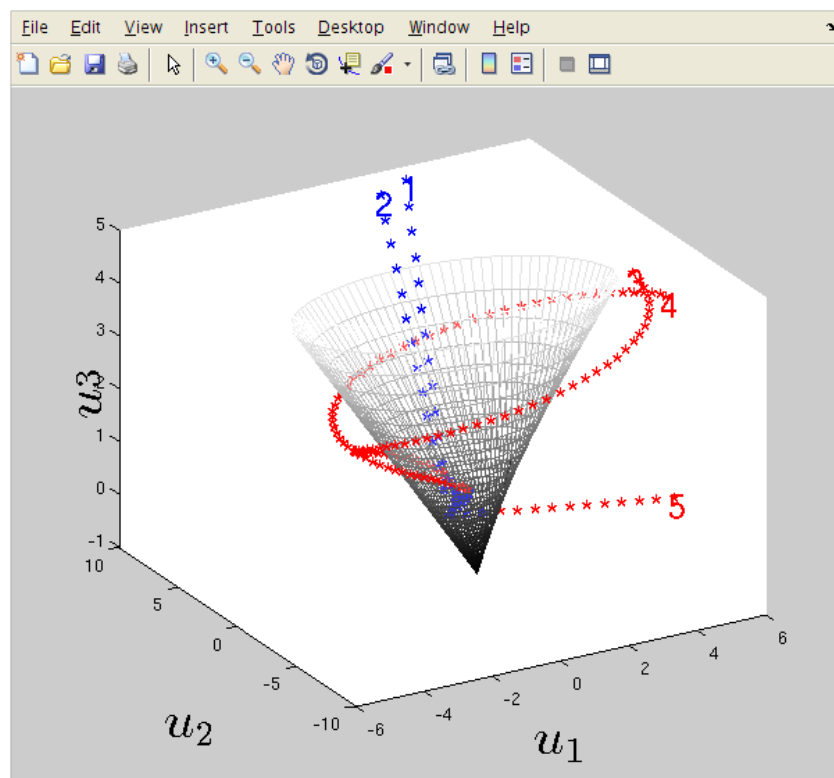


Рис. 1.11. Графический вывод результатов вычисления.

вость уравнения, то будут показаны все необходимые промежутки (рис. 1.9), если нет — будет выдано сообщение о неустойчивости уравнения при всех значениях запаздывания (рис. 1.10).

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 «Заккрыть 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. 1.12).

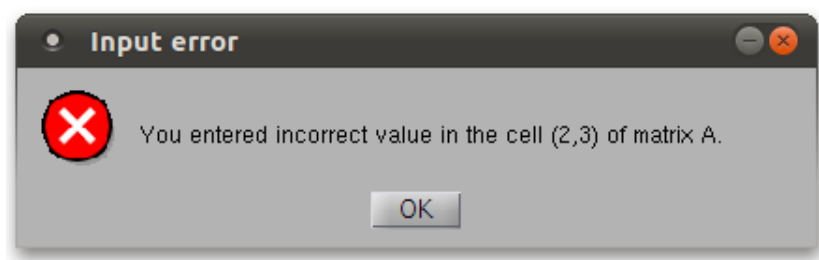


Рис. 1.12. Сообщение об ошибке: введены некорректные данные

1.6.3 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 msc, входящего в дистрибутив. Для запуска программы нужен установленный Matlab соответствующей версии либо библиотеки Matlab, которые можно получить на официальном сайте (<http://www.mathworks.com/>).

Программа зарегистрирована в Объединенном фонде электронных ресурсов «Наука и образование» (ОФЭРНиО), информация о ней представлена в [?], а пользовательская версия программы доступна по электронному адресу [?].

1.7 Сравнение результатов главы 1 с известными результатами

Идейным источником метода конусов устойчивости явились работы Рехлицкого, в частности, статья [?] (1956), в которой впервые появился овал устойчивости для линейного уравнения с запаздыванием $\dot{x}(t) = Ax(t - \tau)$ в банаховом пространстве. Овал устойчивости для разностного матричного

уравнения $x_n = x_{n-1} + Ax_{n-k}$ с запаздыванием $k \in \mathbb{N}$ введен I. Levitskaya [?] (2006), позднее для матричного уравнения $x_n = \alpha x_{n-1} + Ax_{n-k}$, $0 \leq \alpha \leq 1$ изучен в работе Е. Kaslik [?] (2009). Конусы устойчивости для разностных матричных уравнений введены в статьях М.М. Kipnis, V.V. Malygina [?] (2011), S.A. Ivanov, М.М. Kipnis, V.V. Malygina [?] (2011) и S.A. Ivanov, М.М. Kipnis, V.V. Malygina [?] (2011).

Устойчивость скалярных дифференциальных уравнений с запаздываниями с комплексными коэффициентами изучали А.И. Кирьянен и К.В. Галунова [?] (1989), но овалы или конусы устойчивости в их работе не появлялись.

В работе В. Cahlon, D. Schmidt [?] (2000) рассматривалась задача об устойчивости класса уравнений вида $\dot{x}(t) = \alpha Ax(t) + (1 - \alpha)Ax(t - \tau)$, $0 \leq \alpha \leq 1$ с 2×2 матрицей A . Годом позднее те же авторы [?] (2001) усилили свой результат, рассматривая уравнение $\dot{x}(t) = \alpha Ax(t) + \beta Ax(t - \tau)$ с произвольными действительными α, β , и снова для 2×2 матрицы A . Поскольку матрицы αA и βA с действительными α и β , очевидно, приводятся совместно к треугольному виду, мы заключаем, что результаты главы 1 диссертации сильнее результатов этих статей как по размерности рассматриваемых задач, так и по охвату изучаемых уравнений, даже если ограничиться 2×2 матрицами.

Н. Matsunaga [?] (2007) дал критерий устойчивости уравнения $\dot{x}(t) = Ax(t - \tau)$ с 2×2 матрицей A в терминах следа и детерминанта матрицы A . Эти результаты легко вытекают из результатов диссертации, поскольку для 2×2 матриц собственные числа однозначно определяются по следу и детерминанту. На самом деле результаты Х. Мацунаги вытекают даже из результатов Рехлицкого [?] 50-летней давности.

S. Sakata [?] (1998) рассматривал уравнения вида $\dot{x}(t) = \alpha x(t) + Bx(t - \tau)$ с 2×2 матрицей B и действительным α . Метод конуса устойчивости применим к значительно более широкому классу уравнений.

Относительно условия (1.42), гарантирующего устойчивость, независимую от запаздывания, автору неизвестно, есть ли оно в литературе, кроме статьи автора совместно с научным руководителем и В.В. Малыгиной [?]. Другое, не столь удобное условие устойчивости, независимой от запаздывания (delay-independent stability) дано в работе J. Chen, H. Latchman [?] (1995). А именно, признак Чена-Латчмана таков: в обозначениях теоремы 1.4 уравнение (1.32) асимптотически устойчиво при любом $\tau > 0$ тогда и только тогда, когда для всех $j = 1, 2 \dots n$ верны неравенства $\operatorname{Re} \lambda_{jj} > 0$ и $\max_{s \in \mathbb{R}} |(is - \lambda_{jj})^{-1} \mu_{jj}| < 1$. Очевидно, максимум достигается при $s = \operatorname{Im} \lambda_{jj}$, так что условие Чена-Латчмана превращается в $\operatorname{Re} \lambda_{jj} > |\mu_{jj}|$, что в точности совпадает с нашим условием (1.42). Наше условие проще проверяется и имеет естественное геометрическое объяснение, данное в разделе 1.4.

Алгоритмов для определения интервалов значений запаздывания, обеспечивающих устойчивость матричного уравнения (1.6), в литературе нет. Поэтому алгоритм и программа разделов 1.5, 1.6 не имеют аналогов в литературе.

Глава 2

Численный и теоретический анализ устойчивости моделей нейронных сетей кольцевой и линейной конфигурации с неограниченным количеством нейронов

2.1 Постановка задачи и алгоритм диагностирования устойчивости модели кольцевой сети с неограниченным количеством нейронов

Много работ посвящено проблеме устойчивости нейронных сетей, состоящих из двух нейронов (например, [1, 2]). Проблема устойчивости нейронной сети круговой архитектуры с запаздывающим взаимодействием при количестве нейронов больше двух исследуется, например, в работах [3, 4, 5, 6, 7]. Изучаемая нами в этой главе модель наиболее близка к модели работ [8, 9]. В отличие от работ [10, 11], 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 [12]), но требуют изменения в связи с спецификой задачи о неограниченности количестве нейронов в кольце. Аналогичные методы применены в [13] для изучения устойчивости дискретных моделей круговой системы нейронов (см. также [14, 15]).

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. 2.1) с запаздыванием τ_1 во взаимодействии нейрона с левым соседним нейроном и τ с правым:

$$\dot{x}_j(t) + x_j(t) + a x_{j-1}(t - \tau_1) + b x_{j+1}(t - \tau) = 0 \quad (j \bmod n). \quad (2.1)$$

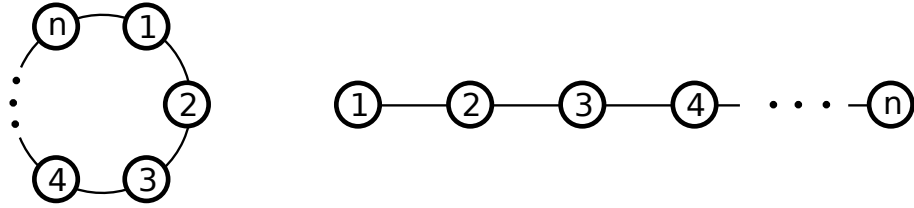


Рис. 2.1. Кольцевая и линейная система нейронов

Здесь x_j ($1 \leq j \leq n$) 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 (2.1) вряд ли возможно, поскольку даже скалярное уравнение $\dot{x}(t) + a x(t - \tau_1) + b x(t - \tau) = 0$ имеет, как выяснилось (см. [? ?]), очень сложную структуру области устойчивости в пространстве параметров. Поэтому основными моделями в нашем исследовании будут два упрощенных варианта систем (2.1), в первом из которых $\tau_1 = 0$, во втором $\tau_1 = \tau$:

$$\dot{x}_j(t) + x_j(t) + a x_{j-1}(t) + b x_{j+1}(t - \tau) = 0 \quad (j \bmod n), \quad (2.2)$$

$$\dot{x}_j(t) + x_j(t) + a x_{j-1}(t - \tau) + b x_{j+1}(t - \tau) = 0 \quad (j \bmod n). \quad (2.3)$$

Уравнения (2.2), (2.3) вместе с вышеуказанными интерпретациями параметров a, b, τ, n мы называем моделями кольцевой сети, имея в виду, что они получаются в результате линеаризации известных нелинейных моделей Хопфилда-Маркуса-Вестервельта [? ?] или Коэна-Гроссберга [?] или Чуа-Янга-Роска

[? ?] вокруг некоторого решения. Модель (2.2) соответствует малым запаздываниям взаимодействия нейронов с правыми соседними нейронами, а (2.3) — 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 [? ?]. Еще одно отличие в том, что у нас отсутствует запаздывание в реакции нейрона на собственный сигнал (selfconnection).

Системы (2.2), (2.3) принадлежат классу систем вида

$$\dot{x}(t) + Ax(t) + Bx(t - \tau) = 0, \quad (2.4)$$

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

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

$$P = \begin{pmatrix} 0 & 0 & \dots & 0 & 1 \\ 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \end{pmatrix}. \quad (2.5)$$

В матричном виде уравнения (2.2), (2.3) соответственно будут иметь вид

$$\dot{x}(t) + (I + aP)x(t) + bP^{n-1}x(t - \tau) = 0, \quad (2.6)$$

$$\dot{x}(t) + Ix(t) + (aP + bP^{n-1})x(t - \tau) = 0, \quad (2.7)$$

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

Пусть матрицы A, B в уравнении (2.4) приводятся одной трансформирующей матрицей к треугольным формам A_T, B_T с диагональными элементами λ_j, μ_j ($1 \leq j \leq n$) соответственно.

Построим n точек $M_j = (u_{1j}, u_{2j}, u_{3j}) \in \mathbb{R}^3$, так что

$$u_{1j} + iu_{2j} = \tau \mu_j \exp(i\tau \operatorname{Im} \lambda_j), \quad u_{3j} = \tau \operatorname{Re} \lambda_j, \quad 1 \leq j \leq n. \quad (2.8)$$

В силу результатов главы 1 диссертации (см. также [?]), система (2.4) асимптотически устойчива тогда и только тогда, когда все точки M_j ($1 \leq j \leq n$) лежат внутри конуса устойчивости. Если хотя бы одна из точек M_j лежит вне конуса, то (2.4) неустойчива.

В разделах 1.5, 1.6 диссертации мы описали алгоритм и программный продукт «Анализ устойчивости» (см. также [?]), предназначенный для диагностирования устойчивости системы (2.4). Программа по согласованному спектру λ_j, μ_j $1 \leq j \leq$ матриц A, B возвращает объединение интервалов

$$T = \cup_{k=1}^N (\tau_k, \tau_{k+1}), \quad (2.9)$$

такое что если $\tau \in T$, то система (2.4) асимптотически устойчива, а если $\tau \notin [\tau_k, \tau_{k+1}]$ при любом k ($1 \leq k \leq N$), то (2.4) неустойчива.

Для проверки устойчивости систем (2.2) и (2.3) при неограниченном порядке n требуется модификация алгоритма, первоначально предназначенного для общей системы (2.4). В настоящем разделе мы указываем такой алгоритм.

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

$$\lambda'_j = 1 + a \exp(i\frac{2\pi j}{n}), \quad \mu'_j = b \exp(-i\frac{2\pi j}{n}), \quad 1 \leq j \leq n, \quad (2.10)$$

$$\lambda''_j = 1, \quad \mu''_j = a \exp(i\frac{2\pi j}{n}) + b \exp(-i\frac{2\pi j}{n}), \quad 1 \leq j \leq n. \quad (2.11)$$

Для изучения устойчивости уравнений (2.5), (2.6) строим систему точек $M'_j = (u'_{1j}, u'_{2j}, u'_{3j}) \in \mathbb{R}^3$, определенных равенствами (см. (2.8), (2.10))

$$u'_{1j} + iu'_{2j} = \tau b \exp(i(-\frac{2\pi j}{n} + a\tau \sin \frac{2\pi j}{n})), \quad u'_{3j} = \tau(1 + a \cos \frac{2\pi j}{n}), \quad 1 \leq j \leq n. \quad (2.12)$$

Аналогично, для системы (2.5), (2.7) точки $M''_j = (u''_{1j}, u''_{2j}, u''_{3j}) \in \mathbb{R}^3$, определены равенствами (см. (2.8), (2.11))

$$u''_{1j} + iu''_{2j} = \tau a (\exp(i\frac{2\pi j}{n}) + b \exp(-i\frac{2\pi j}{n})), \quad u''_{3j} = \tau, \quad 1 \leq j \leq n. \quad (2.13)$$

Для анализа устойчивости систем (2.5), (2.6) 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 (2.14):

$$u'_1(t) + iu'_2(t) = \tau b \exp(i(-t + a\tau \sin t)), \quad u'_3(t) = \tau(1 + a \cos t), \quad 0 \leq t \leq 2\pi. \quad (2.14)$$

Аналогично, для (2.5), (2.7) строим кривую $M''(t) = (u''_1(t), u''_2(t), u''_3(t))$:

$$u''_1(t) + iu''_2(t) = \tau(a \exp(it) + b \exp(-it)), \quad u''_3(t) = \tau, \quad 0 \leq t \leq 2\pi. \quad (2.15)$$

При $n \rightarrow \infty$ точки M_j , определенные равенствами (2.12), образуют плотное на кривой (2.14) множество. Поэтому вопрос об устойчивости системы (2.5), (2.6) сводится к геометрической проблеме. Если все точки кривой (2.14) лежат внутри конуса устойчивости (1.21), (1.22) (см. Определение 1.2), то система (2.5), (2.6) устойчива при любом n , а если хотя бы одна точка кривой (2.14) лежит вне конуса устойчивости, то система (2.5), (2.6) неустойчива при всех достаточно больших значениях n (рис. 2.2).

В свою очередь, эта геометрическая задача может быть решена численными методами. Для системы (2.5), (2.6) при различных значениях $t \in [0, 2\pi]$ мы вычисляем $\arg(u'_1(t) + iu'_2(t))$ согласно (2.14) и находим на конусе устой-

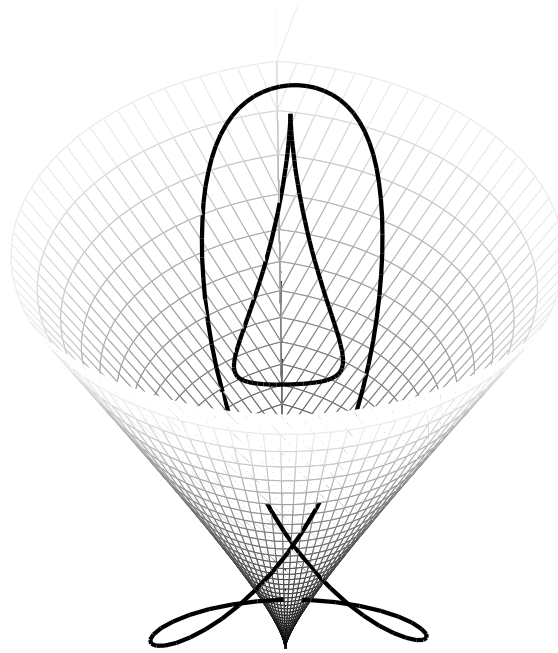


Рис. 2.2. Конус устойчивости и две кривые (2.14). Одна кривая для $\tau = 1.5$, $a = -1.4$, $b = 0.7$, находится частично вне конуса устойчивости, следовательно, система (2.5), (2.6) неустойчива при достаточно больших n . Вторая кривая для $\tau = 2$, $a = 0.5$, $b = -0.2$, находится полностью внутри конуса устойчивости, следовательно, система (2.5), (2.6) устойчива при любом n .

чивости на высоте $h = u'_3(t)$ точку или две точки с тем же аргументом. Сравнение модулей точек на кривой и конусе дает ответ на вопрос, устойчива ли система (2.5), (2.6). Аналогично изучается система (2.5), (2.7).

2.2 Программный продукт «Устойчивость нейронных сетей»

2.2.1 Функциональное назначение и область применения программы

На основе модификации алгоритма исследования устойчивости для систем с неограниченным количеством нейронов, описанной в предыдущем разделе, автором была разработана программа «Устойчивость нейронных сетей»,

которая по коэффициентам уравнений (2.2), (2.3) и величине запаздывания выдаёт сообщение об устойчивости соответствующей кольцевой системы нейронов. Продукт может применяться для получения вывода об устойчивости конкретных систем, а также для регулирования коэффициентов модели с целью получения устойчивого решения.

