

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ



Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Магнитогорский государственный технический университет им. Г.И. Носова»



УТВЕРЖДАЮ
Директор ИЕиС
И.Ю. Мезин

30.01.2023 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ (МОДУЛЯ)

КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ В ПРИБОРОСТРОЕНИИ

Направление подготовки (специальность)
12.03.01 Приборостроение

Направленность (профиль/специализация) программы
Приборы и методы контроля качества и диагностики

Уровень высшего образования - бакалавриат

Форма обучения
очная

Институт/ факультет	Институт естествознания и стандартизации
Кафедра	Физики
Курс	4
Семестр	7

Магнитогорск
2023 год

Рабочая программа составлена на основе ФГОС ВО - бакалавриат по направлению подготовки 12.03.01 Приборостроение (приказ Минобрнауки России от 19.09.2017 г. № 945)

Рабочая программа рассмотрена и одобрена на заседании кафедры Физики
16.01.2023, протокол № 4

Зав. кафедрой _____ М.Б. Аркулис

Рабочая программа одобрена методической комиссией ИЕиС
30.01.2023 г. протокол № 5

Председатель _____ И.Ю. Мезин

Рабочая программа составлена:

доцент кафедры Физики, канд. физ.-мат. наук _____ В.В. Мавринский

Рецензент:

зав. кафедрой ПМИИ, д-р техн. наук _____ Ю.А. Извеков

Лист актуализации рабочей программы

Рабочая программа пересмотрена, обсуждена и одобрена для реализации в 2024 - 2025 учебном году на заседании кафедры Физики

Протокол от _____ 20__ г. № ____
Зав. кафедрой _____ М.Б. Аркулис

Рабочая программа пересмотрена, обсуждена и одобрена для реализации в 2025 - 2026 учебном году на заседании кафедры Физики

Протокол от _____ 20__ г. № ____
Зав. кафедрой _____ М.Б. Аркулис

Рабочая программа пересмотрена, обсуждена и одобрена для реализации в 2026 - 2027 учебном году на заседании кафедры Физики

Протокол от _____ 20__ г. № ____
Зав. кафедрой _____ М.Б. Аркулис

Рабочая программа пересмотрена, обсуждена и одобрена для реализации в 2027 - 2028 учебном году на заседании кафедры Физики

Протокол от _____ 20__ г. № ____
Зав. кафедрой _____ М.Б. Аркулис

1 Цели освоения дисциплины (модуля)

Приобретение обучающимися знаний, умений и навыков, связанных с использованием современных информационных технологий и программного обеспечения в области неразрушающего контроля.

2 Место дисциплины (модуля) в структуре образовательной программы

Дисциплина Компьютерные технологии в приборостроении входит в обязательную часть учебного плана образовательной программы.

Для изучения дисциплины необходимы знания (умения, владения), сформированные в результате изучения дисциплин/ практик:

Приборы и методы ультразвукового контроля

Приборы и методы магнитного контроля

Основы языка программирования MatLab

Математика

Физика

Физические основы ультразвукового контроля

Знания (умения, владения), полученные при изучении данной дисциплины будут необходимы для изучения дисциплин/практик:

Подготовка к сдаче и сдача государственного экзамена

3 Компетенции обучающегося, формируемые в результате освоения дисциплины (модуля) и планируемые результаты обучения

В результате освоения дисциплины (модуля) «Компьютерные технологии в приборостроении» обучающийся должен обладать следующими компетенциями:

Код индикатора	Индикатор достижения компетенции
ОПК-4	Способен понимать принципы работы современных информационных технологий и использовать их для решения задач профессиональной деятельности
ОПК-4.1	Осуществляет поиск, анализ и синтез информации с использованием информационных технологий
ОПК-4.2	Применяет технологии обработки данных, выбора данных по критериям; строит типичные модели решения предметных задач по изученным образцам
ОПК-4.3	Использует современные информационные технологии для решения задач профессиональной деятельности

4. Структура, объём и содержание дисциплины (модуля)

Общая трудоемкость дисциплины составляет 3 зачетных единиц 108 акад. часов, в том числе:

- контактная работа – 58,05 акад. часов;
- аудиторная – 57 акад. часов;
- внеаудиторная – 1,05 акад. часов;
- самостоятельная работа – 49,95 акад. часов;
- в форме практической подготовки – 0 акад. час;

Форма аттестации - зачет

Раздел/ тема дисциплины	Семестр	Аудиторная контактная работа (в акад. часах)			Самостоятельная работа студента	Вид самостоятельной работы	Форма текущего контроля успеваемости и промежуточной аттестации	Код компетенции
		Лек.	лаб. зан.	практ. зан.				
1. Введение								
1.1 Роль и задачи компьютерных технологий в процессе разработки и эксплуатации приборов	7	4	4/2И		4	подготовка к семинарским занятиям; изучение литературы по теме занятий, в том числе и с помощью ЭБС.	семинар №1, 2	ОПК-4.1, ОПК-4.2
Итого по разделу		4	4/2И		4			
2. Программные средства моделирования								
2.1 Программные средства моделирования, анализа и оптимизации процессов, связанных с НК	7	8	12/4И		5	подготовка к лабораторным занятиям; изучение литературы по теме занятий, в том числе и с помощью ЭБС	отчет по лабораторным работам №1, 2, 3, 4, 5, 6;	ОПК-4.1, ОПК-4.2
Итого по разделу		8	12/4И		5			
3. Моделирование процессов								
3.1 Моделирование процессов прохождения УЗ волны в твердых телах	7	7	22		6,1	подготовка к лабораторным и семинарским занятиям; изучение литературы по теме занятий, в том числе и с помощью ЭБС	отчет по лабораторной работе № 7	ОПК-4.1, ОПК-4.2
Итого по разделу		7	22		40,95			
Итого за семестр		19	38/6И		15,1		зачёт	
Итого по дисциплине		19	38/6И		49,95		зачет	

5 Образовательные технологии

Для формирования компетенций и реализации предусмотренных видов учебной работы в учебном процессе используются традиционная, интерактивная и информационно-коммуникационные технологии.

Используются следующие виды лекций:

Информационная лекция – последовательное изложение материала в дисциплинарной логике, осуществляемое преимущественно вербальными средствами (монолог преподавателя).

Проблемная лекция – изложение материала, предполагающее постановку проблемных и дискуссионных вопросов, освещение различных научных подходов, авторские комментарии, связанные с различными моделями интерпретации изучаемого материала.

Лекция-визуализация – изложение содержания сопровождается презентацией (демонстрацией учебных материалов, представленных в различных знаковых системах, в т.ч. иллюстративных, графических, аудио- и видеоматериалов).

Теоретический материал закрепляется в ходе лабораторных занятий с применением IT-технологий.

6 Учебно-методическое обеспечение самостоятельной работы обучающихся

Представлено в приложении 1.

7 Оценочные средства для проведения промежуточной аттестации

Представлены в приложении 2.

8 Учебно-методическое и информационное обеспечение дисциплины (модуля)

а) Основная литература:

1. Берлине, Э. М. САПР технолога машиностроителя [Электронный ресурс]: учебник / Э. М. Берлинер, О. В. Таратынов. - М.: Форум, НИЦ ИНФРА-М, 2015. - 336 с.: 60x90 1/16. - (Высшее образование) (Переплёт). - Режим доступа: <http://znanium.com/bookread2.php?book=501435>. - Загл. с экрана.

2. Градов, В. М. Компьютерное моделирование [Электронный ресурс]: учебник / В. М. Градов, Г. В. Овечкин, П. В. Овечкин, И. В. Рудаков — М. : КУРС : ИНФРА-М, 2018. — 264 с. — Режим доступа: <http://znanium.com/bookread2.php?book=911733>. — Загл. с экрана.

б) Дополнительная литература:

1. Якушенков, Ю. Г. Теория и расчет оптико-электронных приборов [Электронный ресурс]: учебник / Ю. Г. Якушенков. - 6-е изд., перераб. и доп. - М. : Логос, 2011. - 568 с. - ISBN 978-5-98704-533-6 - Режим доступа: <http://znanium.com/catalog/product/469679>. — Загл. с экрана.

2. Калинеченко, А.В. Справочник инженера по контрольно-измерительным приборам в автоматике [Электронный ресурс] / А.В. Калинеченко, Н.В. Уваров, В.В. Дойников. - М.: Инфра-Инженерия, 2015. - 576 с. - ISBN 978-5-9729-0017-6 - Режим доступа: <http://znanium.com/catalog/product/520694>. — Загл. с экрана.

3. Безруков, А.И. Математическое и имитационное моделирование [Электронный ресурс]: учеб. пособие / А.И. Безруков, О.Н. Алексенцева. — М. : ИНФРА-М, 2017. — 227 с. + Доп. материалы; — (Высшее образование: Бакалавриат). — Режим доступа <http://znanium.com/bookread2.php?book=811122>. — Загл. с экрана.

в) Методические указания:

Методические указания к лабораторным работам приведены в приложении 3

г) Программное обеспечение и Интернет-ресурсы:

Программное обеспечение

Наименование ПО	№ договора	Срок действия лицензии
MS Office 2007 Professional	№ 135 от 17.09.2007	бессрочно
7Zip	свободно	бессрочно
Scilab Computation Engine	свободно распространяемое ПО	бессрочно
MathCAD v.15 Education University	Д-1662-13 от 22.11.2013	бессрочно
MathWorks MathLab v.2014 Classroom License	К-89-14 от 08.12.2014	бессрочно
FAR Manager	свободно	бессрочно

Профессиональные базы данных и информационные справочные системы

Название курса	Ссылка
Поисковая система Академия Google (Google Scholar)	URL: https://scholar.google.ru/
Информационная система - Единое окно доступа к информационным	URL: http://window.edu.ru/
Электронные ресурсы библиотеки МГТУ им. Г.И. Носова	https://magtu.informsystema.ru/Marc.html?locale=ru

9 Материально-техническое обеспечение дисциплины (модуля)

Материально-техническое обеспечение дисциплины включает:

1. Лекционная аудитория включает:
Мультимедийные средства хранения, передачи и представления информации
2. Аудитория для лабораторных работ включает:
Персональные компьютеры с пакетом MS Office, с выходом в Интернет, установленными пакетами MatCad и Scilab.
3. Аудиторий для групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации включает:
Интерактивная доска, проектор;
Мультимедийный проектор, экран
4. Помещения для самостоятельной работы обучающихся, оснащенных компьютерной техникой с возможностью подключения к сети «Интернет» и наличием доступа в электронную информационно-образовательную среду организации включает:
Персональные компьютеры с пакетом MS Office, с выходом в Интернет и с доступом в электронную информационно-образовательную среду университета.
5. Помещения для хранения и профилактического обслуживания учебного оборудования включает:
Стеллажи, сейфы для хранения учебного оборудования. Инструменты для ремонта

Примерная структура и содержание раздела:

По дисциплине «Компьютерные технологии в приборостроении» предусмотрена аудиторная и внеаудиторная самостоятельная работа обучающихся.

Аудиторная самостоятельная работа студентов предполагает семинарские и лабораторные занятия.

Вопросы к семинару №1:

1. Роль и задачи ИТ в процессе разработки ПС.
2. Способы проектирования. Аспекты, описания и иерархические уровни проектирования.
3. Определение САПР. Процесс проектирования ПС (на примере жизненного цикла). Типовые проектные процедуры (синтез, анализ, оптимизация).
4. Маршруты проектирования. Виды обеспечения САПР.
5. Определения схемы, конструкции и технологии ПС.
6. Уровни разукрупнения ПС по функциональной и конструктивной сложности.
7. ПС как методологическая система. Признаки системного подхода. Основы системного анализа.
8. Условная формализация технического процесса как системы.
9. Показатели параметрической чувствительности.
10. Задачи проектирования ПС, решаемые на основе исследования параметрической чувствительности с применением ИТ (настройка, регулировка, стабильность выходных характеристик и т. п.).
11. Методы расчета ФПЧ (метод вариаций, метод непосредственного дифференцирования, метод преобразованной и/или метод сопряженной /присоединенной/ модели). Теорема взаимности
12. Классификация расчетных моделей.
13. Аналитические модели – вектор функция, дифференциальные уравнения, матричные уравнения.
14. Структурные модели – направленные графы, блок-схемы.
15. Топологические модели ненаправленные графы, эквивалентные цепи.
16. Применение современных компьютерных измерительных технологий для проведения измерения физических величин, в том числе в режиме удаленного доступа.

Вопросы к семинару №2:

1. Какой вид имеет приглашение, выводимое при загрузке MATLAB?
2. Для чего служит команда HELP? DEMO? INFO ?
3. Перечислите основные команды MATLAB для работы в режиме прямых вычислений.
4. Для чего осуществляется перелистывание?

5. С помощью какой команды устанавливается формат чисел?
6. Для чего служит команда `format hex`?
7. Перечислите основные системные переменные MATLAB.
8. Приведите примеры математических функций системы MATLAB.
9. Как осуществляется ввод вектора-строки? Как осуществляется ввод вектора-столбца? Как осуществляется ввод матрицы?
10. Для чего служит команда `zeros`? Для чего служит команда `ones`? Для чего служит команда `rand`? Для чего служит команда `eye`?
11. Как определяется число строк и столбцов матрицы? Какие операции служат для определения минимального и максимального элемента матрицы?
12. С помощью каких операций осуществляется суммирование и умножение элементов матрицы?
13. Как осуществляется диалоговый ввод и вывод?
14. Для чего используются условные операторы? Для чего служит команда `diary`?
15. Как осуществляется переход во внешнюю среду? Какая команда служит для удаления переменных?
16. Для чего служит команда `dir`? Как осуществляется сохранение переменных в файл? Для чего используется команда `what`?
17. Какая команда служит для загрузки переменных из файла? Перечислите основные команды для работы с файлами.
18. Какие шкалы могут иметь графики? С помощью какой команды осуществляется построение графиков в полярной системе координат?
19. Для чего служит команда `mesh`? Как осуществляется построение столбцовой диаграммы? Как осуществляется задание надписей? Для чего используется команда `grid`? С помощью какой команды устанавливается масштаб?
20. Для чего служит команда `contour`? Как осуществляется разбивка окна на меньшие окна? Для чего используется команда `hold`?
21. Что называется функцией? Для чего служит функция `fmin`? Приведите пример описания функции. Для чего служит функция `ode 23`?
22. Перечислите основные математические функции системы MATLAB. Для чего служит функция `fzero`? Перечислите основные внешние расширения системы MATLAB для поиска экстремумов функций. Какая функция служит для вычисления определенного интеграла? Для чего служит функция `fmins`?
23. Перечислите основные внешние расширения системы MATLAB для решения системы дифференциальных уравнений.

а) Планируемые результаты обучения и оценочные средства для проведения промежуточной аттестации:

Код компетенции	Индикатор достижения компетенции	Оценочные средства
ОПК-4: Способен понимать принципы работы современных информационных технологий и использовать их для решения задач профессиональной деятельности		
ОПК-4.1	Осуществляет поиск, анализ и синтез информации с использованием информационных технологий	<p>Перечень вопросов для подготовки к экзамену:</p> <ol style="list-style-type: none"> 17. Роль и задачи ИТ в процессе разработки ПС. 18. Способы проектирования. Аспекты, описания и иерархические уровни проектирования. 19. Определение САПР. Процесс проектирования ПС (на примере жизненного цикла). Типовые проектные процедуры (синтез, анализ, оптимизация). 20. Маршруты проектирования. Виды обеспечения САПР. 21. Определения схемы, конструкции и технологии ПС. 22. Уровни разукрупнения ПС по функциональной и конструктивной сложности. 23. ПС как методологическая система. Признаки системного подхода. Основы системного анализа. 24. Условная формализация технического процесса как системы. 25. Показатели параметрической чувствительности. 26. Задачи проектирования ПС, решаемые на основе исследования параметрической чувствительности с применением ИТ (настройка, регулировка, стабильность выходных характеристик и т. п.). 27. Методы расчета ФПЧ (метод вариаций, метод непосредственного дифференцирования, метод преобразованной и/или метод сопряженной /присоединенной/ модели). Теорема взаимности

Код компетенции	Индикатор достижения компетенции	Оценочные средства
		<p>28. Классификация расчетных моделей.</p> <p>29. Аналитические модели – вектор функция, дифференциальные уравнения, матричные уравнения.</p> <p>30. Структурные модели – направленные графы, блок-схемы.</p> <p>31. Топологические модели ненаправленные графы, эквивалентные цепи.</p> <p>32. Применение современных компьютерных измерительных технологий для проведения измерения физических величин, в том числе в режиме удаленного доступа.</p> <p>33. Какой вид имеет приглашение, выводимое при загрузке MATLAB?</p> <p>34. Для чего служит команда HELP? DEMO? INFO ?</p> <p>35. Перечислите основные команды MATLAB для работы в режиме прямых вычислений.</p> <p>36. Для чего осуществляется перелистывание?</p> <p>37. С помощью какой команды устанавливается формат чисел?</p> <p>38. Для чего служит команда format hex?</p> <p>39. Перечислите основные системные переменные MATLAB.</p> <p>40. Приведите примеры математических функций системы MATLAB.</p> <p>41. Как осуществляется ввод вектора-строки? Как осуществляется ввод вектора-столбца? Как осуществляется ввод матрицы?</p> <p>42. Для чего служит команда zeros? Для чего служит команда ones? Для чего служит команда rand ? Для чего служит команда eye?</p> <p>43. Как определяется число строк и столбцов матрицы? Какие операции служат для определения минимального и максимального элемента матрицы?</p> <p>44. С помощью каких операций осуществляется суммирование и умножение элементов матрицы?</p> <p>45. Как осуществляется диалоговый ввод и вывод?</p>

Код компетенции	Индикатор достижения компетенции	Оценочные средства
		<p>46. Для чего используются условные операторы? Для чего служит команда diary?</p> <p>47. Как осуществляется переход во внешнюю среду? Какая команда служит для удаления переменных?</p> <p>48. Для чего служит команда dir? Как осуществляется сохранение переменных в файл? Для чего используется команда what?</p> <p>49. Какая команда служит для загрузки переменных из файла? Перечислите основные команды для работы с файлами.</p> <p>50. Какие шкалы могут иметь графики? С помощью какой команды осуществляется построение графиков в полярной системе координат?</p> <p>51. Для чего служит команда mesh? Как осуществляется построение столбцовой диаграммы? Как осуществляется задание надписей? Для чего используется команда grid? С помощью какой команды устанавливается масштаб?</p> <p>52. Для чего служит команда contour? Как осуществляется разбивка окна на меньшие окна? Для чего используется команда hold?</p> <p>53. Что называется функцией? Для чего служит функция fmin? Приведите пример описания функции. Для чего служит функция ode 23?</p> <p>54. Перечислите основные математические функции системы MATLAB. Для чего служит функция fzero? Перечислите основные внешние расширения системы MATLAB для поиска экстремумов функций. Какая функция служит для вычисления определенного интеграла? Для чего служит функция fmins?</p> <p>55. Перечислите основные внешние расширения системы MATLAB для решения системы дифференциальных уравнений.</p>
ОПК-4.2	Применяет технологии обработки данных, выбора данных по критериям; строит типичные модели решения предметных задач по изученным образцам	Комплексные задания :

Код компетенции	Индикатор достижения компетенции	Оценочные средства
		<ol style="list-style-type: none"> 1. Анализ процесса моделирования прохождения УЗВ в твердом теле 2. Определение положения несплошности по результатам моделирования прохождения УЗВ в металле.
ОПК-4.3	Использует современные информационные технологии для решения задач профессиональной деятельности	<p>Примерные практические задания на экзамене:</p> <ol style="list-style-type: none"> 1. Разработка файл-функции вычисления решения дифференциального уравнения в частных производных 2. Разработка программы решения дифференциального уравнения с помощью инструментов визуального моделирования

б) Порядок проведения промежуточной аттестации, показатели и критерии оценивания:

Промежуточная аттестация по дисциплине «Компьютерные технологии в приборостроении» включает теоретические вопросы, позволяющие оценить уровень усвоения обучающимися знаний, и практические задания, выявляющие степень сформированности умений и владений, проводится в форме экзамена и в форме выполнения и защиты курсовой работы.