Программа выполнена в пакете MATLAB 7.11.0 (R2010b) посредством matlab APP для создания графического интерфейса пользователя (GUI). Исходный код программы представлен в приложении Б.

Программа зарегистрирована в Объединенном фонде электронных ресурсов «Наука и образование» (ОФЭРНиО), информация о ней представлена в [?], а пользовательская версия программы доступна по электронному адресу [?]. Примеры работы программы «Устойчивость нейронных сетей» и особенности лежащего в её основе алгоритма представлены также в [?].

2.2.2 Использование

Все входные параметры задаются на одноимённой панели, показанной на рисунке 2.3.

Прежде всего пользователю предлагается выбрать конфигурацию нейронной сети. В программе доступны два варианта кругового соединения нейронов: с малым запаздыванием взаимодействия с правыми соседями (уравнение (2.2)) и с одинаковым запаздыванием взаимодействия с правыми и левыми соседями (уравнение (2.3)). Выбирая первую или вторую радиокнопку, пользователь определяет, для какой именно модели будет проводиться анализ устойчивости. На рисунке 2.3 выбрана первая конфигурация, а на рисунке 2.4 — вторая.

Ниже, на панели ввода представлено название, описание и рисунок выбранной конфигурации. Пользователю необходимо ввести коэффициенты

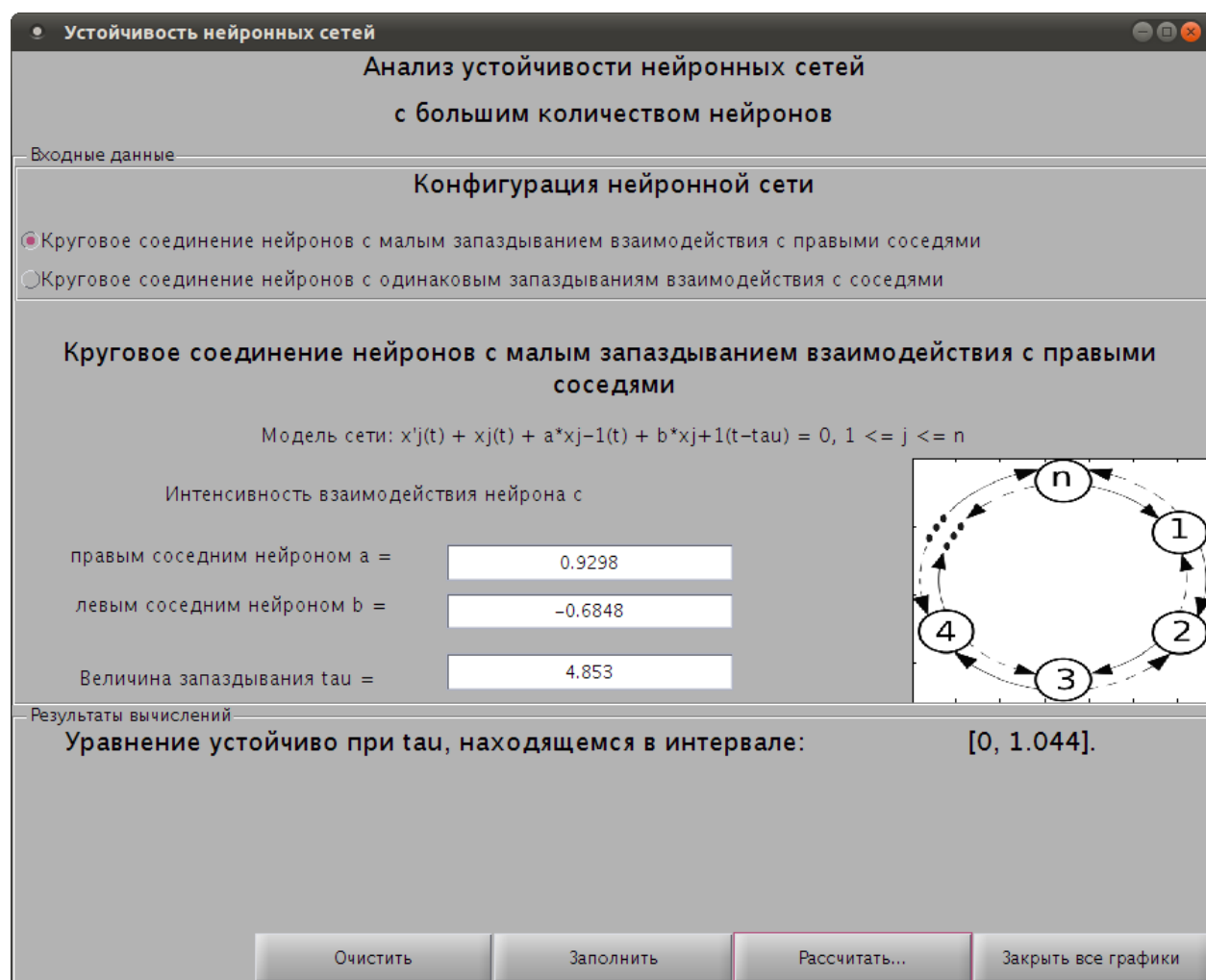


Рис. 2.3. Главное окно программы «Устойчивость нейронных сетей» при выборе первой конфигурации.

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.

После ввода данных пользователю необходимо нажать на кнопку «Рассчитать...». Вспомогательное окно (рис. 2.5) графически представляет результаты расчётов: в трёхмерном пространстве строятся конус устойчивости и системы точек, взаимное расположение которых указывает на устойчивость

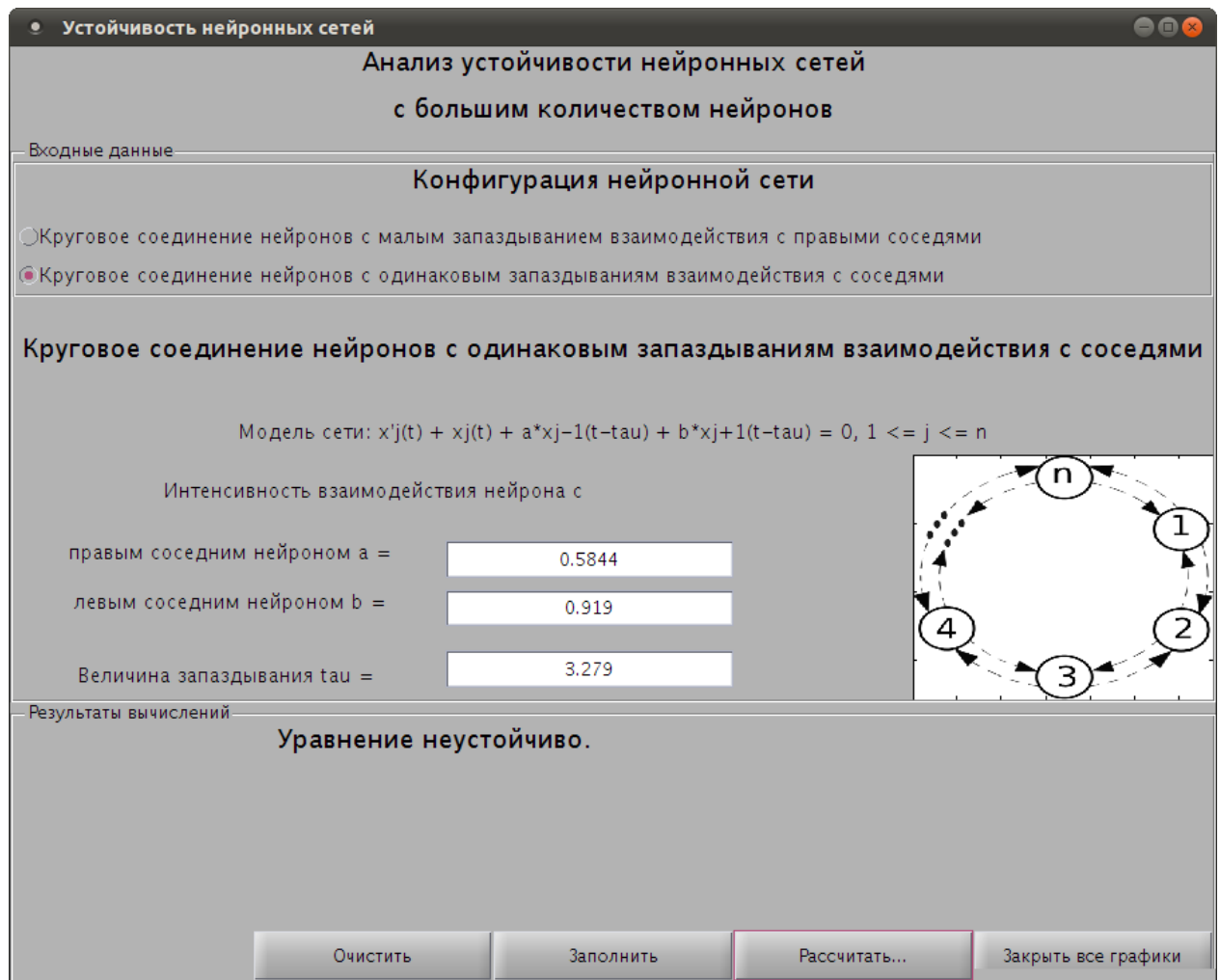


Рис. 2.4. Главное окно программы «Устойчивость нейронных сетей» при выборе второй конфигурации.

или неустойчивость данного уравнения. Чёрным цветом обозначаются точки, находящиеся внутри конуса устойчивости и соответствующие устойчивости, а красным — находящиеся вне конуса и соответствующие неустойчивости. На панели «Результаты вычислений» отображается отчёт о результатах вычислений и выводится сообщение об устойчивости уравнения. Как и в программе «Анализ устойчивости» (см. раздел 1.6 главы 1), выходными данными служат промежутки значений запаздывания τ , обеспечивающих устойчивость исследуемого уравнения (см. рис. 2.3), если такие промежутки существуют. В противном случае будет выдано сообщение о неустойчивости уравнения при всех значениях запаздывания (см. рис. 2.4).

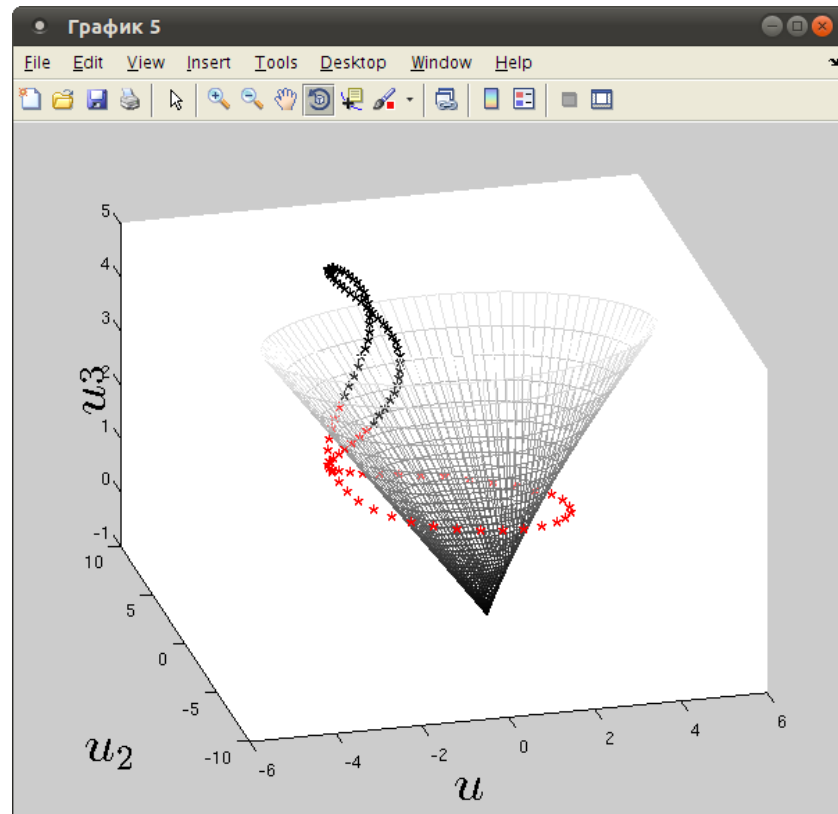


Рис. 2.5. Графический вывод результатов вычисления.

Пользователь имеет возможность изменить входные параметры и провести исследование устойчивости другой системы нейронов с новыми параметрами. При этом появляется ещё одно окно с графиком, а также обновляется отчёт в главном окне. Кнопка «Заккрыть все графики» позволяет закрыть все дополнительные окна.

Программа анализирует входные данные, и при вводе некорректных данных выдаёт сообщения об ошибках. На рис. 2.6 представлено сообщение, которое будет получено при вводе некорректных значений в поле ввода параметра a .

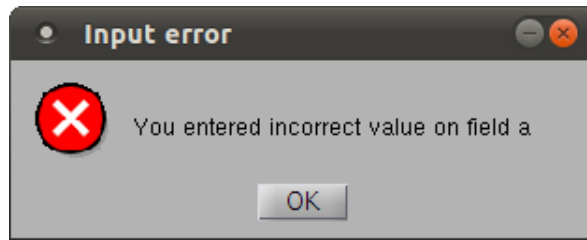


Рис. 2.6. Сообщение об ошибке: введены некорректные данные.

2.3 Результаты исследования устойчивости кольцевой сети нейронов с неограниченным количеством нейронов

В работе Mori и др. [?] доказано, что устойчивость уравнения (2.4), независимая от запаздывания, гарантирована при условии

$$\min_{1 \leq j \leq n} \{ \alpha_{jj} - \sum_{k=1, k \neq j}^n |\alpha_{jk}| \} > \max_{1 \leq j \leq n} \sum_{k=1}^n |\beta_{jk}|, \quad (2.16)$$

где α_{jk}, β_{jk} суть элементы матриц A, B соответственно. Отсюда вытекает следующее Предложение.

Предложение 2.1. *Если $|a| + |b| < 1$, то системы (2.5), (2.6) и (2.5), (2.7) асимптотически устойчивы при любом n и любом $\tau \geq 0$.*

Предложение 2.1 также несложно вывести из Теоремы 1.5 раздела 1.4 диссертации.

Теорема 2.1. *Если $|a + b| > 1$, то системы (2.5), (2.6) и (2.5), (2.7) неустойчивы при любом $\tau > 0$, если n достаточно велико.*

Доказательство теоремы 2.1 будет дано в разделе 2.6. Предложение 2.1 и теорема 2.1 не дает информации о поведении систем (2.5), (2.6) и (2.5), (2.7) для случая, когда выполнены одновременно неравенства $|a + b| < 1$ и

$|a| + |b| > 1$. Для этого случая мы применили исходный код программы «Устойчивость нейронных сетей».

Здесь мы представляем результаты применения программного продукта, Предложения 2.1 и Теоремы 2.1 (см. также [?]).

На рисунках 2.7, 2.8 показаны области D_τ в плоскости параметров (a, b) модели нейронной сети для некоторых значений τ . Область D_τ является расширением области, заданной неравенством $|a| + |b| < 1$, на северо-запад и юго-восток до границ, зависящих от τ . Если точка (a, b) лежит внутри D_τ на Рис. 2.7, то система (2.5), (2.6) асимптотически устойчива. Если (a, b) лежит вне D_τ на рисунке 2.7, то система (2.5), (2.6) неустойчива при достаточно больших n . Аналогичные утверждения верны для рисунка 2.8 и системы (2.5), (2.7). Область D_τ центрально-симметрична: если $(a, b) \in D_\tau$, то $(-a, -b) \in D_\tau$. При одинаковых τ область устойчивости для системы (2.5), (2.6) шире области для (2.5), (2.7).

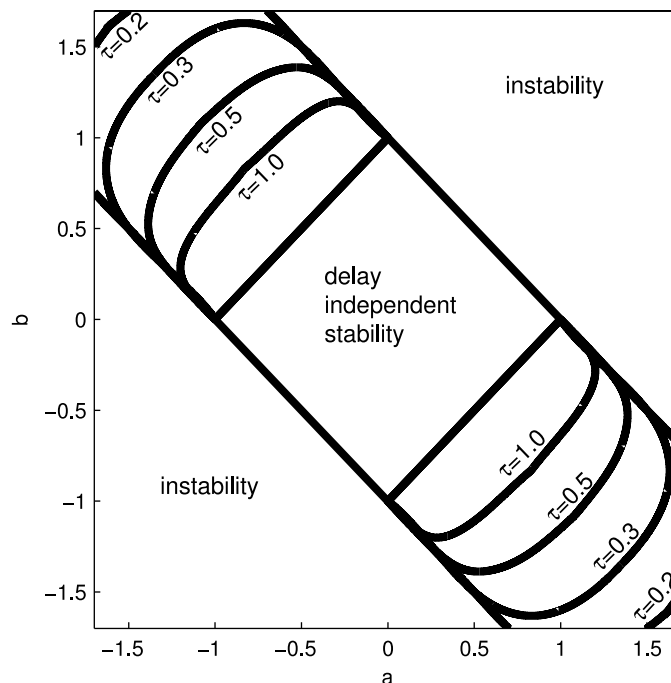


Рис. 2.7. Области устойчивости для системы (2.5), (2.6) с неограниченным числом нейронов.

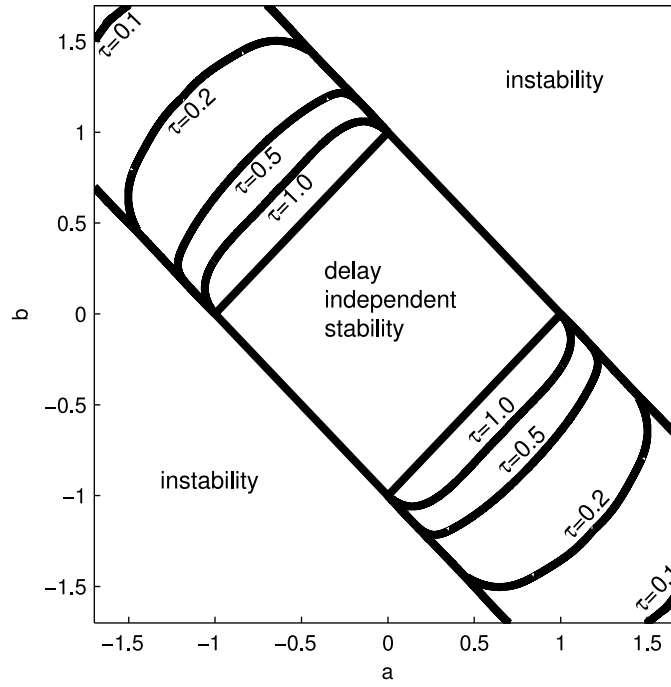


Рис. 2.8. Области устойчивости для системы (2.5), (2.7) с неограниченным числом нейронов.