Экзамен по данной дисциплине проводится в виде теста на ПК и выполнения практического задания с помощью программных средств моделирования.

Показатели и критерии оценивания экзамена:

– на оценку «отлично» (5 баллов) – обучающийся демонстрирует высокий уровень сформированности компетенций, всестороннее, систематическое и глубокое знание учебного материала, свободно выполняет практические задания, свободно оперирует знаниями, умениями, применяет их в ситуациях повышенной сложности.

– на оценку «хорошо» (4 балла) – обучающийся демонстрирует средний уровень сформированности компетенций: основные знания, умения освоены, но допускаются незначительные ошибки, неточности, затруднения при аналитических операциях, переносе знаний и умений на новые, нестандартные ситуации.

– на оценку «удовлетворительно» (3 балла) – обучающийся демонстрирует пороговый уровень сформированности компетенций: в ходе контрольных мероприятий допускаются ошибки, проявляется отсутствие отдельных знаний, умений, навыков, обучающийся испытывает значительные затруднения при оперировании знаниями и умениями при их переносе на новые ситуации.

– на оценку «неудовлетворительно» (2 балла) – обучающийся демонстрирует знания не более 20% теоретического материала, допускает существенные ошибки, не может показать интеллектуальные навыки решения простых задач.

– на оценку «неудовлетворительно» (1 балл) – обучающийся не может показать знания на уровне воспроизведения и объяснения информации, не может показать интеллектуальные навыки решения простых задач.

Методические рекомендации для самостоятельной работы студентов

В процессе выполнения самостоятельной работы студенты должны научиться воспринимать сведения на слух, фиксировать информацию в виде записей в тетрадях, работать с письменными текстами, самостоятельно извлекая из них полезные сведения и оформляя их в виде тезисов, конспектов, систематизировать информацию в виде заполнения таблиц, составления схем. Важно научиться выделять главные мысли в лекции преподавателя либо в письменном тексте; анализировать явления; определять свою позицию к полученным на занятиях сведениям, четко формулировать ее; аргументировать свою точку зрения: высказывать оценочные суждения; осуществлять самоанализ. Необходимо учиться владеть устной и письменной речью; вести диалог; участвовать в дискуссии; раскрывать содержание изучаемой проблемы в монологической речи; выступать с сообщениями и докладами.

Конспект лекции. Смысл присутствия студента на лекции заключается во включении его в активный процесс слушания, понимания и осмысления материала, подготовленного преподавателем. Этому способствует конспективная запись полученной информации, с помощью которой в дальнейшем можно восстановить основное содержание прослушанной лекции.

Для успешного выполнения этой работы советуем:

- подготовить отдельные тетради для каждого предмета. Запись в них лучше вести на одной стороне листа, чтобы позднее на чистой странице записать дополнения, уточнения, замечания, а также собственные мысли. С помощью разноцветных ручек или фломастеров можно будет выделить заголовки, разделы, термины и т.д.
- не записывать подряд все, что говорит лектор. Старайтесь вначале выслушать и понять материал, а затем уже зафиксировать его, не упуская основных положений и выводов. Сохраняйте логику изложения. Обратите внимание на необходимость точной записи определений и понятий.
- оставить место на странице свободным, если не успели осмыслить и записать часть информации. По окончании занятия с помощью однокурсников, преподавателя или учебника вы сможете восстановить упущенное.
- уделять внимание грамотному оформлению записей. Научитесь графически ясно и удобно располагать текст: вычленять абзацы, подчеркивать главные мысли, ключевые слова, помещать выводы в рамки и т.д. Немаловажное значение имеет и четкая структура лекции, в которую входит план, логически выстроенная конструкция освещения каждого пункта плана с аргументами и доказательствами, разъяснениями и примерами, а также список литературы по теме.
- научиться писать разборчиво и быстро. Чтобы в дальнейшем не тратить время на расшифровку собственных записей, следите за аккуратностью почерка, не экономьте бумагу за счет уплотнения текста. Конспектируя, пользуйтесь общепринятыми сокращениями слов и условными знаками, если есть необходимость, то при-думайте собственные сокращения.
- уметь быстро и четко переносить в тетрадь графические рисунки и таблицы. Для этих целей приготовьте прозрачную линейку, карандаш и резинку. Старайтесь как можно точнее скопировать изображение с доски. Если наглядный материал трудно воспроизводим в условиях лекции, то сделайте его словесное описание с обобщающими выводами.
- просмотреть свои записи после окончания лекции. Подчеркните и отметьте разными цветами фломастера важные моменты в записях. Исправьте неточности, внесите необходимые дополнения. Не тратьте время на переписывание конспекта, если он оказался не совсем удачным. Совершенствуйтесь, записывая последующие лекции.

Подготовка к лабораторным занятиям. Они предназначены для углубленного изучения отдельных тем и курсов. По форме проведения обычно представляют собой решение задач по теме лекций или индивидуальных задач.

Подготовка к занятиям заключается, прежде всего, в освоении того теоретического материала. Для этого необходимо в первую очередь перечитать конспект лекции или разделы учебника, в которых присутствует установочная информация. Изучение рекомендованной литературы необходимо сделать максимально творчески – не просто укладывая в память новые сведения, а осмысливая и анализируя материал.

Подготовка к экзамену. Готовиться к экзамену нужно заранее и в несколько этапов. Для этого:

- Просматривайте конспекты лекций сразу после занятий. Это поможет разобраться с непонятными моментами лекции и возникшими вопросами, пока еще лекция свежа в памяти.
 - Бегло просматривайте конспекты до начала следующего занятия. Это позволит «освежить» предыдущую лекцию и подготовиться к восприятию нового материала.
 - Каждую неделю отводите время для повторения пройденного материала.
- Непосредственно при подготовке:
- Упорядочьте свои конспекты, записи, задания.
 - Прикиньте время, необходимое вам для повторения каждой части (блока) материала, выносимого на зачет.
 - Составьте расписание с учетом скорости повторения материала, для чего
 - Разделите вопросы для экзамена на знакомые (по лекционному курсу, семинарам, конспектированию), которые потребуют лишь повторения и новые, которые придется осваивать самостоятельно. Начните с тем хорошо вам известных и закрепите их с помощью конспекта и учебника. Затем пополните свой теоретический багаж новыми знаниями, обязательно воспользовавшись рекомендованной литературой.
 - Правильно используйте консультации, которые проводит преподаватель. Приходите на них с заранее проработанными самостоятельно вопросами. Вы можете получить разъяснение по поводу сложных, не до конца понятых тем, но не рассчитывайте во время консультации на исчерпывающую информацию по содержанию всего курса.

Методические материалы к лабораторным работам

Лабораторная работа 1. Вектора.

Теоретический материал.

Базовой структурой данных в Scilab является матрица. Всякая матрица характеризуется своим размером (т.е. числом строк и столбцов) и типом содержащихся в ней значений. Элементами матрицы могут являться вещественные, комплексные или целые числа, логические значения, строки или полиномы. Если две матрицы имеют совпадающее число строк и столбцов, говорят, что они имеют одинаковый размер.

Частным случаем матриц являются векторы, в которых число строк либо столбцов равно 1. Собственно скалярные величины в Scilab отсутствуют - скалярное значение представляется матрицей 1×1 .

Вектор в Scilab — это упорядоченная совокупность элементов (одномерный массив) одного типа данных. Упорядоченность для пользователя в этом смысле проявляется в том, что к каждому элементу вектора можно обратиться по его уникальному порядковому номеру или индексу. В среде Scilab все индексы начинаются с единицы, что немного не привычно, так как например в программировании на языке Си те же индексы массивов начинаются с нуля.

Для ввода вектора используются квадратные скобки, элементы вектора отделяются друг от друга:

- точкой с запятой, если требуется получить вектор–столбец;
- пробелом или запятой, если необходимо разместить элементы в вектор–строке.

Занесите вектор–столбцы и вектор–строки:

$$a = \begin{bmatrix} 0.2 \\ -3.9 \\ 4.6 \end{bmatrix} \quad b = \begin{bmatrix} 7.6 \\ 0.1 \\ 2.5 \end{bmatrix} \quad u = [0.1 \ 0.5 \ -3.7 \ 8.1] \quad v = [5.2 \ 9.7 \ 3.4 \ -0.2]$$

в соответствующие массивы, набрав в командной строке:

```
-->a = [0.2; -3.9; 4.6];    --> b=[7.6; 0.1; 2.5];    --> u=[0.1 0.5 -3.7 8.1];    --> v=[5.2 9.7 3.4 -0.2];
```

Точка с запятой в конце каждой строки поставлена для подавления вывода на экран, она никак не связана с точкой с запятой, которая является разделителем элементов в вектор–столбцах. Вектор в Scilab представляются двумерными массивами, один из размеров которых равен единице.

Для получения длины вектора предназначена функция `length`, вектор указывается в качестве ее входного аргумента:

```
--> L=length(a)
L =3
```

Вектор-столбцы с одинаковым числом элементов можно складывать и вычитать друг из друга при помощи знаков "+" и "-". Аналогичное верно и для вектор-строк:

```
--> c=a+b;
```

```
--> w=u-v;
```

Сложение и вычитание вектор-строки и вектор-столбца или векторов разных размеров приводит к ошибке. Операция * предназначена для умножения векторов по правилу матричного умножения. Поскольку Scilab различает вектор-строки и вектор столбцы, то допустимо либо умножение вектор-строки на такой же по длине вектор-столбец (скалярное произведение), либо умножение вектор-столбца на вектор-строку (внешнее произведение, в результате которого получается прямоугольная матрица).

Векторное произведение двух векторов cross:

```
--> c=cross(a,b)
```

Векторное произведение определено только для векторов из трех элементов.

Для операции транспонирования зарезервирован апостроф '. Если вектор содержит комплексные числа, то операция ' приводит к комплексно-сопряженному вектору.

Scilab поддерживает поэлементные операции с векторами. Наряду с умножением по правилу матричного умножения, существует операция поэлементного умножения .* (точка со звездочкой). Данная операция применяется к векторам одинаковой длины и приводит к вектору той же длины, что исходные, элементы которого равны произведениям соответствующих элементов исходных векторов. Например, для векторов a и b, введенных выше, поэлементное умножение дает следующий результат:

```
--> c=a.*b
```

```
c =1.52
```

```
-0.39
```

```
11.5
```

Аналогичным образом работает поэлементное деление ./ (точка с косой чертой). Кроме того, операция \ (точка с обратной косой чертой) осуществляет обратное поэлементное деление, то есть выражения a./b и b.\a эквивалентны. Возведение элементов вектора a в степени, равные соответствующим элементам b, производится с использованием ^. Для транспонирования вектор-строк или вектор-столбцов предназначено сочетание .' (точка с апострофом). Операции ' и .' для вещественных векторов приводят к одинаковым результатам. Не обязательно применять поэлементные операции при умножении вектора на число и числа на вектор, делении вектора на число, сложении и вычитании вектора и числа. При выполнении, например, операции a*2, результат представляет собой вектор того же размера, что и a, с удвоенными элементами.

Вектор, элементы которого представляют собой числовую последовательность в виде арифметической прогрессии, может быть создан особым образом. Для этого используется конструкция <начальное значение>:<шаг>:<конечное значение>

Например, создадим вектор с начальным значением -5, конечным значением 10 и шагом между элементами 2

```
-->-5:2:10
```

```
ans = -5. -3. -1. 1. 3. 5. 7. 9.
```

Вообще говоря, шаг можно не указывать. В этом случае его значение по умолчанию принимается равным 1. Если шаг не указывается, то начальное значение всегда должно быть меньше конечного, иначе Scilab создаст пустой вектор.

Векторы могут быть аргументами встроженных математических функций, таких, как sin, cos и т. д. В результате получается вектор с элементами, равными значению вызываемой функции от соответствующих элементов исходного вектора, например:

```
-->q=sin(u)
```

Однако для вычисления более сложной функции от вектора значений, скажем

$$\frac{v \sin v + v^2}{v + 1}$$

выражение f=(v*sin(v)+v^2)/(v+1) вызовет ошибку уже при попытке умножения v на sin(v). Дело в том, что v является вектор-строкой длиной четыре, т. е. хранится в двумерном массиве размером один на четыре. Точно также представлен и sin(v), следовательно, умножение при помощи звездочки (по правилу матричного умножения) лишено смысла. Аналогичная ситуация возникает и при возведении вектора v в квадрат, т. е., фактически, при вычислении v*v. Правильная запись выражения в Scilab требует использования поэлементных операций:

```
-->f=(v.*sin(v)+v.^2)/(v+1)
```

Если вы хотите обратиться к последнему элементу вектора, но не знаете его размер, вы можете воспользоваться служебным символом в виде знака доллара '\$'.

```
-->b($)
```

Так же, для обращения к последнему элементу можно воспользоваться функцией длины: `b(length(b))`.

Очень часто требуется обработать только часть вектора, или обратиться к некоторым его элементам. Разберем правила Scilab, по которым производится индексация векторных данных. Для доступа к элементу вектора необходимо указать его номер в круглых скобках сразу после имени переменной, в которой содержится вектор. Например, сумма первого и третьего элементов вектора `v` находится при помощи выражения

```
--> s=v(1)+v(3);
```

Указание номеров элементов вектора можно использовать и при вводе векторов, последовательно добавляя новые элементы (не обязательно в порядке возрастания их номеров). Команды:

```
-->h=5
```

```
-->h(2)=3
```

```
-->h(4)=1
```

приводят к образованию вектора:

```
h =5.
```

```
3.
```

```
0.
```

```
1.
```

Заметьте, что для ввода первого элемента `h` не обязательно указывать его индекс, т.к. при выполнении оператора `h=1` создается вектор (массив размера один на один). Следующие операторы присваивания приводят к автоматическому увеличению длины вектора `h`, а пропущенные элементы (в нашем случае `h(3)`) получают значение ноль.

В любой момент вы можете удалить элемент вектора. Для этого достаточно на желаемой позиции создать, так называемую, пустую матрицу в виде конструкции `[]`.

```
-->b(1)=[] // удаляет первый элемент вектора b
```

Индексация двоеточием позволяет выделить идущие подряд элементы в новый вектор. Начальный и конечный номера указываются в круглых скобках через двоеточие, например:

```
-->z=[0.2 -3.8 7.9 4.5 7.2 -8.1 3.4];
```

```
-->znew=z(3:6)
```

```
znew =7.9 4.5 7.2 -8.1
```

Scilab обладает большим набором встроенных функций для обработки векторных данных, часть из них приведена в табл. 2.1. Для получения подробной информации о каждой функции требуется указать ее имя в качестве аргумента команды `help`. Обратите внимание на то, что ряд функций допускает обращение к ним как с одним, так и с двумя выходными аргументами. В случае нескольких выходных аргументов они заключаются в квадратные скобки и отделяются друг от друга запятой.

Таблица. 2.1 Функции обработки данных

Функции	Назначение
<code>s=sum(a)</code>	Сумма всех элементов вектора <code>a</code>
<code>p=prod(a)</code>	Произведение всех элементов вектора <code>a</code>
<code>m=max(a)</code>	Нахождение максимального значения среди элементов вектора <code>a</code>
<code>[m, k]=max(a)</code>	Второй выходной аргумент <code>k</code> содержит номер максимального элемента в векторе <code>a</code>
<code>m=min(a)</code>	Нахождение минимального значения среди элементов вектора <code>a</code>
<code>[m, k]=min(a)</code>	Второй выходной аргумент <code>k</code> содержит номер минимального элемента в векторе <code>a</code>
<code>m=mean(a)</code>	Вычисление среднего арифметического элементов вектора <code>a</code>
<code>a1= gsort(a)</code>	Упорядочение элементов вектора по возрастанию
<code>[a1, ind]=gsort(a)</code>	Второй выходной аргумент <code>ind</code> является вектором из целых чисел от 1 до <code>length(a)</code> , который соответствует проделанным перестановкам

Применение встроенных функций обработки данных к некоторым последовательно расположенным элементам вектора не представляет труда. Следующий вызов функции `prod` вычисляет произведение элементов вектора `z` со второго по шестой:

```
--> p=prod(z(2:6))
```

Индексация вектором служит для выделения элементов с заданными индексами в новый вектор. Индексный вектор должен содержать номера требуемых элементов, например:

```
--> ind=[3 5 7]
ind =
    3.    5.    7.
```

```
--> znew=z(ind)
znew =
    7.9    7.2    3.4
```

Подумайте, как найти сумму элементов произвольного вектора `z` с четными индексами. Вот правильное решение:

```
--> ind=2:2:length(z)
ind =
    2.    4.    6.
--> s=sum(z(ind))
s =
    - 7.4
```

Конструирование новых векторов из элементов имеющихся векторов производится при помощи квадратных скобок. Следующий оператор приводит к образованию вектора, в котором пропущен пятый элемент вектора `z`

```
--> znew=[z(1:4) z(6:$)]
znew =
    0.2 - 3.8    7.9    4.5 - 8.1    3.4
```

Задания для самостоятельной работы

Часть 1. Для заданных векторов `a` и `b` длины `n`:

1. вычислить их сумму, разность и скалярное произведение;
2. образовать вектор $c = [a_1, a_2, \dots, a_n, b_1, b_2, \dots, b_n]$, определить его максимальный и минимальный элементы и поменять их местами;
3. упорядочить вектор `c` по возрастанию и убыванию;
4. переставить элементы вектора `c` в обратном порядке и записать результат в новый вектор;
5. найти векторное произведение $u = [a_1, a_3, a_4]$ и $v = [b_2, b_3, b_4]$.

Варианты:

$a = [0.5 \ 3.7 \ 6.0 \ -4.3 \ 1.2 \ -2.7 \ 2.4 \ 2.2];$	$b = [3.6 \ 7.0 \ 7.0 \ 5.4 \ 2.6 \ -2.7 \ -6.4 \ 0.3].$
$a = [-4.8 \ -1.3 \ -1.0 \ 0.7 \ 4.0 \ 5.8 \ 4.3 \ -8.0];$	$b = [-1.1 \ -1.9 \ 7.1 \ -2.1 \ 6.8 \ 2.8 \ 0.3 \ 1.6].$
$a = [1.0 \ -3.9 \ -2.3 \ -3.3 \ -1.7 \ 2.2 \ -0.6 \ 1.8];$	$b = [2.7 \ -2.7 \ -2.2 \ 4.4 \ 0.4 \ -6.0 \ -3.4 \ -5.2].$
$a = [-2.4 \ 3.3 \ -0.1 \ 3.6 \ 7.4 \ -2.8 \ 0.3 \ 2.2];$	$b = [6.3 \ 0.6 \ 4.3 \ -3.7 \ -7.0 \ 3.7 \ 3.7 \ 8.0].$
$a = [8.4 \ -5.9 \ -6.5 \ -0.9 \ 6.9 \ -1.7 \ 1.7 \ 0.8];$	$b = [-0.0 \ 2.0 \ -1.5 \ 7.5 \ -4.0 \ -3.0 \ -6.2 \ 0.0].$
$a = [5.3 \ 6.8 \ -7.1 \ 6.8 \ -4.0 \ -2.3 \ -4.4 \ -0.2];$	$b = [7.5 \ -1.5 \ -4.9 \ -4.6 \ -2.3 \ -5.3 \ 5.5 \ 2.3].$
$a = [1.2 \ -4.1 \ -0.8 \ -0.7 \ -2.2 \ 1.7 \ 3.3 \ -6.1];$	$b = [-1.5 \ 2.2 \ 1.0 \ -4.3 \ -0.0 \ -1.8 \ -1.5 \ 2.4].$
$a = [6.6 \ -5.0 \ -2.7 \ 8.3 \ 3.8 \ 1.9 \ 1.1 \ 2.7];$	$b = [-1.0 \ 3.2 \ 4.2 \ -6.4 \ 1.9 \ -6.5 \ -6.2 \ -8.1].$
$a = [-1.9 \ 0.4 \ 1.8 \ 4.2 \ -3.8 \ -4.7 \ 4.0 \ -2.1];$	$b = [-8.7 \ -4.2 \ -1.4 \ 2.8 \ -2.2 \ 7.8 \ 0.0 \ -0.1].$
$a = [0.9 \ 1.7 \ -3.2 \ -3.8 \ 7.3 \ 6.0 \ -0.2 \ 8.6];$	$b = [0.6 \ -0.4 \ -6.9 \ -2.2 \ 1.6 \ 3.8 \ -3.2 \ 0.4].$

Часть 2. Вычислить значения функции на отрезке в заданном числе N равномерно отстоящих друг от друга точек.

Варианты.

1.	$y(x) = \frac{\sin x \cos x}{x^2 + 1}$	$[0, 2\pi]$	$N = 10$
2.	$y(x) = \ln(x+1) \sqrt{e^x + e^{-x}}$	$[-0.2, 4]$	$N = 8$
3.	$y(x) = x^2 \operatorname{tg} \sqrt{\arcsin x}$	$[0, \frac{1}{3}]$	$N = 9$
4.	$y(x) = x \sin x + x^3 \frac{e^x}{x+1}$	$[0, 1]$	$N = 7$
5.	$y(x) = \frac{1}{1 + \frac{x}{\sqrt{1+x}}}$	$[0, 3]$	$N = 9$
6.	$y(x) = \frac{e^{\sin x} + e^{\cos x}}{x^2}$	$[\pi, 3\pi]$	$N = 11$
7.	$y(x) = \operatorname{ctg}(x^2 + 1) \cdot (\sin 2x + \cos 2x)$	$[-1, 1]$	$N = 7$
8.	$y(x) = \log_2(x^2 + 1) \cdot \sin \frac{1}{x^2 + 1}$	$[-1, 1]$	$N = 10$
9.	$y(x) = x^3 + 2x^2 - 3 \sin \pi x$	$[-2, 2]$	$N = 7$
10.	$y(x) = \frac{\sqrt[3]{x+1}}{\sqrt{ x + \frac{1}{2}}} \cdot \frac{\sin x + 1}{\cos x + 2}$	$[-2, 2]$	$N = 9$

Лабораторная работа 2. Матрицы.

Теоретический материал.

Базовой структурой данных в Scilab является матрица. Scilab создавался в первую очередь для работы с матрицами вещественных значений, и поэтому содержит большое число функций для работы с вещественно-значными матрицами.

В числе задач проектирования Scilab также стояла оптимизация скорости выполнения таких операций. Для этого было разработано специальное внутреннее представление матриц, позволяющее интерпретатору эффективно манипулировать ими. Основные операции линейной алгебры, такие как сложение, вычитание, транспонирование и скалярное произведение, выполняются оптимизированными встроенными функциями. Эти операции обозначаются в Scilab символами "+", "-", "*", ".*" и ".*".

При использовании высокоуровневых операторов и функций практически отпадает необходимость в реализации циклов, которые, помимо прочего, выполняются существенно медленнее (от 10 до 100 раз), нежели встроенные функции. Данное свойство Scilab носит название векторизации.

Матрицы небольших размеров удобно вводить из командной строки. Существует три способа ввода матриц. Например, матрицу

$$A = \begin{bmatrix} 0.7 & -2.5 & 9.1 \\ 8.4 & 0.3 & 1.7 \\ -3.5 & 6.2 & 4.7 \end{bmatrix}$$

можно ввести следующим образом: набрать в командной строке (разделяя элементы строки матрицы пробелами): `A=[0.7 -2.5 9.1` и нажать <Enter>. Курсор перемещается в следующую строку (символ приглашения командной строки >> отсутствует). Элементы каждой следующей строки матрицы набираются через пробел, а ввод строки завершается нажатием на <Enter>. При вводе последней строки в конце ставится закрывающая квадратная скобка:

```
-->A=[0.7 -2.5 9.1
-->8.4 0.3 1.7
-->-3.5 6.2 4.7]
```

Если после закрывающей квадратной скобки не ставить точку с запятой для подавления вывода в командное окно, то матрица выведется в виде таблицы.

Другой способ ввода матрицы основан на том, что матрицу можно рассматривать как вектор-столбец, каждый элемент которого является строкой матрицы. Поскольку точка с запятой используется для разделения элементов вектор-столбца, то ввод, к примеру, матрицы

$$B = \begin{bmatrix} 6.1 & 0.3 \\ -7.9 & 4.4 \\ 2.5 & -8.1 \end{bmatrix}$$

осуществляется оператором присваивания:

```
-->B=[6.1 0.3; -7.9 4.4; 2.5 -8.1]
```

Введите матрицу и отобразите ее содержимое в командном окне, набрав в командной строке `B` и нажав <Enter>.

Очевидно, что допустима такая трактовка матрицы, при которой она считается вектор-строкой, каждый элемент которой является столбцом матрицы. Следовательно, для ввода матрицы

$$C = \begin{bmatrix} 0.4 & -7.2 & 5.3 \\ 0.1 & -2.1 & -9.5 \end{bmatrix}$$

достаточно воспользоваться командой:

```
-->C=[[0.4; 0.1] [-7.2; -2.1] [5.3; -9.5]]
```

Обратите внимание, что внутренние квадратные скобки действительно нужны. Оператор `C=[0.4; 0.1 -7.2; -2.1 5.3; -9.5]` является недопустимым и приводит к сообщению об ошибке, поскольку оказывается, что в первой строке матрицы содержится только один элемент, во второй и третьей — по два, а в четвертой — снова один.

Функция `size` позволяет установить размеры массивов, она возвращает результат в виде вектора, первый элемент которого равен числу строк, а второй — столбцов:

```
-->s=size(B)
s =3. 2.
```

Сложение и вычитание матриц одинаковых размеров производится с использованием знаков +, -. Звездочка * служит для вычисления матричного произведения, причем соответствующие размеры матриц должны совпадать, например:

```
-->P=A*B
P =
46.77 - 84.5
53.12 - 9.93
- 58.58 - 11.84
```

Допустимо умножение матрицы на число и числа на матрицу, при этом происходит умножение каждого элемента матрицы на число и результатом является матрица тех же размеров, что и исходная.

Апостроф ' предназначен для транспонирования вещественной матрицы или нахождения сопряженной к комплексной матрице. Для возведения квадратной матрицы в степень применяется знак ^.

Вычислите для тренировки матричное выражение $R = (A - BC)^3 + ABC$, в котором A, B и C — определенные выше матрицы.

```
-->R=(A-B*C)^3+A*B*C
```

```
R =
- 45358.407    166093.21  - 657876.12
   81173.314  - 277025.94    1290559.3
- 42599.757    127425.41  - 787112.23
```

Scilab обладает многообразием различных функций и способов для работы с матричными данными. Для обращение к элементу массива следует указать его строчный и столбцевой индексы в круглых скобках после имени массива, например:

```
-->C(1,2)
```

```
ans =
- 7.2
```

Индексация двоеточием позволяет получить часть матрицы — строку, столбец или блок, например:

```
-->c1=A(2:3,2)
```

```
c1 =
  0.3
  6.2
```

```
-->r1=A(1,1:3)
```

```
r1 =
  0.7  - 2.5    9.1
```

Для обращения ко всей строке или всему столбцу не обязательно указывать через двоеточие начальный (первый) и конечный индексы, то есть операторы $r1=A(1,1:3)$ и $r1=A(1,:)$ эквивалентны. Для доступа к элементам строки или столбца от заданного до последнего можно использовать \$, так же как и для векторов: $A(1,2:$)$. Выделение блока, состоящего из нескольких строк и столбцов, требует индексации двоеточием как по первому измерению, так и по второму. Пусть в массиве T хранится матрица:

$$T = \begin{bmatrix} 1 & 7 & -3 & 2 & 4 & 9 \\ 0 & -5 & -6 & 3 & -8 & 7 \\ 2 & 4 & 5 & -1 & 0 & 3 \\ -6 & -4 & 7 & 2 & 6 & 1 \end{bmatrix}$$

Для выделения ее элементов (обозначенных курсивом) со второй строки по третью и со второго столбца по четвертый, достаточно использовать оператор:

```
-->T1=T(2:3,2:4)
```

```
T1 =
- 5.   - 6.    3.
  4.    5.   - 1.
```

Индексация двоеточием так же очень полезна при различных перестановках в массивах. В частности, для перестановки первой и последней строк в произвольной матрице, хранящейся в массиве A, подойдет последовательность команд:

```
s=A(1,:)
```

```
s =
  0.7  - 2.5    9.1
```

```
-->A(1,:)=A($,:)
```

```
A =
- 3.5    6.2    4.7
  8.4    0.3    1.7
```

```
- 3.5    6.2    4.7
```

```
-->A($,:)=s
```

```
A =
- 3.5    6.2    4.7
  8.4    0.3    1.7
  0.7  - 2.5    9.1
```

Scilab поддерживает такую операцию, как вычеркивание строк или столбцов из матрицы. Достаточно удаляемому блоку присвоить пустой массив, задаваемый квадратными скобками. Например, вычеркивание второй и третьей строк из массива T, введенного выше, производится следующей командой:

```
-->T(2:3,:)=[]
```

```
T =
```

```

1.   7.  -3.   2.   4.   9.
-6.  -4.   7.   2.   6.   1.

```

$$W = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Индексация двоеточием упрощает заполнение матриц, имеющих определенную структуру. Предположим, что требуется создать матрицу

Первый шаг состоит в определении нулевой матрицы размера пять на пять, затем заполняются первая и последняя строки и первый и последний столбцы:

```

-->W(1:5,1:5)=0;
-->W(1,:)=1;
-->W($,:)=1;
-->W(:,1)=1;
-->W(:,5)=1;

```

Проверьте, что в результате получается требуемая матрица. Ряд встроенных функций, приведенных в табл. 3.1, позволяет ввести стандартные матрицы заданных размеров. Обратите внимание, что во всех функциях, кроме `diag`, допустимо указывать размеры матрицы следующими способами:

- числами через запятую (в двух входных аргументах);
- одним числом, результат — квадратная матрица;
- вектором из двух элементов, равных числу строк и столбцов.

Последний вариант очень удобен, когда требуется создать стандартную матрицу тех же размеров, что и некоторая имеющаяся матрица. Если, к примеру, `A` была определена ранее, то команда `I=eye(size(A))` приводит к появлению единичной матрицы, размеры которой совпадают с размерами `A`, так как функция `size` возвращает размеры матрицы в векторе.

Таблица. 3.1 Функции для создания стандартных матриц

Функция	Результат и примеры вызовов
<code>zeros</code>	Нулевая матрица <code>F=zeros(4,5)</code> <code>F=zeros(3)</code> <code>F=zeros([3 4])</code>
<code>eye</code>	Единичная прямоугольная матрица (единицы расположены на главной диагонали) <code>I=eye(5,8)</code> <code>I=eye(5)</code> <code>I=eye([5 8])</code>
<code>ones</code>	Матрица, целиком состоящая из единиц <code>E=ones(3,5)</code> <code>E=ones(6)</code> <code>E=ones([2 5])</code>
<code>rand</code>	Матрица, элементы которой — случайные числа, равномерно распределенные на интервале (0,1) <code>R=rand(5,7)</code> <code>R=rand(6)</code> <code>R=rand([3 5])</code>

<code>diag</code>	1) диагональная матрица, элементы которой задаются во входном аргументе — векторе <code>D=diag(v)</code> 2) диагональная матрица со смещенной на <code>k</code> позиций диагональю (положительные <code>k</code> — смещение вверх, отрицательные — вниз), результатом является квадратная матрица размера <code>length(v)+abs(k)</code> <code>D=diag(v,k)</code> 3) выделение главной диагонали из матрицы в вектор <code>d=diag(A)</code> 4) выделение <code>k</code> -ой диагонали из матрицы в вектор <code>d=diag(A,k)</code>
-------------------	--

Разберем, как получить трехдиагональную матрицу размера семь на семь, приведенную ниже, с использованием функций Scilab.

$$T = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 5 & 2 & -2 & 0 & 0 & 0 & 0 \\ 0 & 5 & 3 & -3 & 0 & 0 & 0 \\ 0 & 0 & 5 & 4 & -4 & 0 & 0 \\ 0 & 0 & 0 & 5 & 5 & -5 & 0 \\ 0 & 0 & 0 & 0 & 5 & 6 & -6 \\ 0 & 0 & 0 & 0 & 0 & 5 & 7 \end{bmatrix}$$

Введите вектор `v` с целыми числами от одного до семи и используйте его для создания диагональной матрицы и матрицы со смещенной на единицу вверх диагональю. Вектор длины шесть, содержащий пятерки, заполняется, например, так: `5*ones(1,6)`. Этот вектор укажите в первом аргументе функции `diag`, а минус единицу — во втором и получите третью вспомогательную матрицу. Теперь достаточно вычесть из первой матрицы вторую и сложить с третьей:

```

--> T=diag(v)-diag(v(1:6),1)+diag(5*ones(1,6),-1)

```

Для создания треугольной матрицы из существующей матрицы используются функции: `triu(A,k)` - формирует из матрицы `A` нижнюю треугольную матрицу начиная с главной или с `k`-ой диагонали; `triu(A,k)` - формирует из матрицы `A` верхнюю треугольную матрицу начиная с главной или с `k`-ой диагонали.

```

-->A=[1 2 3;4 5 6;7 8 9]

```

```

A =
1.   2.   3.
4.   5.   6.
7.   8.   9.

-->tril(A)
ans =
1.   0.   0.
4.   5.   0.

-->tril(A,1)
ans =
7.   8.   9.
1.   2.   0.