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:

$$\dot{x}_j(t) + x_j(t) + a(x_{j-1}(t) - x_{j+1}(t - \tau)) = 0 \quad (j \bmod n), \quad (2.17)$$

$$\dot{x}_j(t) + x_j(t) + a(x_{j-1}(t - \tau) - x_{j+1}(t - \tau)) = 0 \quad (j \bmod n). \quad (2.18)$$

Определение 2.1. Границей устойчивости системы (2.17) для больших n назовем такое число $a_1(\tau) \in \mathbb{R}$, что если $|a| < a_1(\tau)$, то (2.17) устойчива при любом n , а если $|a| > a_1(\tau)$, то (2.17) неустойчива при всех достаточно больших n . Аналогично определим $a_2(\tau)$ как границу устойчивости (2.18) для больших n .

В таблице 2.1 представлены результаты анализа устойчивости систем (2.17), (2.18) с помощью программного продукта «Устойчивость нейронных сетей».

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

Не столь очевидно поведение систем (2.17), (2.18) при $\tau \rightarrow 0$, которое рассматривается в следующей теореме.

Теорема 2.2.

$$\lim_{\tau \rightarrow 0} a_1(\tau)\sqrt{2\tau} = \lim_{\tau \rightarrow 0} a_2(\tau)2\sqrt{\tau} = 1. \quad (2.19)$$

Доказательство теоремы 2.2 дано в разделе 3.6. Таблица 2.1 подтверждает оценки теоремы 2.2. Действительно, согласно таблице 2.1, при $\tau = 0.01$ имеем $a_1(\tau)\sqrt{2\tau} \simeq 1.0017$, $a_2(\tau)2\sqrt{\tau} \simeq 1.0033$.

Таблица 2.1. Границы областей устойчивости для систем (2.17), (2.18)

| τ | 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 |

Для построения областей устойчивости и неустойчивости исследуемых моделей в плоскости параметров (рис. 2.7, 2.8) и составления таблицы 2.1 потребовалось многократное применение программы «Устойчивость нейронных сетей», что послужило толчком к созданию нового программного продукта, который будет подробно описан в главе 3 настоящей диссертации.

2.4 Модели кольцевых сетей нейронов с неединичным коэффициентом демпфирования

До сих пор мы рассматривали модели кольцевых систем нейронов с единичным коэффициентом при $x_j(t)$. В случае, когда этот коэффициент не равен единице, соответствующей заменой переменных мы можем перейти к

рассмотренным системам (2.5), (2.6) и (2.5), (2.7). В данном разделе мы укажем необходимую замену переменных, опишем способ диагностирования устойчивости сетей с неединичным коэффициентом демпфирования и сравним области устойчивости таких сетей при различных коэффициентах демпфирования.

Итак, рассмотрим следующий аналог системы (2.5), (2.6) с $\gamma \in \mathbb{R}$:

$$\dot{x}_j(t) + \gamma x_j(t) + a x_{j-1}(t) + b x_{j+1}(t - \tau) = 0 \quad (j \bmod n). \quad (2.20)$$

Для перехода от уравнения (2.20) к уравнению вида (2.5), (2.6) с единичным коэффициентом при $x_j(t)$ и некоторыми другими коэффициентами при $x_{j-1}(t)$ и $x_{j+1}(t - \tau)$ и новым запаздыванием изменим масштаб времени заменой $t = s\tau$, $s > 0$.

В новых переменных уравнение (2.20) примет вид:

$$\frac{dx_j(s/\gamma)}{d(s/\gamma)} + \gamma x_j\left(\frac{s}{\gamma}\right) + a x_{j-1}\left(\frac{s}{\gamma}\right) + b x_{j+1}\left(\frac{s}{\gamma} - \tau\right) = 0 \quad (j \bmod n). \quad (2.21)$$

Обозначив $x_l(s/\gamma) = y_l(s)$, получим

$$\gamma \frac{dy_j(s)}{ds} + \gamma y_j(s) + a y_{j-1}(s) + b y_{j+1}(s - \gamma\tau) = 0 \quad (j \bmod n). \quad (2.22)$$

Переобозначая $s = t$, перейдем к уравнению с единичным коэффициентом при $y_j(t)$:

$$\dot{y}_j(t) + y_j(t) + \frac{a}{\gamma} y_{j-1}(t) + \frac{b}{\gamma} y_{j+1}(t - \gamma\tau) = 0 \quad (j \bmod n), \quad \gamma \neq 0. \quad (2.23)$$

Система (2.20) устойчива тогда и только тогда, когда устойчива система (2.23), для которой мы можем применить разработанные нами алгоритмы диагностирования устойчивости.

Воспользуемся соотношениями (2.12), заменяя a на a/γ , b на b/γ , а также τ на $\gamma\tau$. Получим систему точек $M'_{j\gamma} = (v'_{1j}, v'_{2j}, v'_{3j}) \in \mathbb{R}^3$, определенных равенствами

$$v'_{1j} + iv'_{2j} = \tau b \exp(i(-\frac{2\pi j}{n} + a\tau \sin \frac{2\pi j}{n})), \quad v'_{3j} = \tau(\gamma + a \cos \frac{2\pi j}{n}), \quad 1 \leq j \leq n. \quad (2.24)$$

В случае конечного n необходимо проверить принадлежность данных точек конусу устойчивости. Система (2.23) асимптотически устойчива тогда и только тогда, когда все n точек $M'_{j\gamma}$ (2.24) лежат внутри конуса устойчивости. Если хотя бы одна из точек $M'_{j\gamma}$ лежит вне конуса, то система (2.23) неустойчива.

В случае неограниченного количества нейронов в сети требуется проверить принадлежность конусу устойчивости кривой $M_\gamma(t) = (v'_1(t), v'_2(t), v'_3(t))$, построенной аналогично (2.14):

$$v'_1(t) + iv'_2(t) = \tau b \exp(i(-t + a\tau \sin t)), \quad v'_3(t) = \tau(\gamma + a \cos t), \quad 0 \leq t \leq 2\pi. \quad (2.25)$$

Если все точки кривой (2.25) лежат внутри конуса устойчивости, то система (2.23) устойчива при любом n , а если хотя бы одна точка кривой (2.25) лежит вне конуса устойчивости, то система (2.23) неустойчива при всех достаточно больших значениях n .

На рисунке 2.9 представлены области устойчивости и неустойчивости системы (2.20) при $\tau = 0.5$ и различных значениях коэффициента γ . Границы областей получены с помощью многократного применения критерия определения устойчивости системы (2.23) (а значит, и системы (2.20)) при неограниченном количестве нейронов, описанного в данном разделе. Для этого применялся исходный код программы «Устойчивость нейронных сетей» с учетом необходимых изменений в задании кривой, указанных в (2.25).

Средняя кривая была получена также в разделе 2.3 и ограничивает область устойчивости исследуемого уравнения при $\gamma = 1$. Остальная часть плос-

кости (a, b) соответствует неустойчивости изучаемого уравнения. Эту нейронную сеть также описывает система (2.5), (2.6), подробно изученная в данной главе.

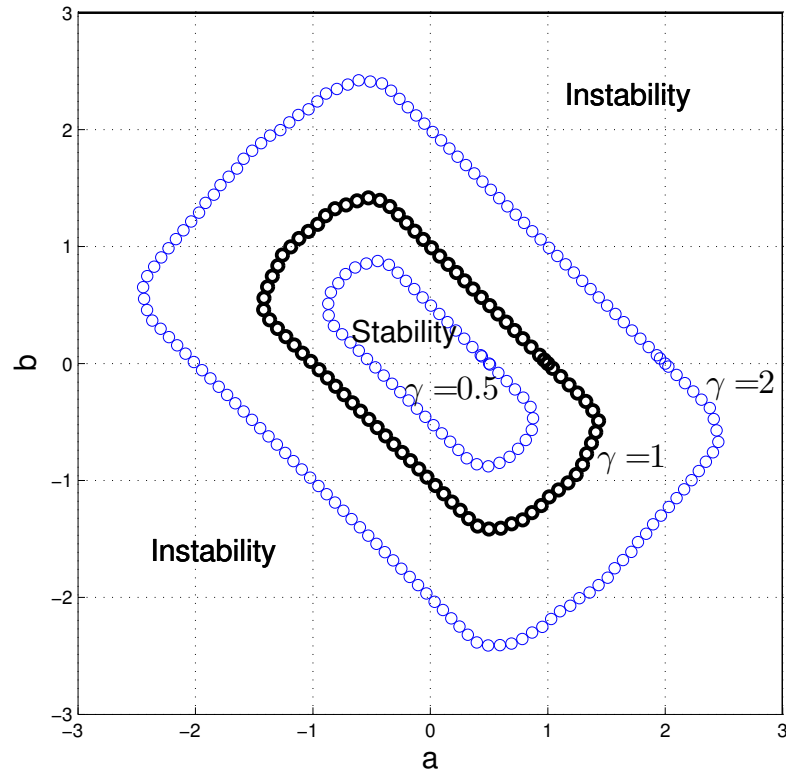


Рис. 2.9. Области устойчивости для системы (2.20) при различных коэффициентах γ

Две другие кривые (при $\gamma = 0.5$ и $\gamma = 2$) находятся внутри и снаружи известной нам границы для $\gamma = 1$. Как видно, соответствующие им области устойчивости сужаются (при $0 < \gamma < 1$) или расширяются (при $\gamma > 1$) относительно основной кривой для $\gamma = 1$.

Таким образом, введение некоторого не равного единице коэффициента γ при $x_j(t)$ не мешает нам применять ранее разработанный алгоритм и не вносит принципиальных изменений в структуру областей устойчивости моделей, что позволяет нам, не теряя общности, проводить анализ устойчивости известных нам уравнений (2.5), (2.6) и (2.5), (2.7).

2.5 Разрыв в кольце: модель сети линейной конфигурации с большим количеством нейронов

Выясним, что происходит с устойчивостью, когда разорвана связь между, скажем, первым и последним нейронами в модели кольцевой сети нейронов (рисунок 2.10).

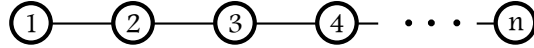


Рис. 2.10. Линейная система нейронов.

В этом случае система (2.5), (2.6) переходит в некоторое уравнение вида (2.4) 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 (2.5), (2.7), подвергнутой разрыву одной связи между нейронами. Итак, рассмотрим следующий аналог модели (2.5), (2.7):

$$\begin{aligned} \dot{x}_1(t) + x_1(t) + b x_2(t - \tau) &= 0, \\ \dot{x}_j(t) + x_j(t) + a x_{j-1}(t - \tau) + b x_{j+1}(t - \tau) &= 0, \quad 2 \leq j \leq (n - 1), \\ \dot{x}_n(t) + x_n(t) + a x_{n-1}(t - \tau) &= 0, \quad n > 2. \end{aligned} \quad (2.26)$$

Матричная форма уравнения (2.26) такова:

$$\dot{x}(t) + I x(t) + D x(t - \tau) = 0, \quad (2.27)$$

где $n \times n$ матрица D имеет вид

$$D = \begin{pmatrix} 0 & b & 0 & \dots & 0 & 0 \\ a & 0 & b & \dots & 0 & 0 \\ 0 & a & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & b \\ 0 & 0 & 0 & \dots & a & 0 \end{pmatrix}. \quad (2.28)$$

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

$$F(\tau) = \frac{1}{4 \sin^2 \omega(\tau)}, \quad (2.29)$$

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

$$\tau = \omega \operatorname{tg} \omega. \quad (2.30)$$

Теорема 2.3. 1. Если $0 \leq ab < \frac{1}{4}$, то система (2.27), (2.28) асимптотически устойчива при любом n и любом $\tau \geq 0$.

2. Если $ab > \frac{1}{4}$, то система (2.27), (2.28) неустойчива при любом $\tau \geq 0$, если n достаточно велико.

3. Если $ab < 0$ и $|ab| < F(\tau)$, то система (2.27), (2.28) асимптотически устойчива при любом n .

4. Если $ab < 0$ и $|ab| > F(\tau)$, то система (2.27), (2.28) неустойчива, если n достаточно велико.

Доказательство Теоремы 2.3 дано в разделе 2.6.

Теорема 2.3 иллюстрируется рисунком 2.11. Сравнивая рисунки 2.8 и 2.11, мы видим, что при разрыве кольца нейронов область устойчивости в пространстве параметров увеличивается, если количество нейронов достаточно велико. Это подтверждается следующей теоремой.

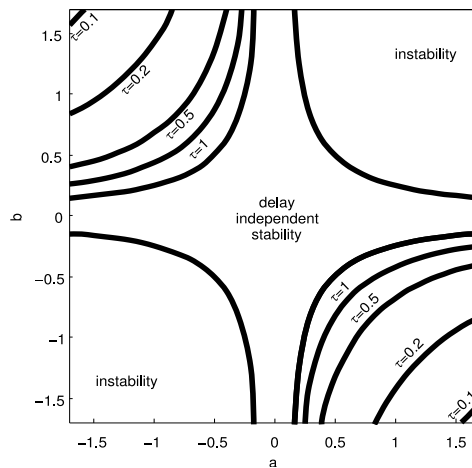


Рис. 2.11. Границы области устойчивости системы (2.27), (2.28) для линейной конфигурации при достаточно большом количестве нейронов n .

Теорема 2.4. *Для любых действительных значений параметров a , b , $\tau > 0$ найдется такое n_0 , что для всех $n > n_0$ верно утверждение: либо система (2.5), (2.7) для кольца нейронов неустойчива, либо система (2.27), (2.28) для сети с линейной конфигурацией нейронов асимптотически устойчива.*

Доказательство Теоремы 2.4 дано в разделе 2.6. В следующей главе мы покажем, что при малом количестве нейронов в кольцевой системе разрыв кольца может, наоборот, сделать устойчивую кольцевую систему неустойчивой. Но это явление редкое (в житейском смысле этого слова), и автору пришлось предпринять специальные усилия, чтобы его обнаружить.

Ещё одно наблюдение: в модели (2.27), (2.28) при $a = 0$ (или $b = 0$) Теорема 2.3 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 2.1 и Рисунки 2.7, 2.8).

Рассмотрим вариант $a = -b$ в (2.27), (2.28). Положим $H(\tau) = \sqrt{F(\tau)}$ (см. (2.29), (2.30)). Очевидно, $\lim_{\tau \rightarrow \infty} H(\tau) = 1/2$ и $\lim_{\tau \rightarrow 0} H(\tau)2\sqrt{\tau} = 1$. В системе (2.27), (2.28) с $a = -b$ функция $H(\tau)$ играет роль границы устойчивости, аналогичную функции $a_2(\tau)$ для системы (2.18) (см. Определение 2.1). Сравнение поведения $H(\tau)$ с $a_2(\tau)$ (см. (2.19)) 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 (2.27), (2.28).

2.6 Доказательства теорем главы 2

2.6.1 Доказательство Теоремы 2.1

Начнем доказательство с замечания, что неравенство

$$u_1 + u_3 \geq 0 \quad (2.31)$$

верно на поверхности конуса устойчивости (1.21), (1.22) (см. определение 1.2) и внутри него. Из (2.14) 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. (2.31)), поэтому при $a + b > 1$ система (2.5), (2.6) неустойчива, если n достаточно велико.

Преобразование $t \rightarrow t + \pi, a \rightarrow -a, b \rightarrow -b$ переводит кривую (2.14) в себя, поэтому область устойчивости системы (2.5), (2.6) центрально-симметрична. Следовательно, и неравенство $a + b < -1$ обеспечивает неустойчивость системы (2.5), (2.6) при достаточно больших n . Аналогично, положив в (2.15) $t = \pi/2$, докажем неустойчивость (2.5), (2.7) при достаточно больших n и $|a + b| > 1$. Теорема 2.1 доказана.

2.6.2 Доказательство Теоремы 2.2

1. Для выяснения асимптотики $a_1(\tau)$ требуется вычислить точки касания в \mathbb{R}^3 конуса устойчивости с кривой (2.14) при $a = a_1(\tau)$, $b = -a_1(\tau)$, $t = \pi/2$. На кривой (2.14)

$$\arg(u_1'(\frac{\pi}{2}) + iu_2'(\frac{\pi}{2})) - \frac{\pi}{2} = \tau a_1(\tau). \quad (2.32)$$

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

$$\arg(u_1 + iu_2) - \frac{\pi}{2} = -\omega + \arg(i\omega - \tau) - \frac{\pi}{2} \sim \frac{\tau}{\omega} - \omega. \quad (2.33)$$

Здесь $\alpha \sim \beta$ означает, что $(\alpha/\beta) \rightarrow 1$. Касание кривой и конуса влечет равенство аргументов (2.32), (2.33), а также равенство $|u_1 + iu_2| = |u_1'(\pi/2) + iu_2'(\pi/2)|$. Поэтому

$$\tau a_1(\tau) \sim \frac{\tau}{\omega} - \omega, \quad \tau a_1(\tau) = \sqrt{\omega^2 + \tau^2} \sim \omega. \quad (2.34)$$

Из (2.34) 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 (2.15). Получим сегмент

$$u_1''(t) = 0, \quad u_2''(t) = 2\tau a_2(\tau) \sin t, \quad u_3''(t) = 0, \quad 0 \leq t \leq 2\pi. \quad (2.35)$$

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

$$u_2 = \omega \cos \omega + \tau \sin \omega \sim \sqrt{\tau}. \quad (2.36)$$

Из (2.36) и (2.35) при $t = \pi/2$ получим $a_2(\tau) \sim 1/(2\sqrt{\tau})$. Теорема 2.2 доказана.

2.6.3 Доказательство Теоремы 2.3

Начнем доказательство Теоремы 2.3 с леммы.

Лемма 2.1. Для любого натурального n собственные числа $n \times n$ матрицы D (см. (2.28)) суть

$$\mu_{jn} = 2\sqrt{ab} \cos \frac{\pi j}{n+1}, \quad 1 \leq j \leq n. \quad (2.37)$$

Доказательство. Характеристический многочлен матрицы D (см. (2.28)) $D_n(\mu) = |\mu I - D|$ удовлетворяет соотношениям ($n = 1, 2, \dots$)

$$D_1(\mu) = \mu, \quad D_2(\mu) = \mu^2 - ab, \quad D_{n+2}(\mu) = \mu D_{n+1}(\mu) - ab D_n(\mu). \quad (2.38)$$

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

$$\mu = 2y\sqrt{ab}, \quad D_n(2y\sqrt{ab}) = (\sqrt{ab})^n U_n(y). \quad (2.39)$$

Из (2.38) и (2.39) выведем

$$U_1(y) = 2y, \quad U_2(y) = 4y^2 - 1, \quad U_{n+2}(y) = 2yU_{n+1}(y) - U_n(y), \quad n = 1, 2, \dots \quad (2.40)$$

Равенства (2.40) описывают многочлены Чебышева второго рода $U_n(y)$, нули которых таковы:

$$y_{jn} = \cos \frac{\pi j}{n+1}, \quad 1 \leq j \leq n. \quad (2.41)$$

Из (2.41) и (2.39) получим требуемое. Лемма 2.1 доказана. ■

Продолжим доказательство Теоремы 2.3. Собственные числа $n \times n$ матрицы E равны $\lambda_{jn} = 1$. Поэтому из (2.37) получим при $|ab| < 1/4$

$$\frac{|\mu_{jn}|}{\operatorname{Re} \lambda_{jn}} = 2\sqrt{ab} \left| \cos \frac{\pi j}{n+1} \right| < 1, \quad (2.42)$$

что достаточно для асимптотической устойчивости, независимой от запаздывания (см. Теорему 1.5). Пункт 1 Теоремы 2.3 доказан.