```


4.	5.	6.	0.	0.	0.	<i>ans</i> =		
7.	8.	9.	4.	0.	0.	1.	2.	3.
-->tril(A,-1)			7.	8.	0.	0.	5.	6.
<i>ans</i> =			-->triu(A)			0.	0.	9.

Для вычисления детерминанта матрицы используется функция det.

-->det(A)

Поэлементные вычисления с матрицами производятся практически аналогично поэлементарным вычислениям с векторами, разумеется, необходимо следить за совпадением размеров матриц:

A.*B, A./B— поэлементные умножение и деление;

A.^p — поэлементное возведение в степень, p — число;

A.^B — возведение элементов матрицы A в степени, равные соответствующим элементам матрицы B;

A.' — транспонирование матрицы (для вещественных матриц A' и A.' приводят к одинаковым результатам);

Допустимо записывать сумму и разность матрицы и числа, при этом сложение или вычитание применяется, соответственно, ко всем элементам матрицы. Вызов математической функции от матрицы приводит к матрице того же размера, на соответствующих позициях которой стоят значения функции от элементов исходной матрицы.

В Scilab определены и матричные функции, например, sqrtm предназначена для вычисления квадратного корня. Найдите квадратный корень из матрицы

$$K = \begin{bmatrix} 3 & 2 \\ 1 & 4 \end{bmatrix}$$

и проверьте полученный результат, возведя его в квадрат (по правилу матричного умножения, а не поэлементно!):

-->K=[3 2; 1 4];	-->S*S
-->S=sqrtm(K)	<i>ans</i> =
S =	3. 2.
1.688165 0.5479029	1. 4.
0.2739515 1.9621165	

Все функции обработки данных, приведенные в табл. 2.1 (лабораторная работа № 2), могут быть применены и к двумерным массивам. Рассмотрим для примера функцию sum. Основное отличие от обработки векторных данных заключается в следующем:

sum(X[,fl]) – вычисляет сумму элементов массива X, имеет необязательный параметр fl; если параметр fl, отсутствует то функция sum(X) возвращает скалярное значение равное сумме элементов массива если ; fl='r' или fl=1, , что то же самое то функция вернет строку равную поэлементной сумме столбцов матрицы X; если fl='c' или fl=2, то результатом работы функции будет вектор столбец каждый элемент которого равен сумме элементов строк матрицы X

9]	-->M=[1 2 3; 4 5 6; 7 8]	-->sum(M)	-->sum(M,2)
	M =	<i>ans</i> =	<i>ans</i> =
	1. 2. 3.	45.	6.
	4. 5. 6.	-->sum(M,1)	15.
	7. 8. 9.	<i>ans</i> =	24.
		12. 15. 18.	

Аналогичным образом работают и другие функции из таблицы 2.1.

Функция cat служит для объединения матриц.

Очень удобной возможностью Scilab является конструирование матрицы из матриц меньших размеров. Для объединения матриц можно пойти двумя путями: воспользоваться функцией cat(n, A, B, [C, ...]) - объединяет матрицы A и B, или все входящие матрицы при n=1, по строкам при n=2; или же объединить матрицы стандартным способом: [A; B] или [A, B].

-->A=[1 2; 3 4]	-->cat(2,A,B)	1. 2.
A =	<i>ans</i> =	3. 4.
1. 2.	1. 2. 5.	5. 6.
3. 4.	3. 4. 7.	7. 8.
6.		
-->B=[5 6; 7 8]	-->cat(1,A,B)	-->[A B]
B =	<i>ans</i> =	<i>ans</i> =
5. 6.	1. 2. 5.	
7. 8.	6.	

8. 3. 4. 7.

-->[A; B]
ans =

1. 2.
3. 4.
5. 6.
7. 8.

Задания для самостоятельной работы

Часть 1. Введите матрицы и найдите значения следующих выражений.

$$A = \begin{bmatrix} -9.8 & 4.4 & 1.3 \\ -5.7 & 0.1 & 0.8 \\ 2.4 & 4.4 & 8.6 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 0 & 2 \\ 3 & 0 & -1 \\ 5 & 2 & 2 \\ 8 & 9 & 0 \end{bmatrix} \quad C = \begin{bmatrix} 0.1 & 0.2 & -1.3 & 0.7 \\ -0.2 & 0.3 & 2.2 & 0.8 \\ 1.9 & 2.3 & 6.5 & 4.9 \end{bmatrix}$$

Варианты

1. $(A^3 + CB)(A^2 - 3CB)^T$. 2. $A^4 + 2A^3 - ACB$.
3. $BAC - 4C^T B^T$. 4. $3BA^3C - BAC + 2BC$.
5. $-3C^T AC - BB^T$. 6. $(BCB - 4C^T)A^4$.
7. $(AB^T - C)(C + AB^T)^T - 3A$. 8. $(AB^T B)^4 - 2A^3 + CC^T$.
9. $C(BB^T + C^T C)C^T - 8A$. 10. $2AA^T - (CB)^2 + 4A$.

Часть 2. При помощи встроенных функций для заполнения стандартных матриц, индексации двоеточием и, возможно, поворота, транспонирования или вычеркивания получите следующие матрицы:

Варианты

1. $\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 0 & 0 & 0 & 7 & 1 \\ 1 & 0 & 2 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 2 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 2 & 0 & 1 \\ 1 & 0 & 0 & 0 & 2 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$ 2. $\begin{bmatrix} 2 & 1 & 0 & 0 & 0 & 0 & 5 \\ -1 & 3 & 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 5 & 1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 6 & 1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 7 & 0 \\ 5 & 0 & 0 & 0 & 0 & -1 & 8 \end{bmatrix}$ 3. $\begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & 6 & 0 & 0 & 0 & 0 \\ 0 & 7 & 0 & 0 & 0 & 0 & 0 \\ 8 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ 7. $\begin{bmatrix} 2 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 2 & 1 & 1 & 1 \end{bmatrix}$ 8. $\begin{bmatrix} 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$
4. $\begin{bmatrix} 4 & 3 & 3 & 3 & 3 & 3 & 9 \\ 3 & 4 & 3 & 3 & 3 & 3 & 3 \\ 3 & 3 & 4 & 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 4 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 & 4 & 3 & 3 \\ 3 & 3 & 3 & 3 & 3 & 4 & 3 \\ 9 & 3 & 3 & 3 & 3 & 3 & 4 \end{bmatrix}$ 5. $\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 1 \\ 2 & 0 & 0 & 0 & 7 & 0 & 1 \\ 3 & 0 & 0 & 7 & 0 & 7 & 1 \\ 4 & 0 & 7 & 0 & 7 & 0 & 1 \\ 5 & 7 & 0 & 7 & 0 & 0 & 1 \\ 6 & 0 & 7 & 0 & 0 & 0 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 & 1 \end{bmatrix}$ 6. $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 5 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$ 9. $\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$ 10. $\begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$

Часть 3. Вычислить значения функции для всех элементов матрицы и записать результат в матрицу того же размера, что и исходная.

Варианты

$$1. f(x) = x^3 - 2x^2 + \sin x - 4; A = \begin{bmatrix} 9.33 & -4.01 & 8.19 & 2.64 \\ 0.55 & 3.81 & 3.32 & 5.07 \end{bmatrix}.$$

$$2. f(x) = \frac{e^x - x}{e^x + x}; A = \begin{bmatrix} 9.32 & 0.21 & -9.89 & 3.11 \\ 0.54 & 4.99 & 5.01 & -0.03 \end{bmatrix}.$$

$$3. f(x) = \sqrt{1 + \sqrt{|x|^3 + 1}}; A = \begin{bmatrix} -1.54 & 0.49 & 3.11 & 2.99 \\ 4.05 & -5.85 & 3.72 & 0.11 \end{bmatrix}.$$

$$4. f(x) = e^x \sin x - e^{-x} \cos x; A = \begin{bmatrix} -9.04 & 3.36 & 3.09 & -2.49 \\ -4.33 & -5.09 & 9.74 & 1.65 \end{bmatrix}.$$

$$5. f(x) = \ln(|x|) \sin \pi x; A = \begin{bmatrix} 0.33 & 0.95 & 7.12 & -9.22 \\ -0.64 & 3.76 & 1.34 & -0.03 \end{bmatrix}.$$

$$6. f(x) = e^{x^2 + x + 1}; A = \begin{bmatrix} -4.53 & -2.12 & -6.54 & -3.21 \\ 3.43 & 7.43 & -0.25 & 1.64 \end{bmatrix}.$$

$$7. f(x) = \frac{\sqrt[3]{x^2 - 1}}{|x| + 3}; A = \begin{bmatrix} 0.23 & 3.89 & -4.23 & -7.25 \\ 5.84 & 5.13 & -0.89 & 3.55 \end{bmatrix}.$$

$$8. f(x) = \frac{1}{1 + \frac{1+x}{1-x^2}}; A = \begin{bmatrix} -5.84 & 9.84 & 0.23 & 1.59 \\ -9.25 & -0.25 & 1.54 & 0.43 \end{bmatrix}.$$

$$9. f(x) = \frac{x^3 + \sin x}{x^3 - \cos x} \sqrt{e^x + 1}; A = \begin{bmatrix} 0.64 & 6.34 & 0.32 & - \\ 1.19 & 3.23 & 1.54 & 0 \end{bmatrix}.$$

$$10. f(x) = \arcsin(\cos x^2); A = \begin{bmatrix} \pi & 2.2\pi & -2\pi & 0.3\pi \\ 3\pi & -\pi & 0.1\pi & 5\pi \end{bmatrix}.$$

Часть 4. Сконструировать блочные матрицы (используя функции для заполнения стандартных матриц) и применить функции обработки данных и поэлементные операции для нахождения заданных величин.

Варианты

$$1. A = \begin{bmatrix} 1 & 0 & 0 & 4 & 0 & 0 \\ 0 & 1 & 0 & 0 & 4 & 0 \\ 0 & 0 & 1 & 0 & 0 & 4 \\ 2 & 2 & 2 & 3 & 3 & 3 \\ 2 & 2 & 2 & 3 & 3 & 3 \\ 2 & 2 & 2 & 3 & 3 & 3 \end{bmatrix} \quad m = \max_{j=1, \dots, 6} \left\{ \sum_{i=1}^6 a_{ij}^2 \right\}.$$

$$6. A = \begin{bmatrix} -1 & 2 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 0 & 4 & 4 \end{bmatrix} \quad s = \sum_{i=1}^6 a_{ii} + \sum_{i=1}^5 a_{i,i+1}$$

$$2. A = \begin{bmatrix} 2 & 2 & -1 & -1 & 2 & 2 \\ 2 & 2 & -1 & -1 & 2 & 2 \\ 2 & 2 & -1 & -1 & 2 & 2 \\ 2 & 2 & -1 & -1 & 2 & 2 \\ 2 & 2 & -1 & -1 & 2 & 2 \\ 2 & 2 & -1 & -1 & 2 & 2 \end{bmatrix} \quad s = \sum_{i=1}^6 \sum_{j=1}^6 |a_{ij}|.$$

$$7. A = \begin{bmatrix} 0 & 0 & 0 & 0 & 3 & 4 \\ 0 & 0 & 0 & 3 & 0 & 4 \\ 0 & 0 & 3 & 0 & 0 & 4 \\ 0 & 3 & 0 & 0 & 0 & 4 \\ 3 & 0 & 0 & 0 & 0 & 4 \\ 1 & 1 & 1 & 1 & 1 & -2 \end{bmatrix} \quad s = \sum_{i=1}^6 \sum_{j=1}^6 \sin\left(\frac{\pi}{6} a_{ij}^2\right)$$

$$3. A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ -1 & -2 & -3 & -4 & -5 & -6 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 4 & 4 & 4 & 4 & 4 & 4 \\ 5 & 5 & 5 & 5 & 5 & 5 \\ 5 & 5 & 5 & 5 & 5 & 5 \end{bmatrix} \quad m = \min_{i,j=1, \dots, 6} a_{ij}^3.$$

$$8. A = \begin{bmatrix} 1 & 2 & 3 & -1 & 0 & 0 \\ 1 & 2 & 3 & 0 & -1 & 0 \\ 1 & 2 & 3 & 0 & 0 & -1 \\ 0 & 0 & 7 & 2 & 2 & 2 \\ 0 & 7 & 0 & 2 & 2 & 2 \\ 7 & 0 & 0 & 2 & 2 & 2 \end{bmatrix} \quad m = \max_{i=1, \dots, 6} \min_{j=1, \dots, 6} a_{ij}$$

$$4. A = \begin{bmatrix} -1 & -1 & -1 & 1 & 1 & 1 \\ -1 & -1 & -1 & 1 & 1 & 1 \\ -1 & -1 & -1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 3 & 0 & 0 \\ 2 & 2 & 2 & 0 & 3 & 0 \\ 2 & 2 & 2 & 0 & 0 & 3 \end{bmatrix} \quad s = \sum_{k=1}^6 a_{kk}^3.$$

$$9. A = \begin{bmatrix} 1 & 0 & 0 & 1 & -3 & -3 \\ 0 & 1 & 1 & 0 & -3 & -3 \\ 0 & 1 & 1 & 0 & -3 & -3 \\ 1 & 0 & 0 & 1 & -3 & -3 \\ -2 & -2 & -2 & -2 & 4 & 4 \\ -2 & -2 & -2 & -2 & 4 & 4 \end{bmatrix} \quad s = \sum_{i=1}^6 \max_{j=1, \dots, 6} (a_{ij} - 3)$$

$$5. A = \begin{bmatrix} 1 & 1 & -3 & -3 & -3 & -3 \\ 1 & 1 & -3 & -3 & -3 & -3 \\ -3 & -3 & 2 & 0 & 0 & 0 \\ -3 & -3 & 0 & 2 & 0 & 0 \\ -3 & -3 & 0 & 0 & 2 & 0 \\ -3 & -3 & 0 & 0 & 0 & 2 \end{bmatrix} \quad s = \sum_{i=1}^5 a_{ii+1}$$

$$10. A = \begin{bmatrix} 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 1 & 1 & 2 & 2 & 2 \\ 1 & 1 & 1 & 2 & 2 & 2 \\ -1 & -1 & -1 & 4 & 0 & 0 \\ -1 & -1 & -1 & 0 & 4 & 0 \\ -1 & -1 & -1 & 0 & 0 & 4 \end{bmatrix} \quad p = \prod_{i=1}^6 \sum_{j=1}^6 (a_{ij})^{a_{ij}}$$

Лабораторная работа 3.
Графика и визуализация данных.
Теоретический материал.

Scilab предоставляет широкие возможности для создания и настройки различных типов графиков и диаграмм, среди которых двумерные, контурные и трехмерные графики, гистограммы, столбиковые и круговые диаграммы и др. Наиболее часто используемые функции для отображения графиков представлены на рисунке 1.

<code>plot</code>	двухмерный график
<code>surf</code>	трехмерный график
<code>contour</code>	контурный график
<code>pie</code>	круговая диаграмма
<code>histplot</code>	гистограмма
<code>bar</code>	столбиковая диаграмма
<code>barh</code>	горизонтальная столбиковая диаграмма
<code>hist3d</code>	трехмерная гистограмма
<code>polarplot</code>	график в полярных координатах
<code>Matplot</code>	цветной двухмерный график матрицы
<code>Sgrayplot</code>	сглаженный контурный график с использованием цвета
<code>grayplot</code>	несглаженный контурный график с использованием цвета

Рисунок 1. Функции для отображения графиков.

Все графики выводятся в графические окна со своими меню и панелями инструментов. Важно понимать, что для построения графиков функций на некоторой области изменения аргументов следует вычислить значения функции в точках области, часто для получения хороших графиков следует использовать достаточно много точек.

Разберем сначала, как получить график функции одной переменной, к примеру:

$$f(x) = e^x \sin \pi x + x^2$$

на отрезке $[-2;2]$. Первый шаг состоит в задании координат точек по оси абсцисс. Заполнение вектора x элементами с постоянным шагом при помощи двоеточия позволяет просто решить эту задачу. Далее необходимо поэлементно вычислить значения $f(x)$ для каждого элемента вектора x и записать результат в вектор f . Для построения графика функции осталось использовать какую-либо из графических функций Scilab. Достаточно универсальной графической функцией является `plot`. В самом простом случае она вызывается с двумя входными аргументами — парой x и f (т. е. `plot` выводит зависимость элементов одного вектора от элементов другого). Последовательность команд, записанная ниже, приводит к появлению графического окна «Графическое окно 0» с графиком функции (рисунок 2).

```
-->x=[-2:0.05:2];
-->f=exp(x).*sin(%pi*x)+x.^2;
-->plot(x,f)
```

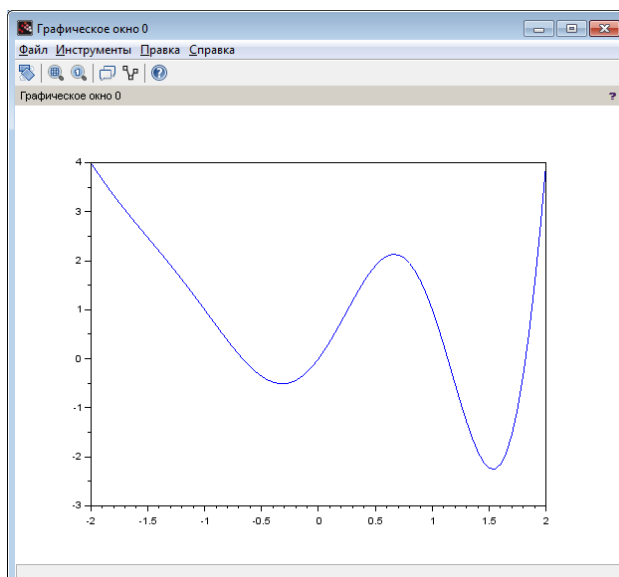


Рисунок 2. График функции

Тип линии, цвет и маркеры определяются значением третьего дополнительного аргумента функции `plot`. Этот аргумент указывается в апострофах, например, вызов `plot(x,f,'ro:')` приводит к построению графика красной пунктирной линией, размеченной круглыми маркерами. Обратите внимание, что абсциссы маркеров определяются значениями элементов вектора `x`. Всего в дополнительном аргументе может быть заполнено три позиции, соответствующие цвету (параметр 1), типу маркеров (параметр 2) и стилю линии (параметр 3). Обозначения для них приведены на таблицах 1, 2, 3. Порядок позиций может быть произвольный, допустимо указывать только один или два параметра, например, цвет и тип маркеров. Посмотрите на результат выполнения следующих команд: `plot(x,f,'g')`, `plot(x,f,'ko')`, `plot(x,f,':')`.