Для доказательства пунктов 2-4 Теоремы 2.3 построим точки $M_{jn} = (u_{1jn}, u_{2jn}, u_{3jn}) \in \mathbb{R}^3$ ($1 \leq j \leq n$), такие что (см. (2.37))

$$\begin{aligned} u_{1jn} + iu_{2jn} &= \tau \mu_{jn} \exp(i\tau \operatorname{Im} \lambda_{jn}) = 2\tau \sqrt{ab} \cos \frac{\pi j}{n+1}, \\ u_{3jn} &= \tau \operatorname{Re} \lambda_{jn} = \tau. \end{aligned} \quad (2.43)$$

Пусть $ab > 1/4$. Тогда по (2.43) найдётся такое n_0 , что для любого $n > n_0$

$$u_{1nn} = 2\tau \sqrt{ab} \cos \frac{\pi n}{n+1} < -\tau. \quad (2.44)$$

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

$$u_1 \geq -u_3 = -\tau. \quad (2.45)$$

Сравнивая (2.45) с (2.44), выясняем, что точка M_{nn} при $n > n_0$ лежит вне конуса устойчивости. Поэтому система (2.27), (2.28) неустойчива при достаточно больших n . Пункт 2 Теоремы 2.3 доказан.

Из (2.43) и Леммы 2.1 при $ab < 0$ получим

$$u_{1jn} = 0, \quad u_{2jn} = 2\tau \sqrt{|ab|} \cos \frac{\pi j}{n+1}, \quad u_{3jn} = \tau. \quad (2.46)$$

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

$$|u_2| = \frac{\omega}{\cos \omega}, \quad (2.47)$$

где параметр ω ввиду $u_1 = 0$ удовлетворяет условию (2.30). Теперь из (2.46), (2.29) и (2.30) получим

$$u_{2jn} = \tau \cos \frac{\pi j}{n+1} \sqrt{\frac{|ab|}{F(\tau) \sin \omega}}, \quad (2.48)$$

Если $|ab| < F(\tau)$, то из (2.48) и неравенства $\mu_{jn} < 2$ (см. Лемму 2.1) вытекает $|u_{2jn}| < |u_2|$ при любых j, n , поэтому все точки M_{jn} лежат внутри

конуса устойчивости, следовательно, система (2.27), (2.28) асимптотически устойчива при любых n . Пункт 3 Теоремы 2.3 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 (2.46) отсюда следует, что

$$u_{21n} > 2\tau\sqrt{F(\tau)}. \quad (2.49)$$

Из (2.49), (2.29) и (2.30) получим $u_{21n} > \frac{\omega}{\cos \omega}$, что ввиду (2.47) дает $u_{21n} > u_2$. Поэтому точка M_{nn} находится вне конуса устойчивости, что влечет неустойчивость системы (2.27), (2.28) при всех $n > n_0$. Теорема 2.3 доказана.

2.6.4 Доказательство Теоремы 2.4

Доказательство. Фиксируем $a, b, \tau > 0$. Рассмотрим эллипс, лежащий в плоскости комплексного переменного z :

$$z = \tau(a + b) \cos \theta + i\tau(a - b) \sin \theta, \quad 0 \leq \theta < 2\pi. \quad (2.50)$$

Его вершины имеют вид

$$z_1 = \tau|a + b|, \quad z_2 = -|a + b|, \quad z_3 = i\tau|a - b|, \quad z_4 = -i|a - b|. \quad (2.51)$$

СЛУЧАЙ 1. Одна из вершин (2.51) лежит вне овала устойчивости на высоте τ . Поскольку собственные числа μ'_{jn} матрицы $aP + bP^{n-1}$ даются формулой $\mu'_{jn} = (a + b) \cos \frac{2\pi j}{n} + i(a - b) \sin \frac{2\pi j}{n}$, в соответствии с теоремой 1.4 построим точки $M'_{jn} = (u'_{1jn}, u'_{2jn})$, такие что

$$u'_{1jn} + iu'_{2jn} = \tau(a + b) \cos \frac{2\pi j}{n} + i\tau(a - b) \sin \frac{2\pi j}{n}. \quad (2.52)$$

Все точки, определенные посредством (2.52), лежат на эллипсе (2.50) и при неограниченном увеличении n плотно его заполняют. Поэтому найдется такое n_0 , что при всех $n > n_0$ найдется такое j , ($1 \leq j \leq n$), что точка M'_{jn}

находится вне овала устойчивости. Поэтому для данных a, b, τ система (2.5), (2.7) для нейронного кольца неустойчива.

СЛУЧАЙ 2. Все вершины (2.51) лежат внутри овала устойчивости на высоте τ или на его границе. Поскольку собственные числа μ_{jn}'' матрицы D (см. (2.28)) равны $\mu_{jn}'' = 2\sqrt{ab} \cos \frac{\pi j}{n+1}$ (см. Лемму 2.1), в соответствии с теоремой 1.4 построим точки $M_{jn}'' = (u_{1jn}'', u_{2jn}'')$, такие что

$$u_{1jn}'' + iu_{2jn}'' = 2\tau\sqrt{ab} \cos \frac{\pi j}{n+1}. \quad (2.53)$$

Ясно, что устойчивость системы (2.27), (2.28) зависит от положения двух точек $z'' = \pm 2\tau\sqrt{ab} \cos \frac{\pi}{n+1}$ относительно овала устойчивости. Если $ab \geq 0$, то из очевидного неравенства $2\sqrt{ab} \leq a + b$ и того, что точки z_1, z_2 лежат внутри овала устойчивости, следует, что обе точки $z'' = \pm 2\tau\sqrt{ab} \cos \frac{\pi}{n+1}$ лежат внутри овала устойчивости, и поэтому система (2.27), (2.28) асимптотически устойчива. Если $ab < 0$, то из неравенства $2\sqrt{|ab|} \leq |a - b|$ и того, что точки z_3, z_4 лежат внутри овала устойчивости, следует, что обе точки $z'' = \pm 2\tau\sqrt{ab} \cos \frac{\pi}{n+1}$ лежат внутри овала устойчивости, и поэтому система (2.27), (2.28) также асимптотически устойчива. Теорема 2.4 доказана. ■

2.7 Сравнение результатов главы 2 с известными результатами

Насколько известно автору, модели кольцевых нейронных сетей с неопределенно большим количеством нейронов не рассматривались в литературе до работы [?] автора диссертации совместно с научным руководителем.

Благодаря рассмотрению большого количества нейронов изложение решения задачи об устойчивости кольцевой сети в диссертации является доста-

точно прозрачным. Вот для сравнения цитата из работы [?] (2004) Ю. Юан и С. Кэмпбелл:

«Области устойчивости при нечетных n сходны с областями при $n = 3$, которые были исследованы в [?] (здесь ссылка на рукопись, которая по каким-то причинам опубликована двумя годами позднее), поэтому мы сосредоточимся на области устойчивости для четных n .»

Таким образом, про области устойчивости нейронных сетей с нечетным количеством нейронов из работ Кэмпбелл с соавторами можно только узнать, что они похожи на область устойчивости для трехнейронного кольца. Естественно, перейти к пределу при количестве нейронов, стремящемся к бесконечности, на основе такой информации невозможно.

Предпринятое в настоящей главе изучение устойчивости моделей линейных конфигураций нейронов не встречается в литературе (кроме работ автора диссертации), хотя вообще публикации по линейным нейронным сетям имеются (например, [?]). Тема устойчивости линейных сетей вполне естественна, так как линейная конфигурация естественно появляется при разрыве кольцевой.

Область устойчивости, независимой от запаздывания, очерченная пунктами 1,3 Теоремы 2.3, шире области $|a| + |b| < 1$, гарантированной достаточным условием (2.16), вытекающим из результата Мори с соавторами [?].

Алгоритмы и программы для анализа устойчивости моделей кольцевых и линейных сетей с неограниченным количеством нейронов в известной автору литературе отсутствуют.

Глава 3

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

3.1 Алгоритм и программа для построения границ областей устойчивости моделей кольцевых нейронных сетей с ограниченным количеством нейронов

В этом разделе мы опишем алгоритм и программу для построения границ области устойчивости моделей кольцевых нейронных сетей с ограниченным количеством нейронов. Программа использует фрагменты предыдущих программ «Анализ устойчивости» (раздел 1.5) и «Устойчивость нейронных сетей» (раздел 2.1).

Мы рассматриваем задачу устойчивости модели (см. формулы (2.2), (2.3) соответственно)

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

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

В дальнейшем кольцевые сети, описываемые уравнением (3.1), будем называть сетями с односторонним запаздыванием, а сети, описываемые уравнением (3.2), — сетями с двусторонним запаздыванием.

3.1.1 Алгоритм нахождения границы области устойчивости

Для построения границ областей устойчивости и неустойчивости данных моделей в плоскости (a, b) применяется отдельный алгоритм. Он предназначен для построения замкнутой кривой, состоящей из равномерного массива точек R_k и соответствующей границе области устойчивости круговых нейронных сетей, с погрешностью δ . Априорно предполагается, что границей области устойчивости является замкнутая кривая, причём область устойчивости находится внутри её.

Данный алгоритм проиллюстрирован на следующем рисунке 3.1.

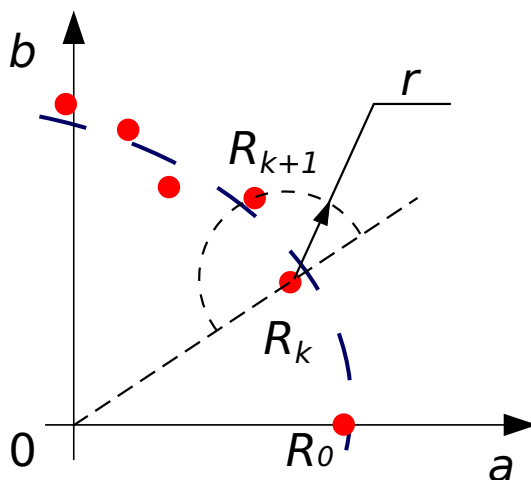


Рис. 3.1. Схема нахождения точек в программе «Построение областей устойчивости круговых нейронных сетей»

Вначале находим первую точку границы, которая образуется при пересечении этой границы с осью $0a$. Для этого сначала определяем две точки на луче $0a$: одну внутри области устойчивости, а другую вне её. В качестве первой точки подходит точка с координатами $(0, 0)$, так как она, очевидно, соответствует устойчивости уравнения 1.6. Вторую точку найдём, двигаясь по лучу $0a$ с шагом 1 (инкрементируя параметр a на единицу) до тех пор, пока точка не окажется вне области устойчивости. Диагностирование принадлежности точек области устойчивости проводится с помощью отдельного

алгоритма, описанного в следующем пункте. Затем методом дихотомии уточняем координаты данной точки до заданной точности и тем самым получаем точку R_0 .

Все остальные точки находятся следующим образом:

1. Переходим в полярную систему координат с центром в только что найденной точке R_k , так что её ось совпадает с осью OR_k .
2. Ищем следующую точку R_{k+1} на расстоянии заданной погрешности δ от точки R_k , проводя итерацию по следующему алгоритму.

Определяем начальные параметры:

$$r = \delta, \phi_0 = 0, \phi_1 = \pi.$$

Так как точка R_k является граничной, то точка (r, ϕ_0) будет лежать вне области устойчивости, а точка (r, ϕ_1) внутри её. Следующую точку R_{k+1} найдём половинным делением данной полуокружности (а именно угла ϕ_s), пока не будет достигнута заданная точность.

Шаг 2 проводим до тех пор, пока кривая не замкнётся.

3.1.2 Алгоритм исследования устойчивости фиксированных точек плоскости параметров

Хотя данная задача уже была решена в программе «Анализ устойчивости» (раздел 2.2), мы намеренно отказались от данного алгоритма. Это связано с тем, что алгоритм решает более глобальную задачу (нахождения всех значений τ , обеспечивающих устойчивость исследуемого уравнения) и является затратным по времени. Время вычисления принадлежности точек области устойчивости в данном случае является критичным, так как для получения границы области устойчивости с заданной точностью требуется

исследовать достаточно большое число точек плоскости (a, b) . Поэтому все исследуемые точки плоскости параметров диагностируются на устойчивость непосредственно с помощью метода годографов, реализованного с помощью функции *straightStabAnalyzer* (см. Приложение В).

1. Для каждой исследуемой пары значений a, b вычисляем собственные числа λ'_j и μ'_j ($j = 1 \dots n$) по формулам (2.10) для уравнения (3.1) и λ''_j и μ''_j ($j = 1 \dots n$) по формулам (2.11) для уравнения (3.2). С их помощью находим координаты точек M'_j по формулам (2.12) для уравнения (3.1) и точек M''_j по формулам (2.13) для уравнения (3.2).
2. Далее требуется определить, находятся ли все n точек M'_j внутри конуса устойчивости или вне его. Для этого прежде всего выполняем простую проверку на непринадлежность конусу: если хотя бы для одного j выполнено неравенство $u'_{3j} < -1$, то точка M'_j находится ниже вершины конуса, что соответствует неустойчивости уравнения (3.1) по теореме 1.4. Дальнейшее выполнение алгоритма прекращается.
3. Если для всех j выполнено неравенство $u'_{3j} \geq -1$, переходим к проверке принадлежности точек M'_j конусу устойчивости. Это трёхмерная задача, которую мы можем свести к двумерной сечением конуса устойчивости плоскостью, параллельной плоскости (u_1, u_2) и находящейся на высоте u'_{3j} . Тогда остаётся определить, находится ли точка внутри овала устойчивости или нет.
4. Для этого для каждого j строим овал устойчивости на уровне u'_{3j} и находим на нём точку, максимально близкую к точке M'_j при заданной точности. Если расстояние от данной точки до оси u_3 меньше расстояния от точки M'_j до этой оси, то исследуемая пара значений a, b является неустойчивой для уравнения (3.1), и алгоритм завершается.

5. Если предыдущий шаг покажет, что все точки M'_j лежат внутри овала устойчивости, то исследуемая пара значений a, b является устойчивой для уравнения (3.1).
6. Аналогично проходит проверка принадлежности конусу устойчивости точек M''_j для исследования уравнения (3.2).

3.1.3 Функциональное назначение и область применения программы

Программа «Построение областей устойчивости круговых нейронных сетей» строит области устойчивости для круговых нейронных сетей (3.1), (3.2) 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 АРУ для создания графического интерфейса пользователя (GUI). Исходный код программы представлен в приложении В.

Программа зарегистрирована в Объединенном фонде электронных ресурсов «Наука и образование» (ОФЭРНиО), информация о ней представлена в [?], а пользовательская версия программы доступна по электронному адресу [?].

3.1.4 Использование

На рисунке 3.2 представлено главное окно программы, в котором задаются входные данные.

Прежде всего пользователю предлагается выбрать конфигурацию нейронной сети. В программе доступны два варианта кругового соединения ней-

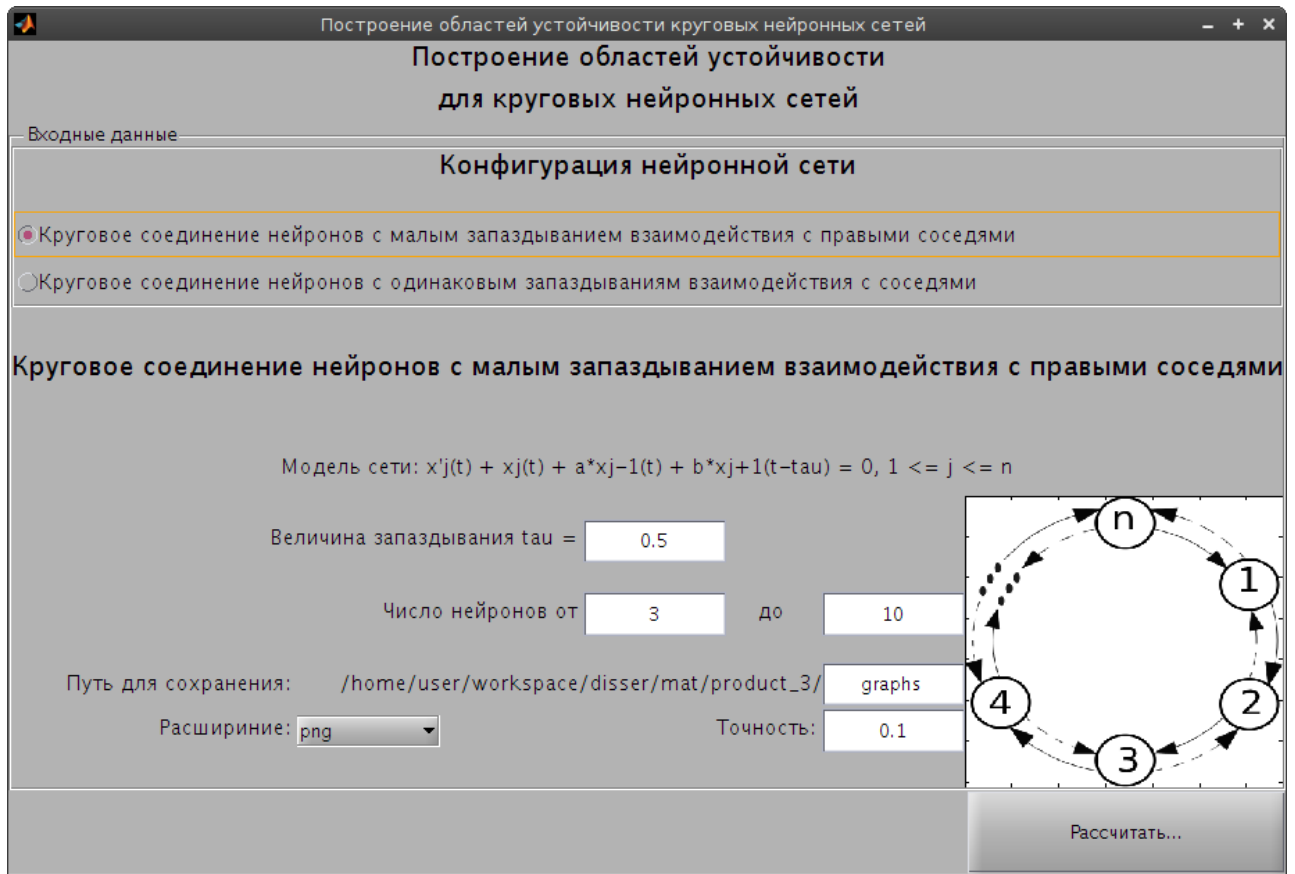


Рис. 3.2. Главное окно программы «Построение областей устойчивости круговых нейронных сетей» при выборе первой конфигурации

ронов: с малым запаздыванием взаимодействия с правыми соседями (уравнение (3.1)) и с одинаковым запаздыванием взаимодействия с правыми и левыми соседями (уравнение (3.2)). Выбирая первую или вторую радиокнопку, пользователь определяет, для какой именно модели будет проводиться анализ устойчивости. На рисунке 3.2 выбрана первая конфигурация, а на рисунке 3.3 — вторая. Ниже, на панели ввода представлено название, описание и рисунок выбранной конфигурации. Пользователю необходимо ввести параметр τ , для которого будут строиться области устойчивости, определить количество нейронов в сети (причём графики будут построены для всех n , определённых пользователем), задать директорию для сохранения, выбрать расширение, с которым будут сохраняться графики, и определить погрешность построения.

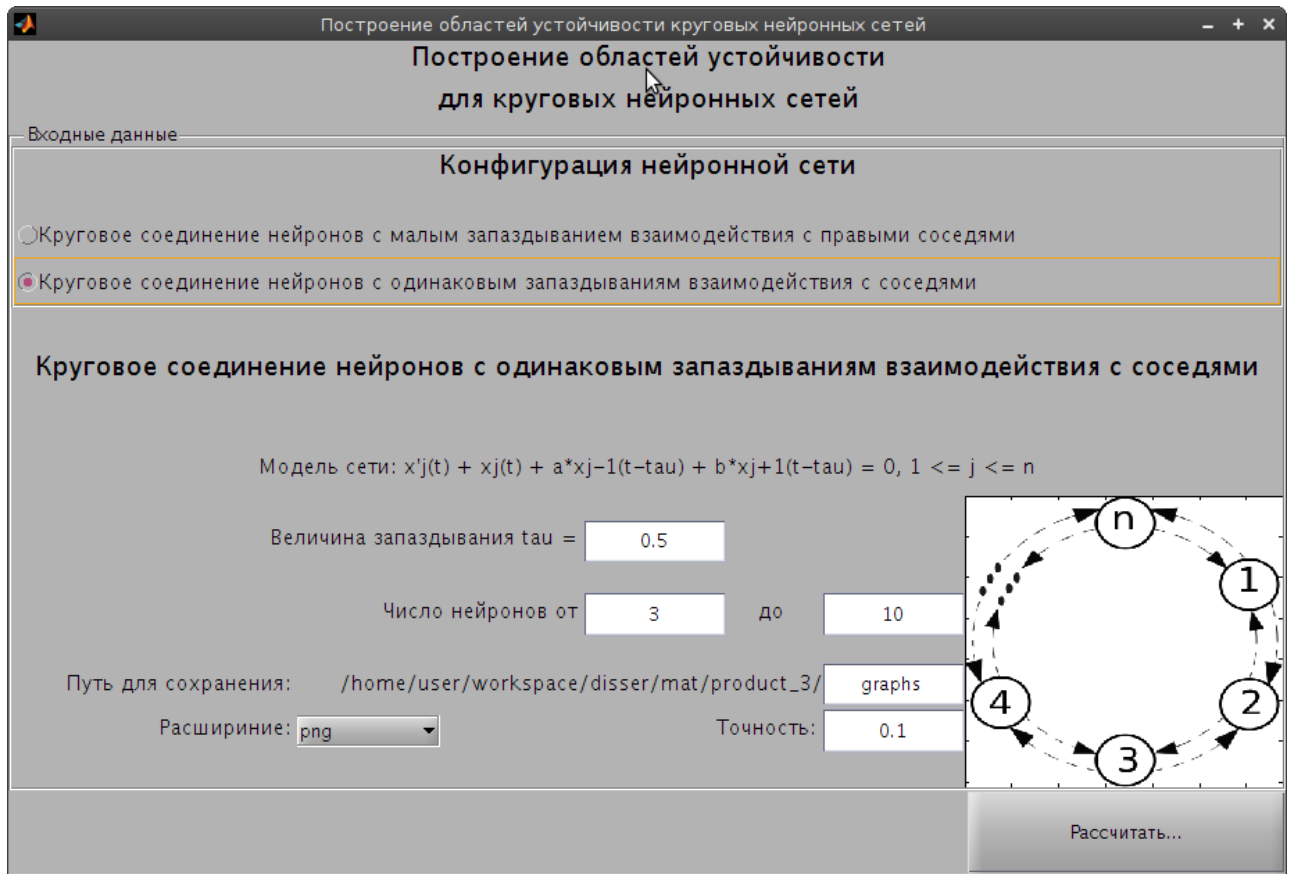


Рис. 3.3. Главное окно программы «Построение областей устойчивости круговых нейронных сетей» при выборе второй конфигурации

После ввода данных в обоих случаях для начала расчета необходимо нажать на кнопку «Рассчитать...». В процессе построения будут появляться вспомогательные окна, представляющие 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;

ext — расширение файла.

Программа также строит область устойчивости для бесконечного числа нейронов.

После проведения расчетов пользователь может изменить входные параметры и ещё раз выполнить построение областей устойчивости. Графики будут сохраняться 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.

При вводе некорректных данных программа выдает различные сообщения об ошибках. 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 3.4.

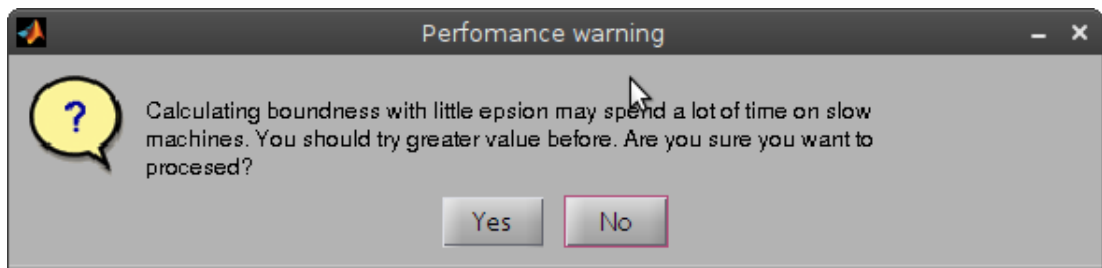


Рис. 3.4. Требование подтверждения необходимости выполнения длительных расчетов

Применение программы «Построение областей устойчивости круговых нейронных сетей» позволило в плоскости параметров построить границы областей устойчивости для систем (3.1), (3.2) с ограниченным количеством нейронов. Полученные области устойчивости и неустойчивости исследуемых моделей для различных значений запаздывания и различного числа нейронов будут представлены в следующих разделах данной главы. Отметим, что исходный код описанной программы с небольшими изменениями может применяться для получения областей устойчивости не только кольцевых нейронных сетей, но и других систем, описываемых уравнением (1.6) с совместно триан-

гулируемыми матрицами. В частности, он применялся автором для построения областей устойчивости полносвязных и звездных структур нейронных сетей (см. [?]), а также двухслойного соединения нейронов (см. [?]).

3.2 Границы областей устойчивости моделей кольцевых сетей с ограниченным количеством нейронов и односторонним запаздыванием

В этом разделе мы укажем результаты поиска границ областей устойчивости для модели (3.1) кольцевой системы нейронов с односторонним запаздыванием. В отличие от предыдущей главы, здесь мы будем рассматривать сети с ограниченным количеством нейронов. Разделим демонстрацию результатов на три группы: сети с малым, средним и большим запаздываниями.

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

На рисунках 3.5, 3.6, 3.7 в плоскости параметров модели (3.1) представлены границы областей устойчивости и неустойчивости соответствующей нейронной сети для $\tau = 0.1$ (малое запаздывание), $\tau = 0.5$ (среднее запаздывание), $\tau = 1$ (большое запаздывание) и различного количества нейронов в сети (от 3 до 8). Результаты получены с помощью программы «Построение областей устойчивости круговых нейронных сетей», описанной в предыдущем разделе.

Сплошной линией на графиках указана граница области устойчивости такой же сети, гарантированной для любого n . Эти границы построены с помощью программы «Устойчивость нейронных сетей», описанной в главе 3 настоящей диссертации.

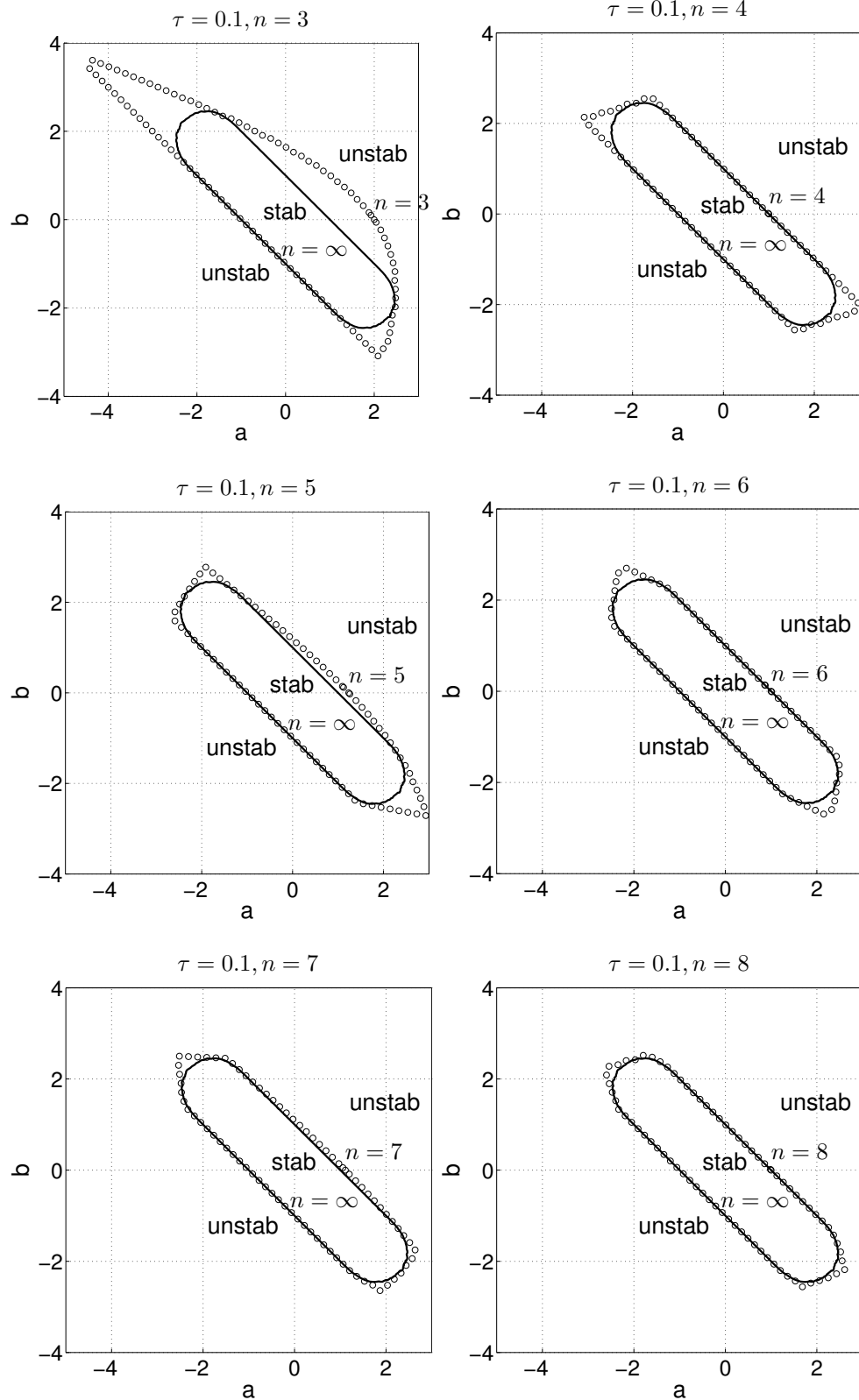


Рис. 3.5. Границы областей устойчивости для системы (3.1) для $\tau = 0.1$ и значений n от 3 до 8 показаны кружками. Сплошная линия — граница области устойчивости, гарантированной для любого n .

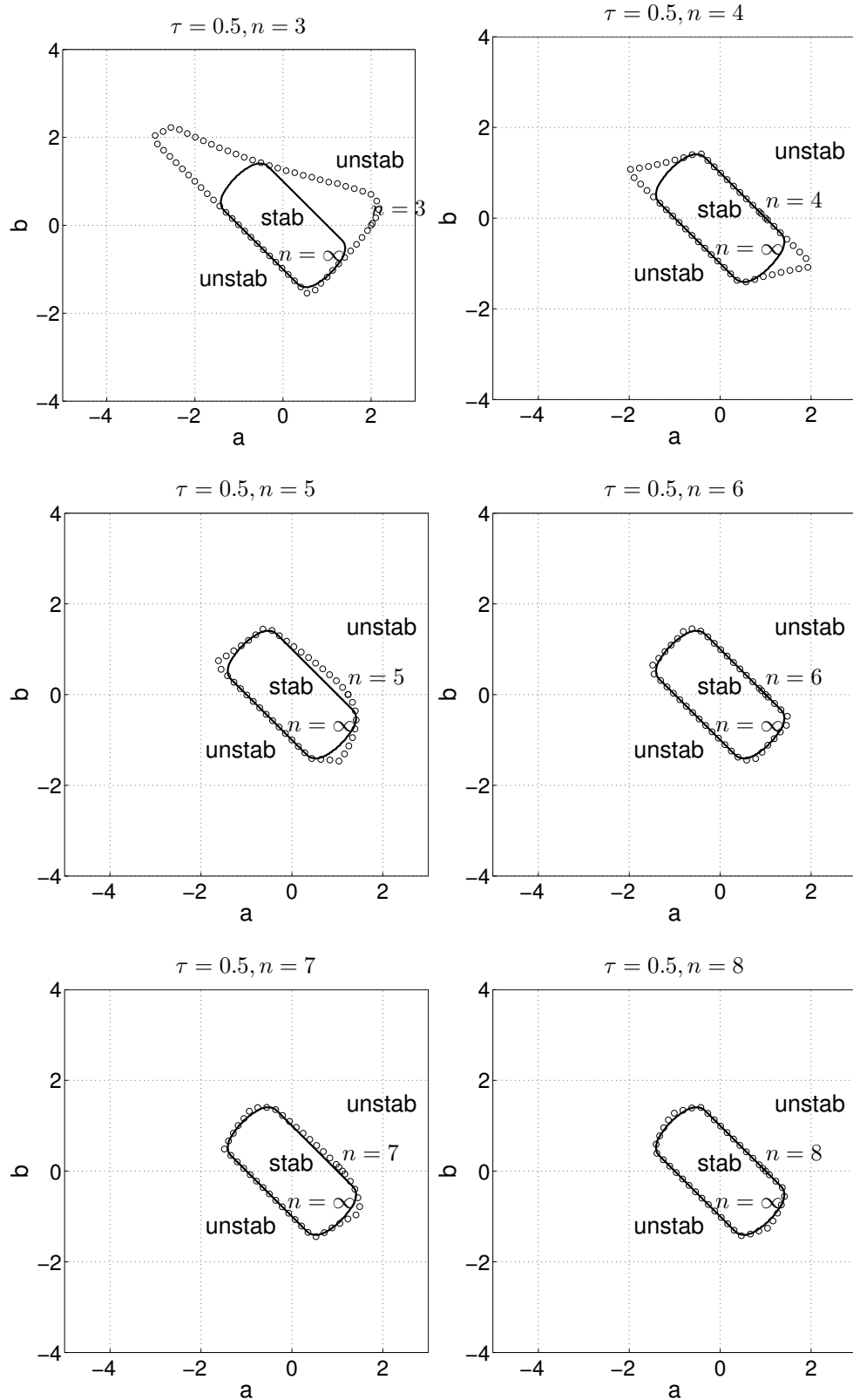


Рис. 3.6. Границы областей устойчивости для системы (3.1) для $\tau = 0.5$ и значений n от 3 до 8 показаны кружками. Сплошная линия — граница области устойчивости, гарантированной для любого n .

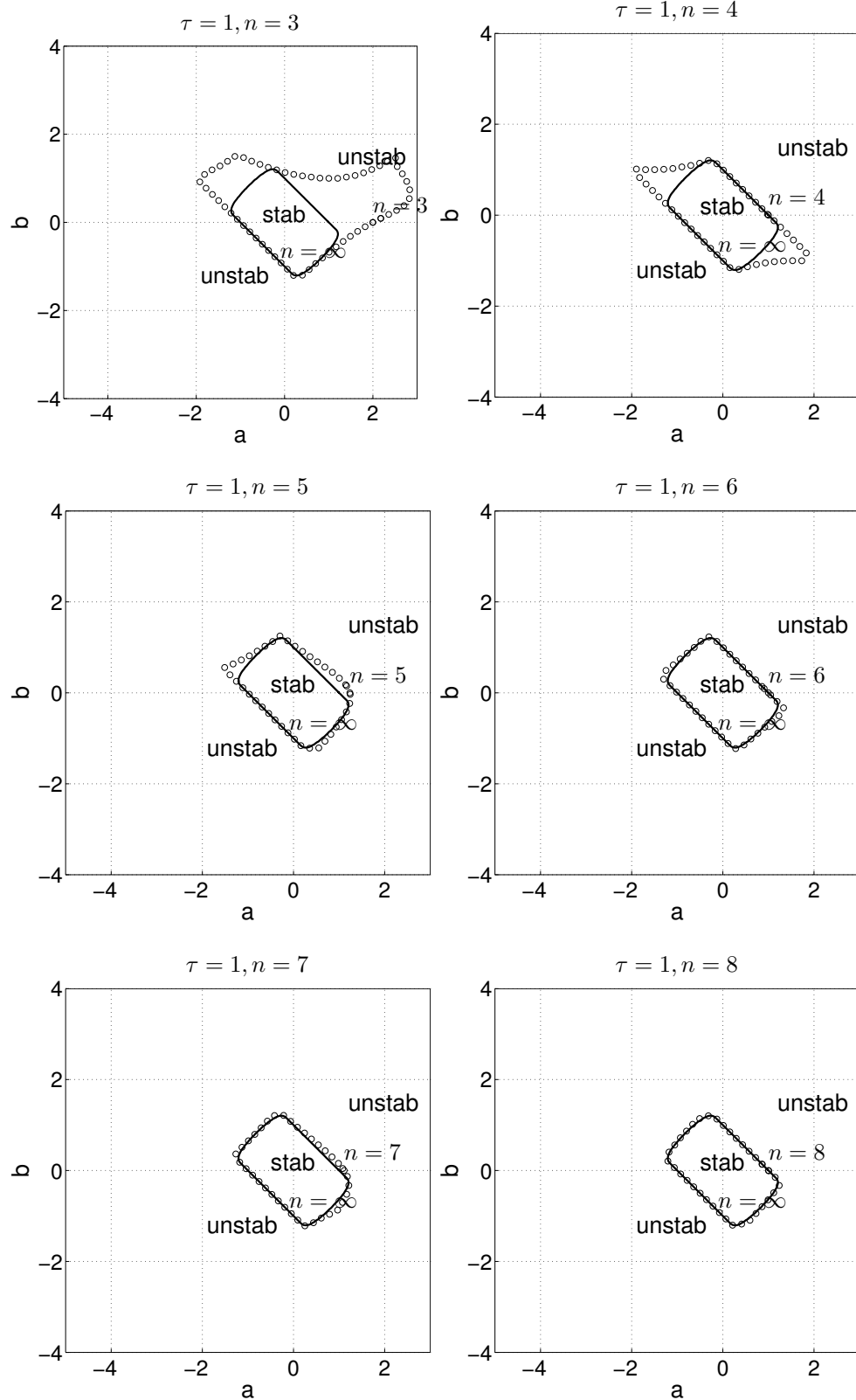


Рис. 3.7. Границы областей устойчивости для системы (3.1) для $\tau = 1$ и значений n от 3 до 8 показаны кружками. Сплошная линия — граница области устойчивости, гарантированной для любого n .