Таблица 1

Значения параметра функции `plot`, определяющего цвет графика

Символ	Описание
y	желтый
m	розовый
c	голубой
r	красный
g	зеленый
b	синий
w	белый
k	черный

Таблица 2.

Значение параметра, определяющего тип маркеров (точек) графика

Символ	Описание
.	точка
o	кружок
x	крестик
+	знак "плюс"
*	звездочка
s	квадрат
d	ромб
v	треугольник вершиной вниз
^	треугольник вершиной вверх
<	треугольник вершиной влево
>	треугольник вершиной вправо
p	пятиконечная звезда
h	шестиконечная звезда

Таблица 3.

Значения параметра, определяющего тип линии графика

Символ	Описание
-	сплошная
:	пунктирная
-.	штрихпунктирная
--	штриховая

Функция plot имеет достаточно универсальный интерфейс, она, в частности, позволяет отображать графики нескольких функций на одних осях. Пусть требуется вывести график не только $f(x)$, но и $g(x) = e^{-x^2} \sin 5\pi x$ на отрезке $[-2, 2]$. Сначала необходимо вычислить значения $g(x)$:

```
-->g=exp(-x.^2).*sin(5*%pi*x);
```

а затем вызвать plot, указав через запятую пары x, f и x, g и, при желании, свойства каждой из линий:

```
-->plot(x,f,'ko-', x,g,'k:')
```

Допускается построение произвольного числа графиков функций, свойства всех линий могут быть различными. Кроме того, области построения каждой из функций не обязательно должны совпадать, но тогда следует использовать разные вектора для значений аргументов и вычислять значения функций от соответствующих векторов. Для получения графика кусочно-заданной функции:

$$y(x) = \begin{cases} \sin x & -4\pi \leq x \leq -\pi \\ 3(x/\pi + 1)^2 & -\pi < x \leq 0 \\ 3e^{-x} & 0 < x \leq 5 \end{cases}$$

достаточно выполнить последовательность команд:

```
-->x1=[-4*%pi:%pi/10:-%pi];
```

```

-->y1=sin(x1);
-->x2=[- %pi: %pi/30:0];
-->y2=3*(x2/%pi+1).^2;
-->x3=[0:0.02:5];
-->y3=3*exp(-x3);
-->plot(x1,y1,x2,y2,x3,y3)

```

Заметьте, что графики ветвей функции отображаются различными цветами. Можно было поступить и по-другому, а именно: после заполнения x_1 , y_1 , x_2 , y_2 , x_3 и y_3 собрать вектор x для значений аргумента и вектор y для значений $y(x)$ и построить зависимость y от x :

```

-->x=[x1 x2 x3];
-->y=[y1 y2 y3];
-->plot(x,y)

```

Несложно догадаться, как построить график параметрически заданной функции, используя то обстоятельство, что `plot` отображает зависимость одного вектора от другого.

Пусть требуется получить график астроида: $x(t) = \cos^3 t$, $y(t) = \sin^3 t$, $t \in [0, 2\pi]$.

Следует задать вектор t , затем в векторы x , y занести значения $x(t)$, $y(t)$ и воспользоваться `plot` для отображения зависимости y от x :

```

-->t=[0:%pi/20:2*%pi];
-->x=cos(t).^3;
-->y=sin(t).^3;
-->plot(x,y)

```

Функция `comet` позволяет проследить за движением точки по траектории параметрически заданной линии. Вызов `comet(x,y)` приводит к появлению графического окна, на осях которого рисуется перемещение точки в виде движения кометы с хвостом. Управление скоростью движения осуществляется изменением шага при определении вектора значений параметра.

Графики оформляются в Scilab специальными командами и функциями. Сетка наносится на оси командой `xgrid`. Заголовок размещается в графическом окне посредством функции `xtitle`:

`xtitle(name, xname, yname)`

здесь `name` — название графика, `xname` — название оси X, `yname` — название оси Y (входные аргументы заключаются в апострофы). Помимо этого существуют функции `title`, `xlabel`, `ylabel` для задания подписей названия графика, оси X и оси Y соответственно.

При наличии нескольких графиков требуется расположить легенду обратившись к `legend`. Обращение к этой функции выглядит следующим образом:

`legend(line1,line2,...,linen,place,frame)`

- `line1` — описание (название) первого графика, `line2` — второго ..., `linen` — имя n -го графика;
- `place` определяет месторасположение описания: если `place=1`, то описание будет расположено в верхнем правом углу графической области (если значение `place` не указано, то по умолчанию `place=1`), 2 — в верхнем левом углу, 3 — в нижнем левом углу, 4 — в нижнем правом углу, 5 — положение определяется пользователем после изображения графика;
- `frame` — может принимать два значения `%t` и `%f` (по умолчанию `frame=%t`), если `frame=%t`, то описание будет заключено в рамку, если же `frame=%f` — рамка отсутствует.

Обратимся теперь к визуализации векторных и матричных данных. Самый простой способ отображения векторных данных состоит в использовании функции `plot` с вектором в качестве входного аргумента. При этом получающийся в виде ломаной линии график символизирует зависимость значений элементов вектора от их индексов. Второй дополнительный аргумент может определять цвет, стиль линии и тип маркеров, например: `plot(x,'ko')`. Вызов функции `plot` от матрицы приводит к нескольким графикам, их число

совпадает с числом столбцов матрицы, а каждый из них является зависимостью элементов столбца от их строчных индексов. Цвет и стиль линий и тип маркеров сразу для всех линий так же определяется вторым дополнительным аргументом.

Наглядным способом представления матричных и векторных данных являются разнообразные диаграммы. Простейшая столбцевая диаграмма строится при помощи функции `bar`:

```
--> x=[0.7 2.1 2.5 1.9 0.8 1.3]
--> bar(x)
```

Дополнительный числовой аргумент `bar` указывает на ширину столбцов (по умолчанию он равен 0.8), а значения большие единицы, например `bar(x,1.2)`, приводят к частичному перекрытию столбцов. Указание матрицы во входном аргументе `bar` приводит к построению групповой диаграммы, число групп совпадает с числом строк матрицы, а внутри каждой группы столбиками отображаются значения элементов строк.

Круговые диаграммы векторных данных получаются с помощью функции `pie`.

```
--> pie([1 2 3])
```

Можно отделить некоторые секторы от всего круга диаграммы, для чего следует вызвать `pie` со вторым аргументом — вектором той же длины, что исходный. Ненулевые элементы второго вектора соответствуют отделяемым секторам. Следующий пример показывает, как отделить от диаграммы сектор, соответствующий наибольшему элементу вектора `x`:

```
--> x=[0.3 2 1.4 0.5 0.9];
--> [m,k]=max(x);
--> [l, n]=size(x);
--> v = zeros(l,n);
--> v(k)=1;
--> pie(x,v)
```

Подписи к секторам диаграммы указываются во втором дополнительном входном аргументе, который заключается в фигурные скобки:

```
--> pie([2400 3450 1800 5100],{'Март','Апрель','Май','Июнь'})
```

Визуализация функций двух переменных в Scilab может быть осуществлена несколькими способами, но все они предполагают однотипные предварительные действия. Рассмотрим здесь только построение графиков функций двух переменных на прямоугольной области определения. Предположим, что требуется получить поверхность

функции $z(x, y) = e^{-x} \sin(\pi y)$ на прямоугольнике $x \in [-1,1]$, $y \in [0,2]$. Первый шаг состоит в задании сетки на прямоугольнике, т. е. точек, которые будут использоваться для вычисления значений функции. Для генерации сетки предусмотрена функция `meshgrid`, вызываемая от двух входных аргументов — векторов, задающих разбиения по осям `x` и `y`. Функция `meshgrid` возвращает два выходных аргумента, являющиеся матрицами.

```
--> [X,Y]=meshgrid(-1:0.1:1,0:0.1:2);
```

Матрица `X` состоит из одинаковых строк, равных первому входному аргументу — вектору в `meshgrid`, а матрица `Y` — из одинаковых столбцов, совпадающих со вторым вектором в `meshgrid`. Такие матрицы оказываются необходимыми на втором шаге при заполнении матрицы `Z`, каждый элемент которой является значением функции $z(x,y)$ в точках сетки. Несложно понять, что использование поэлементных операций при вычислении функции $z(x,y)$ приводит к требуемой матрице:

```
--> Z=exp(-X).*sin(%pi*Y);
```

Для построения графика $z(x,y)$ осталось вызвать подходящую графическую функцию, к примеру:

```
--> mesh(X,Y,Z)
```

На экране появляется графическое окно, содержащее каркасную поверхность исследуемой функции (рисунок 3).

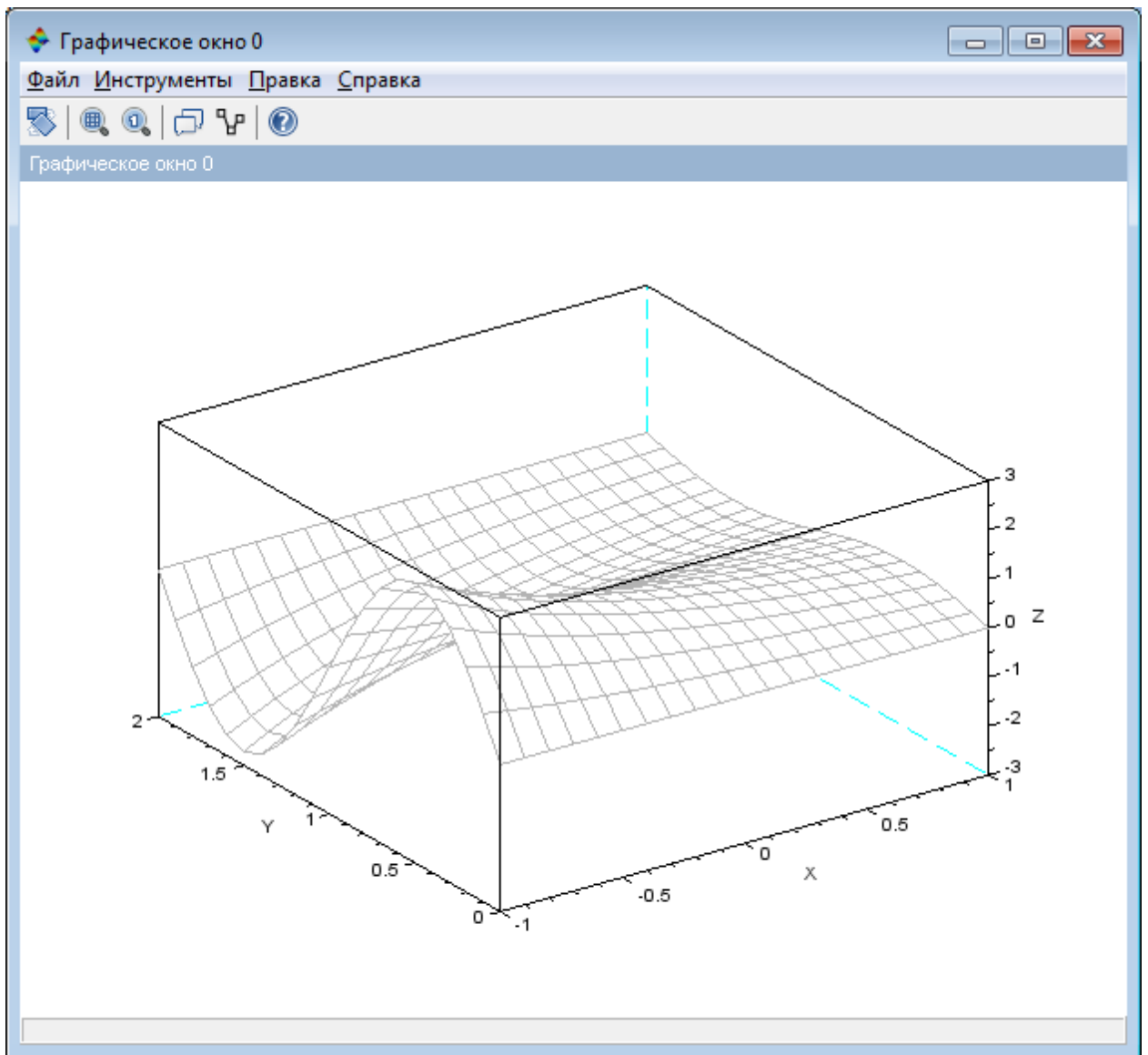


Рисунок 3

Попробуем отобразить этот же график функцией surf:

```
-->surf(X,Y,Z);
```

Появится цветное отображение. Команда colorbar приводит к отображению в графическом окне столбика, показывающего соотношение между цветом и значением $z(x,y)$. В аргументы данной функции необходимо ввести минимальное и максимальное значения:

```
-->zm = min(Z); zM = max(Z);
```

```
-->surf(X,Y,Z); colorbar(zm, zM)
```

Разберем теперь работу с несколькими графиками. Первый вызов любой графической функции приводит к появлению на экране графического окна «Графическое окно 0», содержащего оси с графиком. Однако, при дальнейших обращениях к графическим функциям прежний график пропадает, а новый выводится в тоже самое окно. Команда figure предназначена для создания пустого графического окна. Если требуется получить несколько графиков в разных окнах, то перед вызовом графических функций следует прибегать к figure. Графические окна при этом нумеруются так: «Графическое окно 1», «Графическое окно 2» и т. д.

В одном графическом окне можно расположить несколько осей со своими графиками. Функция subplot предназначена для разбиения окна на части и определения текущей из них. Предположим, что требуется вывести графики на шесть пар осей в одно

графическое окно (две по вертикали и три по горизонтали). Создайте графическое окно при помощи `figure` и выполните команду:

```
--> subplot(2,3,1)
```

В левом верхнем углу окна появились оси. Первые два аргумента в `subplot` указывают на общее число пар осей по вертикали и горизонтали, а последний аргумент означает номер данной пары осей. Нумерация идет слева направо, сверху вниз.

Используйте `subplot(2,3,2)`, ... , `subplot(2,3,6)` для создания остальных пар осей. Вывод любой из графических функций можно направить в нужные оси, указав их при помощи `subplot(2,3,k)`, например:

```
--> subplot(2,3,3)
```

```
--> bar([1.2 0.3 2.8 0.9])
```

```
--> subplot(2,3,6)
```

```
--> surf(X,Y,Z)
```

Задания для самостоятельной работы

Часть 1. Построить графики функций одной переменной на указанных интервалах. Вывести графики различными способами:

1. в отдельные графические окна
2. в одно окно на одни оси
3. в одно окно на отдельные оси.

Дать заголовки, разместить подписи к осям, легенду, использовать различные цвета, стили линий и типы маркеров, нанести сетку.

Варианты.

1. $f(x) = \sin x$; $g(x) = \sin^2 x$; $x \in [-2\pi, 3\pi]$.

$u(x) = 0.01x^2$; $v(x) = e^{-|x|}$; $x \in [-0.2, 9.4]$.

2. $f(x) = \sin x^2$; $g(x) = \cos x^2$; $x \in [-\pi, \pi]$.

$u(x) = x/20$; $v(x) = e^x$; $x \in [-2, 2]$.

3. $f(x) = x^3 + 2x^2 + 1$; $g(x) = (x - 1)^4$; $x \in [-1, 1]$.

$u(x) = \sqrt{x}$; $v(x) = e^{-x^2}$; $x \in [0, 1]$.

4. $f(x) = \ln x$; $g(x) = x \ln x$; $x \in [0.2, 10]$.

$u(x) = x^{1/3}$; $v(x) = \sqrt{x}$; $x \in [0, 8]$.

5. $f(x) = |2x|^3$; $g(x) = |2x|^5$; $x \in [-0.5, 0.5]$.

$u(x) = \sqrt{|x|}$; $v(x) = x^{1/5}$; $x \in [-0.6, 0.5]$.

$$6. f(x) = x^2; g(x) = x^3; x \in [-1, 1].$$

$$u(x) = x^4; v(x) = x^5; x \in [-1, 1].$$

$$7. f(x) = \arcsin x; g(x) = \arccos x; x \in [-1, 1].$$

$$u(x) = \operatorname{arctg} x; v(x) = \operatorname{arctg} 3x; x \in [-1, 1].$$

$$8. f(x) = \operatorname{sh} x; g(x) = \operatorname{ch} x; x \in [-1, 1].$$

$$u(x) = e^x; v(x) = e^{-x}; x \in [-0.6, 0.6].$$

$$9. f(x) = \frac{\sin x}{x}; g(x) = e^{-x} \cos x; x \in [0.01, 2\pi].$$

$$u(x) = \sin(\ln(x+1)); v(x) = \cos(\ln(x+1)); x \in [0, 2\pi].$$

$$10. f(x) = x^x; g(x) = x^{x^x}; x \in [0.1, 1].$$

$$u(x) = \frac{1}{1+x}; v(x) = \frac{1}{1+\frac{1}{1+x}}; x \in [0, 1].$$

Часть 2. Построить график кусочно-заданной функции, отобразить ветви разными цветами и маркерами.

Варианты.

$$1. \quad f(x) = \begin{cases} -1, & -3 \leq x \leq -1 \\ x, & -1 < x \leq 1 \\ e^{1-x}, & 1 < x \leq 3 \end{cases} \quad 2. \quad f(x) = \begin{cases} \sqrt{x}, & 0 \leq x \leq 1 \\ 1, & 1 < x \leq 3 \\ (x-4)^2, & 3 < x \leq 5 \end{cases}$$

$$3. \quad f(x) = \begin{cases} \ln x, & 1 \leq x \leq e \\ x/e, & e < x \leq 9 \\ 9e^{8-x}, & 9 < x \leq 12 \end{cases} \quad 4. \quad f(x) = \begin{cases} \sin x, & -2\pi \leq x \leq 0 \\ -x^3, & 0 < x \leq 1 \\ \cos \pi x, & 1 < x \leq 3\pi \end{cases}$$

$$5. \quad f(x) = \begin{cases} \arcsin x - 1, & 0 \leq x \leq 1 \\ \frac{\pi}{2} - x, & 1 < x \leq \frac{\pi}{2} \\ \cos x, & \frac{\pi}{2} < x \leq \pi \end{cases} \quad 6. \quad f(x) = \begin{cases} |x|, & -2 \leq x \leq 1 \\ \sin \frac{\pi}{2} x, & 1 < x \leq 2 \\ (2-x)^3, & 2 < x \leq 3 \end{cases}$$

$$7. \quad f(x) = \begin{cases} (x-1)^2, & -2 \leq x \leq 1 \\ \cos \frac{\pi}{2} x, & 1 < x \leq 3 \\ 1 - e^{3-x}, & 3 < x \leq 8 \end{cases} \quad 8. \quad f(x) = \begin{cases} e^x, & -2 \leq x \leq -1 \\ \frac{|x|}{e}, & -1 < x \leq 1 \\ e^{-x}, & 1 < x \leq 2 \end{cases}$$

$$9. \quad f(x) = \begin{cases} e^{x+1}, & -2 \leq x \leq -1 \\ x^2, & -1 < x \leq 1 \\ (2-x)^3, & 1 < x \leq 2 \end{cases} \quad 10. \quad f(x) = \begin{cases} x^2 \log_2 x, & 1 \leq x \leq 2 \\ x^3/2, & 2 < x \leq 3 \\ x^x/2, & 3 < x \leq 3.5 \end{cases}$$

Часть 3. Построить график параметрически заданной функции, используя plot и comet.

Варианты

$$1. \quad x(t) = t - \sin t; \quad y(t) = 1 - \cos t \quad 2. \quad x(t) = 2 \sin t - \frac{2}{3} \sin 2t; \quad y(t) = 2 \cos t - \frac{2}{3} \cos 2t$$

$$3. \quad x(t) = 9 \sin \frac{t}{10} - \frac{1}{2} \sin \frac{9}{10} t; \quad y(t) = 9 \cos \frac{1}{10} t + \frac{1}{2} \cos \frac{9}{10} t \quad 4. \quad x(t) = \cos t; \quad y(t) = \sin(\sin t)$$

$$5. \quad x(t) = e^{-t} \cos t; \quad y(t) = \sin t \quad 6. \quad x(t) = e^{-t} \cos t; \quad y(t) = e^t \sin t$$

$$7. \quad x(t) = t(t - 2\pi); \quad y(t) = \sin t \quad 8. \quad x(t) = \sin t(t - 2\pi); \quad y(t) = \sin t$$

$$9. \quad x(t) = \sin t(t - 2\pi); \quad y(t) = \sin t \cdot \cos t \quad 10. \quad x(t) = \sin t + \cos^3 t; \quad y(t) = \sin t \cdot \cos t$$

Часть 4. Визуализировать функцию двух переменных на прямоугольной области определения различными способами:

- каркасной поверхностью;
- залитой цветом каркасной поверхностью;

Расположить графики в отдельных графических окнах и в одно окно на отдельные оси.

1. $z(x,y) = \sin x \cdot e^{-3y} \quad x \in [0, 2\pi] \quad y \in [0, 1]$

2. $z(x,y) = \sin^2 x \cdot \ln y \quad x \in [0, 2\pi] \quad y \in [1, 10]$

3. $z(x,y) = \sin^2(x - 2y) \cdot e^{-|y|} \quad x \in [0, \pi] \quad y \in [-1, 1]$

4. $z(x,y) = \frac{x^2 y^2 + 2xy - 3}{x^2 + y^2 + 1} \quad x \in [-2, 2] \quad y \in [-1, 1]$

5. $z(x,y) = \frac{\sin xy}{x} \quad x \in [0.1, 5] \quad y \in [-\pi, \pi]$

6. $z(x,y) = (\sin x^2 + \cos y^2)^{xy} \quad x \in [-1, 1] \quad y \in [-1, 1]$

7. $z(x,y) = \arctan(x + y)(\arccos x + \arcsin y) \quad x \in [-1, 1] \quad y \in [-1, 1]$

8. $z(x,y) = (1 + xy)(3 - x)(4 - y) \quad x \in [0, 3] \quad y \in [0, 4]$

9. $z(x,y) = e^{-|x|}(x^5 + y^4)\sin(xy) \quad x \in [-2, 2] \quad y \in [-3, 3]$

10. $z(x,y) = (y^2 - 3)\sin \frac{x}{|y| + 1} \quad x \in [-2\pi, 2\pi] \quad y \in [-3, 3]$

Лабораторная работа 4.

Работа с встроенным редактором скриптов. Файл-функции и файл программы.

Теоретический материал.

В Scilab имеется редактор скриптов, для запуска которого следует выбрать пункт меню Инструменты – Текстовый редактор. Так же его можно запустить из консоли с помощью команды `editor()`. На экране появляется окно редактора. Наберите в нем какие-либо команды, например для построения графика (см. листинг 5.1):

Листинг 5.1. Простейшая файл-программа

```
x=[-1:0.01:1];
y=exp(x);
plot(x,y)
xgrid
xtitle('Экспоненциальная функция')
```

Наиболее часто используемые команды редактора располагаются в меню

Выполнение (Execute):

- *Загрузить в Scilab* (Load into Scilab) - позволяет выполнить все команды некоторого скрипта так, как будто эти команды последовательно вводятся в консоли. При этом результат выполнения инструкций, оканчивающихся символом ”;”, не отображается.
- *Вычислить выделенное* (Evaluate Selection) - позволяет выполнить выделенные инструкции.
- *Выполнить файл в Scilab* (Execute File Into Scilab) - загружает на исполнение файл подобно тому, как это делается с использованием функции `exec`. При этом в консоль будут выводиться лишь результаты работы печатающих функций, например, `disp`.

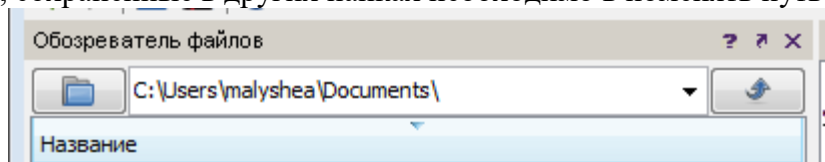
Для быстрого запуска программного кода используется клавиша F5.

Меню Правка (Edit) предлагает полезную возможность автоматического форматирования отступов *Исправить отступы* (Correct Indentation). Эта возможность позволяет структурировать текст программы, что существенно упрощает чтение блоков в таких конструкциях как `if`, `for` и т.д.

Выделив несколько строк и нажав правую кнопку мыши (или комбинацию `Control+Click` в Mac OS), можно отобразить контекстное меню. . Контекстное меню содержит ряд полезных команд:

- *Вычислить выделенное* (Execute selection in Scilab) - выполнить выделенные команды;
- *Править выделенное в новой вкладке* (Edit selection in a new tab) – открыть новую вкладку и скопировать туда выделенный фрагмент;
- *Справка по '...'* (Help about '...') - отобразить страницу помощи, связанную с выделенной командой.

Набранная программа сохраняется в файле с расширением `.sce`. Данный файл можно запустить из окна *Обозреватель файлов*. Для того, чтобы в окне отображались файлы, сохраненные в других папках необходимо в помянуть путь к файлу.



Файл-программа может использовать переменные рабочей среды. Например, если была введена команда:

```
--> a=[0.1 0.4 0.3 1.9 3.3];
```

то файл-программа, содержащая строку `bar(a)`, построит столбцевую диаграмму вектора `a` (разумеется, если он не был переопределен в самой файл-программе).

Файл-функции отличаются от файл-программ тем, что они могут иметь входные и выходные аргументы, а все переменные, определенные внутри файл-функции, являются локальными и не видны в рабочей среде.

Простейший синтаксис вызова функции выглядит следующим образом:

outvar = myfunction (invar)

Значение каждого из трех элементов вызова функции приведено в списке:

- *myfunction* представляет собой наименование вызываемой функции,
- *invar* обозначает входные аргументы,
- *outvar* соответствует выходным аргументам.

Значения переменных, указанных при вызове в качестве фактических параметров, функция изменить не может.

Листинг 5.2 содержит пример простейшей файл-функции с двумя входными и одним выходным аргументами.

Листинг 5.2. Файл-функция mysum
function c=mysum(a,b)

c=a+b;

Наберите этот пример в новом файле в редакторе и сохраните его. Обратите внимание, что Scilab предлагает в качестве имени файла название файл-функции, т.е. *mysum*. Всегда сохраняйте файл-функцию в файле, имя которого совпадает с именем файл-функции! Файл функции сохраняются в файлах с расширением *.sci*. **Обязательно откомпилируйте** данный файл, без компиляции он не будет запускаться!!!

Для компиляции можно так же воспользоваться командой:

-->exec ('C:\Users\malyshea\Documents\mysum.sci'

Убедитесь, что каталог с файлом *mysum.sci* является текущим и вызовите файл-функцию *mysum* из командной строки:

--> s=mysum(2,3)

s =

5

При вызове файл-функции *mysum* произошли следующие события:

- входной аргумент *a* получил значение 2;
- входной аргумент *b* стал равен 3;
- сумма *a* и *b* записалась в выходной аргумент *c*;
- значение выходного аргумента *c* получила переменная *s* рабочей среды и результат вывелся в командное окно.

Заметьте, что оператор *c=a+b* в файл-функции *mysum* завершен точкой с запятой для подавления вывода локальной переменной *c* в командное окно. Для просмотра значений локальных переменных при отладке файл-функций, очевидно, не следует подавлять вывод на экран значений требуемых переменных.

Практически все функции Scilab являются файл-функциями и хранятся в одноименных файлах. Функция *sin* допускает два варианта вызова: *sin(x)* и *y=sin(x)*, в первом случае результат записывается в *ans*, а во втором — в переменную *y*. Наша функция *mysum* ведет себя точно так же. Более того, входными аргументами *mysum* могут быть массивы одинаковых размеров или массив и число.

Ключевые слова и инструкции Scilab, использующиеся при работе с функциями:

<code>function</code>	открывает определение функции
<code>endfunction</code>	завершает определение функции
<code>argn</code>	количество входных и выходных аргументов в данном вызове функции
<code>varargin</code>	вектор, представляющий переменное число входных аргументов функции
<code>varargout</code>	вектор, представляющий переменное число выходных аргументов функции
<code>fun2string</code>	генерирует текстовое определение (исходный код) функции
<code>get_function_path</code>	возвращает путь к файлу исходного кода функции
<code>getd</code>	отображает полный список функций, определения которых хранятся в заданном каталоге файловой системы
<code>head_comments</code>	отображает комментарии к функции
<code>listfunctions</code>	отображает свойства функций, которые вызывались ранее в данной сессии
<code>macrovar</code>	возвращает списки входных и выходных параметров функции, а также используемых в теле функции внешних переменных, вызовов других функций и локальных переменных

Разберем теперь, как создать файл-функцию с несколькими выходными аргументами. Список выходных аргументов в заголовке файл-функции заключается в квадратные скобки, сами аргументы отделяются запятой. В качестве примера на листинге 5.3 приведена файл-функция `quadeq`, которая по заданным коэффициентам квадратного уравнения находит его корни.

Листинг 5.3. Файл-функция для решения квадратного уравнения

```
function [x1,x2]=quadeq(a,b,c)
D=b^2-4*a*c;
x1=(-b+sqrt(D))/(2*a);
x2=(-b-sqrt(D))/(2*a);
endfunction
```

При вызове `quadeq` из командной строки используйте квадратные скобки для указания переменных, в которые будут занесены значения корней:

```
>> [r1,r2]=quadeq(1,3,2)
r1 =
-1
r2 =
-2
```

Файл-функция может и не иметь входных или выходных аргументов, заголовки таких файл-функций приведены ниже:

```
function noout(a,b), function [v,u]=noin, function noarg()
```

Умение писать собственные файл-функции и файл-программы необходимо как при программировании в Scilab, так и при решении различных задач средствами Scilab (в частности, поиска корней уравнений, интегрирования, оптимизации).

Разберем еще один пример файл-функции, связанный с построением графика. Запрограммируйте файл-функцию `myfun` для вычисления $f(x)$ (см. листинг 5.4).

Листинг 5.4. Файл-функция myfun

```
function y=myfun(x);
y= sin(x);
endfunction
```

В командной строке, для получения графика функции, воспользуемся командой `plot`:

```
-->x=[0:0.01:10];
-->y=myfun(x);
-->plot(x,y)
```

Задания для самостоятельной работы

Часть 1. Написать файл-функции и построить графики на заданном отрезке при помощи plot (с шагом 0.05) для следующих функций:

Варианты

1. $f(x) = \sin \frac{1}{x} \quad x \in [0.05, 1]$. 2. $f(x) = e^{3x \sin 5\pi x} + e^{3x \cos 5\pi x} \quad x \in [0, 1]$.

3. $f(x) = \frac{10}{11 - 10 \sin 21\pi x} \quad x \in [0.05, 1]$. 4. $f(x) = \sqrt{\frac{|\sin 21\pi x|}{2 + \sin 20\pi x}} \quad x \in [0, 1]$.

5. $f(x) = \frac{1}{\operatorname{arctg}\left(\frac{1}{1.1 + \sin 5\pi x}\right) - \frac{3}{2}} \quad x \in [0, 1]$. 6. $f(x) = \cos\left(\frac{1}{\frac{2\pi}{11} - \operatorname{arctg} x^x}\right) \quad x \in [0, 1]$.

7. $f(x) = \sin\left(6\pi\left|x - \frac{2}{3}\right|x^3\right) \quad x \in [0, 1]$. 8. $f(x) = \sin 2\pi \sqrt{\left|\sqrt{1-x^3} - \frac{4}{7}\right|} \quad x \in [0, 1]$.

9. $f(x) = |\sin 20\pi x| \quad x \in [0.05, 1]$.

10. $f(x) = \frac{1}{\sin(e^{2x} - e^{-2x}) + \cos(e^{2x} - e^{-2x}) - \frac{3}{2}} \quad x \in [-1, 1]$.

Часть 2. Написать файл-функцию для решения поставленной задачи.

Варианты

1. Написать файл-функцию, которая по заданному вектору определяет номер его элемента с наибольшим отклонением от среднего арифметического всех элементов вектора.
2. Написать файл-функцию, возвращающую сумму всех элементов вектора с нечетными индексами.
3. Написать файл-функцию, вычисляющую максимальное значение среди диагональных элементов заданной матрицы.
4. Написать файл-функцию, переставляющую первый столбец квадратной матрицы с ее диагональю
5. Написать файл-функцию, которая суммирует все внедиагональные элементы заданной матрицы.
6. Написать файл-функцию, заменяющую максимальный элемент вектора средним значением всех его элементов.
7. Написать файл-функцию, заменяющую элемент матрицы с индексами 1,1 произведением всех элементов матрицы.
8. Написать файл-функцию, которая строит многоугольник (замкнутый) по заданным векторам x и y с координатами вершин
9. Написать файл-функцию, которая отображает элементы заданного вектора синими маркерами, а максимальный элемент — красным и возвращает значение и номер максимального элемента.
10. Написать файл-функцию, переводящую время в секундах в часы, минуты и секунды.

Лабораторная работа 5. Программирование.

Теоретический материал.

В Scilab имеется встроенный язык программирования, который необходимо знать для эффективной работы в среде. Данный язык является структурным с поддержкой объектов.

В структурном программировании сценарий представляет последовательность инструкций, которые выполняются последовательно друг за другом, словно на конвейере.

В структурном программировании существует ряд особых конструкций. К ним относятся:

- **Ветвление** — однократное выполнение одной из двух или более операций в зависимости от заданного условия;
- **Цикл** — многократное выполнение одной или нескольких инструкций до тех пор, пока не выполнится некоторое условие. Один проход по всем инструкциям внутри цикла называется итерацией.

Данные конструкции кодируются с помощью специальных служебных слов, которые не могут быть использованы для имен объектов.

В языке Scilab две конструкции, организующие **ветвление**:

1. Конструкция **if ... else**;
2. Конструкция **select ... case**.

Общий синтаксис конструкции **if ... else** имеет следующий вид

```
if <условие 1> then
  <операторы 1>
elseif <условие 2> then
  <операторы 2>
...
elseif <условие n> then
  <операторы n>
else
  <операторы>
end
```

Общий синтаксис конструкции **select ... case** имеет вид:

```
select <значение>
case <значение 1>
  <операторы 1>
case <значение 2>
  <операторы 2>
...
case <значение n>
  <операторы n>
else
  <операторы>
end
```

В языке Scilab всего два вида **циклов**:

1. Цикл **for**;
2. Цикл **while**.

Общий синтаксис для цикла **for** имеет вид

```
for <счетчик>=<начальное значение>:<шаг>:<конечное значение>
<операторы>
end
```

Для цикла **while** синтаксис имеет следующий вид

```
while <условие>
<операторы>
end
```

Иногда довольно сложно записать точное условие выхода из цикла явно. В этом случае нарочно создают бесконечный цикл, а условие выхода задают уже в теле с помощью специальных управляющих операторов.

Управление циклом в его теле возможно с помощью следующих специальных операторов:

- **continue** — передача управления следующей итерации;
- **break** — прекращение текущей итерации и выход из цикла.