На представленных графиках значения параметров a и b , лежащие в области с подписью «Stability», обеспечивают устойчивость системы (3.1) при данных τ и n . Области с подписью «Instability» соответствуют неустойчивости данной нейронной сети.

Легко заметить, что границы областей устойчивости исследуемой сети для ограниченного количества нейронов с ростом их числа сближаются с границей области устойчивости данной нейронной сети при $n = \infty$. При этом во всех случаях границы области устойчивости для $n = 3$ в наибольшей степени отличаются от соответствующей границы для $n = \infty$ и охватывают достаточно большие участки в первой и второй четвертях плоскости параметров. (эти области с ростом n заметно уменьшаются).

Также нетрудно заметить некоторое родство построенных границ для нечетных n (левый столбец графиков на всех рисунках) и четных n (правый столбец графиков). Кроме того, границы областей устойчивости для четного числа нейронов центрально-симметричны. Эти свойства говорят о непосредственном влиянии числа нейронов в исследуемой сети (при их ограниченном количестве) на её устойчивость. Однако это актуально лишь для небольших сетей, так как с увеличением n соответствующие границы быстро сближаются с границей области устойчивости для неограниченного числа нейронов.

Применение разработанного автором программного продукта позволило быстро построить серии графиков для различных значений запаздывания и различного числа нейронов и существенно помогло при анализе полученных результатов.

3.3 Границы областей устойчивости моделей кольцевых сетей с ограниченным количеством нейронов и двусторонним запаздыванием

В этом разделе мы представим результаты поиска границ областей устойчивости для модели (3.2) кольцевой системы нейронов с двусторонним запаздыванием и ограниченным количеством нейронов. Результаты получены с помощью программы, описанной в первом разделе данной главы.

Как и в предыдущем разделе, полученные области указаны кружками в трёх сериях рисунков для различных значений запаздывания (см. соответственно рисунок 3.8 для $\tau = 0.1$, рисунок 3.9 для $\tau = 0.5$ и рисунок 3.10 для $\tau = 1$).

Как и в случае кольцевых сетей с односторонним запаздыванием, рассмотренных в предыдущем разделе, наблюдается стремление границ областей устойчивости для ограниченного числа нейронов к границе для неограниченного числа нейронов, а также некоторое сходство полученных кривых для нечетного и четного числа нейронов в сети (левый и правый столбцы соответственно). В отличие от сетей с односторонним запаздыванием, области устойчивости для сетей с двусторонним запаздыванием и нечетным числом нейронов не выдаются в первую и вторую четверти и симметричны относительно прямой $a = b$.

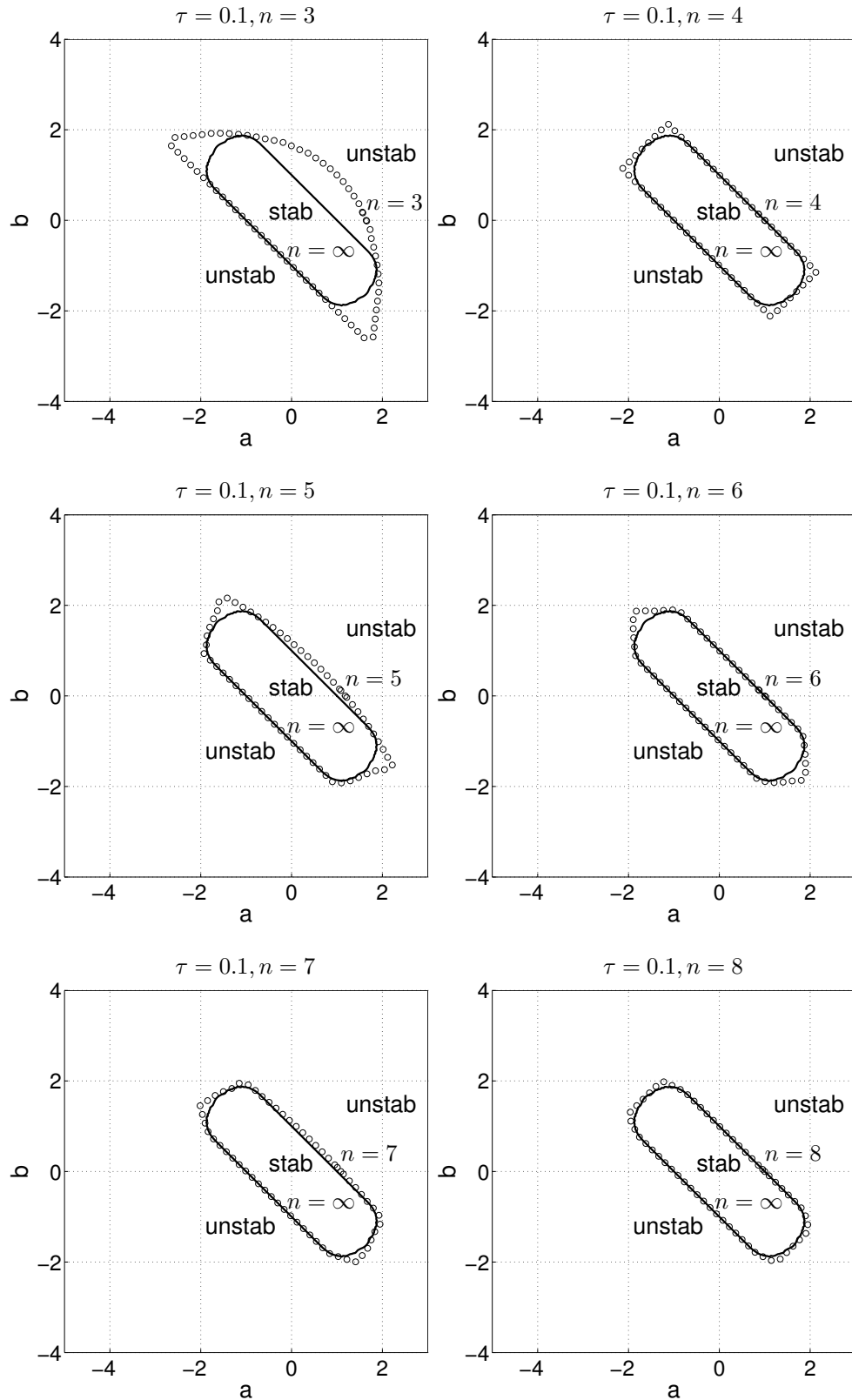


Рис. 3.8. Границы областей устойчивости для системы (3.2) для $\tau = 0.1$ и значений n от 3 до 8 показаны кружками. Сплошная линия — граница области устойчивости, гарантированной для любого n .

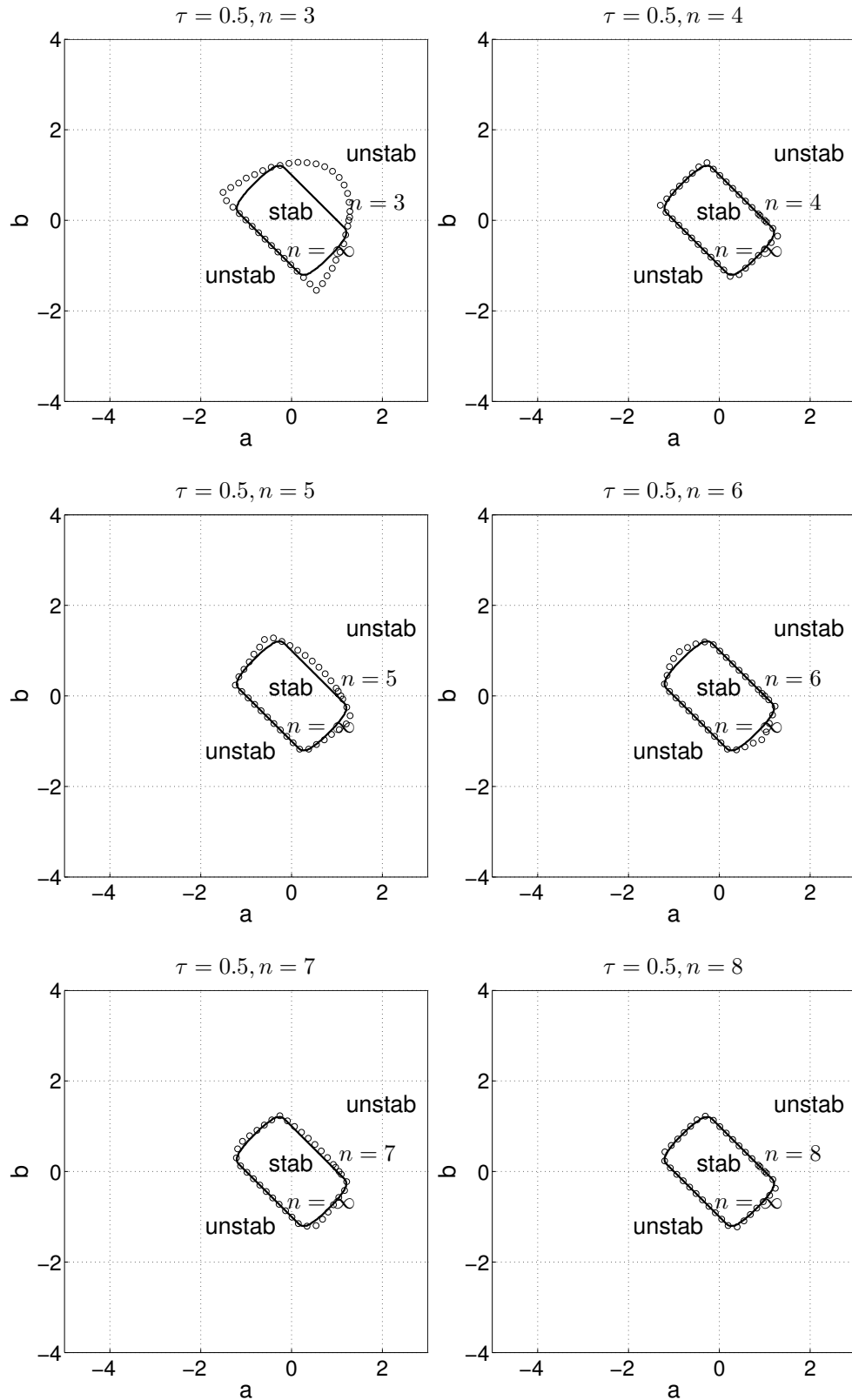


Рис. 3.9. Границы областей устойчивости для системы (3.2) для $\tau = 0.5$ и значений n от 3 до 8 показаны кружками. Сплошная линия — граница области устойчивости, гарантированной для любого n .

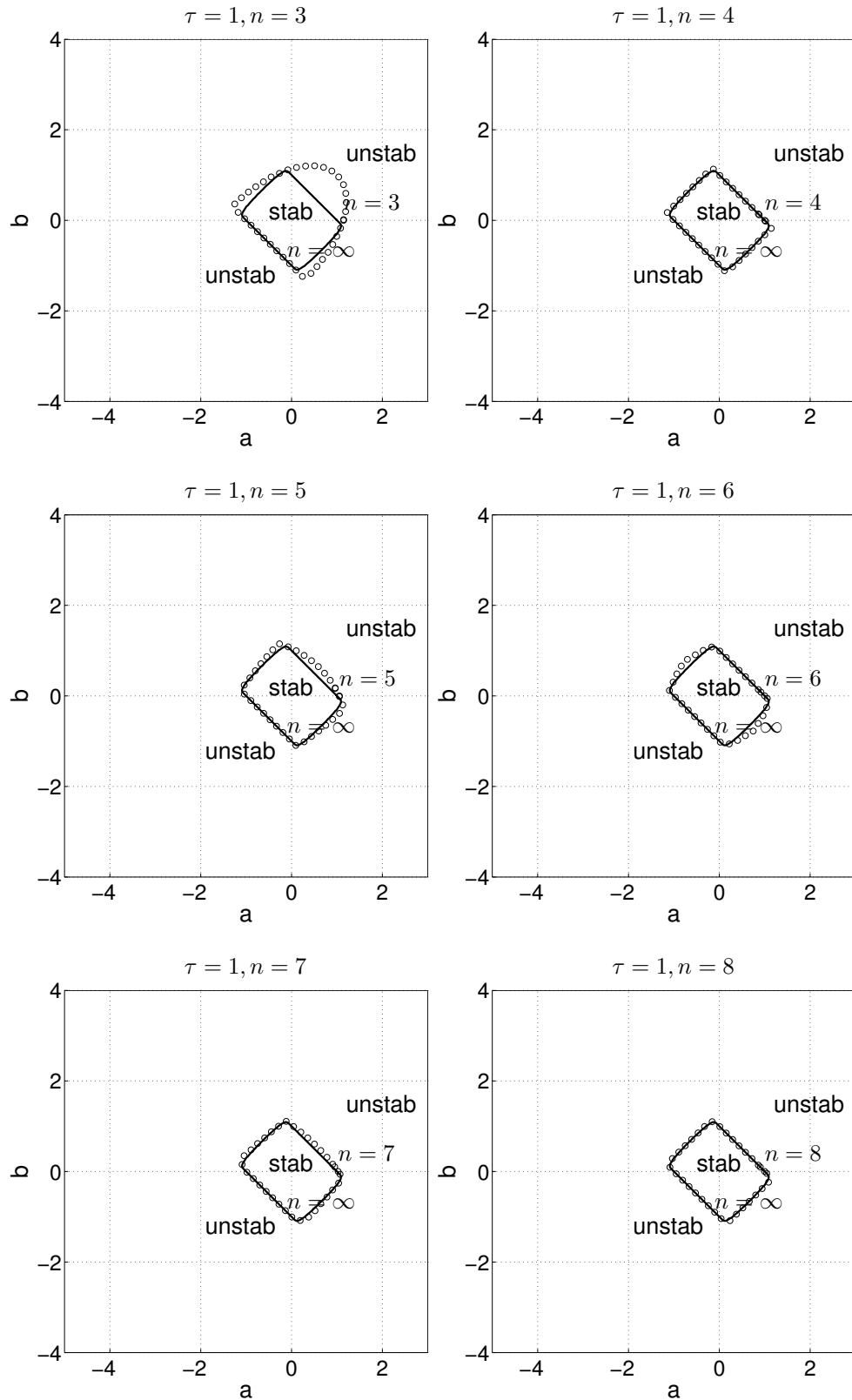


Рис. 3.10. Границы областей устойчивости для системы (3.2) для $\tau = 1$ и значений n от 3 до 8 показаны кружками. Сплошная линия — граница области устойчивости, гарантированной для любого n .

3.4 Устойчивость модели нейронной сети линейной конфигурации с ограниченным количеством нейронов

3.4.1 Область устойчивости

В разделе 2.5 мы строили область устойчивости модели линейной сети с неограниченным количеством нейронов, и эти области не зависела от величины n в системе уравнений (2.27), (2.28). В настоящем разделе мы дадим полное описание области устойчивости для линейной системы из n нейронов. Эти области, естественно, зависят от n .

Определим функцию $F1(\tau, n)$ (ср. с формулой (2.29)) от запаздывания $\tau \in (0, \infty)$ и количества нейронов в сети $n \in \mathbb{N}$:

$$F1(\tau, n) = \frac{1}{4 \sin^2 \omega(\tau) \cos^2 \frac{\pi}{n+1}}, \quad (3.3)$$

где $\omega(\tau)$ есть наименьший положительный корень уравнения (2.30).

Следующая теорема (ср. с Теоремой 3.1) дает полное описание области устойчивости линейной конфигурации n нейронов.

- Теорема 3.1.** 1. Если $0 \leq ab < \frac{1}{4 \cos^2 \frac{\pi}{n+1}}$, то система (2.27), (2.28) асимптотически устойчива при любом $\tau \geq 0$.
2. Если $ab > \frac{1}{4 \cos^2 \frac{\pi}{n+1}}$, то система (2.27), (2.28) неустойчива при любом $\tau \geq 0$.
3. Если $ab < 0$ и $|ab| < F1(\tau, n)$, то система (2.27), (2.28) асимптотически устойчива.
4. Если $ab < 0$ и $|ab| > F1(\tau, n)$, то система (2.27), (2.28) неустойчива.

Доказательство. Будем пользоваться обозначениями, введенными в доказательстве Теоремы 2.3.

1. Согласно Теореме 1.5, для асимптотической устойчивости, независимой от запаздывания, достаточно выполнения неравенств (2.42), что гарантировано условием пункта 1 теоремы. Пункт 1 Теоремы 3.1 доказан.

2. Построим точки $M_{jn} = (u_{1jn}, u_{2jn}, u_{3jn}) \in \mathbb{R}^3$ по формулам (2.43). Пусть $ab > \frac{1}{4 \cos^2 \frac{\pi}{n+1}}$. Тогда $\mu_j \in \mathbb{R}$ (Лемма 2.1), при этом выполняются неравенства (см. (2.43))

$$u_{1nn} + iu_{2nn} = \tau \mu_{nn} \exp(i\tau \operatorname{Im} \lambda_{nn}) = 2\tau \sqrt{ab} \cos \frac{\pi n}{n+1} < -\tau, \quad (3.4)$$

и поэтому точка M_{nn} находится вне овала устойчивости. Пункт 2 Теоремы 3.1 доказан.

3-4. Пусть $ab < 0$. Тогда числа μ_j являются чисто мнимыми (Лемма 2.1), поэтому $u_{1jn} = 0$, при этом

$$u_{2jn} = 2\tau \sqrt{|ab|} \cos \frac{\pi j}{n+1}. \quad (3.5)$$

На поверхности конуса устойчивости на высоте $u_3 = \tau$ при $u_1 = 0$ имеем (см. (1.21), (1.22))

$$|u_2| = \frac{\omega}{\cos \omega} = \frac{\tau}{\sin \omega}, \quad (3.6)$$

где ω есть наименьший положительный корень уравнения (2.30). Поэтому из (3.5) получим

$$u_{21n} = 2\tau \sqrt{|ab|} \cos \frac{\pi}{n+1} = \frac{\tau}{\sin \omega(\tau)} \sqrt{\frac{ab}{F1(\tau, n)}} = |u_2| \sqrt{\frac{ab}{F1(\tau, n)}}. \quad (3.7)$$

Из (3.7) следует, что при $|ab| < F1(\tau, n)$ система (2.27), (2.28) асимптотически устойчива, а при $|ab| > F1(\tau, n)$ неустойчива. Пункты 3 и 4 Теоремы 3.1 доказаны. ■

Теорема 3.1 проиллюстрирована рисунком 3.11.

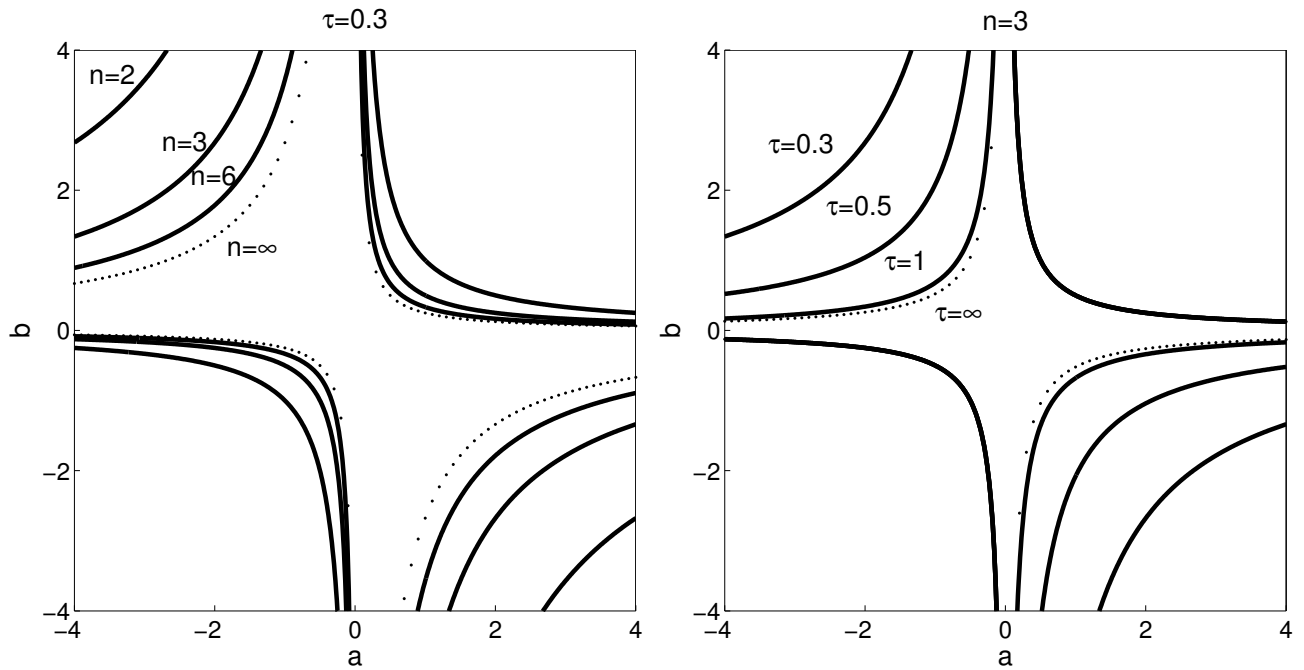
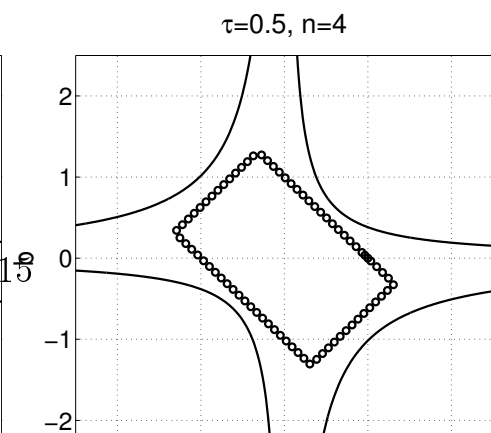
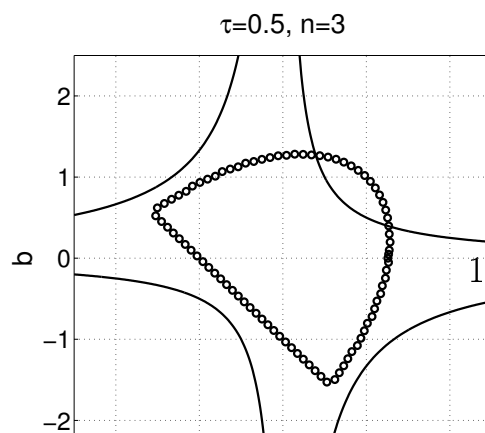
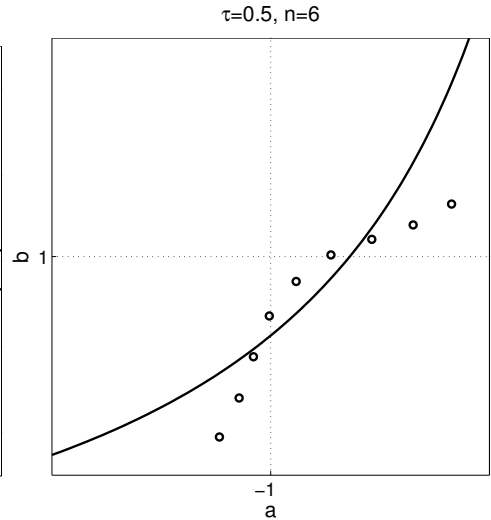
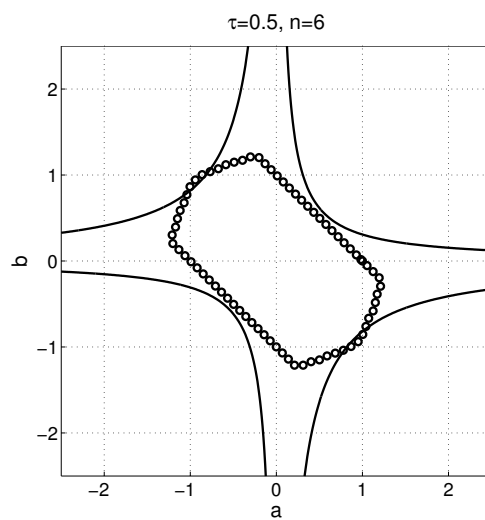
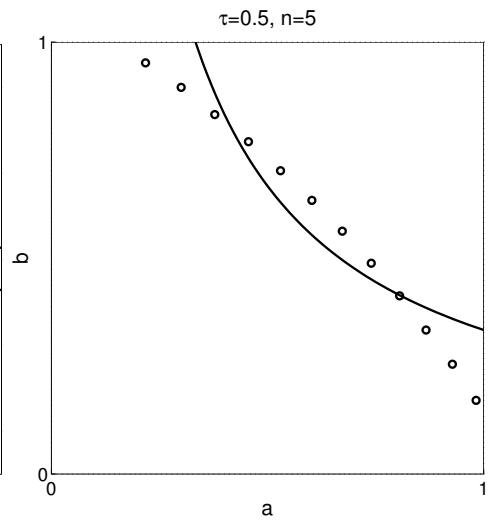
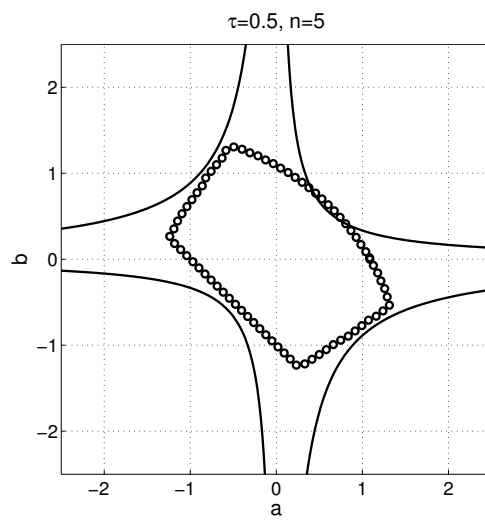
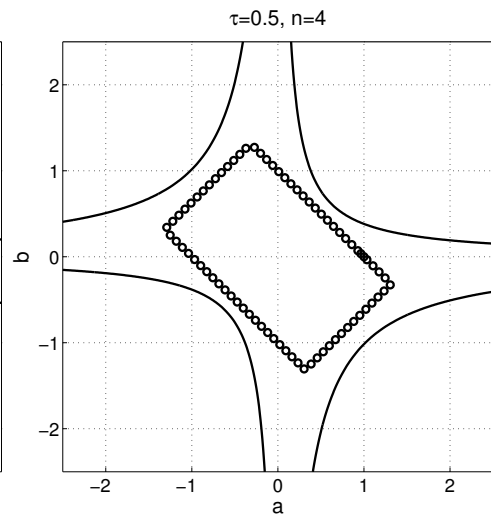
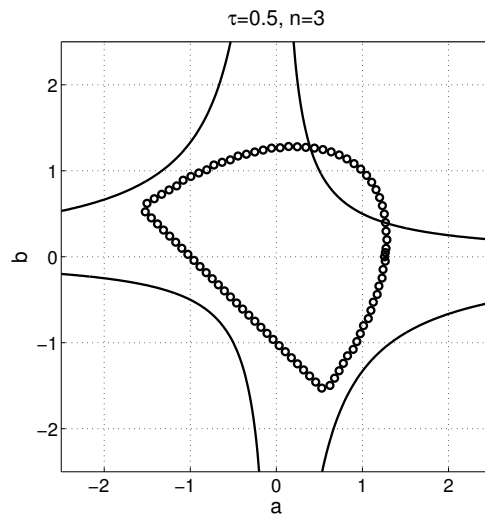


Рис. 3.11. Границы областей устойчивости для линейной конфигурации нейронов (система (2.27), (2.28)). На рисунке слева: запаздывание фиксировано, количество нейронов меняется. Справа: количество нейронов фиксировано, запаздывание меняется.

3.4.2 Сравнение областей устойчивости моделей нейронной сети кольцевой и линейной конфигурации с ограниченным количеством нейронов с одинаковыми параметрами

В разделе 2.5 было показано, что разрыв кольцевой сети может только улучшить ее устойчивость, если количество нейронов в сети достаточно велико (Теорема 2.3). Наши численные исследования с помощью программ показывают, что это же явление, за некоторыми исключениями, имеет место и при небольшом количестве нейронов.

Область тех значений параметров (a, b) , при которых кольцевая система нейронов устойчива, а линейная с теми же параметрами неустойчива, назовем парадоксальной областью при данных τ, n . Таким образом, парадоксальная область это область значений параметров, в которых нарушается принцип «разрыв кольца благоприятен для устойчивости». На следующем рисунке ??



показано, что при $\tau = 0,5$ парадоксальная область заметна в сети с количеством нейронов $n = 3$, отсутствует при $n = 4$ и весьма мала при $n = 5, n = 6$. Мы не приводим здесь результаты экспериментов при $n > 6$. В них парадоксальная область либо отсутствует, либо пренебрежимо мала.

Численные эксперименты, не отраженные в диссертации, показывают, что при увеличении запаздываний и сохранении других параметров парадоксальная область уменьшается. Технически это связано с тем, что при увеличении запаздывания τ точки M_j (см. (2.43)) перемещаются в верхние слои конуса устойчивости (см. Теорему 2.3), которые больше похожи на окружности, чем нижние.

3.5 Динамика области устойчивости в процессе разрыва кольцевой сети нейронов

В данном разделе мы рассмотрим модель (3.2) кольцевой сети с двусторонним запаздыванием из шести нейронов и проведем численное исследование изменения областей устойчивости и неустойчивости в плоскости параметров при постепенном уменьшении связи между первым и шестым нейронами.

Пусть в кольцевой сети взаимодействие между нейронами осуществляется с запаздыванием τ в обоих направлениях, а реакция нейрона на изменение собственного состояния мгновенна и обладает интенсивностью, равной единице. Пусть интенсивности взаимодействия между всеми нейронами, кроме первого и шестого, равны a и b (как в модели (3.2)), а интенсивности воздействия шестого нейрона на первый и первого нейрона на шестой равны соответственно ac и bc , где c – некоторый неотрицательный параметр.

Тогда взаимодействие нейронов в описанной модели задается уравнением (1.6) с матрицами

$$A = I, \quad B = \begin{pmatrix} 0 & b & 0 & 0 & 0 & ac \\ a & 0 & b & 0 & 0 & 0 \\ 0 & a & 0 & b & 0 & 0 \\ 0 & 0 & a & 0 & b & 0 \\ 0 & 0 & 0 & a & 0 & b \\ bc & 0 & 0 & 0 & a & 0 \end{pmatrix}. \quad (3.8)$$

При $c = 1$ система (1.6), (3.8) описывает кольцевую сеть с двусторонним запаздыванием (3.2). С изменением c от единицы до нуля интенсивность взаимодействия между первым и шестым нейронами в системе (1.6), (3.8) постепенно ослабевает, и при $c = 0$ кольцо нейронов размыкается. На рисунке 3.13 построены области устойчивости и неустойчивости рассматриваемой сети при различных значениях параметра c . С уменьшением c границы областей устойчивости вытягиваются в четырех концах и при полном разрыве связи переходят в гиперболы с асимптотами $a = 0$ и $b = 0$ (см. последний график в серии рисунков 3.13).

Как показывает рис. 3.13, ослабление одной связи между нейронами в кольцевой сети из 6 нейронов расширяет область устойчивости всей системы.

Для всех систем (1.6), (3.8) области неустойчивости занимают достаточно большую часть плоскости (a, b) , особенно при больших по модулю значениях параметров.

Все графики получены на основе исходного кода программы «Построение областей устойчивости круговых нейронных сетей» с учетом необходимых изменений в элементах матрицы B . Отметим также, что для кольцевой модели с односторонним запаздыванием (3.1) ослабление связи в одной паре нейронов приведет к изменениям в структуре соответствующей области устойчивости, аналогичным тем, что были получены в данном разделе для системы с двусторонним запаздыванием (3.2).

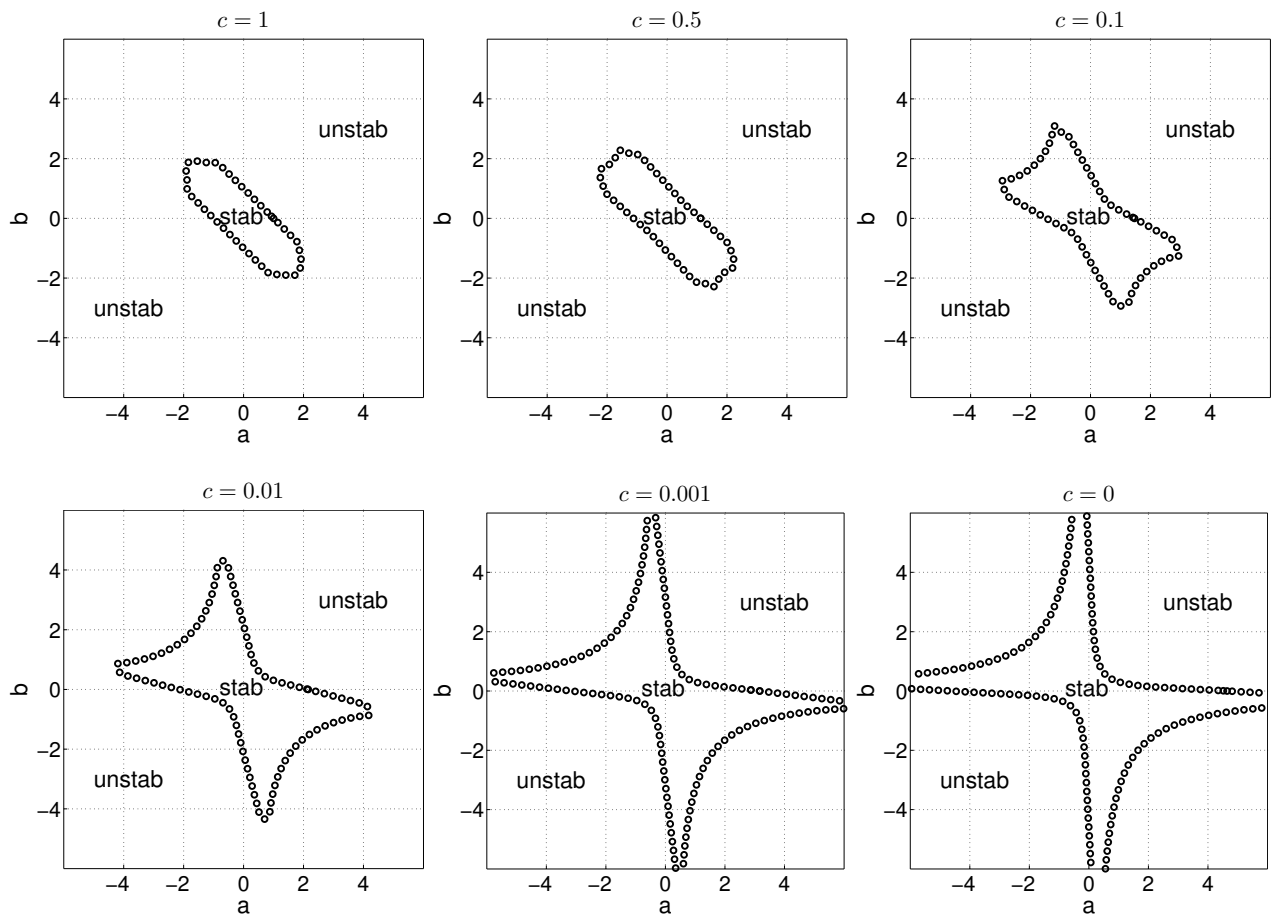


Рис. 3.13. Динамика области устойчивости в процессе разрыва нейронного кольца. Границы областей устойчивости для системы (1.6), (3.8) для $\tau = 0.1$ и $c = 1$, $c = 0.5$, $c = 0.1$, $c = 0.01$, $c = 0.001$, $c = 0$.