Листинг 6.1 содержит файл-программу для вывода графиков функции

$$f(x, \beta) = e^{\beta x} \sin x \quad \text{на отрезке } [-2;2], \text{ для значений параметра } \beta \in [-0.5, 0.5].$$

Листинг 6.1. Графики функции при различных значениях параметра

```
x=[-2:0.01:2];
for beta=-0.5:0.1:0.5
y=exp(beta*x).*sin(x);
plot(x,y)
set(gca(),"auto_clear","off")
end
set(gca(),"auto_clear","on")
```

Здесь строки `set(gca(),"auto_clear","off")` и `set(gca(),"auto_clear","on")` используются для постепенного добавления графиков в одно и то же графическое окно.

Если шаг равен единице, то его указывать не обязательно. Например, для вычисления суммы

$$\sum_{k=1}^{10} \frac{x^k}{k!}$$

при различных значениях x потребуется файл-функция, текст которой приведен на листинге 6.2. Обратите внимание, что `sum10` может быть вызвана как от числа, так и от массива значений, благодаря применению поэлементных операций.

Листинг 6.2. Файл-функция для вычисления суммы

```
function s=sum10(x)
s=0;
for k=1:10
s=s+x.^k/factorial(k);
end
```

Подумайте над тем, как избежать нахождения факториала в каждом слагаемом (при вычислении k -го слагаемого можно использовать значение $(k-1)!$, найденное на предыдущем шаге цикла).

Цикл `for` подходит для повторения заданного числа определенных действий. В том случае, когда число повторов заранее неизвестно и определяется в ходе выполнения блока операторов, следует организовать цикл `while`. Цикл `while` работает, пока выполнено условие цикла. Файл-функция `negsum` (см. листинг 6.3) находит сумму всех первых отрицательных элементов вектора.

Листинг 6.3. Файл-функция negsum

```
function s=negsum(x)
s=0;
k=1;
while x(k)<0
    s=s+x(k);
    k=k+1;
end
```

В качестве операторов отношения используются символы: >, <, >=, <=, == (равно), ~= (не равно). Файл-функция negsum имеет один недостаток: если все элементы массива — отрицательные числа, то k становится больше длины массива x, что приводит к ошибке, например:

```
" b=[-2 -7 -1 -9 -2 -5 -4];
" s=negsum(b)
??? Index exceeds matrix dimensions.
```

Кроме проверки значения x(k) следует позаботиться о том, чтобы значение k не превосходило длины вектора x. Вход в цикл должен осуществляться только при одновременном выполнении условий k<=length(x) и x(k)<0, т. е. необходимо применить логический оператор "и", обозначаемый в Scilab символом &. Замените условие цикла на составное: k<=length(x) & x(k)<0. Если первое из условий не выполняется, то второе условие проверяться не будет, именно поэтому выбран такой порядок операндов. Теперь файл-функция negsum работает верно для любых векторов.

Логический оператор "или" обозначается символом вертикальной черты |, а отрицание — при помощи тильды ~. Ниже приведены логические операции по мере убывания их приоритета:

- отрицание ~ ;
- операторы отношения >, <, >=, <=, ==, ~= ;
- логическое "и" & ;
- логическое "или" | .

Для изменения порядка выполнения логических операторов используются круглые скобки.

Циклы могут быть вложены друг в друга. Например, для поиска суммы элементов матрицы, расположенных выше главной диагонали, следует использовать два цикла for, причем начальное значение счетчика внутреннего цикла зависит от текущего значения счетчика внешнего цикла (см. листинг 6.4).

Листинг 6.4. Использование вложенных циклов

```
function s=upsum(A)
[n m]=size(A);
s=0;
for i=1:n
    for j=i+1:m
        s=s+A(i,j);
    end
end
```

Ветвление в ходе работы программы осуществляется при помощи конструкции if-elseif-else. Самый простой вариант ее использования (без elseif и else) реализован в файл-функции possum (см. листинг 6.5), которая предназначена для нахождения суммы всех положительных элементов вектора.

Листинг 6.5. Файл-функция для суммирования положительных элементов вектора

```
function s=possum(x)
s=0;
for k=1:length(x)
```

```

if x(k)>0
s=s+x(k);
end
end

```

Если ход программы должен изменяться в зависимости от нескольких условий, то следует использовать полную конструкцию if-elseif-else. Каждая из ветвей elseif в этом случае должна содержать условие выполнения блока операторов, размещенных после нее. Важно понимать, что условия проверяются подряд, первое выполненное условие приводит к работе соответствующего блока, выходу из конструкции if-elseif-else и переходу к оператору, следующему за end. У последней ветви else не должно быть никакого условия. Операторы, находящиеся между else и end, работают в том случае, если все условия оказались невыполненными. Предположим, что требуется написать файл-функцию для вычисления кусочно-заданной функции:

$$f(x) = \begin{cases} 1 - e^{-1-x}, & x < -1 \\ x^2 - x - 2, & -1 \leq x \leq 2 \\ 2 - x, & x > 2 \end{cases}$$

Первое условие $x < -1$ проверяется в ветви if. Обратите внимание, что условие $-1 \leq x$ не требуется включать в следующую ветвь elseif (см. листинг 6.6), поскольку в эту ветвь программа заходит, если предыдущее условие ($x < -1$) оказалось не выполнено. Условие $x > 2$ проверять не надо — если не выполнены два предыдущих условия, то x будет больше двух.

Листинг 6.6. Файл-функция для вычисления кусочно-заданной функции

```

function f=pwf(x)
if x<-1
f=1-exp(-1-x);
elseif x<=2
f=x^2-x-2;
else
f=2-x;
end

```

Ход работы программы может определяться значением некоторой переменной (переключателя). Такой альтернативный способ ветвления программы основан на использовании конструкции **select ... case**.

Предположим, что требуется найти количество единиц и минус единиц в заданном массиве и, кроме того, найти сумму всех элементов, отличных от единицы и минус единицы. Следует перебрать все элементы массива в цикле, причем в роли переменной-переключателя будет выступать текущий элемент массива. Листинг 6.7 содержит файл-функцию, которая по заданному массиву возвращает число минус единиц в первом выходном аргументе, число единиц — во втором, а сумму — в третьем.

Листинг 6.7. Файл-функция mpsum

```

function [m, p, s]=mpsum(x)
m=0;
p=0;
s=0;
for i=1:length(x)
select x(i)
case -1
m=m+1;
case 1
p=p+1;

```

```

else
s=s+x(i);
end
end

```

Блок case может быть выполнен не только при одном определенном значении переключателя, но и в том случае, когда переключатель принимает одно из нескольких допустимых значений. В этом случае значения указываются после слова case в фигурных скобках через запятую, например: case {1,2,3}.

Рассмотрим досрочное завершение циклы. Пусть, например, требуется по заданному массиву x образовать новый массив y по правилу $y(k)=x(k+1)/x(k)$ до первого нулевого элемента x(k), т.е. до тех пор, пока имеет смысл операция деления. Номер первого нулевого элемента в массиве x заранее неизвестен, более того, в массиве x может и не быть нулей. Решение задачи состоит в последовательном вычислении элементов массива y и прекращении вычислений при обнаружении нулевого элемента в x. Файл-функция, приведенная на листинге 6.8, демонстрирует работу оператора break.

Листинг 6.8. Использование оператора break для выхода из цикла

```

function y=div(x)
for k=1:length(x)-1
if x(k)==0
break
end
y(k)=x(k+1)/x(k);
end

```

Задания для самостоятельной работы

Написать файл-функцию для решения поставленной задачи.

Варианты

1. Вычислить произведение элементов вектора, не превосходящих среднее арифметическое значение его элементов.
2. Подсчитать число нулей и единиц в заданной матрице.
3. Определить количество положительных элементов вектора, расположенных между его максимальным и минимальным элементами.
4. Просуммировать отрицательные элементы матрицы, лежащие ниже главной диагонали.
5. Заменить положительные элементы вектора суммой всех его отрицательных элементов.
6. Заполнить квадратную матрицу A, каждый элемент которой a_{ij} определяется следующим образом:

$$a_{ij} = \begin{cases} i - j, & i > j \\ i + j, & i = j \\ i^2 + j^2, & i < j \end{cases}.$$

7. Вычислить сумму:

$$s(x) = \sum_{i=1}^n \sum_{j=1}^m \frac{x^{i+j}}{(i+j)^2}.$$

8. Для матрицы $A = a_{ij}$ размера n на m найти значение выражения

$$w(x) = \sum_{i=1}^n \prod_{j=1}^m a_{ij}.$$

9. По заданному x найти максимальное значение n , для которого следующая сумма не превосходит 100:

$$s(x) = \sum_{k=1}^n kx^k$$

10. Вычислить сумму

$$s(x) = \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

с заданной точностью ε . Суммировать следует пока модуль отношения текущего слагаемого к уже накопленной части суммы превосходит ε . Сравнить результат с точным значением, построив графики e^x и $s(x)$ для $x \in [0; 5]$.

Лабораторная работа 6. Работа со строками.

Теоретический материал.

Переменные, содержащие строки, будем называть строковыми переменными. Для ограничения строки используются апострофы, например, оператор присваивания

```
--> str='Hello, World!'
```

приводит к образованию строковой переменной

```
str =
```

```
Hello, World!
```

Длина строковой переменной, т.е. число символов в ней, находится при помощи функции `length`. Допустимо сцепление строк как вектор-строк с использованием квадратных скобок. Создайте еще одну строковую переменную `str1`, содержащую текст 'My name is Igor.', и осуществите сцепление:

```
--> strnew=[str str1]
```

```
strnew =
```

```
Hello, World! My name is Igor.
```

Обратите внимание, что для разделения сцепляемых строк в квадратных скобках следует использовать пробел (или запятую). Применение точки с запятой приведет к образованию вектора-столбца, состоящего из этих строк.

Сцепление строк может быть проведено как с использованием квадратных скобок, так и при помощи функции `strcat([s1,s2])`. Входными аргументами `strcat` являются сцепляемые строки, их число неограничено, а результат возвращается в выходном аргументе.

Поиск позиций вхождения подстроки в строку производится при помощи функции `strindex`. Ее входными аргументами являются строка и подстрока, а выходным — вектор позиций, начиная с которых подстрока входит в строку, например:

```
--> s='abcbcddefbcc';
```

```
--> s1='bc';
```

```
--> p=strindex(s,s1)
```

```
p =
```

```
2 4 10
```

Функция `strcmp` предназначена для сравнения двух строк, которые указываются во входных аргументах: `strcmp(s1,s2)`. Значение больше нуля указывает, что первый символ, который не соответствует, имеет большее значение в `string1`, чем в `string2`, а значение меньше нуля указывает обратное. Для того, чтобы проводить сравнение без учета регистра, т.е., например, символы 'A' и 'a' считаются одинаковыми, необходимо использовать еще один параметр в функции `strcmp(s1,s2, 'i')`.

Для замены в строке одной подстроки на другую служит функция `strsubst`. Пусть, например, требуется заменить в строке `str` подстроку `s1` на `s2` и записать обновленную строку в `strnew`. Тогда вызов функции `strrep` должен выглядеть так: `strnew= strsubst (str,s1,s2)`. Здесь важен порядок аргументов.

Преобразование всех букв строки в строчные (прописные) и наоборот производит функция `convstr(str)`.

Перейдем теперь к изучению массивов строк. Можно считать, что массив строк является вектор-столбцом, каждый элемент которого есть строка, причем длины всех строк одинаковы. В результате получается прямоугольная матрица, состоящая из символов. Например, для переменных `s1='March'`, `s2='April'`, `s3='May'`, операция `S=[s1; s2]` и приводит к массиву строк:

```
S =
```

```
March
```

```
April
```

Аналогичное объединение трех строк $S=[s1; s2; s3]$ создаст вектор-столбец, состоящий из 3 элементов.

Также, для создания массива строк можно использовать функцию `char`:

```
--> S=char(s1,s2,s3)
```

```
S =
```

```
March
```

```
April
```

```
May
```

Для определения размеров массива строк, так же, как и любых массивов, используется `size`.

Для того, чтобы создать матрицу строк, мы можем использовать символ « " » и обычный синтаксис для матриц. В следующем примере мы создаём матрицу строк размером 2 x 3.

```
-->x = [ " 11111 " " 22 " " 333 " ; " 4444 " " 5 " " 666 " ]
```

```
x =
```

```
! 11111    22    333    !
```

```
!          !
```

```
! 4444    5    666    !
```

Функция `size` возвращает размер матрицы, а функция `length`, с другой стороны, возвращает число символов в каждом элементе матрицы.

Для преобразования входного аргумента в строку используется функция `string`. В следующем примере мы определим вектор-строку и используем функцию `string` для преобразования его в строку. Затем мы используем функцию `typeof` и проверим, что переменная `str` действительно является строкой. Наконец, мы используем функцию `size` и проверим, что переменная `str` является матрицей строк размером 1 x 5.

```
-->x = [1 2 3 4 5];
```

```
-->str = string ( x )
```

```
str =
```

```
!1 2 3 4 5 !
```

```
-->typeof ( str )
```

```
ans =
```

```
string
```

```
-->size ( str )
```

```
ans =
```

```
1. 5.
```

Листинг 7.1. Файл-функция для перестановки символов в строке

```
function sout=strinv(s)
```

```
L=length(s);
```

```
for k=1:L
```

```
sout(L-k+1)=s(k);
```

```
end
```

Список команд, которые связаны со строками Scilab'а представлены в таблице:

string	преобразование в строку
sci2exp	преобразовать выражение в строку
ascii	преобразовать строку в коды ascii (и обратно)
blanks	создать строку из пробелов
convstr	преобразовать регистры символов
emptystr	строка нулевой длины
grep	найти соответствия строки в векторе строк
justify	выравнивать символы в столбцах матрицы
length	длина объекта
part	выделение части строки
regexp	найти подстроки, которые соответствуют строке регулярного выражения
strcat	конкатенировать символьные строки
strchr	найти первую встречу символа в строке
strcmp	сравнить символьные строки
strcmpi	сравнить символьные строки (независимо от регистра)
strcspn	получить интервал до указанного символа в строке
strindex	найти позицию символьной строки в другой строке
stripblanks	обрезает пробелы (и табуляторы) в начале и в конце строк
strncpy	копирует символы из строк
strrchr	найти последнюю встречу символа в строке
strrev	возвращает строку задом наперёд
strsplit	разбивает строку на векторы строк
strspn	получить интервал символов в строке
strstr	определить позицию подстроки
strsubst	заменить символьную строку другой символьной строкой
strtod	преобразование строки в число типа double
strtok	разделение строки на лексемы
tokenpos	возвращает позиции лексем в символьной строке
tokens	возвращает лексемы символьной строки
str2code	возвращает целочисленные коды Scilab, связанные с символьной строкой
code2str	возвращает символьную строку, связанную с целочисленными кодами Scilab

Задания для самостоятельной работы

Написать файл-функцию для решения поставленной задачи.

Варианты

1. Подсчитать число вхождений подстроки в строку.
2. Найти количество пробелов в строке.
3. Определить количество цифр в строке.
4. Удалить идущие подряд одинаковые символы в строке.
5. Заменить идущие подряд одинаковые символы в строке на один.
6. Строка является предложением, в котором слова разделены пробелами. Переставить первое и последнее слово.
7. Образовать строку, состоящую из первых букв строк, входящих в массив строк.
8. Вывести номера одинаковых строк в массиве строк.
9. Определить количество символов в каждой строке массива строк без учета пробелов
10. По заданному массиву строк образовать новый, исключив повторяющиеся строки.

Лабораторная работа 7. Массивы структур и массивы ячеек.

Теоретический материал.

Массивы Scilab могут состоять не только из чисел или символов, но содержать и более сложно организованную информацию. Пусть необходимо хранить и обрабатывать информацию о группе из пяти студентов. Данные о каждом студенте включают в себя:

1. фамилию;
2. имя;
3. год рождения;
4. экзаменационные оценки по четырем предметам.

Структура данных, относящихся к каждому из студентов, одинакова (имеет одинаковые поля), а содержание структуры (значения полей структуры) индивидуально для каждого из студентов. Для хранения в Scilab такой однотипно организованной информации предназначен массив структур. Назовем массив структур GR521, а его поля: Fam, Name, Year и Marks.

Обращение к структурам массива производится при помощи индексации, например: GR521(4) — четвертая структура массива GR521. Названия полей отделяются от структуры при помощи точки. Скажем, для получения имени второго студента, следует воспользоваться обращением GR521(2).Name. Заметьте, что поля Fam и Name содержат строки, поле Year — число, а Marks — вектор длины четыре (поскольку записаны оценки по четырем предметам). Создайте файл-программу fillinfo для заполнения массива структур GR521 (см. листинг 8.1).

После запуска fillinfo в рабочей среде образовался массив структур GR521. Функция size позволяет получить его размеры, в данном случае пять на один. Длину одномерного массива структур возвращает функция length. Введите имя массива в командную строку и нажмите <Enter> — содержимое каждой структуры не отображается, выводятся только сведения о размере массива и названия полей. Для отображения в командном окне содержимого каждой структуры следует обратиться непосредственно к ней, например: GR521(2) или disp(GR521(2)).

Листинг 8.1. Файл-программа fillinfo для заполнения массива структур

```
GR521(1).Fam='Алексеев'; GR521(1).Name='Иван';  
GR521(1).Year=1982; GR521(1).Marks=[4 5 5 4];  
GR521(2).Fam='Иванов'; GR521(2).Name='Сергей';  
GR521(2).Year=1981; GR521(2).Marks=[3 4 4 5];  
GR521(3).Fam='Николаев'; GR521(3).Name='Олег';  
GR521(3).Year=1981; GR521(3).Marks=[5 5 5 5];  
GR521(4).Fam='Петрова'; GR521(4).Name='Анна';  
GR521(4).Year=1982; GR521(4).Marks=[5 5 5 4];  
GR521(5).Fam='Федорова'; GR521(5).Name='Елена';  
GR521(5).Year=1982; GR521(5).Marks=[3 3 3 4];
```

При работе с массивами структур требуется производить некоторые операции либо над структурами, либо с их содержимым. Элемент массива структур всегда можно выделить в отдельную структуру и наоборот, заменить его на другую структуру (с аналогичным набором полей). Например, для перестановки первых двух структур в массиве GR521 достаточно использовать вспомогательную структуру:

```
--> H=GR521(2);  
--> GR521(2)=GR521(1);  
--> GR521(1)=H;
```

Доступ к данным структур осуществляется при помощи полей, причем обязательно учитывать, что именно содержит поле: строку, число или массив. Предположим, что требуется по заданной структуре вида GR521 сформировать массив строк с фамилиями и именами каждого из студентов. Файл-функция namesgroup, приведенная на листинге 8.2,

решает поставленную задачу. Предполагается, что входной аргумент GR является структурой вида GR521. Сначала в строковые переменные f и n выделяются фамилия и имя первого студента, из которых формируется строка N. Затем в цикле из каждой структуры массива GR извлекаются фамилия и имя текущего студента, они объединяются в строку str, которая добавляется в массив строк при помощи функции char.

Листинг 8.2. Файл-функция, формирующая массив строк с фамилиями и именами

```
function N=namesgroup(GR)
f=GR(1).Fam;
n=GR(1).Name;
N=[f, ' ', n];
for k=2:length(GR)-1
f=GR(k).Fam;
n=GR(k).Name;
str=[f, ' ', n];
N=char(N, str);
end
```

Проверьте работу файл-функции namesgroup, вызвав ее от массива GR521 с выходным аргументом NAMES, а затем отобразите содержимое NAMES в командное окно при помощи функции disp:

```
--> NAMES=namesgroup(GR521);
--> disp(NAMES)
NAMES =
!Иванов Сергей      !
!                   !
!Алексеев Иван     !
!                   !
!Николаев Олег     !
!                   !
!Петрова Анна      !
!                   !
!Федорова Елена   !
```

Структуры массива можно дополнить новым полем, для чего следует присвоить значение этому полю в какой-нибудь структуре массива, например, первой. Добавьте в массив GR521 поле NBook, предназначенное для хранения номера зачетной книжки студента:

```
--> GR521(1).NBook=599001;
```

Созданное поле, разумеется, образуется во всех структурах массива, но только в первой из них будет содержать заданное значение (проверьте!). Поле NBook остальных структур массива следует заполнить отдельно.

Для удаления созданного поля воспользуемся функцией null:

```
-->GR521.NBook = null();
```

Функция struct заполняет структуру по строкам с названием полей. Входными ее аргументами являются пары 'название поля' - значение, а выходным — структура:

```
--> GR521(6)=struct('Fam','Комов','Name','Петр','Year',1980,...
'Marks',[3 5 4 4],'NBook',59908)
```

Определение названий полей в заданном массиве структур или структуре производится при помощи функции fieldnames. Ее входным аргументом является массив структур или структура, а выходным — названия полей, записанные в массив ячеек. Про массив ячеек сказано ниже, приведем здесь только примеры, демонстрирующие получение названий полей с использованием fieldnames:

```
--> Fields=fieldnames(GR521);
--> field3=Fields(3)
```

```
field3 =  
'Year'
```

Массив ячеек является самым универсальным способом хранения разнородных данных. Его элементами могут быть числа, числовые массивы, строки, структуры и массивы строк и структур. Присваивание значений ячейкам массива требует заключения индексов в фигурные скобки, а при обращении к ячейкам можно использовать как фигурные, так и круглые скобки. Листинг 8.3 содержит пример заполнения двумерного массива ячеек CMAS размера два на два. Ячейка с номером (1,1) содержит матрицу, (1,2) — строку, (2,1) — массив строк, (2,2) — структуру с полями Data и Time.

В отличие от массивов, к элементу cell нельзя обратиться по индексам для заполнения, как мы это знаем. Чтобы заполнить позицию в cell-матрице, нужно воспользоваться специальным служебным словом

Листинг 8.3. Заполнение массива ячеек

```
CMAS = cell(2,2);  
CMAS(1,1).entries=[-2.2 0.9; 5.6 -8.3];  
CMAS(1,2).entries='This is a string';  
CMAS(2,1).entries=char('first string','second string');  
CMAS(2,2).entries.Data=[3.981 9.765 4.442 0.003];  
CMAS(2,2).entries.Time=[0.11 0.12 0.13 0.14];
```

Для отображения данных, хранящихся в массиве ячеек, также, используется функция *entries*.

```
-->CMAS(2,2).entries.Data
```

```
ans =  
    3.981    9.765    4.442    0.003
```

Функция *iscellstr* проверяет, является ли значение строкой или массивом строк. *iscellstr*(CMAS(1,2)) — равно T (true), если CMAS(1,2) является строкой или массивом строк, F – в противном случае.

Задания для самостоятельной работы

Часть 1. Задан массив структур вида GR521 (см. выше) с информацией о группе студентов. Написать файл-функцию для решения следующей задачи.

Варианты

1. Подсчитать средний балл каждого студента и вывести столбцевую диаграмму успеваемости.
2. Найти фамилию наиболее успевающего студента.
3. Сформировать матрицу, строки которой содержат оценки каждого из студентов.
4. Определить, есть ли в группе студент с заданной фамилией.
5. Расположить структуры массива в соответствии с успеваемостью студентов.

Часть 2. Задан одномерный массив ячеек, который может содержать данные различных типов. Написать файл-функцию для решения следующей задачи: объединить в один массив все строки и массивы строк, входящие в массив ячеек.

Лабораторная работа 8. Текстовые файлы.

Теоретический материал.

Работа с текстовыми файлами состоит из трех этапов:

1. открытие файла;
2. считывание или запись данных;
3. закрытие файла.

Для открытия файла служит функция **mopen**, которая вызывается с двумя входными аргументами: именем файла и строкой, задающей способ доступа к файлу. Выходным аргументом **mopen** является дескриптор файла, т.е. переменная, которая впоследствии используется при любом обращении к файлу. Функция **mopen** возвращает значения от -1 до -5 , если при открытии файла возникла ошибка. Существует несколько основных режимов открытия файла:

- **r** открывает файл для чтения (по умолчанию). Файл должен существовать, в противном случае ничего не получится.
- **w** открывает файл на запись. Если этот файл существует, то его содержимое будет уничтожено.
- **a** открывает файл для добавления записи. Создает файл если он не существует.
- **r+** открывает файл как для чтения, так и для записи. Файл должен существовать, иначе ничего не получится.
- **w+** открывает файл как для чтения, так и для записи. Если файл существует, то его содержимое будет уничтожено.
- **a+** открывает файл как на чтение, так и на добавление записи. Создает файл если он не существует.

После открытия файла появляется возможность считывать из него информацию, или заносить ее в файл. По окончании работы с файлом необходимо закрыть его при помощи **mclose(f)**.

Построчное считывание информации из текстового файла производится при помощи функции **mgetl**. Входным аргументом **fgetl** является идентификатор файла и количество строк, которые необходимо считать, а выходным — сами строки. Каждый вызов **fgetl** приводит к считыванию строк и переводу текущей позиции в файле на начало следующей строки (если количество строк не задано, то функция считает из файла все строки и текущая позиция в файле будет установлена на конец файла). Команды, приведенные на листинге 9.1, считывают из файла `myfile.dat` первые три строки в переменную `str`.

Листинг 9.1. Считывание трех первых строк из текстового файла

```
f=mopen('myfile.dat','r');  
str=mgetl(f, 3);  
mclose(f);
```

Лучше всего перед считыванием проверять, не является ли текущая позиция в файле последней. Для этого предназначена функция **meof**, которая вызывается от идентификатора файла и возвращает ненулевое значение, если текущая позиция последняя и ноль, в противном случае. Обычно последовательное считывание организуется при помощи цикла `while`.

Строки записываются в текстовый файл при помощи функции **mfprintf**, ее первым входным аргументом является идентификатор файла, а вторым — добавляемая строка. Символ `\n` служит для перевода строки. Если поместить его в конец добавляемой строки, то следующая команда **mfprintf** будет осуществлять вывод в файл с новой строки, а если `\n` находится в начале, то текущая команда **mfprintf** выведет текст с новой строки.

Например, последовательность команд (листинг 9.3) приведет к появлению в файле `my.txt` текста из двух строк (листинг 9.4).

Листинг 9.3. Вывод строк в текстовый файл

```
f=fopen('my.txt','w+');
mfprintf(f,'текст ');
mfprintf(f,'еще текст, \n');
mfprintf(f,'а этот текст с новой строки \n');
fclose(f);
```

Листинг 9.4. Результат работы операторов листинга 9.3

текст еще текст

а этот текст с новой строки

Аналогичного результата можно добиться, переместив `\n` из конца строки второй функции `mfprintf` в начало строки третьей функции `mfprintf`. Аргументом `mfprintf` может быть не только строка, но и строковая переменная. В этом случае для обеспечения вывода с новой строки ее следует сцепить со строкой `\n` (`mfprintf(f,str+'\n');`)

Строка, предназначенная для вывода в текстовый файл, может содержать как текст, так и числа. Часто требуется выделить определенное количество позиций под число и вывести его в экспоненциальном виде или с плавающей точкой и с заданным количеством цифр после десятичной точки. Здесь не обойтись без форматного вывода при помощи `mfprintf`, обращение к которой имеет вид:

fprintf(идентификатор файла, 'форматы', список переменных)

В списке переменных могут быть как числовые переменные (или числа), так и строковые переменные (или строки). Второй аргумент является строкой специального вида с форматами, в которых будут выводиться все элементы из списка. Каждый формат начинается со знака процента. Для вывода строк используется формат `s`, а для вывода чисел — `f` (с плавающей точкой) или `e` (экспоненциальный). Число перед `s` указывает количество позиций, отводимых под вывод строки. При выводе значений числовых переменных перед `f` или `e` ставятся два числа, разделенных точкой. Первое из них означает количество позиций, выделяемых под все значение переменной, а второе — количество знаков после десятичной точки. Таким образом, под каждый из элементов списка отводится поле определенной длины, выравнивание в котором по умолчанию производится по правому краю. Для выравнивания по левому краю следует после знака процента поставить знак минус. Ниже приведены варианты вызова `mfprintf` и результаты, незаполненные позиции после вывода (пробелы) обозначены символом `o`.

```
fprintf(f,'%6.2f', pi) oo3.14
fprintf(f,'% -6.2f', pi) 3.14oo
fprintf(f,'%14.4e', exp(-5)) oo6.7379e-003
fprintf(f,'% -14.4e', exp(-5)) 6.7379e-003ooo
fprintf(f,'%8s', 'текст') oooтекст
fprintf(f,'% -8s', 'текст') текстooo
```

При одновременном выводе чисел и текста пробелы могут появляться из-за неполностью заполненных полей, выделенных как под числа, так и под строки. Приведенные ниже операторы и результат их выполнения демонстрируют расположение полей в строке текстового файла. Символ `o` по-прежнему обозначает пробелы.

```
x=0.55;
fprintf(f,'% -5s %6.2f %6s %20.8e', 'x=',x,'y=',exp(x))
x=ooooo0.55oooooy=ooooo1.73325302e+000
```

Форматный вывод удобен при формировании файла с таблицей результатов. Предположим, что необходимо записать в файл `f.dat` таблицу значений функции

$f(x) = x^2 \sin x$ для заданного числа n значений $x \in [a, b]$, отстоящих друг от друга на одинаковое расстояние. Файл с таблицей значений должен иметь такую структуру, как показано на листинге 9.5.

Листинг 9.5. Текстовый файл с таблицей значений функции

```
f(0.65)=2.55691256e-001  
f(0.75)=3.83421803e-001  
f(0.85)=5.42800093e-001  
f(0.95)=7.34107493e-001  
f(1.05)=9.56334106e-001
```

Очевидно, что следует организовать цикл от начального значения аргумента до конечного с шагом, соответствующим заданному числу точек, а внутри цикла вызывать `mfprintf` с подходящим списком вывода и форматами. Символ `\n`, предназначенный для перевода строки, помещается в конец строки с форматами (см. листинг 9.6) .

Листинг 9.6. Файл-функция `tab`, выводящая таблицу значений функции

```
function tab(a, b, n)  
h=(b-a)/(n-1);  
f=mopen('f.dat','wt');  
for x=a:h:b  
    fprintf(f,'%2s%4.2f%2s%15.8e\n', f('x')=',x^2*sin(x)')  
end  
mclose(f);
```

Обратимся теперь к считыванию данных из текстового файла. Функция `mfscanf` осуществляет обратное действие по отношению к `mfprintf`. Каждый вызов `mfscanf` приводит к занесению данных, начинающихся с текущей позиции, в переменную. Тип переменной определяется заданным форматом. В общем случае, обращение к `mfscanf` имеет вид:

`a=mfscanf(идентификатор файла, 'формат')`

Для считывания строк используется формат `%s`, для целых чисел — `%d`, а для вещественных — `%g`.

Предположим, что файл `exper.dat` содержит текст, приведенный на листинге 9.7. Данные отделены друг от друга пробелами. Требуется считать дату проведения эксперимента (число, месяц и год) в подходящие переменные, а результаты занести в числовые массивы `TIME` и `DAT`.

Листинг 9.7. Файл с данными `exper.dat`

Результаты экспериментов 10 мая 2002

```
t= 0.1 0.2 0.3 0.4 0.5  
G= 3.02 3.05 2.99 2.84 3.11
```

Считывание разнородных данных (числа, строки, массивы) требует контроля, который достигается применением форматов. Первые два элемента файла `exper.dat` являются строками, следовательно можно сразу занести их в одну строковую переменную `head`. Очевидно, надо указать формат `%s` дважды (два считываемых элемента). Далее требуется считать целое число 10, строку 'мая' и снова целое число 2002. Обратите внимание, что перед заполнением массива `TIME` еще придется предусмотреть считывание строки 't='. Теперь все готово для занесения пяти вещественных чисел в вектор-столбец `TIME` при помощи формата `%g`. Данные из последней строки файла `exper.dat` извлекаются аналогичным образом (листинг 9.8).

Листинг 9.8. Применение `mfscanf` для считывания данных

```
f=mopen('exper.dat','r');  
head=mfscanf(f,'%s%s')  
data=mfscanf(f,'%d')  
month=mfscanf(f,'%s')  
year=mfscanf(f,'%d')  
str1=mfscanf(f,'%s')  
TIME=mfscanf(f,'%g%g%g%g%g')  
str2=mfscanf(f,'%s')  
DAT=mfscanf(f,'%g%g%g%g%g')
```

```
mclose(f);
```

Информация в текстовом файле может быть представлена в виде таблицы. Для считывания такой информации следует использовать массив структур с подходящим набором полей. Считывание всей информации реализуется в цикле `while`, в условии которого производится проверка на достижение конца файла при помощи функции `feof`.

Рассмотрим считывание числовых данных из файла `matr.txt` (листинг 9.9).

Листинг 9.9. Текстовый файл `matr.txt` с матрицей

```
1.1 2.2 3.3 4.4
```

```
5.5 6.6 7.7 8.8
```

```
0.2 0.4 0.6 0.8
```

Считываем значения матрицы построчно. Затем полученные строки соединяем в одну матрицу, получая матрицу размером [3 4]:

Листинг 9.10. Считывание матрицы из текстового файла

```
f=fopen('matr.txt','r');
```

```
M1=mfscanf(f,'%g%g%g%g')
```

```
M2=mfscanf(f,'%g%g%g%g')
```

```
M3=mfscanf(f,'%g%g%g%g')
```

```
M = [M1; M2; M3]
```

```
mclose(f);
```

Обратите внимание, что массив может иметь размеры не только 3 на 4. Поскольку данные считываются подряд, то они могут быть занесены и в массив 2 на 6, и 4 на 3 и т. д.

Задания для самостоятельной работы

Часть 1. Написать файл-функцию для считывания данных из файла в структуру или массив структур с подходящими полями.

Вариант 1. Алексеев Сергей 1980 5 4 4 5 3 5

Иванов Константин 1981 3 4 3 4 3 5

Петров Олег 1980 5 5 5 4 4 5

Вариант 2. 21 марта 2002 0.56 0.58 0.49 0.44

23 марта 2002 0.36 0.32 0.28 0.25

25 марта 2002 1.62 1.68 1.71 1.91

Вариант 3. 195251 СПб Политехническая 29

195256 СПб Науки 49

195256 СПб Науки 24

Вариант 4. Результаты наблюдений

Time= 0.0 0.1 0.2 0.3 0.4 0.5 0.6

Mass=

2.1 2.3 2.3 1.9 1.8 2.4 2.9

0.8 0.7 0.5 1.1 3.2 0.3 0.4

Вариант 5. Алексеев Иван 121-22-04

Сидоров Николай 101-21-99

Тимофеев Сергей 570-00-03

(номера телефонов должны быть записаны в поля структур как целые числа).

Часть 2. Считать матрицы и вектора из файла в подходящие по размеру массивы. Обратите внимание, что в файлах содержится рядом две или три матрицы или вектора, их следует занести в разные массивы.

Вариант 1.

0.1 0.2 0.3 9.91

1.9 0.4 0.1 8.01

4.7 5.1 3.9 7.16

Вариант 2.

1.399 2.001 9.921 3.21 0.12
0.129 1.865 8.341 9.33 8.01
9.136 8.401 7.133 3.12 3.22

Вариант 3.

1 2 3 4 99 80
5 6 7 8 33 21
15 90

Вариант 4.

10 20 40 50 12 19 21 32 44
-1 -2 -3 -4 32
10

Вариант 5.

1 2 3 4 100
6 7 8 9 0.1 0.2 0.3 0.4 200
0.5 0.6 0.7 0.8 300