3.6 Сравнение результатов главы 3 с известными результатами

В большинстве работ по устойчивости моделей кольцевых нейронных сетей рассматривается задача об устойчивости сети из двух [?], трёх [?] или четырёх [? ?] нейронов. Задачу об устойчивости кольцевой системы произвольного количества нейронов рассматривали Y. Yuan, S. Campbell [?]. В [?] изучена следующая модель кольцевой нейронной сети:

$$\dot{x}_j(t) + x_j(t) + \alpha x_j(t - \tau_s) + \beta (x_{j-1}(t) + x_{j+1}(t - \tau)) = 0 \quad (j \bmod n), \quad (3.9)$$

В уравнении (3.9) реакция нейрона на собственный сигнал (selfconnection) разделена на две части: мгновенная реакция с интенсивностью 1 и запаздывающая на время τ_s с интенсивностью α . Реакции нейрона на сигнал правого и левого соседа имеет одинаковые запаздывания τ и одинаковые интенсивности β . Если положить $\alpha = 0$, то система (3.9) совпадет с частным случаем нашей системы (2.3) с $a = b$. Как показывают Предложение 2.1 и Теорема 2.1, а также рисунок 2.8, поведение таких систем несложно: при $|\beta| < 1/2$ система устойчива независимо от запаздывания, при $|\beta| > 1/2$ система неустойчива независимо от запаздывания. Сложность динамики в модели Юан-Кэмпбелл (3.9) происходит именно из-за запаздывания в реакции нейрона на собственный сигнал.

Это свидетельствует о недостаточной адекватности модели (3.9). В нашей модели источником сложности поведения моделей (2.2), (2.3) 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 [?] 2006 года работы С. Кэмпбелл по нейронным сетям были посвящены только сетям с распределенным запаздыванием.

В работе [?] изучена модель кольцевой сети (3.9) с $\tau_s = \tau$, в которой $n = 4$. В этой статье указаны только области устойчивости, независимой от запаздывания, статья посвящена в основном проблемам симметрии в поведении нелинейной системы.

Устойчивость моделей линейных конфигураций нейронов не была исследована никем. Предпринятое в данной главе сравнение областей устойчивости кольцевой и линейной конфигураций с сопоставимыми параметрами в литературе отсутствует.

Алгоритмы и программы для построения области устойчивости моделей нейронных сетей, описанные в данной главе, не имеют аналогов в известной автору литературе.

Заключение

В соответствии с целями настоящего диссертационного исследования автором разработан метод конуса устойчивости и основанные на этом методе алгоритм и программа для анализа устойчивости матричного дифференциального уравнения с запаздываниями. Указанный метод дает результаты сильнее известных в литературе результатов B. Cahlon и D. Schmidt [?] (2000), а также Н. Matsunaga [?] (2007) и S. Sakata [?] (1998). Указанные алгоритмы и программы в следующих главах использованы для анализа устойчивости кольцевых и линейных нейронных сетей. Пространство параметров нейронных сетей многомерно. Мы считаем основными параметрами силы взаимодействия соседних нейронов и через все разделы диссертации проводим исследование области устойчивости именно в плоскости этих параметров. Остальные параметры, такие как величина запаздывания, количество нейронов в сети, показатель демпфирования собственных колебаний нейрона, рассматриваются с точки зрения их воздействия на изменение области устойчивости в плоскости основных параметров.

Такой подход позволил построить ясные графические иллюстрации влияния свойств кольца и линии нейронов на устойчивость системы. Проведенное в диссертации сравнение с рисунками областей устойчивости в статьях S. Campbell, S. Ruan и Y. Yuan [?] (1999, 2004), касающихся сходных моделей, показывает преимущества нашего подхода.

Две темы, исследованные в диссертации, не затрагивались ранее в научной литературе. Во-первых, это задача об устойчивости нейронных сетей с неограниченным количеством нейронов. Во-вторых, задача об изменении области устойчивости кольцевой сети при ее разрыве и переходе в линейную сеть. Отмеченный в диссертации эффект увеличения области устойчивости

больших кольцевых нейронных сетей при разрыве кольца согласуется с известным явлением стабилизации психики человека при лоботомии.

Направление дальнейших исследований связано с тем, что разработанный автором метод конуса устойчивости, алгоритмы и программы обладают потенциалом значительно более широким, чем использованный в диссертации. В дальнейшем с их помощью автор планирует исследовать устойчивость других конфигураций нейронных сетей.

Приложение А

Исходный код программы «Анализ устойчивости»

application.m

```
function application
    MAX_DIM = 5;
    fontsize0 = 12;
    fontsize1 = 15;
    % Строковые ресурсы.
    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]);
sliderPanel = uipanel('Parent',inputPanel, ...
    'bordertype', 'none', 'Position',[0 .9 1 .1]);
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 = 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]);
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]);
uicontrol(matrixPanel, 'style', 'pushbutton', ...
    'string', label_clear, 'tooltipString', tip_clear,...
    'units', 'normalized', 'position', [0.2, 0.4, .1, .2], ...
    'visible', 'off', 'callback', @clear_callback);
uicontrol(matrixPanel, 'style', 'pushbutton', ...
    'string', label_rand, 'tooltipString', tip_rand,...
    'units', 'normalized', 'position', [0.2, 0.2, .1, .2], ...
    'visible', 'off', 'callback', @random_callback);
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]);
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]);
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]);
resultChildPanel = uipanel('Parent',resultPanel,'bordertype', 'none',...
    'Position',[0 0.2 .7 .8]);
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]);
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]);
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

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');
    [intervals, intervalsCell]= pointsPlotter(eigA, eigB);
    set (close_btn, 'enable', 'on') ;
    dim = str2double(get(dimText, 'string'));
    for n=1:dim
        set(resultLabels(n), 'string', buildResult(intervalsCell{n}));
    end
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 me
            display(me);
        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);
    end
end

```

```

        if length(parseValue) ~= 1
            str = [err_incorrect_cell '(' num2str(j) ')'].';
            errordlg(str, label_error, 'on');
            vector = [];
            return;
        end
        vector(j)=str2double(val);
    end
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
    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
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
end
function clearMatrix(children)

```

```

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

```

pointsPlotter.m

```

function [result, intervalsCell] = pointsPlotter(lambda, mu, maxdistanse, resolution)
    addPath(' ../common');
    % Построение конуса устойчивости.
    cone();
    % Построение точек.
    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);
    end
    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;
    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));
    end
end

```

```
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;
end

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

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
```

Приложение Б

Исходный код программы «Устойчивость нейронных сетей»

application2.m

```
function application2
    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');
    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 запаздыванием взаимодействия с соседями';
    label_model1 = 'Модель сети:  $x''_j(t) + x_j(t) + a*x_{j-1}(t) + b*x_{j+1}(t-\tau) = 0, 1 \leq j \leq n$ ';
    label_model2 = 'Модель сети:  $x''_j(t) + x_j(t) + a*x_{j-1}(t-\tau) + b*x_{j+1}(t-\tau) = 0, 1 \leq j \leq n$ ';
    label_intens = 'Интенсивность взаимодействия нейрона с';
    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]);
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]);
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]);
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]);
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 = pointsPlotter2(a, b, tau, 1);
    elseif get(group,'SelectedObject') == rb2
        result = pointsPlotter2(a, b, tau, 2);
    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 e
            display(e);
        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
end

```

pointsPlotter2.m

```

function [result] = pointsPlotter2(a, b, tau, geometry)
    addPath(' ../common');
    % Построение конуса устойчивости.
    cone();
    % Построение точек и вычисления.
    % Выбираем несколько (10) точек кривой и определяем находятся ли они внутри
    % конуса. Для этого находим собственные числа.
    result = [0 Inf];
    n = 100;
    t = 0:2*pi/(n-1):2*pi;
    j=1:n;
    if geometry == 1
        lambda = 1 + a*exp(1i*2*pi*j/n);
        mu = b*exp(-1i*2*pi*j/n);
        % Calculating point for displaying.
        U = [tau*b*cos(-t + a*tau*sin(t))
            tau*b*sin(-t + a*tau*sin(t))
            tau*(1 + a*cos(t))];
    elseif geometry == 2
        lambda = ones(1,n);
        mu = a*exp(1i*2*pi*j/n) + b*exp(-1i*2*pi*j/n);
        % Calculating point for displaying.
    end
end

```

```
U = [tau*abs(b+a)*cos(t)
      tau*abs(b-a)*sin(t)
      tau*ones(1, n)];
end
for j=1:n
    intervals = getIntervals(lambda(j), mu(j));
    result = intersection(result, intervals);
    % инвертируем результат, чтоб внешние точки отобразить красными.
    text(U(1,j), U(2,j), U(3,j), '*', 'color', [~isInside(intervals, tau) 0 0], 'fontsize', 20);
end
end
```

Приложение В

Исходный код программы «Построение областей устойчивости круговых нейронных сетей»

application3.m

```
function application3
fontsize0 = 12;
fontsize1 = 15;
image1 = imread('circles_right.png');
image2 = imread('circles.png');
% Error strings
label_error = 'Ошибка';
err_incorrect_tau = 'Вы ввели некорректное значение tau.';
err_incorrect_epsilon = 'You entered incorrect epsilon.';
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 запаздыванием взаимодействия с соседями';
label_model1 = 'Модель сети:  $x''_j(t) + x_j(t) + a*x_{j-1}(t) + b*x_{j+1}(t-\tau) = 0, 1 \leq j \leq n$ ';
label_model2 = 'Модель сети:  $x''_j(t) + x_j(t) + a*x_{j-1}(t-\tau) + b*x_{j+1}(t-\tau) = 0, 1 \leq j \leq n$ ';
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_theSameScale = 'Одинаковый масштаб';
label_calc = 'Рассчитать...';
label_calc_tip = 'Определяет интервалы устойчивости системы, заданной матрицами A и B';
% 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]);
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]);
% 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', '3', '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]);
% use the same scale for all plots
checkboxScale = uicontrol(leftPanel1, 'style', 'checkbox', ...
        'string', label_theSameScale, ...
        'units', 'normalized', 'position', [0 .1 .25 .15]);
% extension
uicontrol(leftPanel1, 'style', 'text', 'fontsize', fontsize0, ...
        'string', label_ext, 'horizontalAlignment', 'right', ...
        'units', 'normalized', 'position', [0.25, 0.08, .2, .15]);
extCombo = uicontrol(leftPanel1, 'style', 'popupmenu', 'string', {'eps', 'png'}, ...
        'tooltipString', label_calc_tip,...
        'units', 'normalized', 'position', [0.45 .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]);
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]);
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, ~)
    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(~, ~)
    [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 neurons. ' ...
            '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 neurons. ' ...
            '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 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 get(group,'SelectedObject') == rb1
    type = 1;
elseif get(group,'SelectedObject') == rb2
    type = 2;
end
mkdir(dir);
[minX, maxX] = plotterBoundness(type, tau, minNumber, epsilon, dir, ext);
if get(checkboxScale, 'value')
    for number=minNumber+1:maxNumber
        plotterBoundness(type, tau, number, epsilon, dir, ext, minX, maxX);
    end
else
    for number=minNumber+1:maxNumber
        plotterBoundness(type, tau, number, epsilon, dir, ext);
    end
end
end
end
end
end

```

plotterBoundness.m

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Чертит границу устойчивости.
% type - тип соединения нейронов (1 для несимметричного, 2 для симметричного
% взаимодействия),
% tau - запаздывание,
% number - количество нейронов,
% epsilon - точность,
% dir - директория для сохранения графиков,
% ext - расширение файлов для сохранения,
% minX, maxX - границы для графиков.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [minX, maxX] = plotter_boundness(type, tau, number, epsilon, dir, ext, minX, maxX)

```

```

% Создание окна для графика
figure1 = figure('InvertHardcopy','off','Color',[1 1 1], ...
    'position', [680 0 600 570]);
hold on;
grid on;
% Загрузка или создание данных для границы устойчивости бесконечного
% числа нейронов.
datafile = [num2str(type) '_' num2str(tau) '.mat'];
try
    load(datafile);
catch me
    display(me);
    % предполагается, что файла данных нет.
end
% Если данные не загрузились, то создадим их.
if exist('infiniteR', 'var') == 0 || exist('infinitePhi', 'var') == 0
    infiniteNumber = 100;
    [infinitePhi, infiniteR] = solverBoundnessSmart(type, tau, infiniteNumber, .02);
    save(datafile, 'infinitePhi', 'infiniteR');
end
h = polar(infinitePhi, infiniteR, '-');
set(h, 'LineWidth', 3, 'color', 'k') ;
% Вычисление области устойчивости для ограниченного числа нейронов.
[phi, r] = solverBoundnessSmart(type, tau, number, epsilon);
polar(phi, r, 'o');
title(['$\tau = ' num2str(tau) ', n = ' num2str(number) '$' ],...
    'FontSize',20, 'interpreter', 'latex');
% Создание надписей
xlabel({'a'}, 'FontSize',16);
ylabel({'b'}, 'FontSize',16);
text('String','Stability',...
    'Position',[-.5 .2],...
    'FontSize',16);
text('String','Instability',...
    'Position',[-2.5 -1.2],...
    'FontSize',16);
text('String','Instability',...
    'Position',[1.2 2.5],...
    'FontSize',16);
text('Interpreter','latex',...
    'String',{'$n = ' num2str(number) '$'},...
    'Position',[r(1) -.2],...
    'FontSize',16);
text('Interpreter','latex',...
    'String',{'$n = \infty$'},...
    'Position',[infiniteR(1) 0.2],...
    'FontSize',16);
% Масштабирование
ax=axis;

```

```

x1=min(ax(1), ax(3));
x2=max(ax(2), ax(4));
if exist('minX', 'var')
    x1=minX;
end
if exist('maxX', 'var')
    x2=maxX;
end
minX = x1;
maxX = x2;
axis([x1-.5 x2+.5 x1-.5 x2+.5]);
% Создание дополнительных границ
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-.5 x2+.5 x1-.5 x2+.5]);
% Сохранение построенной фигуры
if type == 1
    typeName = 'asym';
else
    typeName = 'sym';
end
strTau = num2str(tau);
if length(strTau) < 3
    strTau = [strTau '_0'];
end
%mkdir(dir);
figureName = [typeName '-' strTau '-' num2str(number)];
% Заменяем точку на '_', чтоб не было проблем в файловой системе.
figureName = strrep(figureName, '.', '_');
set(figure1, 'PaperPositionMode', 'auto');
if strcmp(ext, 'png')
    saveas(figure1, [dir filesep figureName '.png'], 'png');
else
    saveas(figure1, [dir filesep figureName '.eps'], 'psc2');
end
close;
end

```

solverBoundnessSmart.m

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Вычисляет точки области устойчивости с заданной точностью epsilon, так
% чтобы они находились на одинаковом расстоянии друг от друга.
% type - тип соединения нейронов (1 для несимметричного, 2 для симметричного
% взаимодействия),
% tau - запаздывание,
% number - количество нейронов,
% epsilon - точность,

```

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [phi r] = solver_boundness_smart(type, tau, number, epsilon)
    radius = 2*epsilon;
    r = zeros(1, 1000);
    phi = zeros(1, 1000);
    % Вычисляем первую точку на луче '0a'.
    phi(1) = 0;
    [r(1)] = searchPointInRowGeom(type, number, tau, 0, radius);
    % Все другие точки находим, переходя в локальную полярную систему
    % координат, на заданном расстоянии radius от предыдущей.
    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);
    % Переводит точку от полярной СК к декартовой.
    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
    % Переводит точку от декартовой СК к полярной.
    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
    % Ищет следующую точку на заданном расстоянии radius от предыдущей.
    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 = straightStabAnalyzer(type, xHalf, yHalf, number, tau);
        if stabHalf == 1
            phi1 = phiHalf;
        else
            phi0 = phiHalf;
        end
    end
    [rEnd phiEnd] = toPolar(xHalf, yHalf);
end
end

```

searchPointInRowGeom.m

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Находит точку границы области устойчивости на луче, определённом углом
% phi в полярной системе координат с заданной точностью epsilon.
% Возвращает координату точки на заданном луче с точностью epsilon.
% type - тип соединения нейронов (1 для несимметричного, 2 для симметричного
% взаимодействия).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [half] = searchPointInRowGeom(type, number, tau, phi, epsilon)
r = [0 2];
stab = straightStabAnalyzer(type, r(2)*cos(phi), r(2)*sin(phi), number, tau);
while stab
    r(2) = r(2) + 1;
    stab = straightStabAnalyzer(type, r(2)*cos(phi), r(2)*sin(phi), number, tau);
end
for iter = 1:20
    %disp(' ');
    %disp(['Step ' num2str(iter) ' =====>'])
    half = sum(r)/2;
    stab = straightStabAnalyzer(type, half*cos(phi), half*sin(phi), number, tau);
    if stab
        r(1) = half;
    else
        r(2) = half;
    end
    if abs(r(1) - r(2)) < epsilon
        %disp(['Stopped on ' num2str(iter)]);
        %return;
    end
end
end
end

```

straightStabAnalyzer.m

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Проверяет данную точку на устойчивость непосредственно, используя
% годограф.
% type - тип соединения нейронов (1 для несимметричного, 2 для симметричного
% взаимодействия),
% a, b - коэффициенты уравнения,
% number - количество нейронов,
% tau - запаздывание.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function [stab] = straightStabAnalyzer(type, a, b, number, tau)

    j = 1:number;
    % Вычисление собственных чисел.
    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

    % Находим w, которая является параметром для точки первого пересечения
    % годографа с осью 0a.
    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) ;
    end

    % Для каждой из точек Mj срезаем конус на высоте её третьей компоненты
    % (переводим задачу на плоскость), затем определяем, лежит ли эта точка
    % в овале устойчивости или нет.
    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));
    end

```

```

z = tau*real(la);
xcone = w.*sin(w) - z.*cos(w);
ycone = -w.*cos(w) - z.*sin(w) ;
% Находим расстояние от каждой точки овала устойчивости до Mj.
distArr =(x-xcone).^2 +(y-ycone).^2 ;
% Находим точку овала, наиболее близкую к точке Mj.
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;
% Если расстояние от начала координат до точки Mj больше, чем до
% ближайшей точки овала, то исследуемая пара значения a, b является
% неустойчивой и дальнейшее выполнение функции не имеет смысла.
if distPoint > distCone
    stab = 0;
    return;
end
end
% Если протестированы все точки Mj и среди них не оказалось
% неустойчивых, то исследуемая пара значения a, b является устойчивой.
stab = 1;
end

```

Приложение Г

Общие исходные коды программ

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

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Возвращает массив чётной длины, представляющий собой упорядоченный набор
% множеств (отрезков) и являющийся объединением наборов множеств set1 и set2.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function result = coalition(set1, set2)

% проверка входных параметров.
validate(set1);
validate(set2);
```



```

% проверка на пустое множество.
if isempty(set1)
    result = set2;
    return;
elseif isempty (set2)
    result = set1;
    return;
end
% Now dimension 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)];
    end
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
% Проверяет, что массив имеет чётную длину и состоит из неубывающих
% значений.
function validate(set)
    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
end
end

```

getIntervals.m

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Вычисляет интервалы устойчивости по tau.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
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 abs(result_set(1)-result_set(2)) < 0.000001    %result_set(1) == result_set(2)
        cleanSet = cleanSet(2:end);
    end
    for k = 2:length(result_set)-1
        if abs(result_set(k)-result_set(k+1)) < 0.000001 %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;
end

```

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 length.
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
end

```

isInside.m

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Определяет принадлежность tau множеству интервалов '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
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;
end

```

cone.m

```

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Строит конус устойчивости.
% 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 cone
    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);
        end
    end
end

```

```
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);
end
```